

webMethods Adapter for JDBC Installation and User's Guide

Version 9.0

November 2016

This document applies to webMethods Adapter for JDBC Version 9.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2006-2016 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Table of Contents

About this Guide.....	11
Documentation Installation.....	11
Online Information.....	11
Overview of the Adapter.....	13
About Adapter for JDBC.....	14
Architecture Overview.....	14
Package Management.....	18
Adapter Connections.....	18
Using JDBC Drivers to Connect to Databases.....	19
Transaction Management of Adapter Connections.....	19
Transaction Isolation Level Settings.....	20
Connection Pools.....	20
Run-Time Behavior of Connection Pools.....	21
Built-In Services for Connections.....	21
Adapter Services.....	21
Using Adapter Services.....	24
Using a Connection from the Connection Pool Within a Java or Flow Service.....	26
Changing the Connection Associated with an Adapter Service or Notification at Design Time.....	26
Changing the Connection Associated with an Adapter Service at Run Time.....	26
Changing the User Credentials of a Service's Associated Connection at Run Time.....	27
Adapter Service Transaction Processing.....	27
Adapter Notifications.....	29
Choice of Publish Destinations.....	30
Adapter Notification Templates.....	30
Exactly Once Notification Feature.....	31
Notification Types.....	32
Insert Notifications, Update Notifications, and Delete Notifications.....	32
Using Insert, Update, and Delete Notifications.....	33
Retrieving Old and New Values as Output for an UpdateNotification.....	34
Insert, Update, and Delete Notifications Transaction Processing.....	34
Basic Notifications.....	36
Using Basic Notifications.....	37
Basic Notifications Transaction Processing.....	38
Stored Procedure Notifications.....	40
Using Stored Procedure Notifications.....	40
Stored Procedure Notifications Transaction Processing.....	42
Ordered Notifications.....	45
Considerations when Using Ordered Notifications.....	45
Configuring an Integration Server Trigger and Flow Service.....	48

Using Ordered Notifications.....	48
Ordered Notifications Transaction Processing.....	50
Polling Notification Support in Clusters.....	53
Polling Notifications and States.....	53
Support for Synonyms.....	54
Synonym Support for Oracle Database.....	54
Synonym Support for DB2 UDB.....	55
Forcing a Timeout During Long-Running SQL Operations in Services and Notifications.....	56
Using Version Control Systems to Manage Adapter Elements.....	56
Infrastructure Data Collector Support for Adapter for JDBC.....	57
Viewing the Adapter's Update Level.....	57
Controlling Pagination.....	57
Installing, Upgrading, and Uninstalling Adapter for JDBC.....	59
Overview.....	60
Requirements.....	60
The Integration Server Home Directory.....	60
Installing Adapter for JDBC.....	60
Upgrading to Adapter for JDBC 9.0.....	61
Archiving.....	62
Upgrading.....	62
Reverting.....	62
Migrating adapter services from 6.5 to 9.0.....	63
Uninstalling Adapter for JDBC.....	63
Package Management.....	65
Overview.....	66
Adapter for JDBC Package Management.....	66
Package Dependency Requirements and Guidelines.....	67
Enabling Packages.....	68
Disabling Packages.....	69
Importing and Exporting Packages.....	69
Group Access Control.....	70
Adapter for JDBC in a Clustered Environment.....	70
Polling Notification Support in Integration Server Clusters.....	71
Considerations for Polling Notifications Executing via Scheduled Tasks.....	71
Adapter Service Support in Clusters.....	72
Replicating Packages to Integration Servers.....	72
Disabling the Redirection of Administrative Services.....	72
Clustering Considerations and Requirements.....	73
Requirements for Each Integration Server in a Cluster.....	73
Considerations When Installing Adapter for JDBC Packages.....	74
Considerations When Configuring Connections with Connection Pooling Enabled.....	74
Adapter for JDBC Connections.....	77

Overview.....	78
Before Configuring or Managing Adapter Connections.....	78
Setting the Environment Variable for Oracle JDBC OCI Drivers.....	79
Installing a JDBC Driver on Integration Server.....	79
Configuring Adapter for JDBC Connections.....	80
Required Connection Property Fields.....	93
Dynamically Changing a Service's Connection at Run Time.....	95
Dynamically Changing the User Credentials of a Service's Connection at Run Time.....	96
Viewing Adapter Connection Parameters.....	96
Using Integration Server Administrator to View Adapter Connection Parameters.....	96
Using Designer to View Adapter Connection Parameters.....	97
Editing Adapter Connections.....	97
Copying Adapter Connections.....	98
Deleting Adapter Connections.....	98
Enabling Adapter Connections.....	99
Disabling Adapter Connections.....	99
Adapter Services.....	101
Overview.....	102
Before Configuring or Managing Adapter Services.....	102
Configuring SelectSQL Services.....	103
Configuring InsertSQL Services.....	107
Configuring UpdateSQL Services.....	110
Configuring BatchInsertSQL Services.....	114
Configuring BatchUpdateSQL Services.....	117
Configuring DeleteSQL Services.....	122
Configuring CustomSQL Services.....	124
Considerations before using Fill in all rows to the table feature in CustomSQL Adapter Service.....	124
Creating a CustomSQL service.....	126
Configuring DynamicSQL Services.....	129
Using Input and Output Parameters.....	129
Configuring a DynamicSQL Statement.....	129
Creating a DynamicSQL Service.....	130
Configuring StoredProcedure Services.....	133
Considerations when Configuring StoredProcedure Adapter Services.....	133
Creating StoredProcedure Adapter Services.....	133
Specifying the Maximum Number of Result Set Indexes for a StoredProcedure Adapter Service.....	137
Configuring StoredProcedureWithSignature Services.....	137
Considerations when Configuring StoredProcedureWithSignature Adapter Services.....	138
Creating StoredProcedureWithSignature Adapter Services.....	138
Configuring ExecuteService Services.....	144
Considerations When Creating a Java or Flow Service that Uses a Connection from the Connection Pool.....	145
Testing Adapter Services.....	146

Viewing Adapter Services.....	147
Editing Adapter Services.....	147
Deleting Adapter Services.....	148
Validating Adapter Service Values.....	148
Validate Data for a Single Adapter Service.....	149
Validating Data for All Adapter Services.....	149
Reloading Adapter Values.....	149
Reloading the Values for a Single Adapter Service.....	150
Reloading the Values for All Adapter Services.....	150
Adapter Notifications.....	151
Overview.....	152
Before Configuring or Managing Notifications.....	152
Configuring InsertNotifications.....	153
Considerations when Configuring InsertNotifications.....	153
Creating an InsertNotification.....	153
Configuring UpdateNotifications.....	158
Considerations when Configuring UpdateNotifications.....	158
Creating an UpdateNotification.....	159
Configuring DeleteNotifications.....	164
Considerations when Configuring DeleteNotifications.....	164
Creating a DeleteNotification.....	165
Configuring BasicNotifications.....	170
Configuring StoredProcedureNotifications.....	175
Configuring StoredProcedureNotificationWithSignature.....	179
Configuring OrderedNotifications.....	184
Considerations when Configuring OrderedNotifications.....	184
Creating an OrderedNotification.....	185
Managing Polling Notifications.....	191
Using the Exactly Once Notification Feature.....	193
Enabling Exactly Once Notification.....	193
Exporting Configured Adapter Notifications.....	193
Viewing Notifications.....	195
Editing Notifications.....	195
Deleting Notifications.....	196
Validating Adapter Notification Values.....	196
Validate Data for a Single Adapter Notification.....	197
Validating Data for All Adapter Notifications.....	197
Reloading Adapter Values.....	197
Reloading the Values for a Single Adapter Notification.....	198
Reloading the Values for All Adapter Notifications.....	198
Data Type Configuration.....	199
Overview.....	200
The Default Data Type Mapping File.....	200
javaType.....	200

typeMapping.....	201
The Database-Specific Data Type Mapping Files.....	201
extTypeMapping.....	201
typeClassName.....	201
columnTypeMapping.....	201
alternativeName.....	202
includeColumnSize.....	202
resizeFactor.....	202
includePrecision.....	202
Support for SQLXML Data Type.....	202
Logging and Exception Handling.....	203
Overview.....	204
Adapter for JDBC Logging Levels.....	204
Accessing Adapter Logging Levels.....	204
Changing Logging Levels.....	204
Adapter for JDBC Message Logging.....	205
Adapter for JDBC Exception Handling.....	206
AdapterException.....	206
AdapterConnectionException.....	206
SQLException.....	207
Customizing the Adapter's List of Fatal Error Codes.....	207
Overriding the Adapter's List of Fatal Error Codes.....	208
Suppressing the Logging of Errors.....	209
Adapter for JDBC Error Codes.....	210
Data Type Mapping.....	217
JDBC Data Type to Java Data Type Mappings.....	218
Important Considerations When Using BLOB and CLOB Data Types.....	222
Important Considerations When Using the Array and Struct Database Specific Data Types.....	222
Using the SetAsString Data Type in Adapter for JDBC.....	223
JDBC Data Type to Java Data Type Mapping Constraints.....	223
SQL Data Type to JDBC Data Type Mappings.....	224
Advanced Server Type to JDBC Data Type Mappings.....	224
Advanced Server Type Constraints.....	225
Built-In Services.....	227
Overview.....	228
pub.jdbcAdapter:updateNotificationSchema.....	228
Configuring the UpdateNotificationSchema Service.....	230
Setting Input Fields.....	230
Setting All Schemas to the <current schema>.....	230
Setting All Schemas to Schema A.....	231
Changing Occurrences of the <current schema> to Schema A.....	231
Changing Occurrences of Schema A to the <current schema>.....	232

Changing Occurrences of Schema A to Schema A1, and Occurrences of Schema B to the <current schema>.....	232
pub.jdbcAdapter:updateServiceSchema.....	233
Configuring the updateServiceSchema Service.....	234
Setting Input Fields.....	234
Setting All Schemas to the <current schema>.....	234
Setting All Schemas to Schema A.....	235
Changing Occurrences of the <current schema> to Schema A.....	235
Changing Occurrences of Schema A to the <current schema>.....	236
Changing Occurrences of Schema A to Schema A1, and Occurrences of Schema B to the <current schema>.....	236
pub.jdbcAdapter:updateConnectionPassword.....	237
pub.jdbcAdapter:createConnectionNodes.....	237
pub.pollingNotificationUtils:dropDatabaseObjects.....	240
pub.pollingNotificationUtils:getDatabaseObjectsForNotification.....	240
Built-In Transaction Management Services.....	243
Transaction Management Overview.....	244
Transactions.....	244
Transaction Types.....	244
XA Transactions.....	245
Implicit and Explicit Transactions.....	245
Implicit Transactions.....	245
Explicit Transactions.....	246
Implicit and Explicit Transaction Examples.....	247
Flow Example: ValidMixed.....	249
Flow Example: SingleLocalInsert.....	249
Flow Example: ValidMixed2.....	250
Flow Example: ValidDoubleLocal.....	251
Built-In Transaction Management Services.....	252
pub.art.transaction:commitTransaction.....	252
pub.art.transaction:rollbackTransaction.....	253
pub.art.transaction:setTransactionTimeout.....	254
pub.art.transaction:startTransaction.....	254
Changing the Integration Server Transaction Timeout Interval.....	255
Adapter Configuration Parameters Appendix.....	257
Overview.....	258
watt.adapter.JDBC.AutomaticNotification.Joincolumn.Buffer Table.....	258
watt.adapter.JDBC.DateWithTimestamp.....	258
watt.adapter.JDBC.DateWithTimestampAndMilliseconds.....	258
watt.adapter.JDBC.DisableEmptyResult.....	259
watt.adapter.JDBC.StoredProcedure.customRSColNames.....	259
watt.adapter.JDBC.UsePlainString.....	259
Database Driver Known Limitations.....	261

Driver Limitations.....	262
Index.....	273

About this Guide

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

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

- The basic concepts and tasks for working with relational databases
- Creating flow or Java services
- Terminology and basic operations of your operating system
- The setup and operation of webMethods Integration Server.
- How to perform basic tasks with Software AG Designer.

Documentation Installation

For products that are installed using the Software AG Installer, you can also download the product documentation using the Software AG Installer. The documentation is downloaded to a central directory named `_documentation` in the main installation directory (SoftwareAG by default).

Online Information

You can find additional information about Software AG products at the locations listed below.

If you want to...	Go to...
Access the latest version of product documentation.	For Customers “Documentation web site” in Software AG's customer portal For Non-Customers “Documentation web site” in the Technical Communities

If you want to...	Go to...
<p>Find information about product releases and tools that you can use to resolve problems.</p> <p>See the “Knowledge Center” to:</p> <ul style="list-style-type: none"> ■ Read technical articles and papers. ■ Download fixes and service packs (9.0 SP1 and earlier). ■ Learn about critical alerts. <p>See the “Products area” to:</p> <ul style="list-style-type: none"> ■ Download products. ■ Download certified samples. ■ Get information about product availability. ■ Access older versions of product documentation. ■ Submit feature/enhancement requests. 	<p>Empower Product Support website</p> <p>“https://empower.softwareag.com”</p>
<ul style="list-style-type: none"> ■ Access additional articles, demos, and tutorials. ■ Obtain technical information, useful resources, and online discussion forums, moderated by Software AG professionals, to help you do more with Software AG technology. ■ Use the online discussion forums to exchange best practices and chat with other experts. ■ Expand your knowledge about product documentation, code samples, articles, online seminars, and tutorials. ■ Link to external websites that discuss open standards and many web technology topics. ■ See how other customers are streamlining their operations with technology from Software AG. 	<p>Software AG Developer Community</p> <p>“http://communities.softwareag.com/”</p>

1 Overview of the Adapter

■ About Adapter for JDBC	14
■ Architecture Overview	14
■ Package Management	18
■ Adapter Connections	18
■ Adapter Services	21
■ Adapter Notifications	29
■ Support for Synonyms	54
■ Forcing a Timeout During Long-Running SQL Operations in Services and Notifications	56
■ Using Version Control Systems to Manage Adapter Elements	56
■ Infrastructure Data Collector Support for Adapter for JDBC	57
■ Viewing the Adapter's Update Level	57
■ Controlling Pagination	57

About Adapter for JDBC

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

For a list of the database versions, JDBC drivers, and platforms that Adapter for JDBC supports, see *webMethods Adapters System Requirements*.

For a list of known driver limitations, see [“Database Driver Known Limitations” on page 261](#).

Architecture Overview

Adapter for JDBC 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 Integration Server. For detailed installation instructions, see [“Installing, Upgrading, and Uninstalling Adapter for JDBC” on page 59](#). For software requirements, see *webMethods Adapters System Requirements*.

Because Adapter for JDBC 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 Integration Server. For more details, see [“Installing a JDBC Driver on Integration Server” on page 79](#).

Adapter for JDBC enables you to configure the following components:

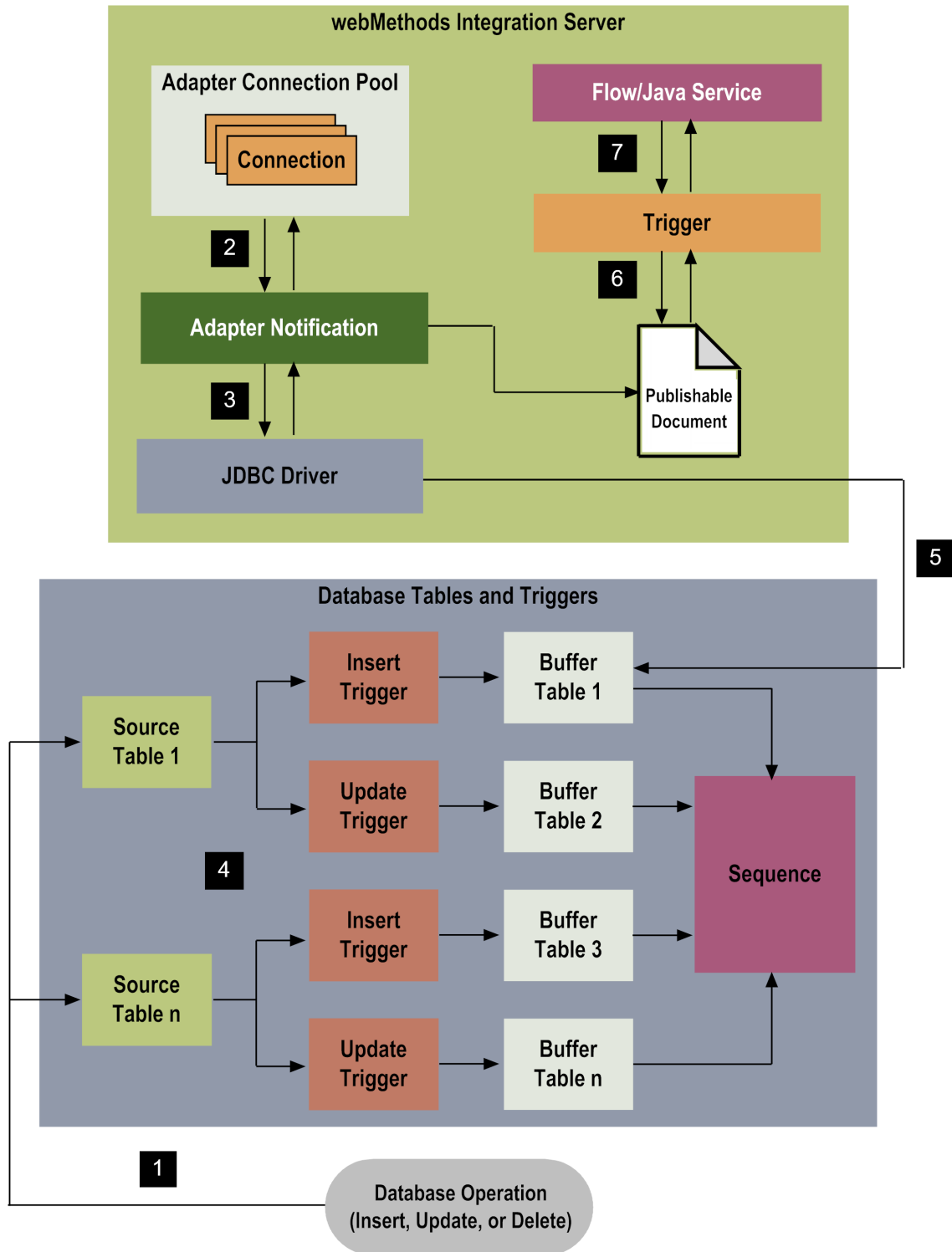
- **Adapter connections.** Enable Integration Server to connect to database systems at run time. You must configure an adapter connection before you can configure adapter services or adapter notifications. For a detailed description of adapter connections, see [“Adapter Connections” on page 18](#).
- **Adapter services.** Enable 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 Adapter for JDBC. For a detailed description of adapter services, see [“Adapter Services” on page 21](#).

- **Adapter notifications.** Monitor a database and notify Integration Server when an action (not initiated by Integration Server) occurs on a particular database table. For example, an adapter notification could notify Integration Server when an update operation was performed on a particular database table. For a detailed description of adapter notifications, see [“Adapter Notifications” on page 29](#).

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.

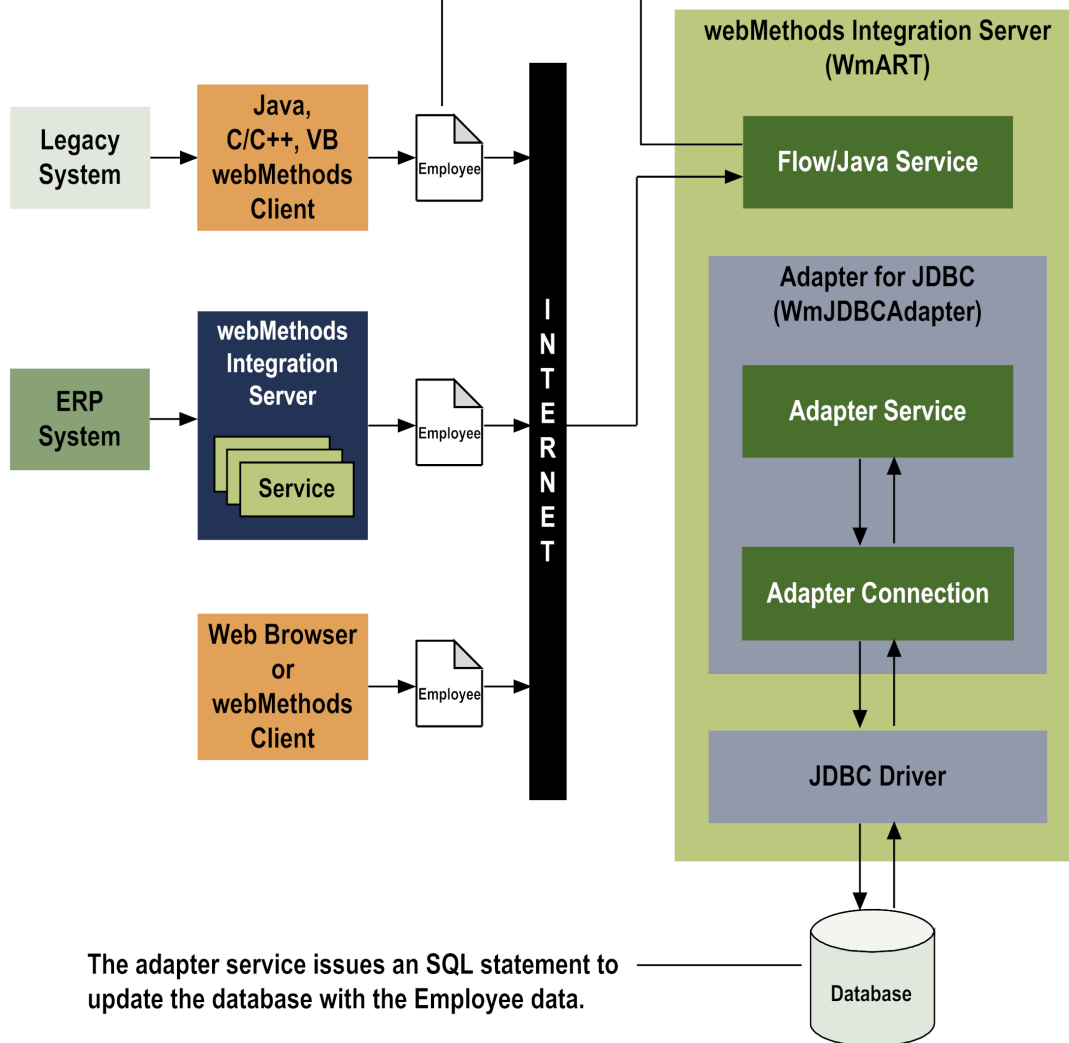
Ordered Notifications



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.

Various backend systems send employee data to Integration Server through an IS client, invoking a service on Integration Server.

A flow or Java service on Integration Server invokes an adapter service created using webMethods Adapter for JDBC. The adapter service uses an adapter connection to connect to the database through a JDBC driver.



The adapter service issues an SQL statement to update the database with the Employee data.

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 Integration Server.

With adapter notifications, you can capture event data from the database and use it to initiate another action within 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. Broker clients could then subscribe to that notification's publishable document.

For more information about the architecture for the different types of adapter notifications, see [“Adapter Notifications” on page 29](#).

Package Management

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

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

- You must create user-defined packages for your connections, adapter services, and notifications. For details, see [“Adapter for JDBC Package Management” on page 66](#).
- You should understand how package dependencies work so you make the best decisions regarding how you manage your adapter services and notifications. For details, see [“Package Dependency Requirements and Guidelines” on page 67](#).
- You control which development groups have access to which adapter services and notifications. For details, see [“Group Access Control” on page 70](#).
- You should understand how clustering, an advanced feature of Integration Server, works to effectively manage your adapter services. For details, see [“Adapter for JDBC in a Clustered Environment” on page 70](#).

Adapter Connections

Adapter for JDBC 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.

Adapter for JDBC connections contain parameters that Integration Server uses to manage connections to the database so that they can be used by the adapter to provide services. You configure connections using Integration Server Administrator. You

must have Integration Server Administrator privileges to access Adapter for JDBC's administrative screens.

For instructions on configuring, viewing, editing, enabling, and disabling Adapter for JDBC connections, see [“Adapter for JDBC Connections” on page 77](#). For information about setting user privileges, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.

Using JDBC Drivers to Connect to Databases

Adapter for JDBC 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. For instructions, see [“Installing a JDBC Driver on Integration Server” on page 79](#).

For more information about the transaction types supported in Adapter for JDBC, see [“Transaction Management of Adapter Connections” on page 19](#).

For a complete list of the JDBC drivers that the adapter supports, see *webMethods Adapters System Requirements*.

For a list of known driver limitations, see [“Database Driver Known Limitations” on page 261](#).

Transaction Management of Adapter Connections

Adapter for JDBC 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 are 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

Transaction Type	Description
	<p>multiple databases. In one transaction boundary, all of the operations on multiple connections are 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.</p> <p>Note: All of the connections involved in a two-phase transaction must support the XA_TRANSACTION transaction type.</p>
<p>Note: Insert Notifications, Update Notifications, Delete Notifications, and Ordered Notifications support LOCAL_TRANSACTION mode only.</p>	

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 transaction types in the preceding table, are managed by the Integration Server transaction manager.

You can also explicitly manage transactions using built-in services. For information about, and examples of, explicitly managing transactions, see [“Built-In Transaction Management Services” on page 243](#).

Transaction Isolation Level Settings

Adapter for JDBC supports the setting of transaction isolation levels on a database. These settings prevent dirty read, repeatable read, and phantom read of the database. The type of isolation levels supported by the adapter depends on the database that you are connecting to. The commonly used isolation levels are TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED, TRANSACTION_REPEATABLE_READ, and TRANSACTION_SERIALIZABLE.

Adapter for JDBC supports all the isolation level settings supported by your database, for example, the isolation level SNAPSHOT for MS SQL Server 2005.

The isolation level settings can be specified in the **Other Properties** field of the Connection Configuration screen, while configuring a connection. For more information about specifying these levels, see [“Configuring Adapter for JDBC Connections” on page 80](#).

Connection Pools

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. Integration Server maintains connection pools in memory. Connection pools improve performance by enabling adapter services to re-use open connections instead of opening new connections.

Run-Time Behavior of Connection Pools

When you enable a connection, 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, 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 Integration Server to obtain a connection, up to the length of time specified in the **Block Timeout** field, until a connection becomes available. Periodically, Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period that you specified in the **Expire Timeout** field.

If initialization of the connection pool fails because of a network connection failure or some other type of exception, you can enable the system to retry the initialization any number of times, at specified intervals. For information about configuring connections, see [“Configuring Adapter for JDBC Connections” on page 80](#).

Built-In Services for Connections

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.

The `setAdapterServiceNodeConnection` and `setPollingNotificationNodeConnection` built-in services enable you to change the connection associated with an adapter service or notification respectively. For more information, see [“Changing the Connection Associated with an Adapter Service or Notification at Design Time” on page 26](#).

For details, see the Integration Server built-in services reference guide for your release. See [“About this Guide”](#) for specific document titles.

Adapter Services

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

then uses adapter connections that you defined earlier to execute the adapter services. For details, see [“Adapter Service Transaction Processing” on page 27](#).

Adapter services are based on templates provided with Adapter for JDBC. 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 Software AG Designer, 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 Designer. Some familiarity with using Designer is required. For more information, see the Designer Service Development online help for your release. See [“About this Guide”](#) for specific document titles.

Adapter for JDBC provides the following adapter service templates:

Adapter Service Type	Adapter Service Template	Description
Select SQL	SelectSQL	Retrieves specified information from a database table and includes a mapping for an output field that stores the number of rows retrieved. For instructions about configuring the service, see “Configuring SelectSQL Services” on page 103 .
Insert SQL	InsertSQL	Inserts new information into a database table. For instructions about configuring the service, see “Configuring InsertSQL Services” on page 107 .
Update SQL	UpdateSQL	Updates the existing information in a database table and includes a mapping for an output field that stores the number of affected rows. For instructions about configuring the service, see “Configuring UpdateSQL Services” on page 110 .
Batch Insert SQL	BatchInsertSQL	Inserts new information into a database table. Use this service when

Adapter Service Type	Adapter Service Template	Description
		<p>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.</p> <p>For instructions about configuring the service, see “Configuring BatchInsertSQL Services” on page 114.</p>
Batch Update SQL	BatchUpdateSQL	<p>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.</p> <p>For instructions about configuring the service, see “Configuring BatchUpdateSQL Services” on page 117.</p>
Delete SQL	DeleteSQL	<p>Deletes rows from a table and includes a mapping for an output field that stores the number of affected rows.</p> <p>For instructions about configuring the service, see “Configuring DeleteSQL Services” on page 122.</p>
Custom SQL	CustomSQL	<p>Defines and executes custom SQL to perform database operations. You can execute almost any SQL statement required by integrations, such as data management statements.</p> <p>For instructions about configuring the service, see on page 124 “Configuring CustomSQL Services”.</p>

Adapter Service Type	Adapter Service Template	Description
Dynamic SQL	DynamicSQL	<p>Defines and executes a SQL statement, part of which you set at run time through the input field.</p> <p>For instructions about configuring the service, see “Configuring DynamicSQL Services” on page 129.</p>
Stored Procedure	StoredProcedure	<p>Calls a stored procedure to perform database operations.</p> <p>For instructions about configuring the service, see “Configuring StoredProcedure Services” on page 133.</p>
Stored Procedure with signature	StoredProcedureWithSignature	<p>Calls a stored procedure to perform database operations. Obtains the stored procedure's parameters by introspecting and listing the signature of the stored procedure at the time you configure the adapter service.</p> <p>For instructions about configuring the service, see “Configuring StoredProcedureWithSignature Services” on page 137.</p>
Execute Service	ExecuteService	<p>Executes a Java or flow service using a JDBC connection object from the Adapter for JDBC connection pool. For more information see “Using a Connection from the Connection Pool Within a Java or Flow Service” on page 26.</p> <p>For instructions about configuring the service, see “Configuring ExecuteService Services” on page 144.</p>

Using Adapter Services

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

For this task...	Use these tools...
<p>1. Create an adapter connection. For details, see “Adapter for JDBC Connections” on page 77.</p>	<p>Integration Server Administrator</p>
<p>2. Select the appropriate adapter service template and configure the adapter service. Depending on the type of adapter service, you specify:</p> <ul style="list-style-type: none"> ■ The adapter connection ■ The database table ■ The SQL expression used to modify or select data, a stored procedure to execute against the database, or a Java or flow service that uses a connection object from the Adapter for JDBC connection pool. ■ The input fields and types as needed ■ The output fields and types as needed <p>The adapter allows you to dynamically configure the data type mapping for a particular database in an XML configuration file. For more information about configuring data types, see “Data Type Configuration” on page 199.</p> <p>For more information about configuring adapter services, see “Adapter Services” on page 101.</p>	<p>Designer</p>
<p>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.</p>	<p>Designer</p>
<p>4. Manage the adapter service.</p> <p>For details, see “Package Management” on page 65, “Adapter Services” on page 101, and “Logging and Exception Handling” on page 203.</p>	<p>Designer and Integration Server Administrator</p>

Using a Connection from the Connection Pool Within a Java or Flow Service

Typically, adapter services use connections from the adapter's connection pool. However, you can also have any custom Java or flow services that perform database operations use Adapter for JDBC connections from the connection pools. By using the `ExecuteService` adapter service template, you can create an adapter service that provides a JDBC connection from a connection pool and then calls the specified Java or flow service. For more information, see [“Configuring ExecuteService Services” on page 144](#).

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

Integration Server provides built-in services that you can use at design time to change the connection associated with an adapter service or notification. The built-in services, `setAdapterServiceNodeConnection` and `setPollingNotificationNodeConnection`, are provided in the WmART package's `pub.art.service` folder and `pub.art.notification` folder, respectively. Using this function, you can change the specific connection associated with an adapter service or an adapter notification at design time so that you do not need to create and maintain multiple adapter services and notifications.

Note: The `setAdapterServiceNodeConnection` and `setPollingNotificationNodeConnection` services can be run at design time only. Do not use them within an Integration Server flow or Java service. You must run the services directly from Designer by selecting a service and running it.

For details, see the Integration Server built-in services reference guide for your release. See [“About this Guide”](#) for specific document titles.

Other built-in services enable you to control connections. For more information, see [“Built-In Services for Connections” on page 21](#).

Changing the Connection Associated with an Adapter Service at Run Time

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 run time 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 run time, see [“Dynamically Changing a Service's Connection at Run Time”](#) on page 95.

Changing the User Credentials of a Service's Associated Connection at Run Time

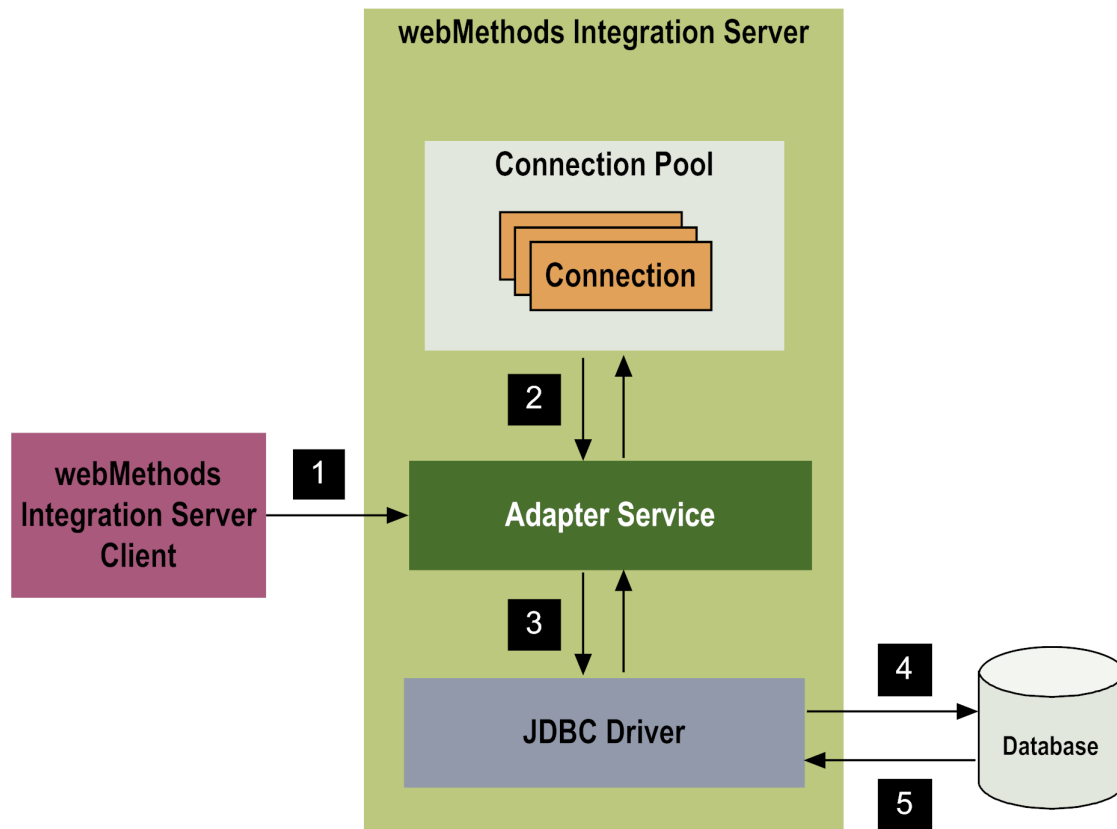
Adapter for JDBC enables you to dynamically change the user credentials of a connection associated with an adapter service at run time. This feature enables you to interact with a database with different user privileges.

For example, consider a service's associated adapter connection that uses an administrator's credentials at design time to define a connection to a database. At run time, you can override the administrator's account credentials with individual user credentials to limit access to the database according to the permission level each user has. This capability also enables you to keep track of the database operations by the user initiating the service.

For more information, see [“Dynamically Changing the User Credentials of a Service's Connection at Run Time”](#) on page 96.

Adapter Service Transaction Processing

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



Step	Description
1	An Integration Server client, typically using a flow or Java service, invokes a Adapter for JDBC service on Integration Server to perform an operation on a database. You configured the adapter service earlier using Designer.
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 Integration Server Administrator.
4	All adapter services except ExecuteService perform a SQL operation against the database.

Step	Description
	<ul style="list-style-type: none"> <li data-bbox="363 323 1365 426">■ For SelectSQL, InsertSQL, UpdateSQL, DeleteSQL, CustomSQL, and DynamicSQL services, the adapter service executes a SQL statement against the database. <li data-bbox="363 447 1365 646">■ 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. <li data-bbox="363 667 1365 737">■ For StoredProcedure and StoredProcedureWithSignature services, the adapter service executes a stored procedure against the database. <li data-bbox="363 758 1365 894">■ The ExecuteService adapter service executes a Java or flow service that needs a connection object from the connection pool. For more information see “Using a Connection from the Connection Pool Within a Java or Flow Service” on page 26.
5	<p data-bbox="363 940 1365 1010">Depending on the adapter service type, such as a SelectSQL service, the adapter service may return data to Integration Server.</p> <ul style="list-style-type: none"> <li data-bbox="363 1031 1365 1100">■ 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. <li data-bbox="363 1268 1365 1409">■ 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 “Logging and Exception Handling” on page 203.

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 Integration Server.

Adapter for JDBC notifications are polling-based. That is, 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 191](#).

Important: Software AG recommends using the same polling notification on multiple Integration Server instances only in an Integration Server cluster. For more information about using polling notifications in a cluster, see [“Polling Notification Support in Clusters” on page 53](#).

Adapter notifications vary somewhat in how they work, depending on the type of the adapter notification. Be sure to review [“Notification Types” on page 32](#) to understand how their operations differ.

Choice of Publish Destinations

You can choose the destination to which asynchronous notifications should publish messages. Specifically, you can choose whether the asynchronous notification templates use JMS APIs to publish messages to Integration Server or webMethods Broker APIs to publish notification messages to webMethods Broker.

Note: To use the JMS protocol with asynchronous notifications, you must first configure a JMS connection alias on Integration Server. For more information, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.

For steps for selecting a publish destination for asynchronous notification messages, see [“Adapter Notifications” on page 151](#).

Adapter Notification Templates

Adapter for JDBC provides the following adapter notification templates:

Notification Type	Notification Template	Description
Insert Notification	InsertNotification	<p>Publishes notification of insert operations on a database table.</p> <p>For instructions, see “Configuring InsertNotifications” on page 153.</p>
Update Notification	UpdateNotification	<p>Publishes notification of update operations on a database table.</p> <p>For instructions, see “Configuring UpdateNotifications” on page 158.</p>
Delete Notification	DeleteNotification	<p>Publishes notification of delete operations on a database table.</p>

Notification Type	Notification Template	Description
		For instructions, see “Configuring DeleteNotifications” on page 164.
Basic Notification	BasicNotification	<p>Polls a database table for data using a SQL Select operation.</p> <p>For instructions, see “Configuring BasicNotifications” on page 170.</p>
Stored Procedure Notification	StoredProcedure Notification	<p>Publishes notification data by calling a stored procedure inside of a database.</p> <p>For instructions, see “Configuring StoredProcedureNotifications” on page 175.</p>
Stored Procedure Notification with Signature	StoredProcedure NotificationWith Signature	<p>Publishes notification data by calling a stored procedure inside of a database. Obtains the stored procedure's parameters by introspecting and listing the signature of the stored procedure at the time you configure the notification.</p>
Ordered Notification	OrderedNotification	<p>Publishes notification data for multiple insert, update, or delete operations on multiple tables for a given database.</p> <p>For instructions, see “Configuring OrderedNotifications” on page 184.</p>

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 is not duplicated even if a failure occurs during processing. This is achieved by assigning unique IDs for each publishable document. After a processing failure, 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.

For more details, see [“Using the Exactly Once Notification Feature”](#) on page 193.

Notification Types

There are seven types of notifications: Insert, Update, Delete, Basic, Stored Procedure, StoredProcedureNotificationWithSignature, 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 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 Integration Server.
- Disable the package containing the enabled notification.
- Reload the package containing the enabled notification.
- Suspend the notification.

In the meantime, the trigger continues to monitor the table and to insert data into the buffer table. Integration Server invokes the enabled notification when it restarts, or when it enables or reloads the package that contains this notification. For more information about how these types of notifications work, see [“Insert, Update, and Delete Notifications Transaction Processing” on page 34](#).

For instructions for configuring this type of adapter notification, see [“Configuring InsertNotifications” on page 153](#), [“Configuring UpdateNotifications” on page 158](#), or [“Configuring DeleteNotifications” on page 164](#).

For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer's guide for your release. See [“About this Guide”](#) for specific document titles.

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. For details, see “Adapter for JDBC Connections” on page 77.	Integration Server Administrator
2. Configure the notification and specify the: <ul style="list-style-type: none"> ■ 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, see the Integration Server publish-subscribe developer’s guide for your release. See “About this Guide” for specific document titles. ■ Output data fields contained in the publishable document ■ Database trigger and buffer table For instructions for configuring notifications, see “Adapter Notifications” on page 151.	Designer
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. For details, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.	Designer
4. Schedule and enable the adapter notification. When you enable the notification:	Integration Server Administrator

For this task...	Use these tools...
<ul style="list-style-type: none"> ■ 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. <p>For instructions for scheduling and enabling notifications, see “Managing Polling Notifications” on page 191.</p>	
<p>5. Manage the notification. For details, see “Package Management” on page 65, “Adapter Notifications” on page 151, and “Logging and Exception Handling” on page 203.</p>	<p>Designer and Integration Server Administrator</p>

Retrieving Old and New Values as Output for an UpdateNotification

Using an UpdateNotification, you can retrieve either the old value, the new value, or both the old and new values from the database table as output values.

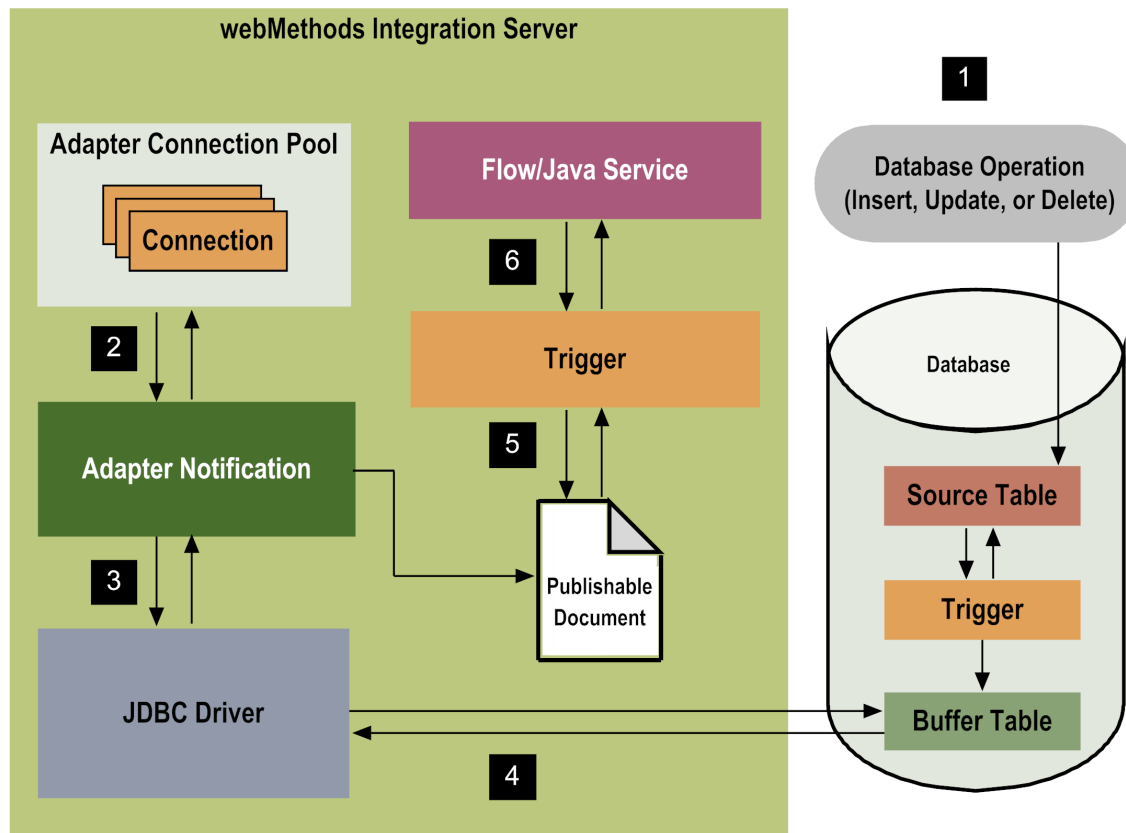
Note: The old value is the value that exists before a value is updated in the selected column of the database table.

You can set the required output value options while configuring the UpdateNotification in the **Select** tab of the UpdateNotification template. For more information about setting the output value options, see the [“Configuring UpdateNotifications”](#) on page 158.

Insert, Update, and Delete Notifications Transaction Processing

The following diagram illustrates what happens when these types of notifications are invoked. 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 Integration Server Administrator.
4	The notification retrieves the rows of data from the buffer table.

Step	Description
	The buffer table holds the data selected by the trigger. While the adapter notification remains enabled, the trigger continues to monitor the database table and insert data into the buffer table.
5	<p>The notification creates the publishable document, which contains a row of data from the buffer table. The notification publishes the publishable document.</p> <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer's guide for your release. See "About this Guide" for specific document titles.</p>
6	<p>Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on Integration Server is invoked to react to the data changes contained in the publishable document.</p> <p>After the data is published, the data is removed from the buffer table.</p>

Basic Notifications

In contrast with Insert Notifications, Update Notifications, and Delete Notifications, Basic Notifications require that you 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 want to delete the data from the buffer table, and you also do not want the adapter to publish the data more than once, you can mark the processed data as published. The notification then only processes the data that is not published. To use this option, you need to:

1. Add a column of CHAR(1) data type with any name in the database table that you are trying to use. This column is required to hold the status of the data, that is, whether the data is processed or not processed.
2. While configuring the Basic Notification, select the newly added column in the **Mark ID Column**, see step 1. For instructions about configuring the Basic Notification, see ["Configuring BasicNotifications" on page 170](#).

For more information about how Basic Notifications work, see ["Basic Notifications Transaction Processing" on page 38](#).

Using Basic Notifications

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

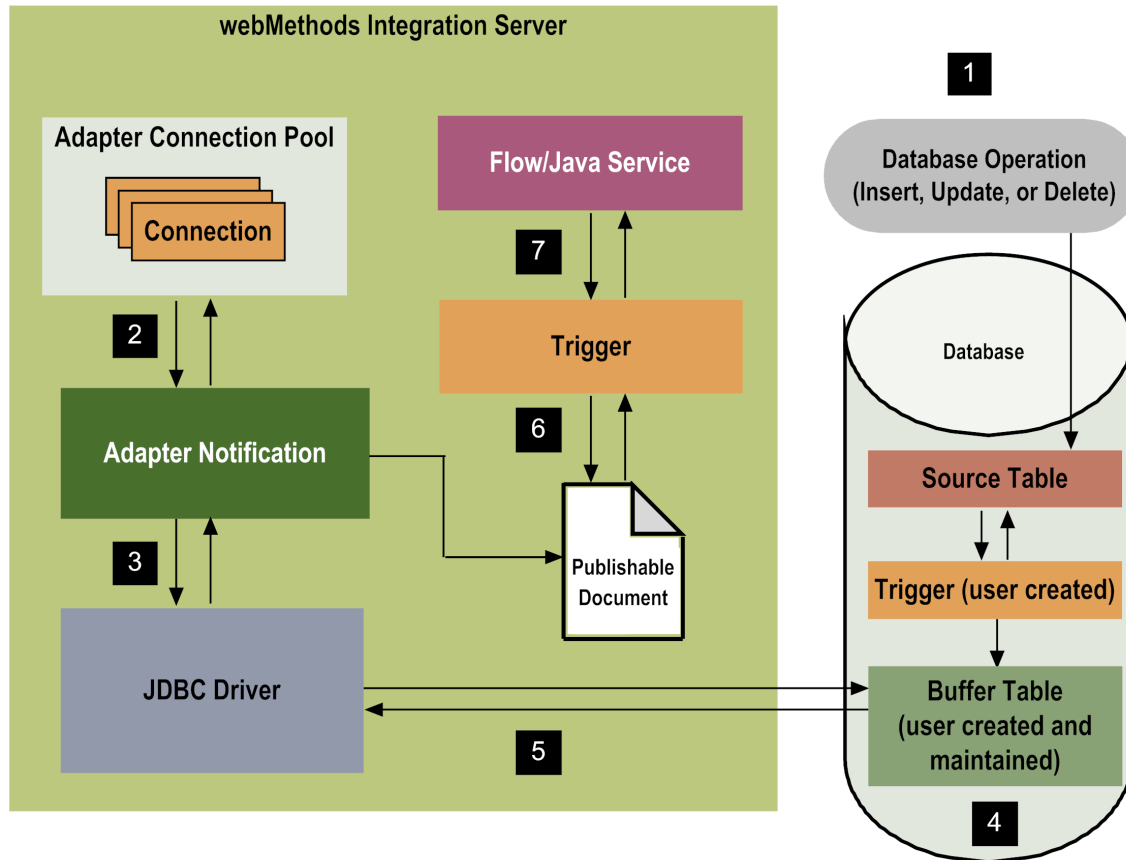
For this task...	Use these tools...
1. If needed, create your own buffer table and database trigger (or other means) to monitor for database changes.	User-defined
2. Create an adapter connection. For details, see “Adapter for JDBC Connections” on page 77 .	Integration Server Administrator
3. Configure the notification and specify the: <ul style="list-style-type: none"> ■ 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. <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See “About this Guide” for specific document titles.</p> <ul style="list-style-type: none"> ■ Output data fields contained in the publishable document <p>For instructions for configuring this type of notification, see “Configuring BasicNotifications” on page 170.</p>	Designer
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. <p>For details, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.</p>	Designer

For this task...	Use these tools...
<p>5. Schedule and enable the adapter notification.</p> <p>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.</p> <p>For instructions for scheduling and enabling notifications, see “Managing Polling Notifications” on page 191.</p>	<p>Integration Server Administrator</p>
<p>6. Manage the notification. For details, see “Package Management” on page 65, “Adapter Notifications” on page 151, and “Logging and Exception Handling” on page 203.</p>	<p>Designer and Integration Server Administrator</p>

Basic Notifications Transaction Processing

The following diagram and steps illustrate what happens when a Basic Notification is invoked. 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.
3	The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using 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

Step	Description
	<p>database changes. The diagram and steps listed here assume you are creating your own buffer table and trigger to monitor for changes.</p> <p>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.</p>
5	The notification retrieves the rows of data from the buffer table.
6	<p>The notification creates the publishable document, which contains a row of data from the buffer table. The notification publishes the publishable document.</p> <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer's guide for your release. See "About this Guide" for specific document titles.</p>
7	<p>Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on Integration Server is invoked to react to the data changes contained in the publishable document.</p> <p>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.</p>

Stored Procedure Notifications

A Stored Procedure Notification calls a stored procedure you created earlier to publish notification data in the notification's publishable documents. For more information about how Stored Procedure Notifications work, see ["Stored Procedure Notifications Transaction Processing"](#) on page 42.

For information about configuring this type of adapter notification, see ["Configuring StoredProcedureNotifications"](#) on page 175.

Note: Stored Procedure Notifications do not support the Exactly Once notification feature because they do not use publishable document unique IDs. For details about this feature, see ["Using the Exactly Once Notification Feature"](#) on page 193.

Using Stored Procedure Notifications

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

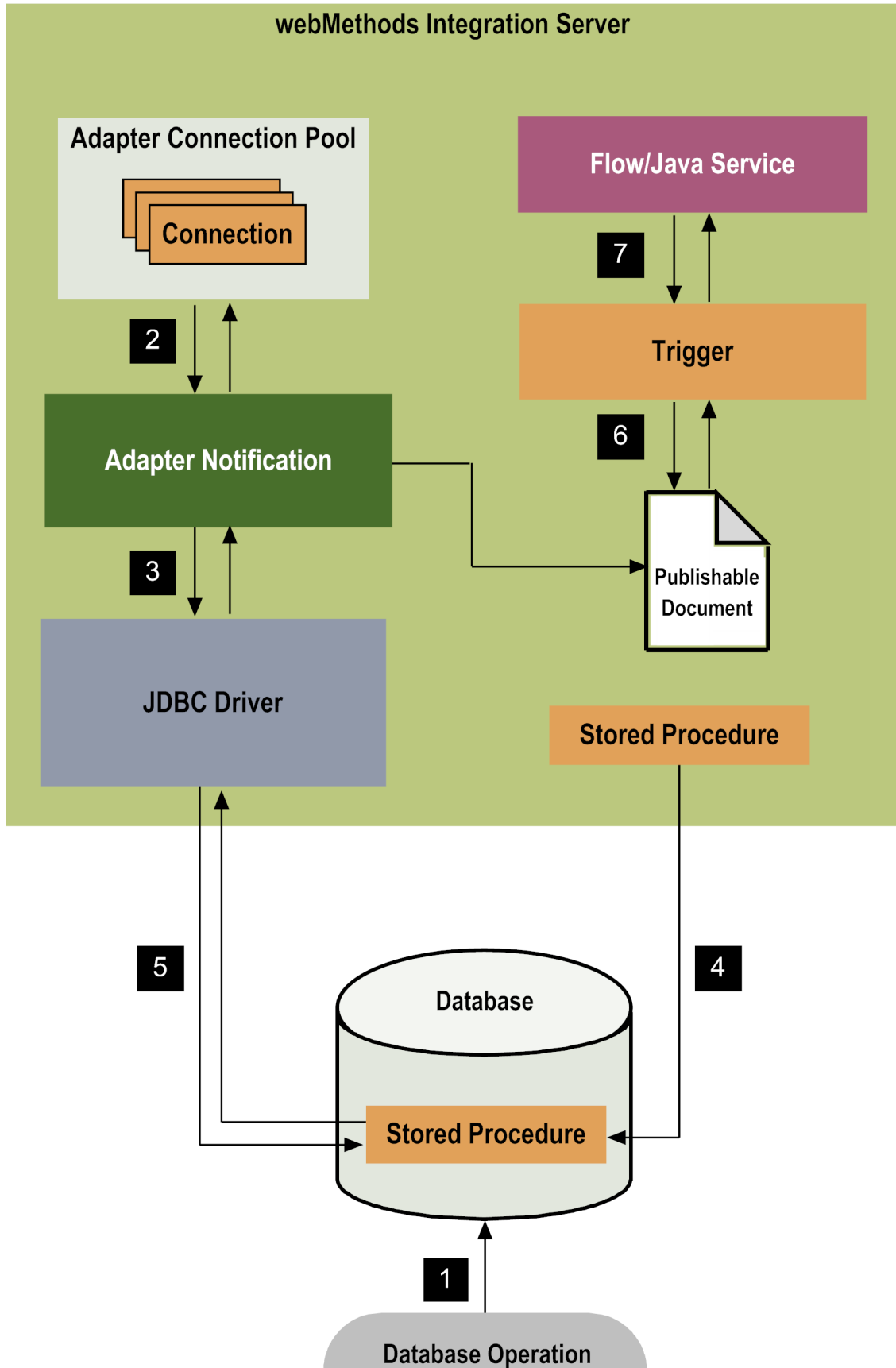
For this task...	Use these tools...
<p>1. To ensure that the same data is not published multiple times, design and test your stored procedure so that whenever the stored procedure is invoked, you are assured that it provides the correct data.</p>	User-dependent
<p>2. Create an adapter connection. For details, see “Adapter for JDBC Connections” on page 77.</p>	Integration Server Administrator
<p>3. Configure the notification and specify the:</p> <ul style="list-style-type: none"> ■ Adapter connection ■ Stored procedure ■ Publishable document to contain the data. There is a single publishable document used for all events associated with the notification. <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See “About this Guide” for specific document titles.</p> <ul style="list-style-type: none"> ■ Any output data fields to be contained in the publishable document <p>For instructions for configuring this type of notification, see “Configuring StoredProcedureNotifications” on page 175.</p>	Designer
<p>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.</p> <p>For details, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.</p>	Designer
<p>5. Schedule and enable the adapter notification. When you enable the notification, the Integration Server Scheduler invokes the</p>	Integration Server Administrator

For this task...	Use these tools...
<p>notification and continues to do so periodically, based on the polling schedule parameters you created earlier.</p> <p>For instructions for scheduling and enabling notifications, see “Managing Polling Notifications” on page 191.</p>	
<p>6. Manage the notification. For details, see “Package Management” on page 65, “Adapter Notifications” on page 151, and “Logging and Exception Handling” on page 203.</p>	<p>Designer and Integration Server Administrator</p>

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	<p>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.</p> <p>When the Stored Procedure Notification calls the stored procedure, it stores any output in the notification's publishable documents.</p>
2	<p>The notification gets a connection from the service's connection pool.</p> <p>Adapter connections contain connection information for the database, including JDBC driver parameters.</p>
3	<p>The notification uses the JDBC driver to connect to the database.</p> <p>You created and enabled the adapter connection earlier using Integration Server Administrator.</p>
4	<p>Integration Server calls the stored procedure.</p>
5	<p>The notification retrieves each row of data from the stored procedure.</p>
6	<p>Each row of data is published using the notification's publishable document. Depending on the stored procedure, the Stored Procedure Notification's publishable documents can contain any of the following:</p> <ul style="list-style-type: none">■ Output parameters: if the called stored procedure has any output parameters, they are contained in any publishable documents for the Stored Procedure Notification.■ Return values: 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. The result set can contain nested cursors. 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. <p>Note: When using the result set that contains nested cursors, the performance of Adapter for JDBC could degrade. Since the</p>

Step	Description
	<p>nested cursors are recursively processed, Adapter for JDBC may also return data that may not be required.</p> <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer's guide for your release. See "About this Guide" for specific document titles.</p>
7	<p>Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on Integration Server is invoked to react to the data changes contained in the publishable document.</p>

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 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. 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, the buffer table, and the sequence.

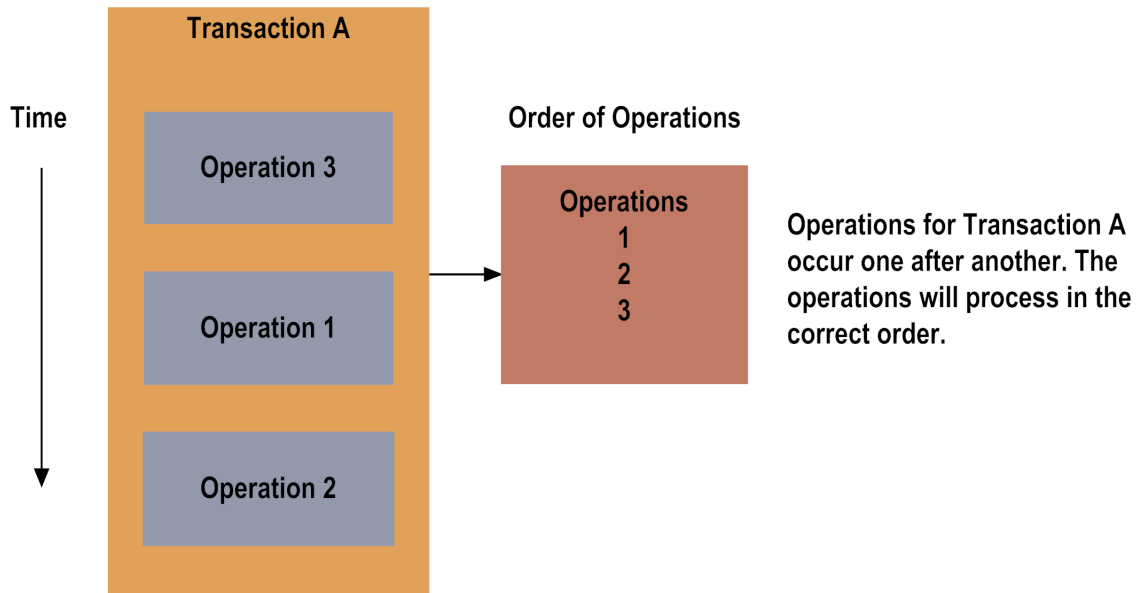
For more information about how Ordered Notifications work, see "[Ordered Notifications Transaction Processing](#)" on page 50.

Considerations when Using Ordered Notifications

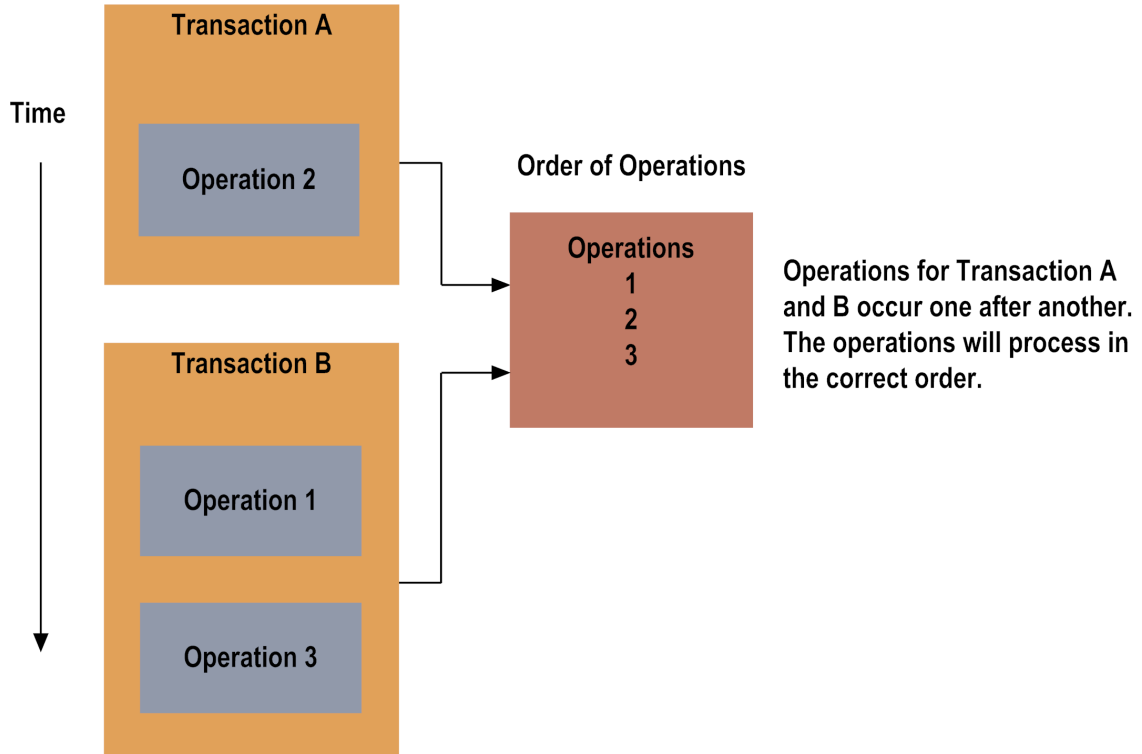
- Use the Ordered Notification only if you need to preserve the order in which the operations occur; otherwise, use Insert Notifications, Update Notifications, and Delete Notifications because they have better performance.
- Ordered Notifications ensure that the operations process in the correct order 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.

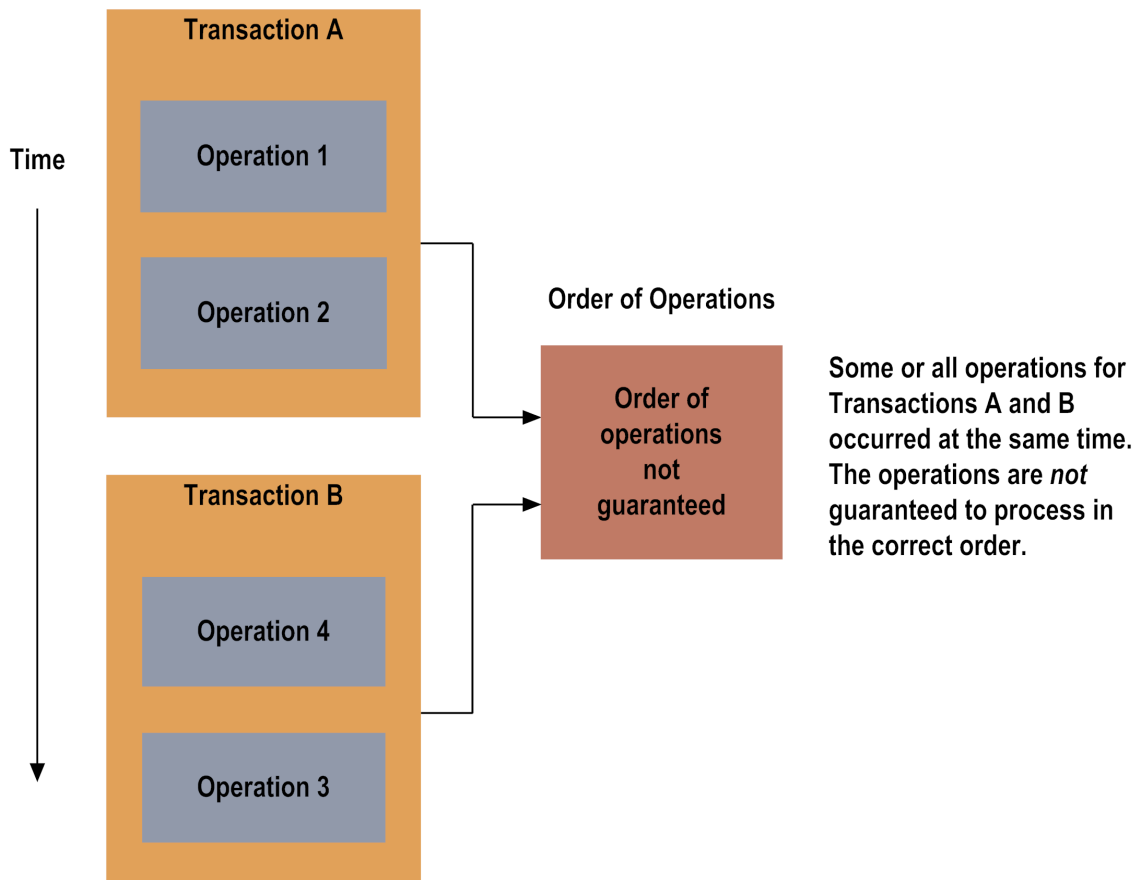
Example A



Example B



Example C

**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 **Processing Mode** option in Designer when you create the trigger and flow service.

For more information about using configuring Integration Server triggers and flow services, see the Designer Service Development online help for your release. See "About this Guide" for specific document titles.

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:

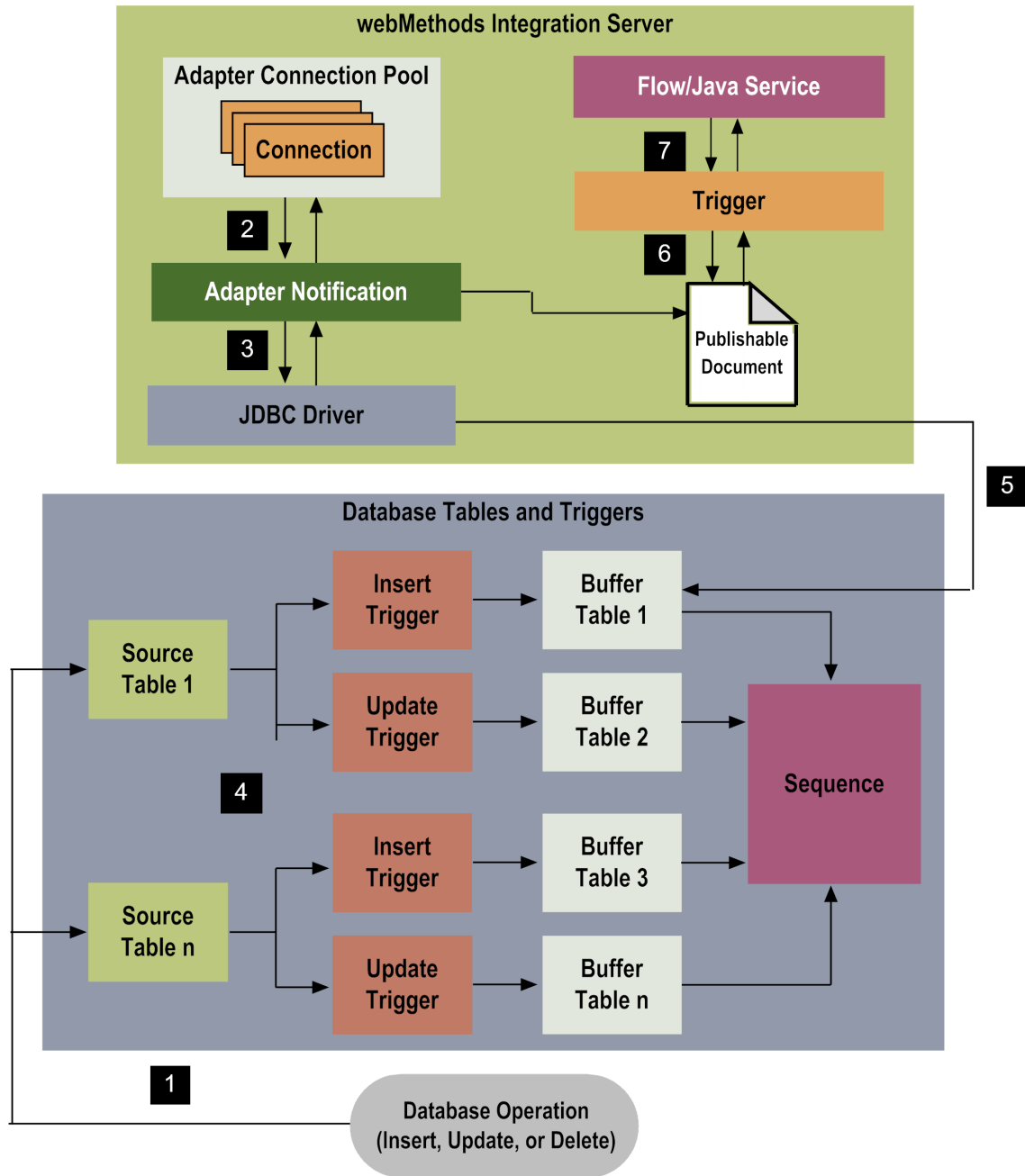
For this task...	Use these tools...
<p>1. Create an adapter connection. For details, see “Adapter for JDBC Connections” on page 77.</p>	<p>Integration Server Administrator</p>
<p>2. Configure the notification and specify the:</p> <ul style="list-style-type: none"> ■ Adapter connection ■ Source tables ■ 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 <p>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. For more details, see “Ordered Notifications” on page 45.</p> ■ Publishable document to contain the data from the buffer table. There is a single publishable document used for all events associated with the notification. <p>For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See “About this Guide” for specific document titles.</p> <p>For instructions for configuring this type of notification, see “Configuring OrderedNotifications” on page 184.</p> 	<p>Designer</p>
<p>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.</p>	<p>Designer</p>

For this task...	Use these tools...
<p>For details, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.</p> <p>If you use a trigger, be sure to set the Processing mode option to Serial. For details, see “Ordered Notifications” on page 45.</p>	
<p>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 periodically, based on the polling schedule parameters you created earlier, and continues to do so.</p> <p>For instruction for scheduling and enabling notifications, see “Managing Polling Notifications” on page 191.</p>	<p>Integration Server Administrator</p>
<p>5. Manage the notification. For details, see “Package Management” on page 65, “Adapter Notifications” on page 151, and “Logging and Exception Handling” on page 203.</p>	<p>Designer and Integration Server Administrator</p>

Ordered Notifications Transaction Processing

The following diagram and steps illustrate what happens when an Ordered Notification is invoked. 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.

Step	Description
2	<p>The notification gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.</p>
3	<p>The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using Integration Server Administrator.</p>
4	<p>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.</p> <p>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 tables and inserts data into the buffer table. When 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.</p>
5	<p>The notification retrieves the rows of data from the buffer table.</p> <p>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.</p>
6	<p>The notification creates the publishable document, which contains a row of data, including the Operation ID and Operation Type, from the buffer table.</p> <p>The notification publishes the publishable document.</p>
7	<p>Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on Integration Server is invoked to react to the data changes contained in the publishable document.</p> <p>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</p>

Step	Description
	<p>value is INSERT, the flow service picks up data from the INSERT record as input and processes accordingly.</p> <p>For more information about using triggers and flow services with Ordered Notifications, see “Ordered Notifications” on page 45.</p> <p>After the data is published, the data is deleted from the buffer table.</p>

Polling Notification Support in Clusters

Adapter for JDBC 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 about how to use polling notifications in a clustered environment, see [“Polling Notification Support in Integration Server Clusters” on page 71](#).

Polling Notifications and States

The following table summarizes the states in which polling notifications can exist 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.	<p>You can suspend polling notifications in an Enabled state. You cannot suspend polling notifications in a Disabled state.</p> <p>You can copy or export suspended polling notifications. You cannot move,</p>

State name	Status of trigger and buffer table when polling notification enters this state	Data processing while in this state	Comments
			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 does not apply to Basic Notifications or Stored Procedure Notifications because with these, the resource administrator (not Adapter for JDBC) 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 in [“Managing Polling Notifications” on page 191](#).

Support for Synonyms

Adapter for JDBC provides support for database synonyms.

Important: Not all JDBC drivers for backends that the adapter supports return synonyms. You can use synonyms only with some of the supported backends.

For information about working with and creating synonyms, see the documentation of your database vendor.

Synonym Support for Oracle Database

To enable synonym support for an Oracle database, you must specify `connectionproperties={includeSynonyms=true}` in the **Other Properties** field for the adapter connection. For information about configuring connections, see [“Configuring Adapter for JDBC Connections” on page 80](#).

The following table lists the adapter services and notifications that support synonyms.

Adapter Services	Adapter Notifications
SelectSQL	Basic Notification
DeleteSQL	Delete Notification
InsertSQL	Insert Notification
UpdateSQL	Ordered Notification
StoredProcedure	Stored Procedure Notification
	Update Notification

Consider the following limitations when using synonyms with adapter services:

- If you want to use synonyms for stored procedures, you cannot use the StoredProcedureWithSignature adapter service. Use the StoredProcedure service instead.
- The SelectSQL, DeleteSQL, InsertSQL, and Update SQL services support synonyms only for database tables and views.

For information about configuring adapter services, see [“Adapter Services” on page 101](#). For information about configuring adapter notifications, see [“Adapter Notifications” on page 151](#).

Synonym Support for DB2 UDB

The following table lists the adapter services and notifications that support synonyms for a DB2 Universal Database (UDB).

Adapter Services	Adapter Notifications
SelectSQL	Basic Notification
DeleteSQL	Delete Notification
InsertSQL	Insert Notification
UpdateSQL	Ordered Notification
	Update Notification

Consider the following limitations when using synonyms with adapter services:

- The StoredProcedure and StoredProcedureWithSignature adapter services do not support synonyms because synonyms are treated as an alias table type in DB2 UDB. However, aliases cannot be created for stored procedures.
- The SelectSQL, DeleteSQL, InsertSQL, and Update SQL services support synonyms only for database tables and views.

For information about configuring adapter services, see [“Adapter Services” on page 101](#). For information about configuring adapter notifications, see [“Adapter Notifications” on page 151](#).

Forcing a Timeout During Long-Running SQL Operations in Services and Notifications

In Adapter for JDBC services or notifications, some of the SQL operations may take a long time to execute. You can force these services or notifications to time out after a specific amount of time. You specify the number of seconds with the `watt.adapter.JDBC.QueryTimeout` property.

To set this property, use Integration Server Administrator and select **Settings > Extended > Edit Extended Settings**. Enter this property in the Extended Settings box:

```
watt.adapter.JDBC.QueryTimeout=value
```

where *value* is the number of seconds the adapter waits for the service or the notification to execute before stopping the SQL operation and throwing an exception. For more information about setting the watt properties, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.

Using Version Control Systems to Manage Adapter Elements

The adapter supports the Version Control System (VCS) Integration feature provided by Designer. When you enable the feature in Integration Server, you can check adapter packages or elements into and out of your version control system from Designer. For more information about the VCS Integration feature, see the *Administering the VCS Integration Feature*.

Beginning with Integration Server 8.2 SP3, the adapter supports the local service development feature in Designer. This feature extends the functionality of the VCS Integration feature to check package elements and their supporting files into and out of a VCS directly from Designer. For more information about local service development and how it compares to the VCS Integration feature, see the *webMethods Service Development Help*.

Infrastructure Data Collector Support for Adapter for JDBC

Optimize Infrastructure Data Collector monitors the system and operational data associated with webMethods run-time components such as Integration Servers, Broker Servers, Brokers, and adapters, and reports the status of these components on Optimize for Infrastructure or other external tools. When you start monitoring an Integration Server, Infrastructure Data Collector automatically starts monitoring all ART-based adapters that are installed on the Integration Server.

For information about monitored key performance indicators (KPIs) collected for the monitored adapter components, see the Optimize documentation for your release. See “About this Guide” for specific document titles.

Viewing the Adapter's Update Level

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 Integration Server Administrator.

Controlling Pagination

When using the adapter on Integration Server 9.0 and later, you can control the number of items that are displayed on the adapter Connections screen and Notifications screen. By default, 10 items are displayed per page. Click **Next** and **Previous** to move through the pages, or click a page number to go directly to a page.

To change the number of items displayed per page, set the `watt.art.page.size` property and specify a different number of items.

To set the number of items per page

1. From Integration Server Administrator, click **Settings > Extended**.
2. Click **Edit Extended Settings**. In the Extended Settings editor, add or update the `watt.art.page.size` property to specify the preferred number of items to display per page. For example, to display 50 items per page, specify:

```
watt.art.page.size=50
```

3. Click **Save Changes**. The property appears in the Extended Settings list.

For more information about working with extended configuration settings, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

2 Installing, Upgrading, and Uninstalling Adapter for JDBC

■ Overview	60
■ Requirements	60
■ The Integration Server Home Directory	60
■ Installing Adapter for JDBC	60
■ Upgrading to Adapter for JDBC 9.0	61
■ Migrating adapter services from 6.5 to 9.0	63
■ Uninstalling Adapter for JDBC	63

- [“Migrating adapter services from 6.5 to 9.0” on page 63](#)

Overview

This chapter explains how to install, upgrade, and uninstall webMethods Adapter for JDBC 9.0. The instructions use the Software AG Installer and the Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the webMethods installation guide for your release. See “About this Guide” for specific document titles.

Requirements

For a list of operating systems, RDBMSs, and webMethods products supported by Adapter for JDBC, see *webMethods Adapters System Requirements*.

Adapter for JDBC has no hardware requirements beyond those of its host Integration Server.

The Integration Server Home Directory

Beginning with Integration Server 9.6, you can create and run multiple Integration Server instances under a single installation directory. Each Integration Server instance has a home directory under *Integration Server_directory\instances\instance_name* that contains the packages, configuration files, log files, and updates for the instance.

For more information about running multiple Integration Server instances, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

If you are using Integration Server 9.5 and lower, the Integration Server home directory is *Integration Server_directory*. For example, on Integration Server 9.5 the adapter package is installed in the *Integration Server_directory\packages* directory.

This guide uses the *packages_directory* as the home directory in Integration Server classpaths. For Integration Server 9.6 and above, the *packages_directory* is *Integration Server_directory\instances\instance_name\packages* directory. For Integration Server 9.5 and lower, the *packages_directory* is *Integration Server_directory\packages* directory.

Installing Adapter for JDBC

Note: If you are installing Adapter for JDBC in a clustered environment, you must install the adapter on each Integration Server in the cluster, and each installation must be identical. For more information about working with

Adapter for JDBC in a clustered environment, see [“Adapter for JDBC in a Clustered Environment”](#) on page 70.

To install Adapter for JDBC

1. Download Installer from the [“Empower Product Support website”](#).
2. If you are installing the adapter on an existing Integration Server, shut down the Integration Server.
3. Start the Installer wizard.
4. Choose the webMethods release that includes the Integration Server on which you want to install the adapter. For example, if you want to install the adapter on Integration Server 9.8, choose the 9.8 release.
5. Specify the installation directory as follows:
 - If you are installing on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server.
 - If you are installing both the host Integration Server and the adapter, specify the installation directory to use.
6. In the product selection list, select **Adapters >webMethods Adapter for JDBC 9.0**.

If you are using Integration Server 9.6 and above, you can choose to install the package in the default instance. In this case, Software AG Installer installs the adapter in both locations, *Integration Server_directory*\packages and the default instance packages directory located in *Integration Server_directory*\instances\default\packages.
7. To download the documentation for the adapter, go to [“Software AG Documentation website”](#).
8. After the installation completes, close the Installer and start the host Integration Server.
9. See [“Installing a JDBC Driver on Integration Server”](#) on page 79 for instructions on installing a compatible JDBC driver.


Upgrading to Adapter for JDBC 9.0

You can upgrade to Adapter for JDBC 9.0 from Adapter for JDBC 6.5.

Before upgrading you can choose to archive the existing adapter package. Archiving creates a copy of the adapter package which enables you to revert to the earlier adapter package later if necessary.

Archiving

To archive the existing adapter

1. Navigate to **Packages > Management** in Integration Server Administrator.
2. Locate WmJDBCAdapter and click the icon  in the Archive column.

The Archive page is displayed in Integration Server Administrator with the list of all files to be archived.

By default, **Full** Archive Type is selected.

3. Click **Create Archive**.

Integration Server creates a copy of the adapter package in the *Integration Server_directory\replicate\outbound* directory.

Upgrading

To upgrade to Adapter for JDBC 9.0

1. Uninstall the existing adapter and delete the package using the instructions in on page 63 “Uninstalling Adapter for JDBC”.
2. Install Adapter for JDBC using the instructions in “[Installing Adapter for JDBC](#)” on page 60.
3. Upgrade the Adapter for JDBC assets. For details, refer to “[Migrating adapter services from 6.5 to 9.0](#)” on page 63.

Reverting

To revert to the earlier adapter

1. Uninstall the existing adapter and delete the package using the instructions in on page 63 “Uninstalling Adapter for JDBC”.
2. Copy the *packages_directory \replicate\outbound\WmJDBCAdapter* to *packages_directory \replicate\inbound* directory.
3. Navigate to **Packages > Management** in Integration Server Administrator.
4. Click **Install Inbound Releases**.

The Inbound Releases page is displayed in Integration Server Administrator.

5. Select the Release file name from the drop-down list and click **Install Release**.

Migrating adapter services from 6.5 to 9.0

In Adapter 9.0 for JDBC, the template for SelectSQL, UpdateSQL, DeleteSQL, and CustomSQL adapter service types are modified. The adapter now auto-populates,

- the selected columns in the input type of the WHERE tab for SelectSQL, UpdateSQL, and DeleteSQL service types.
- the input and output types in a CustomSQL service type.

The migration utility service applies the changes to the older adapter service types.

Use the `wm.adapter.wmjdbadmin:migrate_65_To_90` migration utility service to migrate the adapter service from version 6.5 to Adapter 9.0 for JDBC.

Uninstalling Adapter for JDBC

To uninstall Adapter for JDBC

1. Shut down the host Integration Server. You do not need to shut down any other webMethods products or applications that are running on your machine.
2. Start Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server.
3. In the product selection list, select **Adapters > webMethods Adapter for JDBC**. You can also choose to uninstall documentation.
4. After Uninstaller completes, restart the host Integration Server.

Uninstaller removes all Adapter for JDBC-related files that were installed. However, Uninstaller does not delete files created after you installed the adapter (for example, user-created or configuration files), nor does it delete the adapter directory structure. You can go to the *Integration Server_directory*\packages directory and *Integration Server_directory*\instances\default\packages directory. Delete the WmJDBCAdapter directory.

3 Package Management

■ Overview	66
■ Adapter for JDBC Package Management	66
■ Group Access Control	70
■ Adapter for JDBC in a Clustered Environment	70

Overview

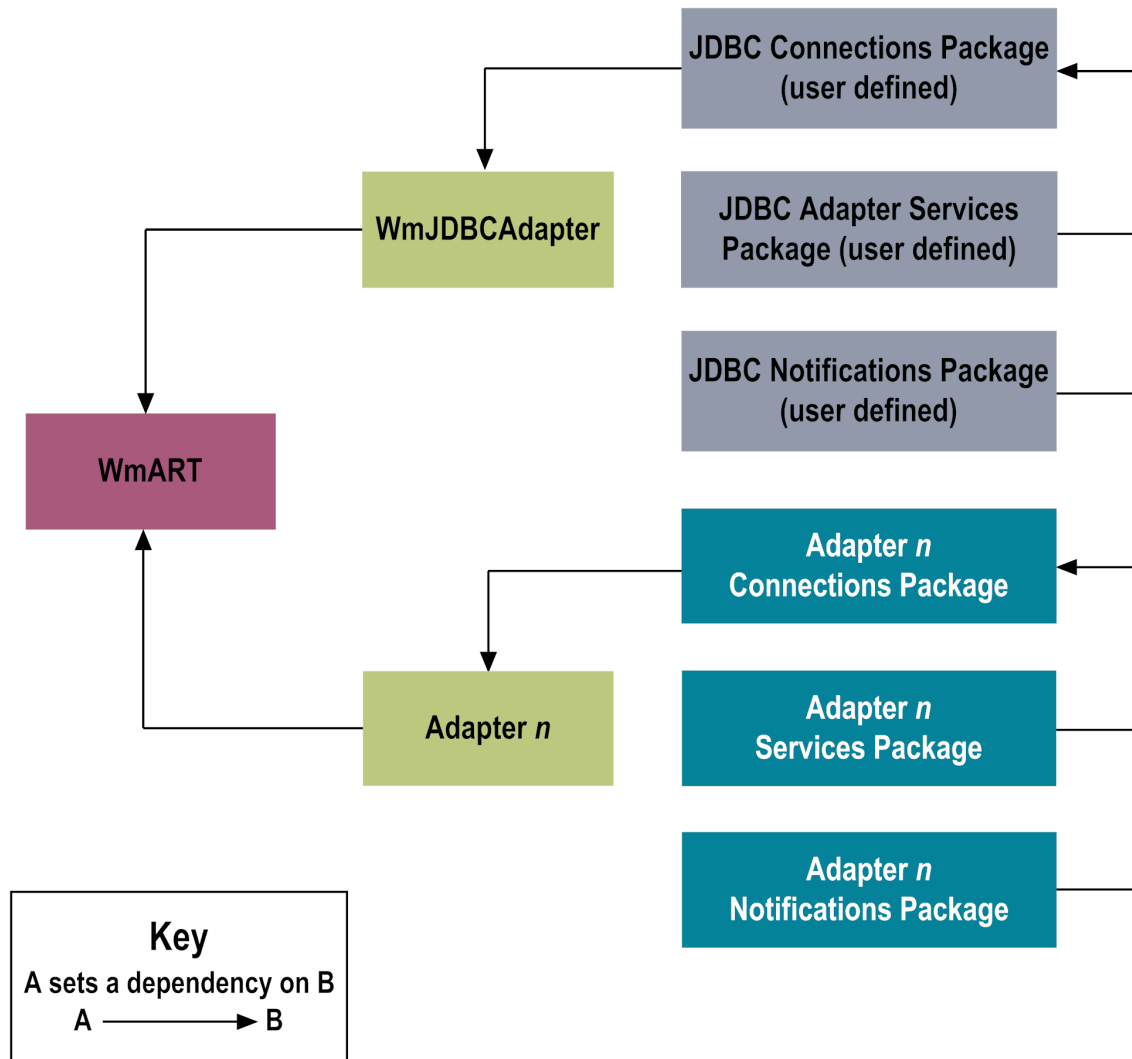
The following sections describe how to set up and manage your Adapter for JDBC packages, set up Access Control Lists (ACLs), and use the adapter in a clustered environment.

Adapter for JDBC Package Management

Adapter for JDBC is provided as a package called WmJDBCAdapter. You manage the WmJDBCAdapter package as you would manage any package on webMethods 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 Software AG Designer 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 \(see “Package Dependency Requirements and Guidelines” on page 67\)](#)
- [“Enabling Packages” on page 68](#)
- [“Importing and Exporting Packages” on page 69](#)
- [“Group Access Control” on page 70](#)

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 Designer Service Development online help for your release. ‘See “About this Guide” for specific document titles.

- 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 packages last. The WmART package is automatically installed when you install 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 connections 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.
- 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 Packages” on page 68](#).
- 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 [“Disabling Packages” on page 69](#).
- You can name connections, adapter services, and notifications the same name provided that they are in different folders and packages.

Enabling Packages

All packages are automatically enabled by default. Use the following procedure when you want to enable a package that was previously disabled.

To enable a package

1. Open 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 packages to be reloaded. For information about reloading packages, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.

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

Disabling Packages

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 Integration Server from loading that package at startup.

Important: If your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package first (WmJDBCAdapter). Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages.

To disable a package

1. Open 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 disable 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 Integration Server Administrator.
- Not be listed in Designer.

Importing and Exporting Packages

You import and export packages using Designer. 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 Designer Service Development online help for your release. See “About this Guide” for specific document titles.

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 Designer Service Development online help for your release. See “About this Guide” for specific document titles.

Adapter for JDBC in a Clustered Environment

Clustering is an advanced feature of the webMethods product suite that substantially extends the reliability, availability, and scalability of Integration Server. Clustering accomplishes this by providing the infrastructure and tools to deploy multiple Integration Servers as if they were a single virtual server and to deliver applications that leverage that architecture. Because this activity is transparent to the client, clustering makes multiple servers look and behave as one.

Integration Server 9.0 and higher supports the caching and clustering functionality provided by Terracotta. Caching and clustering are configured at the Integration Server level and Adapter for JDBC uses the caching mechanism that is enabled on Integration Server. Adapter for JDBC does not explicitly implement any clustering or caching beyond what is already provided by Integration Server.

With clustering, you get the following benefits:

- **Load balancing.** This feature, provided automatically when you set up a clustered environment, allows you to spread the workload over several servers, thus improving performance and scalability.
- **Failover support.** Clustering enables you to avoid a single point of failure. If a server cannot handle a request, or becomes unavailable, the request is automatically redirected to another server in the cluster.

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

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

For details on Integration Server clustering, see the Integration Server clustering guide for your release. See “About this Guide” for specific document titles.

Polling Notification Support in Integration Server Clusters

Adapter for JDBC enables the coordinated execution of polling notifications within an Integration Server cluster. Adapter for JDBC 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.

Important: Adapter for JDBC supports enabling the same polling notification on multiple Integration Server instances connecting to the same backend database to achieve automated failover, *only* when the multiple Integration Servers share the same ISInternal database. If you attempt to use the same polling notification on multiple Integration Servers pointing to the same backend database but using separate ISInternal databases, you may encounter abnormal results.

Beginning with Integration Server 9.0, Adapter for JDBC uses Integration Server Scheduler to support polling notifications. On enabling a polling notification, a new Integration Server scheduled task is created, which polls the backend resource at the given interval. Do not manually edit or change scheduled tasks. Each polling notification creates an Integration Server scheduled task. When a notification is disabled, the scheduled task in Integration Server is removed.

Important: All adapter polling notifications must be in a disabled state on all nodes in the Integration Server cluster before you disable the cluster.

Considerations for Polling Notifications Executing via Scheduled Tasks

With polling notifications executing via scheduled tasks, ensure that:

- Each notification is present in all cluster nodes at all times.
- The Overlap function for the polling notifications is disabled.
- Polling notifications names do not exceed 400 characters.
- The value of the Integration Server `watt.server.scheduler.threadThrottle` property should not be lower than the number of total polling notifications and scheduled tasks. By default the value is 75% of the total threads.
- The IS Internal functional alias (specified on the Settings > JDBC Pools screen) is configured with a database.

Note: You can make scheduled notification tasks visible in the Server > Scheduler page in Integration Server Administrator by setting `watt.pkg.art.scheduler.notificationtask.display=true`
If the parameter is not shown, add it.

Configuring this property is required only for debugging or for editing the polling notification schedule interval.

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.

For more details about adapter services in clusters, see [“Clustering Considerations and Requirements” on page 73](#).

Replicating Packages to Integration Servers

Every Integration Server in the cluster should contain an identical set of packages that you define using Adapter for JDBC; that is, you should replicate the Adapter for JDBC services, the connections they use, and the adapter 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 Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.

Disabling the Redirection of Administrative Services

As mentioned in [“Adapter for JDBC in a Clustered Environment” on page 70](#), a server that cannot handle a client's service request can automatically redirect the request to another server in the cluster. However, Adapter for JDBC 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 Adapter for JDBC 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 Integration Servers in the cluster.

To disable the redirection of administrative services

1. Shut down Integration Server Administrator. For the procedure to do this, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.
2. Open the following file:

Integration Server_directory\config\redir.cnf

3. Add the following line to the file:

```
<value name="wm.art">false</value>
```

4. Save the file and restart Integration Server.

Clustering Considerations and Requirements

Note: The following sections assume that you have already configured the Integration Server cluster. For details about webMethods clustering, see the Integration Server clustering guide for your release. See “About this Guide” for specific document titles.

The following considerations and requirements apply to Adapter for JDBC 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.
Adapter connections	<p>If you configure a connection to the database, this connection must appear on all servers in the cluster so that any Integration Server in the cluster can handle a given request identically.</p> <p>If you plan to use connection pools in a clustered environment, see “Considerations When Configuring Connections with Connection Pooling Enabled” on page 74.</p>
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.

All Integration Servers in a given cluster must have identical...	For Example...
Adapter notifications	<p>If you allow different Integration Servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides on only one server, and that server is unavailable, the request cannot be successfully redirected to another server.</p> <p>If you configure a specific Insert notification, this same adapter notification must appear on all servers in the cluster so that any Integration Server in the cluster can handle the request identically.</p> <p>If you allow different Integration Servers to contain different notification, you might not derive the full benefits of clustering. For example, if a notification executes on only one server, and that server is unavailable, the notification cannot be successfully redirected to another server.</p>

For information about replicating adapter packages, connections, adapter services, and adapter notifications across multiple Integration Servers in a cluster, see [“Replicating Packages to Integration Servers”](#) on page 72.

Considerations When Installing Adapter for JDBC Packages

For each Integration Server in the cluster, use the standard Adapter for JDBC installation procedures for each machine, as described in [“Installing, Upgrading, and Uninstalling Adapter for JDBC”](#) on page 59.

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 Adapter for JDBC, see [“Adapter for JDBC Connections” on page 77](#).

For more general information about connection pools, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.

4 Adapter for JDBC Connections

■ Overview	78
■ Before Configuring or Managing Adapter Connections	78
■ Setting the Environment Variable for Oracle JDBC OCI Drivers	79
■ Installing a JDBC Driver on Integration Server	79
■ Configuring Adapter for JDBC Connections	80
■ Dynamically Changing a Service's Connection at Run Time	95
■ Dynamically Changing the User Credentials of a Service's Connection at Run Time	96
■ Viewing Adapter Connection Parameters	96
■ Editing Adapter Connections	97
■ Copying Adapter Connections	98
■ Deleting Adapter Connections	98
■ Enabling Adapter Connections	99
■ Disabling Adapter Connections	99

Overview

This chapter describes how to configure and manage Adapter for JDBC connections. For more information about how adapter connections work, see [“Adapter Connections” on page 18](#).

Before Configuring or Managing Adapter Connections

Perform the following steps before configuring or managing adapter connections.

To prepare to configure or manage adapter connections

1. Install webMethods Integration Server and Adapter for JDBC on the same machine. For details, see [“Installing, Upgrading, and Uninstalling Adapter for JDBC” on page 59](#).
2. Install a compatible JDBC driver. For instructions, see [“Installing a JDBC Driver on Integration Server” on page 79](#). For a list of supported drivers, see the *webMethods Adapters System Requirements*.
3. Make sure you have Integration Server administrator privileges so that you can access Adapter for JDBC's administrative screens. For information about setting user privileges, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.
4. Be sure to check [“Database Driver Known Limitations” on page 261](#) for a list of known driver limitations because it may affect how you configure your connections.
5. Start your Integration Server and Integration Server Administrator, if they are not already running.
6. Using Integration Server Administrator, make sure the WmJDBCAdapter package is enabled. For instructions, see [“Enabling Packages” on page 68](#).
7. Using Designer, 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 [“Adapter for JDBC Package Management” on page 66](#).
8. If you use Oracle JDBC OCI drivers, you must set an environment variable before you can configure the connection. For details, see [“Setting the Environment Variable for Oracle JDBC OCI Drivers” on page 79](#).

If you use Oracle JDBC OCI drivers with Oracle OCI Instant Client, then copy the following client library files before you set the environment variable:

- `ojdbc5.jar` and `ojdbc6.jar` files to the two locations, *Integration Server_directory*\lib\jars and *Integration Server_directory*\instances*instance_name*\lib\jars.

- All the other client library files to the two locations, *Integration Server_directory*\lib and *Integration Server_directory*\instances*instance_name* \lib.

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 a JDBC Driver on Integration Server

You must install a JDBC driver on Integration Server before you can specify connections. Integration Server requires access to the Java classes for each JDBC driver that it will use to connect to a database.

For a list of supported drivers, see the *webMethods Adapters System Requirements*.

To install a JDBC driver

1. Place the Java classes for the JDBC driver in a location that Integration Server can access, typically the server's classpath.

To place the classes in the server's classpath, place the .zip or .jar file containing the classes in the *Integration Server_directory*\instances*instance_name* \packages \WmJDBCAdapter\code\jars directory.

2. Restart Integration Server.

The server automatically adds the .zip or .jar libraries to its classpath after the restart.

Configuring Adapter for JDBC Connections

When you configure Adapter for JDBC connections, you specify information that Integration Server uses to connect to a JDBC system. You can configure Adapter for JDBC connections either manually using the Integration Server Administrator screen or programmatically using the [pub.jdbcAdapter:createConnectionNodes](#) service.

Note: If you use Oracle JDBC OCI drivers with Adapter for JDBC, you must add an environment variable setting before you configure adapter connections. For details, see [“Setting the Environment Variable for Oracle JDBC OCI Drivers” on page 79](#).

To configure an adapter connection

1. In the **Adapters** menu in Integration Server Administrator's navigation area, click **Adapter for JDBC**.
2. On the Connections screen, click **Configure New Connection**.
3. On the Connection Types screen, click **Adapter for JDBC Connection** to display the Configure Connection Type screen.
4. In the **Adapter for JDBC Connection** section, use the following fields:

Field	Description/Action
Package	<p>The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.</p> <p>Note: Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for JDBC, see “Adapter for JDBC Package Management” on page 66.</p>
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 Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the Designer Service Development online

Field	Description/Action
	help for your release. See “About this Guide” for specific document titles.

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

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

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

Field	Description/Action
Transaction Type	<p>The type of transaction support that the connection provides. Select one of the following transaction types:</p> <ul style="list-style-type: none"> ■ 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. <p>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. For information about these specific configuration options, see “Configuring BasicNotifications” on page 170.</p> <ul style="list-style-type: none"> ■ XA_TRANSACTION: The connection uses XA transactions. <p>Note: Do not use XA_TRANSACTION connections with Teradata databases.</p> <p>Note: Adapter for JDBC 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.</p>

For a more detailed description of the transaction support provided by Adapter for JDBC, see “[Transaction Management of Adapter Connections](#)” on page 19.

Field	Description/Action
DataSource Class Name	<p>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.</p>
	<p>Oracle JDBC OCI Driver</p>
	<p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p>
	<pre>oracle.jdbc.pool.OracleDataSource</pre>
	<p>For XA_TRANSACTION:</p>
	<pre>oracle.jdbc.xa.client.OracleXADataSource</pre>
	<p>Oracle JDBC Thin Driver</p>
	<p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p>
	<pre>oracle.jdbc.pool.OracleDataSource</pre>
	<p>For XA_TRANSACTION:</p>
	<pre>oracle.jdbc.xa.client.OracleXADataSource</pre>
	<p>DataDirect Connect for JDBC (for Microsoft SQLServer)</p>
	<p>For all transaction types:</p>
	<pre>com.dd.jdbcx.sqlserver.SQLServerDataSource</pre>
	<p>DataDirect Connect for JDBC (for Oracle)</p>
	<p>For all transaction types:</p>
	<pre>com.wm.dd.jdbcx.oracle.OracleDataSource</pre>
	<p>webMethods BigData Driver for Apache Hive</p>
	<p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p>
	<pre>wm.jdbcx.hive.HiveDataSource40</pre>
	<p>webMethods BigData Driver for MongoDB</p>
	<p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p>
	<pre>wm.jdbcx.mongodb.MongoDBDataSource40</pre>
	<p>webMethods BigData Driver for Apache Cassandra</p>
	<p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p>
	<pre>wm.jdbcx.cassandra.CassandraDataSource40</pre>
	<p>webMethods BigData Driver for Apache Impala</p>
	<p>For NO_TRANSACTION:</p>

Field	Description/Action
	<code>wm.jdbcx.impala.ImpalaDataSource</code>
	<p>Microsoft SQL Server DataSource for JDBC driver</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.microsoft.sqlserver.jdbc.SQLServerDataSource</code></p> <p>For XA_TRANSACTION: <code>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</code></p>
	<p>MySQL JDBC type 4 driver</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.mysql.jdbc.jdbc2.optional.MysqlDataSource</code></p> <p>For XA_TRANSACTION: <code>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</code></p>
	<p>JTOpen v4.1 (for DB2 for AS/400)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.ibm.as400.access.AS400JDBCDataSource</code></p> <p>For XA_TRANSACTION: <code>com.ibm.as400.access.AS400JDBCXADataSource</code></p>
DataSource Class Name (continued)	<p>DataDirect Connect (for UDB)</p> <p>For all transaction types: <code>com.ddtek.jdbcx.db2.DB2DataSource</code></p>
	<p>DB2 Universal type 2 and type 4 (for UDB)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.ibm.db2.jcc.DB2SimpleDataSource</code></p> <p>For XA_TRANSACTION: <code>com.ibm.db2.jcc.DB2XADataSource</code></p>
	<p>jCONNECT 5.5 type 4 (for Sybase)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.sybase.jdbc2.jdbc.SybDataSource</code></p> <p>For XA_TRANSACTION: <code>com.sybase.jdbc2.jdbc.SybXADataSource</code></p>
	<p>jCONNECT 6.05 type 4 (for Sybase)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION: <code>com.sybase.jdbc3.jdbc.SybDataSource</code></p>

Field	Description/Action
	<p>For XA_TRANSACTION:</p> <pre>com.sybase.jdbc3.jdbc.SybXADataSource</pre>
DataSource Class Name (continued)	<p>Informix JDBC 2.21 type 4 (for Informix)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p> <pre>com.informix.jdbcx.IfxDataSource</pre> <p>Note: If you use the com.informix.jdbcx.IfxDataSource DataSource class with Integration Server, 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 “Adapter for JDBC Package Management” on page 66. For more detailed information, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.</p> <p>For XA_TRANSACTION:</p> <pre>com.informix.jdbcx.IfxXADataSource</pre>
	<p>Teradata type 4 (for Teradata)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p> <pre>com.ncr.teradata.TeraDataSource</pre>
	<p>Teradata type 4 (for Teradata)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p> <pre>com.teradata.jdbc.TeraDataSource</pre>
	<p>Advanced Server JDBC Connector (edb-jdbc16.jar)</p> <p>For NO_TRANSACTION and LOCAL_TRANSACTION:</p> <pre>com.edb.ds.PGSimpleDataSource</pre> <p>For XA_TRANSACTION:</p> <pre>com.edb.xa.PGXDataSource</pre>
	<p>SAP Hana</p> <pre>com.sap.db.jdbcext.DataSourceSAP</pre>

- 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 93](#).

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

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 . <div data-bbox="581 768 1370 947" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: You can also update the password programmatically using the pub.jdbcAdapter:updateConnectionPassword service.</p> </div>
Retype password	Retype the password you just entered.
databaseName	The database to which the connection will connect.
portNumber	The port number that the connection will use to connect to the database. <div data-bbox="581 1247 1370 1381" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>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.</p> </div>
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 <code>tcp</code> , which is the name of the network protocol.

Note: *For Microsoft SQL Server databases, the user name 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. You can specify the following settings in the **Other Properties** field:

- For an Oracle database, specify the following property to enable synonym support:

```
connectionproperties={includeSynonyms=true}
```

- For DB2 AS/400 and DB2 OS390 databases, specify the clischema property as follows:

```
clischema= schema name
```

- With DB2 AS/400, use the clischema property to enable or disable adapter polling notifications when the systriggers view is not in the default schema. For example, if you specify `clischema=QSYS2`, when enabling or disabling a notification the adapter issues the following query: `SELECT trigger_name FROM QSYS2.systriggers`
- With DB2 OS390, use the clischema property to enable an adapter polling notification if the triggers for the notification already exist. First you must create a systriggers view in the sysibm.systriggers table and then specify the schema in which the view was created against the clischema property. Thus the adapter redirects the query for triggers to the appropriate schema.
- For a DB2 AS/400 database, if the StoredProcedure adapter service calls a stored procedure with a procedure name that is different from its specific name, specify the following property:

```
metadatasource=1
```

- Specify table filter property settings to 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.

Note: Use a ; (semi-colon) to delimit table filter, transaction isolation level, and driver-dependent property settings. Do not enter spaces after the semi-colon. For example: `TableFilter='<current catalog>'. 'Accounting';driverType=oci`

Use the following format to enter table filter property settings in the **Other Properties** field:

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

For 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 the following rules:

Use...	To...
<code><current catalog></code>	Use the <i>catalog</i> for the default login catalog. Note: Informix databases automatically access the current catalog only.
<code><current schema></code>	Use the <i>schema</i> for the login user.
<i>table</i>	Optional. The table name pattern. If you do not specify a <i>table</i> value, Adapter for JDBC loads all of the tables for the schema. The following example lists all the tables under the Accounting schema: <code>TableFilter='<current catalog>'.Accounting</code>
<code>%</code> (percent)	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: <code>TableFilter='<current catalog>'.Accounting'. 'Finance%'</code>
<code>,</code> (commas)	List multiple TableFilter settings. Do not enter spaces after the comma. For example: <code>TableFilter='<current catalog>'.Accounting'. 'Finance_','<current catalog>'.Employee%</code>
<code>_</code> (underscore)	Match any one character. The following example lists all the tables under the Accounting schema named Finance1, Finance2, Finance3, and so forth: <code>TableFilter='<current catalog>'.Accounting'. 'Finance_'</code>
<code>;</code> (semi-colons)	Delimit multiple property settings: TableFilter settings, transaction isolation level settings, and driver-dependent settings. Do not enter spaces after the semi-colon. For example: <code>TableFilter='<current catalog>'.Accounting'. 'Finance_','<current catalog>'.Accounting';transactionIsolation=2;driverType=thin</code>

- Specify transaction isolation level settings to set the transaction isolation level for a database. This setting prevents dirty read, repeatable read, and phantom read of the database, prevent dirty read, repeatable read, and phantom read of a database. For more information about the transaction isolation level settings in Adapter for JDBC, see [“Transaction Isolation Level Settings” on page 20](#).

Note: Use a ; (semi-colon) to delimit table filter, transaction isolation level, and driver-dependent property settings. Do not enter spaces after the semi-colon. For example: `TableFilter='<current catalog>'. 'Accounting';driverType=oci`

Use the following format to specify the transaction isolation levels of the database:

```
transactionIsolation = value
```

where *value* is the integer value of the transaction isolation level.

For example, `transactionIsolation = 2`, where 2 sets the TRANSACTION_READ_COMMITTED isolation level.

Note: For DB2 on AS/400, if you are accessing a table with the `<current catalog>.<current schema>` qualifier, set the translation isolation level and also specify the Libraries property as follows:
`TransactionIsolation=none;Libraries=QGPL`

You can specify only one transaction isolation level for a connection. The commonly used transaction isolation levels and their values are:

Transaction Isolation Settings	Value
TRANSACTION_READ_UNCOMMITTED	1
TRANSACTION_READ_COMMITTED	2
TRANSACTION_REPEATABLE_READ	4
TRANSACTION_SERIALIZABLE	8

For information about the transaction isolation levels supported by your database, refer to your database documentation.

If you do not specify the isolation level in the **Other Properties** field, the default isolation level of the database is considered. If you specify an isolation level that is not supported by the database, an error is thrown while enabling the connection.

Note: Consider the following database limitations:

- For Teradata databases, dirty read does not function when you set the transaction isolation level to 1, and non repeatable read and phantom read cannot be avoided even after you set the transaction isolation level to 8.

- For DB2 for OS/390, non repeatable read does not function when you set the transaction isolation level to 2.
 - For Informix, phantom read does not function when you set the transaction isolation level to 4.
- Specify driver-dependent property settings to provide additional JDBC driver DataSource properties depending on the driver that you use.

Use the following format:

```
propertyName=value
```

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

```
TableFilter='<current catalog>'. 'Accounting'. 'Finance';
selectMethod=cursor;transactionIsolation=2
```

Use { } to delimit a combination of multiple key value pairs that use ; (semi-colons) as delimiters. Do not enter spaces after the semi-colon. For example:

```
TableFilter='<current catalog>'. 'Accounting';driverType=oci;
connectionProperties={oracle.jdbc.V8Compatible=true,includeSynonymns=true
};transactionIsolation=2
```

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)	<p>DataDirect Connect for JDBC 3.2 for DB2 UDB 7.2, 8.1, 9.1, and 9.5:</p> <pre>PackageName=value</pre> <p>Where <i>value</i> is the name of the package you created earlier in the database. For information about creating packages, see your DataDirect Connect for JDBC documentation.</p> <hr/> <p>DB2 Universal type 4:</p> <ul style="list-style-type: none"> ■ <code>driverType=4</code> Required setting for this driver. If the <code>driverType</code> is not set to 4, then Type 2 connectivity is selected by default. ■ <code>readOnly=true</code> Creates a read only connection. ■ <code>currentSchema=YourSchemaName</code> Specifies the default schema name used to qualify unqualified database objects in dynamically prepared SQL statements.

Field	Description/Action
	<ul style="list-style-type: none"> ■ <code>loginTimeout=number</code> The maximum time in seconds to wait for the DataSource object to connect to a data source. ■ <code>traceFile=fileName</code> Specifies the name of a file into which this driver writes trace information. ■ <code>traceFileAppend=true</code> Appends, instead of overwriting, the file that is specified by the <code>traceFile</code> property. ■ <code>traceLevel=number</code> Specifies what level to trace. This property's data is INTEGER. Set <code>number</code> to -1 to TRACE_ALL or 2 to TRACE_STATEMENT_CALLS. For more information, see your vendor's driver documentation.
	DB2 Universal type 2:
	<ul style="list-style-type: none"> ■ <code>driverType=2</code> Required setting for this driver. ■ <code>readOnly=true</code> Creates a read only connection. ■ <code>currentSchema=YourSchemaName</code> Specifies the default schema name used to qualify unqualified database objects in dynamically prepared SQL statements. ■ <code>loginTimeout=number</code> The maximum time in seconds to wait for the DataSource object to connect to a data source.

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
```

Field	Description/Action
	<p>Oracle JDBC OCI Driver version 8i:</p> <pre>driverType=oci8</pre>
	<p>Oracle JDBC OCI Driver version 9i:</p> <pre>driverType=oci</pre>
	<p>Teradata Type 4:</p> <pre>DSName=value</pre> <p>where <i>value</i> is the Teradata database server name.</p>
	<p>Informix JDBC 2.21 type 4 (for NO_TRANSACTION, LOCAL_TRANSACTION, and XA_TRANSACTION):</p> <pre>IfxIFXHOST=hostname</pre> <p>where <i>hostname</i> is the machine name of the database server.</p>
	<p>Cloudera JDBC Driver 2.5 for Apache Impala</p> <pre>url={jdbc:impala://hostname:portnumber/databasename;AuthMech=value;SSL=value;UseSasl=value}</pre> <p>Where other database related properties are provided as part of the <code>url</code> values.</p> <p>Note: The keys such as <code>AuthMech</code>, <code>SSL</code>, and <code>UseSasl</code> are required only for secured connections.</p>

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

Field	Description/Action
Enable Connection Pooling	<p>Enables the connection to use connection pooling. For more information about connection pooling, see “Adapter Connections” on page 18.</p> <p>Note: If you plan to enable connection pooling in a clustered environment, consider the connection pool size. For details, see “Considerations When Configuring Connections with Connection Pooling Enabled” on page 74.</p>
Minimum Pool Size	<p>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</p>

Field	Description/Action
	connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	<p>If connection pooling is enabled, this field specifies the number of milliseconds that 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.</p>
Expire Timeout	<p>If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the Minimum Pool Size. The inactivity timer for a connection is reset when the connection is used by the adapter.</p> <p>If you set the Expire Timeout value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on</p>

Field	Description/Action
	<p>your backend resource. This could have an adverse effect if your resource has a limited number of connections.</p> <p>If you set the Expire Timeout value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the Minimum Pool Size to avoid excessive opening/closing of connections during normal processing.</p>
Startup Retry Count	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	The number of seconds that the system should wait between attempts to initialize the connection pool.

8. Click **Save Connection**.

The connection you created appears on the adapter's Connections screen and in Designer.

You can enable a connection only if the parameters for the connection are valid.

Required Connection Property Fields

The following table lists the required **Connection Properties** tab fields by driver type. For additional information, see [“Using JDBC Drivers to Connect to Databases” on page 19](#).

Driver Name	server Name	user	pass word	database Name	port Number	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 2 (for UDB 8.1, 9.1, and 9.5)	Yes	Yes	Yes	Yes	Yes	No
DB2 Universal type 4 (for UDB 7.2, 8.1, 9.1, and 9.5)	Yes	Yes	Yes	Yes	Yes	No
Teradata Type 4 v03.04.00 (for V2R5 and V2R6).	No	Yes	Yes	No	No	No

Driver Name	server Name	user	pass word	database Name	port Number	network Protocol
Teradata Type 4 v12.00.00.01 (for R12.0)	No	Yes	Yes	Yes	No	No
Teradata Type 4 v13.00.00.20 (for R13.0, R13.10, and R14.0)	No	Yes	Yes	Yes	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
MySQL-AB JDBC Driver mysql-connector-java-5.0.8	Yes	Yes	Yes	Yes	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, v5r2, v5r3, and v5r4)	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 and 6.05 type 4 (for Sybase v. 11.x, 12.x, and 15.x)	Yes	Yes	Yes	Yes	Yes	No

Driver Name	server Name	user	pass word	database Name	port Number	network Protocol
Advanced Server JDBC Connector (edb-jdbc16.jar)	Yes	Yes	Yes	Yes	Yes	No
Apache Impala Cloudera JDBC Driver 2.5	No	Yes	Yes	No	No	No

Dynamically Changing a Service's Connection at Run Time

You can run a service using a connection other than the default connection that was associated with the service when the service was created.

Important: At run time, you can change either the credentials (user name and password) or the connection name associated with a specific service, but not both at the same time. If you override both the credentials and the connection name, Adapter for JDBC takes into account only the connection name override.

To override the default connection, 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.

For more information, see [“Changing the Connection Associated with an Adapter Service at Run Time” on page 26](#).

Dynamically Changing the User Credentials of a Service's Connection at Run Time

In Adapter for JDBC, you can dynamically provide the user name and password credentials associated with a specific adapter service at run time. This capability enables you to override the connection that is associated with the adapter service at design time. If you provide the user name and password credentials in an adapter service at run time, Adapter for JDBC connects to the database using the new credentials, along with the other connection parameters associated with the service's associated connection. If you do not provide any user credentials at run time, Adapter for JDBC connects to the database using the user credentials provided at design time.

For more information, see [“Changing the User Credentials of a Service's Associated Connection at Run Time”](#) on page 27.

Viewing Adapter Connection Parameters

You can view a connection's parameters from Integration Server Administrator and Designer.

Using Integration Server Administrator to View Adapter Connection Parameters


Perform the following steps to view adapter connection parameters in Integration Server Administrator.

To view the parameters for a connection using Integration Server Administrator

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

When using the adapter with Integration Server 8.0 and later, you can sort and filter the list of connections that appears on the Connections screen.

- To sort information on the Connections screen, click the **Up** and **Down** arrows at the top of the column you want to sort.
- To filter the list of connections:
 - i. On the Connections screen, click **Filter Connections**.
 - ii. Type the criterion by which you want to filter into the **Filter criteria** box. Filtering is based on the node name, not the connection alias. To locate all connections containing specific alphanumeric characters, use asterisks (*) as wildcards. For example, if you want to display all connections containing the string "abc", type *abc* in the **Filter criteria** box.

- iii. Click **Submit**. The Connections screen displays the connections that match the filter criteria.
 - iv. To re-display all connections, click **Show All Connections**.
The Connections screen appears, listing all the current connections. You can control the number of connections that are displayed on this screen. For more information, see [“Controlling Pagination” on page 57](#).
2. On the Connections screen, click the  icon for the connection you want to see.
The View Connection screen displays the parameters for the connection. For descriptions of the connection parameters, see [“Configuring Adapter for JDBC Connections” on page 80](#).
 3. Click **Return to JDBC Adapter connections** to return to the main connections screen.

Using Designer to View Adapter Connection Parameters

Perform the following steps to view adapter connection parameters in Designer.


To view the parameters for a connection using Designer

1. From the Designer navigation area, open the package and folder in which the connection is located.
2. Double-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 Adapter for JDBC Connections” on page 80](#).

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 Integration Server Administrator.

To edit a connection

1. In the **Adapters** menu in Integration Server Administrator's navigation area, click **JDBC Adapter**.
2. Make sure that the connection is disabled before editing it. For instructions, see [“Disabling Adapter Connections” on page 99](#).
3. On the Connections screen, click the  icon for the connection you want to edit.
The Edit Connection screen displays the current parameters for the connection. Update the connection's parameters by typing or selecting the values you want to specify.


For descriptions of the connection parameters, see [“Configuring Adapter for JDBC Connections” on page 80](#).

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

Copying Adapter Connections

You can copy an existing Adapter for JDBC 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 Integration Server Administrator.

To copy a connection

1. In the **Adapters** menu in Integration Server Administrator's navigation area, click **Adapter for JDBC**.
2. On the Connections screen, click the  icon for the connection you want to copy.

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

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

For descriptions of the connection parameters, see [“Configuring Adapter for JDBC Connections” on page 80](#).


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

Deleting Adapter Connections

If you no longer want to use a particular Adapter for JDBC connection, you can delete it. You delete adapter connections using Integration Server Administrator.

If you delete a Adapter for JDBC connection, the adapter services or notifications that are defined to use the connection will no longer work. However, you can assign a different connection to an adapter service and re-use the service. To do this, use the `setAdapterServiceNodeConnection` built-in service. For more information, see [“Changing the Connection Associated with an Adapter Service or Notification at Design Time” on page 26](#).

To delete a connection

1. In the **Adapters** menu in the Integration Server Administrator navigation area, click **Adapter for JDBC**.
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.
Integration Server deletes the adapter connection.


Enabling Adapter Connections

A Adapter for JDBC 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 Integration Server 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 Integration Server Administrator navigation area, click **Adapter for JDBC**.
2. On the Connections screen, click **No** in the **Enabled** column for the connection you want to enable.

Integration Server Administrator enables the adapter connection and displays a  and **Yes** in the **Enabled** column.

Disabling Adapter Connections

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

To disable a connection

1. In the **Adapters** menu in the Integration Server Administrator navigation area, click **Adapter for JDBC**.
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.

5 Adapter Services

■ Overview	102
■ Before Configuring or Managing Adapter Services	102
■ Configuring SelectSQL Services	103
■ Configuring InsertSQL Services	107
■ Configuring UpdateSQL Services	110
■ Configuring BatchInsertSQL Services	114
■ Configuring BatchUpdateSQL Services	117
■ Configuring DeleteSQL Services	122
■ Configuring CustomSQL Services	124
■ Configuring DynamicSQL Services	129
■ Configuring StoredProcedure Services	133
■ Configuring StoredProcedureWithSignature Services	137
■ Configuring ExecuteService Services	144
■ Testing Adapter Services	146
■ Viewing Adapter Services	147
■ Editing Adapter Services	147
■ Deleting Adapter Services	148
■ Validating Adapter Service Values	148
■ Reloading Adapter Values	149

Overview

This chapter describes how to configure and manage Adapter for JDBC services. For detailed descriptions of the available Adapter for JDBC services, see [“Adapter Services” on page 21](#).

Before Configuring or Managing Adapter Services

Perform the following steps before configuring or managing adapter services.

To prepare to configure or manage Adapter for JDBC services

1. Start your Integration Server and Integration Server Administrator, if they are not already running.
2. Make sure you have Integration Server Administrator privileges so that you can access Adapter for JDBC's administrative screens. For information about setting user privileges, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.
3. Be sure to check [“Database Driver Known Limitations” on page 261](#) 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 Integration Server Administrator, make sure the WmJDBCAdapter package is enabled. For instructions, see [“Enabling Packages” on page 68](#).
6. Using Integration Server Administrator, configure an adapter connection to use with the adapter service. For instructions, see [“Adapter for JDBC Connections” on page 77](#).

Note: Integration Server provides 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 26](#).

7. Start Software AG Designer if it is not already running.
8. Using Designer, 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 [“Package Management” on page 65](#).

Configuring SelectSQL Services

A SelectSQL service retrieves specified information from a database table. You configure Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services” on page 24](#).

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.


To configure a SelectSQL service



1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **SelectSQL** template and 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.

6. 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 <code>t1</code> .
Table Name	Select a table name. The default for the associated catalog name is <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.</p> </div>
Type	The type displays automatically based on the table you select.


7. 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  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.
8. 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  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.
 - 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.

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 .
Output Field Type	The data type of the output field. Adapter for JDBC 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 218 .
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.

Field	Description/Action
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.
Query Time Out	<p>Specify the query time-out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is -1. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out.</p> <p>If you specify a value greater than 0, the service executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>
Result Field	Specify a name for the output field that must contain the total number of rows affected by the SQL statement. Do not use <i>results</i> as the value of the Result Field .
Result Field Type	The data type of the Result Field .

9. Use the **WHERE** tab to specify the conditions for selecting information:
- Select the  icon to define new WHERE clause fields.
 - Select a logical operator from the **AND/OR** field, an **Operator**, and separators (the left and right parentheses) as needed, and specify values for 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.

The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of the inserted row.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
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 218 .
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the Parameter and Column fields. However, you can also choose to specify any custom value.

Note: For Oracle users, if you use a CHAR(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.

Note: The WHERE clause does not support the java.sql.Array data type.

- c. If necessary, use the  or  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.

10. From the **File** menu, select **Save**.

Configuring InsertSQL Services

An InsertSQL service inserts new information into a database table. You configure Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services” on page 24](#).

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.

To configure an InsertSQL service



1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **InsertSQL** template and 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.

6. 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error.

Field	Description/Action
	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.

7. 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.
- Use the  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.

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 <code>?</code> , 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.

- 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.

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.

Field	Description/Action
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(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.

- c. Specify the query time out value of the InsertSQL service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard Statement.SetQueryTimeout() method relies on the Statement.cancel() method. When execution takes longer than the specified time-out interval, the monitor thread calls Statement.cancel(). In some cases, because of a limitation in the Statement.cancel() method, the time out does not free the thread that invoked the Statement.execute() method and this may lead to higher waiting times.</p> <p>The default value is -1. Use the default value to have the service use the value indicated on the watt.adapter.JDBC.QueryTimeout property as the time out. If you specify a value equal to 0, or if the watt.adapter.JDBC.QueryTimeout property is not set, the service executes without a time out. If you specify a value greater than 0, the service executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p>

Field	Description/Action
	For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.

- 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.
- From the **File** menu, select **Save**.

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 Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services”](#) on page 24.

Be sure to review the section [“Before Configuring or Managing Adapter Services”](#) on page 102 before you configure adapter services.

To configure an UpdateSQL service



- In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
- Select the parent namespace, type a name for the adapter service, and click **Next**.
- Select **Adapter for JDBC** as the adapter type and click **Next**.
- Select the appropriate **Adapter Connection Name** and click **Next**.
- From the list of available templates, select the **UpdateSQL** template and 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.

- 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error.

Field	Description/Action
	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.

7. 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:
- Use the  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.

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 <code>?</code> , 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.

- 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.

Field	Description/Action
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(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.


- c. Specify the query time out value of the UpdateSQL service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard Statement.SetQueryTimeout() method relies on the Statement.cancel() method. When execution takes longer than the specified time-out interval, the monitor thread calls Statement.cancel(). In some cases, because of a limitation in the Statement.cancel() method, the time out does not free the thread that invoked the Statement.execute() method and this may lead to higher waiting times.</p> <p>The default value is -1. Use the default value to have the service use the value indicated on the watt.adapter.JDBC.QueryTimeout property as the time out. If you specify a value equal to 0, or if the watt.adapter.JDBC.QueryTimeout property is not set, the service executes without a time out. If you specify a value greater than 0, the service executes with the specified value as the time out.</p>

Note: -1 is the only permissible negative value for this field.

For more information about the watt.adapter.JDBC.QueryTimeout property, see [“Forcing a](#)

Field	Description/Action
	Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.

8. Use the **WHERE** tab to specify the conditions for selecting information:
- Select the  icon to define new WHERE clause fields.
 - 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.

The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of the inserted row.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
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 218.

Field	Description
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the Parameter and Column fields. However, you can also choose to specify any custom value.
Note:	For Oracle users, if you use a CHAR(n) data type and enter a value in the Input Field , Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.
Note:	The WHERE clause does not support the java.sql.Array data type.

- 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.
- From the **File** menu, select **Save**.

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 Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services” on page 24](#).

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.

Note: BatchInsertSQL services cannot be used with a Teradata database (any version).

To configure a BatchInsertSQL Service

- In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
- Select the parent namespace, type a name for the adapter service, and click **Next**.
- Select **Adapter for JDBC** as the adapter type and click **Next**.
- 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. For instructions for



creating a LOCAL_TRANSACTION connection, see [“Configuring Adapter for JDBC Connections” on page 80](#). For information about reloading metadata values, see [“Reloading Adapter Values” on page 149](#).

5. From the list of available templates, select the **BatchInsertSQL** template and 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.

6. 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. 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.

7. 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  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.

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 <code>?</code> , 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

Field	Description/Action
	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:

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(n)` data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the `ORACLEFIXED_CHAR` data type as the JDBC data type.

- c. Specify the query time out value of the BatchInsertSQL service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer</p>

Field	Description/Action
	<p>than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is <code>-1</code>. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to <code>0</code>, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than <code>0</code>, the service executes with the specified value as the time out.</p> <p>Note: <code>-1</code> is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

8. 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.
9. From the **File** menu, select **Save**.

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 Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services”](#) on page 24.

Note: BatchUpdateSQL services cannot be used with a Teradata database (any version).

Be sure to review the section [“Before Configuring or Managing Adapter Services”](#) on [page 102](#) before you configure adapter services.

To configure a BatchUpdateSQL Service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** 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. For instructions for creating a LOCAL_TRANSACTION connection, see [“Configuring Adapter for JDBC Connections”](#) on [page 80](#). For information about reloading metadata values, see [“Reloading Adapter Values”](#) on [page 149](#).



5. From the list of available templates, select the **BatchUpdateSQL** template and 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.

6. 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error.
	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.

7. 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  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.

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 **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.
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(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.

- c. Specify the query time out value of the BatchUpdateSQL service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is <code>-1</code>. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to <code>0</code>, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than <code>0</code>, the service executes with the specified value as the time out.</p> <p>Note: <code>-1</code> is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

8. Use the **WHERE** tab to specify the conditions for selecting information:
- Select the  icon to define new WHERE clause fields.
 - 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 218 .
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: The WHERE clause does not support the java.sql.Array data type.

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. From the **File** menu, select **Save**.

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 Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services” on page 24](#).

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.


To configure a DeleteSQL service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **DeleteSQL** template and 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.

6. 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. 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.

7. Use the **WHERE** tab to specify the conditions for selecting information:
 - a. Select the  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.

The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of the inserted row.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
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 218.
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the Parameter and Column fields. However, you can also choose to specify any custom value.

Note: For Oracle users, if you use a CHAR(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.

Note: The WHERE clause does not support the `java.sql.Array` data type.

8. 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.
9. To verify input or output information for this service, use the **Input/Output** tab as needed.
10. From the **File** menu, select **Save**.

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 Adapter for JDBC services using Designer. For more information about adapter services, see [“Using Adapter Services” on page 24](#).

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. For details, see your vendor documentation.

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.

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. For instructions, see [“Configuring StoredProcedure Services” on page 133](#).

Considerations before using Fill in all rows to the table feature in CustomSQL Adapter Service

Before you begin, ensure that the connection to the database is enabled. Also ensure that the Input/Output tab in the Designer is empty. The following table describes the restrictions when creating an SQL query statement using *Fill in all rows to the table* feature in CustomSQL Adapter Service:

Considerations	Invalid Query input	Valid Query input
Specify the table alias	<code>select deptno , empno from dept,emp deptno</code>	<code>select d.deptno, e.empno from dept d,emp e where d.deptno = ?</code>

Considerations	Invalid Query input	Valid Query input
along with the table name if the SQL query contains more than one table.		
Use different column name alias for columns in the SQL query.	<pre>select firstname as name, lastname as name from emp select name,name from emp</pre>	<pre>select firstname as f_name, lastname as l_name from emp select name as name1,name as name2 from emp</pre>
For Sybase databases, specify an alias for the column result if you are using functions on an output column.	<pre>select e.deptno, count(e.job) from employees e group by e.deptno</pre>	<pre>select e.deptno, count(e.job) as count from employees e group by e.deptno</pre>
Column names in the SQL query should not be enclosed in quotes.	<pre>select partno as 'partno' from emp where partno = 1</pre>	<pre>select partno as partno from emp where partno=1</pre>
Use the “as” keyword when you specify an alias for a column.	<pre>select city dummy from emp</pre>	<pre>select city as dummy from emp</pre>
Irrespective of the database type, the query syntax must follow the SQL standards. Standards corresponding to specific database	<p>For MySQL query,</p> <pre>insert into example_default_now set id=?,data=?</pre>	<p>For MySQL query,</p> <pre>insert into example_default_now [(set,data)] values (?,?)</pre>

Considerations	Invalid Query input	Valid Query input
types are not supported.		
<p>Note: If your SQL query has errors or does not follow the considerations specified in the above table, the Fill in all rows to the table icon does not populate the input and output parameters. The the input and output parameters must be configured manually. You can view the error message by clicking the Reload values from adapter icon.</p>		

Creating a CustomSQL service



Use the following instructions to create a CustomSQL adapter service. You configure JDBC Adapter services using Designer.

To create a CustomSQL service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **CustomSQL** template and 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.

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

Use the  icon and to create new rows as needed. You can use the  icon to fill in all rows to the table.

Be sure to review the section [“Considerations before using Fill in all rows to the table feature in CustomSQL Adapter Service”](#) on page 124 before you use the Fill in all rows to the table icon.

Set the SQL parameters as described in the following table:

Note: When using the CustomSQL service for a Select SQL, it is not mandatory to configure the output fields **Output JDBC Type**, **Output Field Type**, and **Output Field**. Adapter for JDBC uses the fields provided in the Select SQL statement as the output parameter fields.

Field	Description/Action
SQL*	<p>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:</p> <pre>select short_col, int_col, float_col, double_col, date_col, date_time_col, varchar_col from -ADAPTER-TEST</pre> <p>For variable names, use the ? variable placeholder for each variable. For example:</p> <pre>select employee_name where StaffID = ? and Dept = ?</pre> <p>Note: If you use the ? variable placeholders 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 above, <code>StaffID</code> would be the first entry in the Input Field and <code>Dept</code> would be the second entry.</p> <p>Note: Do not end your SQL statement with a semi-colon (;) or an exception will be generated at run time.</p> <p>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.</p>
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 218.
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 218.
Output Field	The output field name.

Field	Description/Action
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.
Query Time Out	<p>Specify the query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is -1. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than 0, the service executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>
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 .

- From the **File** menu, select **Save**.

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.

Important: Adapter for JDBC does not validate the input parameters of a DynamicSQL service for any malicious SQL injections. When you use a variable input parameter such as the text `${INPUT_FIELD_NAME}` in a DynamicSQL service, you must take extra measures to avoid potential security risks by, for example, using a wrapper service for your DynamicSQL service that will validate the variable input parameters.

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.

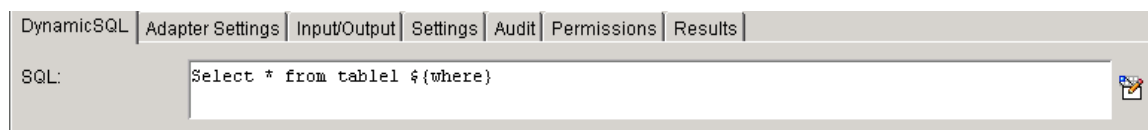
Note: When using the DynamicSQL service for a Select SQL, it is not mandatory to configure the output fields **Output JDBC Type**, **Output Field Type**, and **Output Field**. Adapter for JDBC uses the fields provided in the Select SQL statement as the output parameter fields.

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.

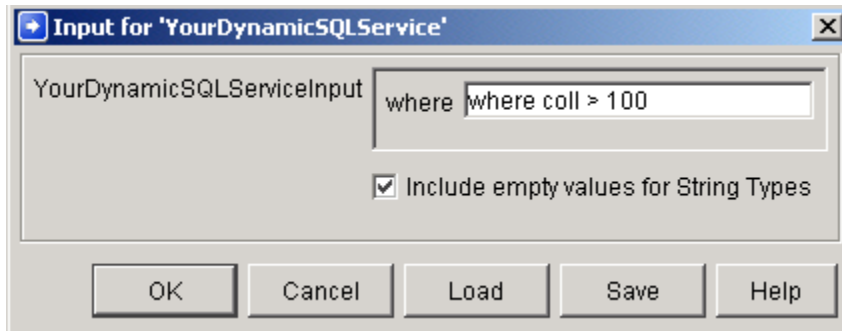
For example, consider the following DynamicSQL statement:

```
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 coll>100`:



The generated SQL statement will be `Select * from table1 where coll>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 Adapter for JDBC services using Designer.

Be sure to review the section [“Before Configuring or Managing Adapter Services” on page 102](#) before you configure adapter services.


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. For instructions, see [“Configuring StoredProcedure Services” on page 133](#).

To create a DynamicSQL service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **DynamicSQL** template and 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.

6. Select the **Dynamic SQL** tab to specify a SQL statement and the associated input and output parameters.

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

Field	Description/Action
SQL	<p>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:</p> <pre>select short_col, int_col, float_col, double_col, date_col, date_time_col, varchar_col from ADAPTER-TEST</pre> <p>For variable names, use the ? variable placeholder for each variable. For example:</p> <pre>select employee_name where StaffID = ? and Dept = ?</pre> <p>Note: If you use the ? variable placeholders 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. In the above example, StaffID would be the first entry in the Input Field and Dept would be the second entry.</p> <p>Note: Do not end your SQL statement with a semi-colon (;) or you will generate an exception.</p> <p>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.</p>
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 218.
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 218.
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.
Query Time Out	<p>Specify the query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is -1. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than 0, the service executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>
Result Field	Name of the output field that contains the total number of rows affected by the SQL statement. Do not use <code>results</code> as the value of the Result Field .

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

- From the **File** menu, select **Save**.

Configuring StoredProcedure Services

A StoredProcedure service calls a stored procedure to perform database operations. 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.

Considerations when Configuring StoredProcedure Adapter Services

You must consider the following restrictions when configuring StoredProcedure services:

- The adapter StoredProcedure service does not support stored procedures that have Array or Struct as OUT parameters. You can use the StoredProcedureWithSignature service instead. For information on configuring StoredProcedureWithSignature service, see [“Configuring StoredProcedureWithSignature Services” on page 137](#).
- When operating on a MySQL database, the adapter StoredProcedure service supports stored procedures, but does not support functions. To call functions, use the StoredProcedureWithSignature service template instead. For more information about configuring StoredProcedureWithSignature services, see [“Configuring StoredProcedureWithSignature Services” on page 137](#).
- MySQL 5.0.x does not support stored procedure and function names containing spaces.

Creating StoredProcedure Adapter Services

You configure Adapter for JDBC services using Designer.





Be sure to review the sections [“Before Configuring or Managing Adapter Services” on page 102](#) and [“Considerations when Configuring StoredProcedure Adapter Services” on page 133](#) before you configure StoredProcedure services.

To configure a StoredProcedure adapter service



1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **StoredProcedure** template and 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.

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

Field	Description/Action
Catalog Name	The name of the catalog. The default for the catalog name is <code>current catalog</code> .
Schema Name	The name of the schema. The default for the schema name is <code>current schema</code> .
Enable Procedure Name Lookup (Optional)	To type in the Procedure Name , set this field to <code>False</code> . To select the Procedure Name from a list, set this field to <code>True</code> . The default is <code>False</code> . 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.
JDBC Type	Specify the JDBC type of the corresponding return field for the stored procedure. Use the  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.
Return Field Name	Add return field names for the stored procedure. Use the  icon to create new rows as needed. You can use the  icon to fill in all rows to the table.

Field	Description/Action
Query Time Out	<p>Specify the query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is <code>-1</code>. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to <code>0</code>, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than <code>0</code>, the service executes with the specified value as the time out.</p> <p>Note: <code>-1</code> is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>


- Use the **Parameter** tab to specify the stored procedure's parameters.
- Use the  icon to create new stored procedure parameters as needed. You can use the  icon to fill in all rows to the table.

Field	Description/Action
Param JDBC Type	The JDBC type of the stored procedure parameter.
Param Name	The stored procedure parameter name.

Field	Description/Action
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 218 .
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 218 .

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

Note: StoredProcedure services can support multiple results sets. The result set can contain nested cursors. When using the result set that contains nested cursors, the performance of Adapter for JDBC could degrade. Since the nested cursors are recursively processed, Adapter for JDBC may also return data that may not be required.

Use the  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.

Field	Description/Action
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 218.

10. From the **File** menu, select **Save**.

Specifying the Maximum Number of Result Set Indexes for a StoredProcedure Adapter Service

The StoredProcedure adapter service can be configured to return multiple result sets. By default, the service can return only a maximum of 20 result sets. A new result set index cannot be added manually. To specify the required maximum number of result sets, you can use the `watt.adapter.JDBC.StoredProcedureMaxCursorIndex` property.

To set this property, use Integration Server Administrator and select **Settings > Extended > Edit Extended Settings**. Enter this property in the Extended Settings box:

```
watt.adapter.JDBC.StoredProcedureMaxCursorIndex=value
```

where *value* is the required maximum number of result sets. For example, to return a maximum number of 30 result sets from the StoredProcedure service, set the property as:

```
watt.adapter.JDBC.StoredProcedureMaxCursorIndex=30
```

The value should be greater than zero with no upper limits. However, it is recommended to provide a value within the practical limits. A large value can lead to an unpredictable behavior of the system. Instead, you can configure another StoredProcedure adapter service for the next set of result sets.

For more information about setting the watt properties, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

Configuring StoredProcedureWithSignature Services

A StoredProcedureWithSignature service calls a stored procedure to perform database operations. Unlike a StoredProcedure adapter service, the StoredProcedureWithSignature service enables you to automatically obtain a stored procedure's parameters by introspecting and listing the signature of the stored procedure at the time you configure the adapter service. This means that you do not need to look up and retype these parameters manually.

Considerations when Configuring StoredProcedureWithSignature Adapter Services

You must consider the following restrictions when configuring StoredProcedureWithSignature adapter services:

- StoredProcedureWithSignature services cannot be used with an Informix or Sybase database (all versions).
- When using Array as IN or OUT parameter in an Oracle database, the input to the IN parameter must be a Java array. The OUT parameter returns a Java array as Java data type, java.lang.Object.
- When operating on a MySQL database, the adapter StoredProcedureWithSignature service supports functions, but does not support stored procedures. To call stored procedures, use the StoredProcedure service template instead. For more information about configuring StoredProcedure services, see [“Configuring StoredProcedure Services” on page 133](#).
- MySQL 5.0.x does not support stored procedure and function names containing spaces.

Creating StoredProcedureWithSignature Adapter Services

You configure Adapter for JDBC services using Designer.

Be sure to review the sections [“Before Configuring or Managing Adapter Services” on page 102](#) and [“Considerations when Configuring StoredProcedure Adapter Services” on page 133](#) before you configure StoredProcedureWithSignature services.

To configure a StoredProcedureWithSignature service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **StoredProcedureWithSignature** template and 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.

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

Field	Description/Action
Catalog Name	The name of the catalog. The default for the catalog name is <code>current catalog</code> .
Schema Name	The name of the schema. The default for the schema name is <code>current schema</code> .
Procedure Name Pattern	To save time, you can type all or part of the procedure name in this field to narrow your search. This is helpful when dealing with a large database that has a long list of procedures. Use % as a multi-character wildcard and _ (underscore) as a single character wildcard. Note: With DB2 databases, functions do not appear in the list of procedure names. Only procedures appear in the list.
Procedure Name	Select the stored procedure name, depending on the how you set the Procedure Name Pattern field. If you select <All Procedures> , this field lists all of the procedures in the selected catalog and schema.
Specific Name (Only for DB2 and DB2AS400)	Select the specific name for a stored procedure in a DB2 or DB2 AS/400 database after specifying the Procedure Name .

7. Based on the signature for the stored procedure, the following fields and values will appear in the top table on the **Call** tab:

Field	Description/Action
Parameter Name	The stored procedure parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the stored procedure parameter.
Parameter Type	Defines the parameter type as IN, INOUT, or OUT. If you select IN or INOUT, you may also set the input expression in the Expression field.

Field	Description/Action
Expression*	Sets the input for the IN or INOUT parameter types only. The RETURN or OUT parameters will appear automatically on the Call tab's bottom table. For a list of the allowable expression settings by parameter type, and how each parameter will map to the input or output fields, see the following table.

The following table shows valid expressions by parameter type:

Parameter Type	Expression	Input or Output Mapping?
RETURN	Empty (default)	Output field
OUT	Empty (default)	Output field
IN	? (default)	Input field
	Fixed value	No mapping
INOUT	? (default)	Input and output field
	Empty	Output field
ORACLE CURSOR (INOUT)	Empty (default)	Output field

8. The middle table on the **Call** tab lists the following input parameters and values for the stored procedure that will map to the input fields of the service:

Field	Description/Action
Input Parameter Name	The stored procedure input parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the input parameter.
Input Field	The name of any input parameters.

Field	Description/Action
Input Field Type	The input parameter Java data type. For a list of JDBC type to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

- a. The bottom table on the **Call** tab lists the following output parameters and values for the stored procedure that will map to the output of the service, including the OUT or INOUT parameters:

Field	Description/Action
Output Parameter Name	The stored procedure output parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the output parameter.
Output Field	The name of the output parameter.
Output Field Type	The output parameter Java data type. For a list of JDBC to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

9. The bottom table on the **Call** tab lists the following output parameters and values for the stored procedure that will map to the output of the service, including the OUT or INOUT parameters:

Field	Description/Action
Output Parameter Name	The stored procedure output parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the output parameter.
Output Field	The name of the output parameter.
Output Field Type	The output parameter Java data type. For a list of JDBC to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

- a. Specify the query time out value of the StoredProcedureWithSignature service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p> <p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is <code>-1</code>. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to <code>0</code>, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than <code>0</code>, the service executes with the specified value as the time out.</p> <p>Note: <code>-1</code> is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

10. Specify the query time out value of the StoredProcedureWithSignature service you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds.</p> <p>This value is the amount of time Adapter for JDBC waits for the service to execute before stopping the SQL operation.</p>

Field	Description/Action
	<p>The time out specified in the Query Time Out field is not guaranteed but depends on the implementation specific to the driver vendor. The JDBC standard <code>Statement.SetQueryTimeout()</code> method relies on the <code>Statement.cancel()</code> method. When execution takes longer than the specified time-out interval, the monitor thread calls <code>Statement.cancel()</code>. In some cases, because of a limitation in the <code>Statement.cancel()</code> method, the time out does not free the thread that invoked the <code>Statement.execute()</code> method and this may lead to higher waiting times.</p> <p>The default value is <code>-1</code>. Use the default value to have the service use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to <code>0</code>, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the service executes without a time out. If you specify a value greater than <code>0</code>, the service executes with the specified value as the time out.</p> <p>Note: <code>-1</code> is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

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

StoredProcedureWithSignature services can support multiple results sets. The result set can contain nested cursors.

Note: When using the result set that contains nested cursors, the performance of Adapter for JDBC could degrade. Since the nested cursors are recursively processed, Adapter for JDBC may also return data that may not be required.

Use the  icon to create additional result sets as needed.

Note: While all the tables in the **Call** tab will be updated automatically if the selected stored procedure changes, the **ResultSet** tab information is not updated automatically. To update this information, you must manually update the fields in the **ResultSet** tab.

Provide values for the following parameters:

Field	Description/Action
Result Set Index	An index is automatically assigned to each result set. The first row default value is 1. Note: When using for Oracle database, this field is not required.
Result Set Name	The name of the result set you want to create. Note: When using for Oracle database, this field is not required.
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 data type of the result column.
Output Type	The Java data type of the result column. For a list of JDBC to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

12. From the **File** menu, select **Save**.

Configuring ExecuteService Services

An ExecuteService allows a Java or a flow service to use a connection from the adapter's connection pool. You can configure the ExecuteService using Designer. For more information on how a service can use a connection, see [“Using a Connection from the Connection Pool Within a Java or Flow Service”](#) on page 26.

To configure an ExecuteService service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.

- From the list of available templates, select the **ExecuteService** template and click **Finish**.

The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm the adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

- Select the **Service to Invoke** tab to specify the Java or the flow service name that needs the connection:

Field	Description/Action
Service Name	The Java or the flow service name that requires a connection from the connection pool. If you specify an incorrect service name, an exception is thrown at run time.

- From the **File** menu, select **Save**.

Considerations When Creating a Java or Flow Service that Uses a Connection from the Connection Pool

- Use the value in the `$db_service_connection` from the pipeline to obtain the connection. You can also use the following Java String constant variable in your Java or flow service code, provided that the package in which you create the service is dependent on the `WmJDBCAdapter` package:

```
com.wm.adapter.wmjdbc.services.ExecuteService.PIPELINE_CONNECTION
```

For information about setting the package dependencies, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.

- Do not use the forbidden methods like `close()`, `setAutoCommit()`, and `commit()` in the service code. Doing so causes the adapter to throw an exception at run time.
- When using the `ExecuteService` in transactions, it is important for the `WmART` package to be aware about any exceptions generated in the service within the transaction `SEQUENCE` block. Only then will a roll back of the transaction happen. You can do this by throwing a `ServiceException` from the service. You can use the following sample code in the service code to do so.

```
java.sql.Connection conn =
(java.sql.Connection) IDataUtil.get (pipeline.getCursor (),
com.wm.adapter.wmjdbc.services.ExecuteService.PIPELINE_CONNECTION);
try
{
java.sql.PreparedStatement pstmt=conn.prepareStatement("insert into Person
values (1, 'Chris')");
pstmt.execute();
}
catch (Throwable e)
{
e.printStackTrace();
}
```

```
throw new ServiceException(e);
}
```

Note: If an error occurs when using the `ExecuteService` in a transaction for a Teradata database, the roll back of transactions does not happen.

Testing Adapter Services

You use Designer to test adapter services.

For more information about testing and debugging services, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.

To test adapter services

1. In Designer, expand the package and folder that contain the service you want to test.
2. Double-click the service you want to test.

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

3. Select **Run > Run As > Run Service**.
4. Specify how you want to connect to the database, using either of the following ways:
 - To connect to a database using a connection that is different than the connection specified during design time, specify the connection name here:

Parameter	Description
<code>\$connectionName</code>	The connection used to connect to the database.

Note: The database schema of the overridden connection must be the same as that of the default connection. For more information, see [“Dynamically Changing a Service's Connection at Run Time” on page 95](#).

- To connect to a database using user credentials of a connection that are different than the user credentials provided when configuring the connection, specify the user name and password here.

Parameter	Description
<code>\$dbUser</code>	The user name used to connect to the database.
<code>\$dbPassword</code>	The password used to connect to the database.

Note: If you do not provide the user credentials here, the user credentials specified when the connection was configured are used. For more information, see [“Dynamically Changing the User Credentials of a Service's Connection at Run Time”](#) on page 96.

Note: You can override either the connection or the user credentials at run time. If you provide both, the connection name and the user credentials, the connection name takes precedence and the service is invoked using the overridden connection.

5. If you defined any other input fields, you will be prompted to type their input values. Type the values for each input field and then click **OK**.
6. Click the **Service Result** tab to view the output from the service.

Viewing Adapter Services

You use Designer to view adapter services.

To view a service

1. In Designer, expand the package and folder that contain the service you want to view.
2. Double-click the service you want to view.

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

Editing Adapter Services

You use Designer to edit adapter services.

To edit an adapter service

1. In Designer, browse to and open the adapter service that you want to edit.
2. Double-click the service that you want to edit.

Designer displays the adapter service in the service template's Adapter Service Editor.

3. Do one of the following:
 - If you have the VCS Integration feature enabled, right-click the service and select **Check Out**.
 - If you do not have the VCS Integration feature enabled, right-click the service and select **Lock for Edit**.

- If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check out the service. The options available in the **Team** menu depend on the VCS client that you use.
4. Modify the values for the adapter 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.
5. After you complete your modifications, save the service and do one of the following:
 - If you have the VCS Integration feature enabled, right-click the service and select **Check In**. Enter a check-in comment and click **OK**.
 - If you do not have the VCS Integration feature enabled, right-click the service and select **Unlock**.
 - If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check in the service. The options available in the **Team** menu depend on the VCS client that you use.
6. Save the service.

Deleting Adapter Services

You use Designer to delete adapter services.

To delete a service

1. In Designer, expand the package and folder that contain the service you want to delete.
2. Right-click the adapter service and click **Delete**.

Validating Adapter Service Values

Designer enables Adapter for JDBC 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 Designer to always validate the values for adapter services. Both options could potentially slow your design-time operations.

If you select the option to always validate values for adapter services, it will do so for all webMethods WmART-based adapters installed on Integration Server.

For more information about the **Adapter Service/Notification Editor** and other Designer menu options and toolbar icons, see the Designer Service Development online help for your release. See "About this Guide" for specific document titles.


Validate Data for a Single Adapter Service

Perform the following procedure to validate data for a single adapter service.

To validate data for a single adapter service

1. In Designer, expand the package and folder that contain the service for which you want to enable automatic validation.
2. Double-click the service for which you want to validate the data.

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

3. Click the  icon.

Validating Data for All Adapter Services

Perform the following procedure to enable Designer to always validate data for all adapter services.

To always validate the values for all adapter services

1. In Designer, select the **Window > Preferences > Software AG > Service Development > Adapter Service/Notification Editor** item.
2. Enable the **Automatic data validation** option.
3. Click **OK**.

Reloading Adapter Values

You can enable Adapter for JDBC to reload and validate user-defined data for adapter services at design time in Designer. You can reload values for a single adapter service or you can configure Designer so it automatically reloads the values for adapter services. Both options could potentially slow your design-time operations.

When you reload adapter values for a single adapter service, Designer compares the service values against the resource data that has already been fetched from the selected adapter.

If you select the option to always reload values for adapter services, it will do so for all webMethods WmART-based adapters installed on Integration Server.

For more information about the **Adapter Service/Notification Editor**, other menu options, and toolbar icons, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.


Reloading the Values for a Single Adapter Service

Perform the following procedure to reload the adapter values for a single adapter service.

To reload the adapter values for a single adapter service

1. In Designer, expand the package and folder that contain the service for which you want to enable automatic validation.
2. Double-click the service for which you want to validate the data.

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

3. Click the  icon.

Reloading the Values for All Adapter Services

Perform the following procedure to reload the adapter values for all adapter services.

To reload the adapter values for all adapter services

1. In Designer, select the **Window > Preferences > Software AG > Service Development > Adapter Service/Notification Editor** item.
2. Enable the **Automatic polling of adapter metadata** option.
3. Click **OK**.

6 Adapter Notifications

■ Overview	152
■ Before Configuring or Managing Notifications	152
■ Configuring InsertNotifications	153
■ Configuring UpdateNotifications	158
■ Configuring DeleteNotifications	164
■ Configuring BasicNotifications	170
■ Configuring StoredProcedureNotifications	175
■ Configuring StoredProcedureNotificationWithSignature	179
■ Configuring OrderedNotifications	184
■ Managing Polling Notifications	191
■ Using the Exactly Once Notification Feature	193
■ Exporting Configured Adapter Notifications	193
■ Viewing Notifications	195
■ Editing Notifications	195
■ Deleting Notifications	196
■ Validating Adapter Notification Values	196
■ Reloading Adapter Values	197

Overview

This chapter describes how to configure and manage Adapter for JDBC notifications. For detailed descriptions of the available Adapter for JDBC notifications, see [“Adapter Notifications” on page 29](#).

Before Configuring or Managing Notifications

Perform the following steps before configuring or managing notifications.

To prepare to configure or manage Adapter for JDBC notifications

1. Start your Integration Server and Integration Server Administrator, if they are not already running.
2. Make sure you have Integration Server Administrator privileges so that you can access Adapter for JDBC's administrative screens. For information about setting user privileges, see the Integration Server administration guide for your release. See [“About this Guide”](#) for specific document titles.
3. Be sure to check [“Database Driver Known Limitations” on page 261](#) 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, for details, see [“Using the Exactly Once Notification Feature” on page 193](#).
6. Using Integration Server Administrator, make sure the WmJDBCAdapter package is enabled. For instructions, see [“Enabling Packages” on page 68](#).
7. Using Integration Server Administrator, configure an adapter connection to use with the notification. For instructions, see [“Configuring Adapter for JDBC Connections” on page 80](#).

Note: Integration Server provides 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 26](#).

8. Start Software AG Designer if it is not already running.
9. Using Designer, 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 [“Package Management” on page 65](#).

10. You must schedule a notification and then enable it before you can use the notification. For instructions, see [“Managing Polling Notifications” on page 191](#).
11. If you want to enable the same polling notification on multiple Integration Server instances connecting to the same source database to achieve automated failover, you have to use the polling notification within an Integration Server cluster. For information how to configure clustered polling notifications, see [“Polling Notification Support in Integration Server Clusters” on page 71](#).

Configuring InsertNotifications

An InsertNotification publishes notification of insert operations on a database table. For more information about notifications, see [“Adapter Notifications” on page 29](#).

Considerations when Configuring InsertNotifications

You must consider the following restrictions when configuring InsertNotifications:

- InsertNotifications cannot be used with a Teradata database (any version).
- MySQL version 5.0.x does not support multiple triggers with the same event (insert, delete, or update) for one table. Hence, with a MySQL database, when using an InsertNotification to monitor a table, you must disable the notification before you can enable another InsertNotification or an OrderedNotification with an insert operation for the same table.

Creating an InsertNotification

You configure notifications using Designer.

Be sure to review the sections [“Before Configuring or Managing Notifications” on page 152](#) and [“Considerations when Configuring InsertNotifications” on page 153](#) before you configure InsertNotifications.

To create an InsertNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **InsertNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.

6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.

- a. 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.
- b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to webMethods Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	<p>The name of the JMS connection alias configured on Integration Server.</p> <p>If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p> </div>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

The information from the **Permissions** tab appears in the **Properties** panel.


8. Select the **Notification Configure** tab and use the following fields:

Field	Description/Action
Base Name	<p>The base name used to generate the Resource Name created by Adapter for JDBC.</p> <p>Note: For OS/390 DB2 7.2, the Base Name you create below must be no more than five characters because triggers on OS/390 name cannot be more than eight characters.</p>
Publish Locally	<p>Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.</p>
Resource Type	<p>Types are buffer table, trigger, and sequence. The base name and resource type determine the following Resource Name.</p>
Resource Name	<p>To ensure uniqueness, the resource name combines the following elements. You cannot edit this name.</p> <ul style="list-style-type: none"> ■ 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	<p>The format of the file record. Optional field used by DB2 for AS/400 V4R5 only.</p>
Database Name	<p>The name of the database where the buffer tables will be created. Optional field used by DB2 for OS/390 only.</p>
Table Space Name	<p>The table space where the buffer tables will be created. Optional field used by DB2 for OS/390 only.</p>



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 <code>t1</code> .
Table Name	Select a table name. The default for the associated catalog name is <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. 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  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 using the following fields:

Note: When using the Join clause, select only the fields of the monitor table in the **Select** tab.

- 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  icon (or the  icon to fill in all rows of the table) 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:




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. Adapter for JDBC 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 218 .
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.
Query Time Out	Specify the query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.

Note: -1 is the only permissible negative value for this field.

For more information about the `watt.adapter.JDBC.QueryTimeout` property, see [“Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56](#).

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

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  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  or  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. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
14. From the **File** menu, select **Save**.
15. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications” on page 191](#).

Configuring UpdateNotifications

An UpdateNotification publishes notification of update operations on a database table. For more information about notifications, see [“Adapter Notifications” on page 29](#).

Considerations when Configuring UpdateNotifications

You must consider the following restrictions when configuring UpdateNotifications:

- UpdateNotifications cannot be used with a Teradata database (any version).
- MySQL version 5.0.x does not support multiple triggers with the same event (insert, delete, or update) for one table. Hence, with a MySQL database, when using an UpdateNotification to monitor a table, you must disable the notification before you can enable another UpdateNotification or an OrderedNotification with an update operation for the same table.
- When using an UpdateNotification with a MySQL database, updating any database field with the same value will not invoke a trigger.

Creating an UpdateNotification

You configure notifications using Designer.

Be sure to review the sections [“Before Configuring or Managing Notifications” on page 152](#) and [“Considerations when Configuring UpdateNotifications” on page 158](#) before you configure an UpdateNotification.

To create an UpdateNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **UpdateNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.

6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	The name of the JMS connection alias configured on Integration Server. If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property

Field	Description/Action
	watt.art.notification.jmsSend.usePublicService and set it to true . Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

8. 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 Adapter for JDBC. Note: For OS/390 DB2V7.2, the Base Name you create below must be no more than five characters because triggers on OS/390 name cannot be more than eight characters.
Publish Locally	Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.
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. <ul style="list-style-type: none"> ■ Resource prefix (WMB, WMT, and WMS for buffer table, trigger, and sequence respectively) ■ The name you typed in the Base Name field

Field	Description/Action
	<ul style="list-style-type: none"> ■ 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.

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. The default for the associated catalog name is <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.</p> </div>
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.
- Select the  icon to create new left and right columns.
 - Select **Left Column** and select the first table's joining column.
 - Select the appropriate **Operator**.
 - 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 as follows:

Note: When using the Join clause, select only the fields of the monitor table in the **Select** tab.

- 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 ALLname fromtablename`. The default value is blank, which corresponds to the SQL statement `SELECTname fromtablename`.
- b. Select the  icon (or the  icon to fill in all rows of the table) 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:




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. Adapter for JDBC 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 218 .
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	<p>Enable this option to indicate for which of the columns specified in the SELECT tab you want a notification, if the column is updated. Select:</p> <ul style="list-style-type: none"> ■ Yes if you want a notification if this column of data has been updated. Yes is the default value. ■ No if you do not want a notification if this column of data has been updated. For example, you configure the following three output fields: MyName, MyNumber, and

Field	Description/Action
	<p>MyLocation. You want a 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.</p>
<p>Output Value Type</p>	<p>Specifies which output value to retrieve from the database table. By default, the UpdateNotification retrieves the new value from the database table. Select either of the following output value types:</p> <ul style="list-style-type: none"> ■ Old: Retrieves the old value from the database table ■ New: Retrieves the new value from the database table <p>To retrieve both the old and the new values, create two rows and then select an Output Value Type as Old in one row and an Output Value Type as New in the other row. While doing so, ensure that the Output Field is unique for both the old and the new values.</p> <p>Note: For Sybase and Microsoft 2000/2005, you cannot retrieve both the old and the new values in the same notification. The notification must retrieve either the old or the new value.</p>
<p>Maximum Row</p>	<p>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.</p>
<p>Query Time Out</p>	<p>Specify the query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a</p>

Field	Description/Action
	Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.

12. 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  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  or  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. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
14. From the **File** menu, select **Save**.
15. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications” on page 191.](#)

Configuring DeleteNotifications

A DeleteNotification publishes notification of delete operations on a database table. For more information about notifications, see [“Adapter Notifications” on page 29.](#)

Considerations when Configuring DeleteNotifications

You must consider the following restrictions when configuring DeleteNotifications:

- DeleteNotifications cannot be used with a Teradata database (any version).
- MySQL version 5.0.x does not support multiple triggers with the same event (insert, delete, or update) for one table. Hence, with a MySQL database, when using a DeleteNotification to monitor a table, you must disable the notification before you

can enable another DeleteNotification or an OrderedNotification with a delete operation for the same table.

Creating a DeleteNotification

You configure notifications using Designer.

Be sure to review the sections [“Before Configuring or Managing Notifications” on page 152](#) and [“Considerations when Configuring DeleteNotifications” on page 164](#) before you configure DeleteNotifications.

To create a DeleteNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **DeleteNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.

6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	The name of the JMS connection alias configured on Integration Server. If the connection alias is a Broker Cluster configured with Multisend Guaranteed

Field	Description/Action
	<p>policy, you must add the watt property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true.</p> <p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

The information from the **Permissions** tab appears in the **Properties** panel.

8. Select the **Notification Configure** tab and use the following fields:


Field	Description/Action
Base Name	<p>The base name used to generate the Resource Name created by Adapter for JDBC.</p> <p>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.</p>
Publish Locally	Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.
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.

Field	Description/Action
	<ul style="list-style-type: none"> ■ 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.

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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name cannot contain a period. If the table name does contain a period, Designer will throw an error. <p>Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.</p>
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  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.

Note: When using the Join clause, select only the fields of the monitor table in the **Select** tab.



- 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  icon (or the  icon to fill in all rows of the table) 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:

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. Adapter for JDBC 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 218 .
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.

Field	Description/Action
Query Time Out	<p>Specify the query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

12. 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  or  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. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
14. From the **File** menu, select **Save**.
15. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications”](#) on page 191.

Configuring BasicNotifications

A BasicNotification polls a database table for data using a SQL Select operation. You configure notifications using Designer. For more information about notifications, see [“Adapter Notifications” on page 29](#).

Be sure to review the section [“Before Configuring or Managing Notifications” on page 152](#) before you configure notifications.

To configure a BasicNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **BasicNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.

6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	The name of the JMS connection alias configured on Integration Server. If the connection alias is a Broker Cluster configured with Multisend Guaranteed



Field	Description/Action
	policy, you must add the watt property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true .
	<p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .



The information from the **Permissions** tab appears in the **Properties** panel.

8. 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 <code>t1</code> .
Table Name	Select a table name. The default for the associated catalog name is <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error.
	<p>Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.</p>
Type	The table type displays automatically based on the table you select.

9. If you are not joining tables, skip this step. Select the **Joins** tab to specify the columns for joining the tables you just configured.
- Select the  icon (or the  icon to fill in all rows of the table) 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.
10. 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  icon (or the  icon to fill in all rows of the table) 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:

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. Adapter for JDBC 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 218.
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 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 the notification's publishable document to be published to the local Integration Server, select the **Publish Locally** option.

By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server.

Selecting the **Publish Locally** option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.

- e. If you want to use the Exactly Once notification feature, you must enable the **Exactly Once Notification** option. For more information, see [“Using the Exactly Once Notification Feature” on page 193](#).
- f. Set the **Delete selected records** flag to automatically delete the selected records from the buffer table (based on their **Record ID Column** value) after the notification. Use this option to prevent publishing the same documents to Integration Server each time polling occurs.

Note: Running a BasicNotification may generate a *duplicate message* error. 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 next in the procedure.

- g. 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. For more information, see [“Using the Exactly Once Notification Feature” on page 193](#).
- To ensure that all values will be unique, choose a table column in the **Record ID Column** field whose values are sequential numbers.
- h. Use the **Mark ID Column** field to mark the records as processed. Select the column that you created in the database table to hold the status of the record. Use this option when you do not want to set the **Delete selected records** flag, and also want to avoid the publishing of duplicate records. For information about creating the column that holds the status of the records and marking the records as published, see [“Basic Notifications” on page 36](#).
- i. 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.

- j. Use the **Query Time Out** field to specify the query time out value of the BasicNotification you are configuring. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation.


The default value is -1. Use the default value to have the notification use the value indicated on the `watt.adapter.JDBC.QueryTimeout` property as the time out.

If you specify a value equal to 0, or if the `watt.adapter.JDBC.QueryTimeout` property is not set, the notification executes without a time out.

If you specify a value greater than 0, the notification executes with the specified value as the time out.



Note: -1 is the only permissible negative value for this field.

For more information about the `watt.adapter.JDBC.QueryTimeout` property, see [“Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56](#).

11. Use the **WHERE** tab to specify the WHERE conditions on the SQL query statement for selecting information:
 - a. Select the  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	Type a fixed value in this field. Be sure that it is a valid value, or an exception will be generated at run time.

Note: For Oracle users, if you use a CHAR(n) data type and enter a value in the **Input Field**, Adapter for JDBC automatically sets the ORACLEFIXED_CHAR data type as the JDBC data type.

- c. If necessary, use the  or  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 the WHERE parameters.
12. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
13. From the **File** menu, select **Save**.

14. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications” on page 191](#) for details.

Configuring StoredProcedureNotifications

A StoredProcedureNotification publishes notification data by calling a stored procedure inside of a database. You configure notifications using Designer. For more information about notifications, see [“Adapter Notifications” on page 29](#).

Be sure to review the section [“Before Configuring or Managing Notifications” on page 152](#) before you configure notifications.

For details and important considerations when using a StoredProcedureNotification, see [“Stored Procedure Notifications” on page 40](#).

To configure a StoredProcedureNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **StoredProcedureNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.


6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:



Field	Description/Action
Connection alias name	<p>The name of the JMS connection alias configured on Integration Server.</p> <p>If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true.</p> <p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

The information from the **Permissions** tab appears in the **Properties** panel.

8. Select the **Call** tab to specify which stored procedure to use with the notification. Use the  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 <code>current catalog</code> .
Schema Name	The name of the schema. The default for the schema name is <code>current schema</code> .
Enable Procedure Lookup (Optional)	To type in the Procedure Name , set this field to <code>False</code> . To select the Procedure Name from a list, set this field to <code>True</code> . The default is <code>False</code> . Set this value to <code>False</code> 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.
Publish Locally	Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to

Field	Description/Action
	Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.
JDBC Type	The JDBC type of the corresponding Return Field Name .
Return Field Name	Name of the return field of the stored procedure.
Query Time Out	Specify the query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.
	<p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

9. Use the **Parameter** tab to specify stored procedure parameters. Use the  icon (or the  icon to fill in all rows of the table) to create new parameters for the stored procedure.

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 218.
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. For information about output fields for stored procedures, see “Stored Procedure Notifications” on page 40 .
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 218 .

10. StoredProcedure notifications can support one result set (or one Oracle REF CURSOR). The result set can contain nested cursors.

Note: When using the result set that contains nested cursors, the performance of Adapter for JDBC could degrade. Since the nested cursors are recursively processed, Adapter for JDBC may also return data that may not be required.

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. Note: When using for Oracle database, this field is not required.
Result Set Name	Type the name of the result set you want to create. For information about result sets, see “Stored Procedure Notifications” on page 40 . Note: When using for Oracle database, this field is not required.
Result Set Name (from second row)	Select a valid result set name.
Column Name	Name of column of the result set.

Field	Description/Action
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 218 .

11. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
12. From the **File** menu, select **Save**.
13. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications” on page 191](#).

Configuring StoredProcedureNotificationWithSignature

A `StoredProcedureNotificationWithSignature` publishes notification data by calling a stored procedure inside of a database. Unlike a `StoredProcedureNotification`, the `StoredProcedureNotificationWithSignature` enables you to automatically obtain a stored procedure's parameters by introspecting and listing the signature of the stored procedure at the time you configure this notification. This means that you do not need to look up and retype these parameters manually.

You configure Adapter for JDBC notifications using Designer. For more information about adapter notifications, including what you need to know before you configure and manage them, see the [“Before Configuring or Managing Notifications” on page 152](#).

To configure a `StoredProcedureNotificationWithSignature`

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **StoredProcedureNotificationWithSignature** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.

The name of the publishable document associated with this notification is displayed.

6. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer's guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	<p>The name of the JMS connection alias configured on Integration Server.</p> <p>If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the <code>watt</code> property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p> </div>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

The information from the **Permissions** tab appears in the **Properties** panel.

8. Select the **Call** tab to specify the stored procedure to call. Provide values for the following call parameters:

Field	Description/Action
Catalog Name	The name of the catalog. The default for the catalog name is <code>current catalog</code> .
Schema Name	The name of the schema. The default for the schema name is <code>current schema</code> .

Field	Description/Action
Procedure Name Pattern	To save time, you can type all or part of the procedure name in this field to narrow your search. This is helpful when dealing with a large database that has a long list of procedures. Use % as a multi-character wildcard and _ (underscore) as a single character wildcard. Note: With DB2 databases, functions do not appear in the list of procedure names. Only procedures appear in the list.
Procedure Name	Type or select the stored procedure name, depending on how you set the Procedure Name Pattern field. If you select <All Procedures> , this field lists all of the procedures in the selected catalog and schema.
Publish Locally	Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.

9. Based on the signature for the stored procedure you specified, the following fields and values will appear in the top table on the **Call** tab:

Field	Description/Action
Parameter Name	The stored procedure parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the stored procedure parameter.
Parameter Type	Defines the parameter type as IN, INOUT, or OUT. If you select IN or INOUT, you may also set the input expression in the Expression field.
Expression*	Sets the input for the IN or INOUT parameter types only. The RETURN or OUT parameters will appear automatically on the Call tab's bottom table. For a list of the allowable expression

Field	Description/Action
	settings by parameter type, and how each parameter will map to the input or output fields, see the following table.

The following table shows valid expressions by parameter type:

Parameter Type	Expression	Input or Output Mapping?
RETURN	Empty (default)	Output field
OUT	Empty (default)	Output field
IN	Default is empty, but it needs to be updated with a fixed value.	No mapping
INOUT	Empty (default)	Output field
ORACLE CURSOR (INOUT)	Empty (default)	Output field. Set the parameters in the ResultSet tab as described later in the procedure.


10. The bottom table on the **Call** tab lists the following output parameters and values for the stored procedure that will map to the output of the notification's publishable document and returns the value of OUT or INOUT parameters:

Field	Description/Action
Output Parameter Name	The stored procedure output parameter name.
SQL Type	The SQL data type of the database column.
JDBC Type	The JDBC data type of the output parameter.
Output Field	The name of the output parameter.
Output Field Type	The output parameter Java data type. For a list of JDBC to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

11. Specify the query time out value of the `StoredProcedureNotificationWithSignature` notification you are configuring in the following field:

Field	Description/Action
Query Time Out	<p>The query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.</p> <p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

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

This type of notification can support multiple results sets. Use the  icon to create additional result sets as needed.

Note: While all the tables in the **Call** tab are updated automatically if the selected stored procedure changes, the **ResultSet** tab information is not updated automatically. To update this information, you must manually update the fields in the **ResultSet** tab.

Provide values for the following parameters:

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.

Field	Description/Action
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 data type of the result column.
Output Type	The Java data type of the result column. For a list of JDBC to Java data type mappings, see “JDBC Data Type to Java Data Type Mappings” on page 218.

13. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
14. From the **File** menu, select **Save**.
15. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications”](#) on page 191.

Configuring OrderedNotifications

An OrderedNotification publishes notification data for multiple insert, update, or delete operations on multiple tables. For more information about notifications, see [“Adapter Notifications”](#) on page 29.

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 set the **Processing Mode** option to Serial on the **Settings** tab in Designer when you create the trigger and flow service. For more information about configuring Integration Server triggers and flow services, see the Designer Service Development online help for your release. See [“About this Guide”](#) for specific document titles.

Considerations when Configuring OrderedNotifications

You must consider the following restrictions when configuring an OrderedNotification:

- OrderedNotifications cannot be used with a Teradata database (any version).
- When using an OrderedNotification with a MySQL database, updating any database field with the same value will not invoke a trigger.

- MySQL version 5.0.x does not support multiple triggers with the same event (insert, delete, or update) for one table. Hence, with a MySQL database, before you can enable an OrderedNotification for a table, you must disable any OrderedNotification or other adapter notifications of the same operation type monitoring the table.

Creating an OrderedNotification

You configure notifications using Designer.

Be sure to review the sections [“Before Configuring or Managing Notifications” on page 152](#) and [“Considerations when Configuring OrderedNotifications” on page 184](#) before you configure an OrderedNotification.

To create an OrderedNotification

1. In Designer, right-click the package in which the notification should be contained and select **New > Adapter Notification**.
2. Select the parent namespace, type a name for the adapter notification, and click **Next**.
3. Select **Adapter for JDBC** as the adapter type and click **Next**.
4. Select the **OrderedNotificaton** template and click **Next**.
5. Select the appropriate **Adapter Connection Name** and click **Next**.
6. 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 29](#). For more details about the Integration Server publishable documents, see the Integration Server publish-subscribe developer’s guide for your release. See [“About this Guide”](#) for specific document titles.

7. Designer creates the notification, and the editor for the adapter notification appears.
 - a. 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.
 - b. In the **Publish Document** section, you can specify how you want the notification document to be published:
 - To publish documents to Broker, select **webMethods Broker/Local**. This is the default option.
 - To publish documents to a JMS provider, select **JMS Provider**, and provide values for the following input fields:

Field	Description/Action
Connection alias name	The name of the JMS connection alias configured on Integration Server.

Field	Description/Action
	<p>If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property <code>watt.art.notification.jmsSend.usePublicService</code> and set it to true.</p> <p>Note: Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.</p>
Destination name	The name of the destination from which you want the JMS trigger to receive messages.
Destination type	Whether the destination is a Queue (default) or a Topic .

The information from the **Permissions** tab appears in the **Properties** panel.

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

Field	Description/Action
Base Name	<p>The base name used to generate the Resource Name created by Adapter for JDBC.</p> <p>Note: For OS/390 DB2V7.2, the Base Name you create below must be no more than five characters because triggers on OS/390 name cannot be more than eight characters.</p>
Publish Locally	Specifies whether to publish the notification's publishable document to the local Integration Server. By default, this option is not selected, that is, if the Broker is configured to Integration Server, the publishable document is published to the Broker; otherwise the publishable document is published to the local Integration Server. Selecting the Publish Locally option reduces performance problems, if Integration Server is connecting to a remotely located Broker that is in turn triggering a service on the local Integration Server.
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.



Field	Description/Action
	<ul style="list-style-type: none"> ■ 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.

9. 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 <code>current catalog</code> . The default for the associated schema name is <code>current schema</code> . The table name must not contain a period. If the table name does contain a period, Designer will throw an error. <p>Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.</p>
Type	The table type displays automatically based on the table you select.
Operation Type	Select INSERT, UPDATE, or DELETE operation.

Field	Description/Action
Operation ID	Assign an ID to uniquely identify the given operation for the notification.

10. Use the **SELECT** tab to define the columns and fields to be selected using the following fields:
- 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`.
 - Select the  icon (or the  icon to fill in all rows of the table) 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.

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. Adapter for JDBC 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 218 .
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:

Field	Description/Action
	<ul style="list-style-type: none"> ■ Yes if you want notification if this column of data has been updated. Yes is the default value. ■ No if 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.
Output Value Type	<p>Used for Update operations only. Specifies which output value to retrieve from the database table. By default, the notification retrieves the new value from the database table. Select either of the following output value types:</p> <ul style="list-style-type: none"> ■ Old: Retrieves the old value from the database table ■ New: Retrieves the new value from the database table <p>To retrieve both the old and the new values, you need to create two rows and then select an Output Value Type as Old in one row and an Output Value Type as New in the other row. While doing so, ensure that the Output Field is unique for both the old and the new values.</p> <p>Note: For Sybase and Microsoft 2000/2005, you cannot retrieve both the old and the new values in the same notification. The notification must retrieve either the old or the new value.</p>
Maximum Row	<p>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.</p>
Query Time Out	<p>Specify the query time out value in seconds. This value is the amount of time Adapter for JDBC waits for the notification to execute before stopping the SQL operation. The default value is -1. Use the default value to have the notification use the value indicated on the <code>watt.adapter.JDBC.QueryTimeout</code> property as the time out. If you specify a value equal to 0, or if the <code>watt.adapter.JDBC.QueryTimeout</code> property is not set, the notification executes without a time out. If you specify a value greater than 0, the notification executes with the specified value as the time out.</p>

Field	Description/Action
	<p>Note: -1 is the only permissible negative value for this field.</p> <p>For more information about the <code>watt.adapter.JDBC.QueryTimeout</code> property, see “Forcing a Timeout During Long-Running SQL Operations in Services and Notifications” on page 56.</p>

11. 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  icon to define new WHEN clause fields.
 - b. Select the column in the **Expression** field. The corresponding **Operation ID** is displayed.
 - 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  or  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.
12. The information about using the **Permissions** tab to assign an access control list (ACL) to an element appears in the Properties panel.
13. From the **File** menu, select **Save**.
14. You must schedule and enable the notification using Integration Server Administrator before you can use it. For details, see [“Managing Polling Notifications” on page 191.](#)

Note: While enabling the Ordered Notification that monitors a Sybase database, if you encounter an error that indicates data definition commands are not allowed in transactions, then you need to run the following command in your Sybase database: `sp_dboption database_name, "ddl in tran", true.`

Managing Polling Notifications

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


Note: You must have Integration Server Administrator privileges to access Adapter for JDBC's administrative screens. For information about setting user privileges, see the Integration Server administration guide for your release. See "About this Guide" for specific document titles.


To manage polling notifications

1. Start Integration Server Administrator.
2. From the **Adapters** menu in the navigation area of Integration Server Administrator, select **Adapter for JDBC**.
3. From the navigation area, select **Polling Notifications**.
4. From the **Adapter for JDBC 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. If the table name exceeds 10 characters, 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.
State	After you schedule a polling notification, you can use this option's dropdown list to set the polling notification's state: <div data-bbox="560 1633 1372 1774" style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: You must schedule a polling notification before you can enable it. To schedule a polling notification, use the  icon described in these procedures.</p> </div> <ul style="list-style-type: none"> ■ Enabled: The polling notification performs as scheduled.

Field	Description/Action
	<ul style="list-style-type: none"> ■ 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. <p>The Suspend all enabled and Enable 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 53. If there is no polling notification scheduled for a given adapter notification, control for this field is disabled. Use the  icon to schedule a polling notification.</p>


Edit Schedule

Click the  icon to create or modify polling notification parameters.

Note: You must disable a polling notification before you can edit it.

View Schedule

Click on the **View Schedule** icon to review the parameters for the selected polling notification. Click **Return to Adapter for JDBC Notifications** to go back to the main polling notification page.

- To create or modify schedule parameters for the selected adapter notification, click on the  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 Integration Server to fail.
Immediate	Enable this option to start polling immediately.

- Click **Save Schedule**.

7. After you create a polling notification, you can enable it. Use the **State** field 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, 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 Integration Server. For details, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

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.

After you export or deploy an adapter notification, Software AG recommends reloading the adapter values in the notification template in Designer if all of the following conditions are met:

1. The connection for the notification in the source and target Integration Server connects to the same database.
2. The connection for the notifications is configured with different schemas on the source and target Integration Server.
3. The two schemas have access to each other’s database objects.

Software AG recommends using Basic Notifications in critical environments.

With Insert Notifications, Update Notifications, Delete Notifications, and Ordered 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, Delete Notifications, and Ordered 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.

For more details, see [“Insert Notifications, Update Notifications, and Delete Notifications” on page 32](#).

When exporting the configured notifications, Software AG recommends that you export them in a Suspended state. The trigger and buffer table are not dropped in the Suspended state.

After exporting the configured notifications, the following scenarios can occur in Adapter for JDBC if the notifications are not exported in a Suspended state:

- While reloading the package containing exported enabled notifications, or while restarting Integration Server, if some or all of the database objects of a notification are missing, the adapter throws a warning indicating that the database objects are missing, and then disables that notification.
- When enabling an exported disabled notification, if some or all of the database objects for that notification exist, the adapter throws an error indicating that the database objects exist.
- When disabling an exported notification, if some of the database objects are missing, the adapter disables the notification but does not delete the existing database objects. The next time you enable the same notification, the adapter throws an error indicating that incomplete database objects exist.

To help you recover from these scenarios, Adapter for JDBC provides the following services:

- `pub.pollingNotificationUtils:getDatabaseObjectsForNotification` to list the existing database objects of a notification.
- `pub.pollingNotificationUtils:dropDatabaseObjects` to delete the existing database objects of a notification.

For more information about these services, see [`pub.pollingNotificationUtils:getDatabaseObjectsForNotification`](#) and [`pub.pollingNotificationUtils:dropDatabaseObjects`](#).

For information about polling notifications and their states, see [“Polling Notifications and States” on page 53](#).

For more information about exporting packages, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

Viewing Notifications

You use Designer to view notifications.

To view a notification

1. In Designer, expand the package and folder that contain the notification you want to view.
2. Double-click the notification you want to view.

Designer displays the notification in the notification template's Adapter Notification Editor.

Editing Notifications

You use Designer to edit notifications. You may be able to change the connection associated with an adapter using 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 26](#).

To edit a notification

1. In Designer, expand the package and folder that contain the notification you want to edit.
2. Select the notification you want to edit.

Designer displays the notification in the notification template's Adapter Notification Editor.

3. Do one of the following:
 - If you have the VCS Integration feature enabled, right-click the notification and select **Check Out**.
 - If you do not have the VCS Integration feature enabled, right-click the notification and select **Lock for Edit**.
 - If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check out the notification. The options available in the **Team** menu depend on the VCS client that you use.

4. Modify the values for the 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.
5. After you have completed your modifications, save the notification and do one of the following:
 - If you have the VCS Integration feature enabled, right-click the notification and select **Check In**. Enter a check-in comment and click **OK**.
 - If you do not have the VCS Integration feature enabled, right-click the notification and select **Unlock**.
 - If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check in the notification. The options available in the **Team** menu depend on the VCS client that you use.

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

Deleting Notifications

You use Designer to delete adapter notifications.

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 191](#).

To delete a notification

1. In Designer, expand the package and folder that contain the notification you want to delete.
2. Right-click the notification and click **Delete**.

Validating Adapter Notification Values

Designer enables Adapter for JDBC to validate user-defined data for adapter notifications at design time. You can validate the values for a single notification or you can configure Designer to always validate the values for notifications. Both options could potentially slow your design-time operations.

When you enable data validation for a single adapter notification, Designer compares the notification values against the resource data that has already been fetched from the selected adapter.

If you select the option to always validate values for adapter notifications, it will do so for all webMethods WmART-based adapters installed on Integration Server.

For more information about the Adapter Service/Notification Editor, other Designer menu options, and toolbar icons, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.


Validate Data for a Single Adapter Notification

Perform the following procedure to validate data for a single adapter notification.

To validate data for a single adapter notification

1. In Designer, expand the package and folder that contain the notification for which you want to enable automatic validation.
2. Double-click the notification for which you want to validate the data.

Designer displays the configured adapter notification in the service template's Adapter Notification Editor.

3. Click the  icon.

Validating Data for All Adapter Notifications

Perform the following procedure to enable Designer to always validate data for all adapter notifications.

To enable automatic data validation for all values in adapter notifications

1. In Designer, select the **Window > Preferences > Software AG > Service Development > Adapter Service/Notification Editor** item.
2. Enable the **Automatic data validation** option.
3. Click **OK**.

Reloading Adapter Values

Designer enables Adapter for JDBC to reload and validate user-defined data for notifications at design time. You can reload values for a single notification or you can configure Designer to automatically reload the values for adapter notifications. Both options could potentially slow your design-time operations.

When you reload adapter values for a single adapter notification, Designer compares the notification values against the resource data that has already been fetched from the selected adapter.

If you select the option to always reload values for adapter notifications, it will do so for all webMethods WmART-based adapters installed on Integration Server.

For more information about the Adapter Service/Notification Editor, other menu options, and toolbar icons, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.


Reloading the Values for a Single Adapter Notification

Perform the following procedure to reload the adapter values for a single adapter notification.

To reload the adapter values for a single adapter notification

1. In Designer, expand the package and folder that contain the service for which you want to enable automatic validation.
2. Double-click the service for which you want to validate the data.

Designer displays the configured adapter service in the service template's Adapter Notification Editor.

3. Click the  icon.

Reloading the Values for All Adapter Notifications

Perform the following procedure to reload the adapter values for all adapter notifications.

To automatically reload the values for all adapter notifications

1. In Designer, select the **Window > Preferences > Software AG > Service Development > Adapter Service/Notification Editor** item.
2. Enable the **Automatic polling of adapter metadata** option.
3. Click **OK**.

7 Data Type Configuration

■ Overview	200
■ The Default Data Type Mapping File	200
■ The Database-Specific Data Type Mapping Files	201
■ Support for SQLXML Data Type	202

Overview

webMethods Adapter for JDBC allows you to dynamically configure the data type mapping for a particular database in an XML configuration file. The adapter provides the following data type mapping files in the *Integration Server_directory*\instances*instance_name* \packages\WmJDBC Adapter\config directory:

- a default data type mapping configuration file named `TypeMapping.xml`
- an XML schema file named `TypeMapping.xsd`
- a database-specific data type mapping configuration file for each supported backend, for example `OracleTypeMapping.xml`

Each column in a database table is assigned a SQL type. The JDBC driver then maps each SQL data type to a JDBC data type. The adapter in turn maps each JDBC data type to one or more Java data types that are used as the input or output of adapter services and notifications.

For more information on JDBC data type to Java data type mappings, see on page 217 “Data Type Mapping”.

Based on the XML configuration, Adapter for JDBC selects the exact JDBC type applicable for a particular table column in the database. The adapter service or notification template shows the corresponding Java data types that are mapped to the JDBC data type in the XML configuration file.

In rare cases, you might need to modify the database-specific file if you want to have custom processing of a data type. If validation of the modified file fails, Adapter for JDBC ignores the file, and the adapter connections to the particular backend do not work.

The native SQL data types returned by the JDBC driver that are not part of the predefined JDBC data types must be configured as extended data types in the database-specific configuration file.

The Default Data Type Mapping File

The default `TypeMapping.xml` file contains the `javaType` entries with their setter and getter methods, along with the default `jdbcType` entries that contain the `javaType` mappings.

javaType

Contains:

- the fully qualified class name that is used while accessing the Java object corresponding to a database column

- the setter and getter methods in the prepared statement and result set while accessing a specific column in the database

typeMapping

Contains the Java type mappings for the default JDBC types supported in the `java.sql.Types` class. A single JDBC type can be mapped to one or more Java types. Those Java types appear as drop-down lists in the Adapter for JDBC service and notification templates in Software AG Designer.

If a database requires special handling for a JDBC type and needs to have different `javaType` mapping from the one provided in the default `TypeMapping.xml` file, you can modify the database-specific configuration file accordingly overriding the default configuration.

The Database-Specific Data Type Mapping Files

A database-specific data type mapping configuration file can contain the following elements depending on whether some JDBC types need special handling.

extTypeMapping

Includes additional JDBC types that are not supported in the `java.sql.Types` class, as well as the corresponding code of the JDBC type.

typeClassName

Contains the fully qualified class name that extends the `com.wm.adapter.wmjdbconfig.JDBCType` class when a JDBC type requires special handling.

For example, the new class names for the Oracle BLOB and CLOB data types are included in the `typeClassName` attribute of the BLOB and CLOB JDBC types in the Oracle database configuration file because the Oracle BLOB and CLOB data types require special handling.

columnTypeMapping

Contains the mapping between the native SQL type of a column in a particular database (in the `columnType` element) and the corresponding JDBC type.

In addition, the following attributes provide special handling of a particular column type.

alternativeName

Specifies the value returned by the JDBC driver when a native SQL column type in a database has a different name from the name returned by the driver.

For example, a column of double precision type in a Sybase database is returned as double precis by the driver. In this case, the alternativeName attribute is set to `double precis` in the Sybase type mapping configuration file so that the adapter can recognize it as the double precision type.

includeColumnSize

Specifies whether a column type name is suffixed with the column size when creating an adapter service or notification. When the attribute is set to `false`, the column size is not included. The default value is `true`.

resizeFactor

Indicates that the size of a column is different from the size returned by the JDBC driver. The default value is 1, that is, the column size is the same as the size returned by the JDBC driver.

includePrecision

Specifies whether to include column precision for a Decimal or Numeric JDBC type. If the attribute is set to `true`, precision is included. The default value is `false`.

Support for SQLXML Data Type

Adapter for JDBC provides support for the SQLXML data type for the following databases:

- DB2 for AS/400
- DB2 for OS/390
- Microsoft SQL Server
- Oracle

To enable SQLXML data type support for Oracle, you must download the `xdb6.jar` and `xmlparserv2.jar` files from the Oracle website and copy them to the `Integration Server_directory\instances\instance_name\packages\WmJDBCAdapter\code\jars\static` directory.

Note: After installing a new fix, reapply the changes made to the configuration files manually.

8 Logging and Exception Handling

■ Overview	204
■ Adapter for JDBC Logging Levels	204
■ Adapter for JDBC Message Logging	205
■ Adapter for JDBC Exception Handling	206
■ Customizing the Adapter's List of Fatal Error Codes	207
■ Overriding the Adapter's List of Fatal Error Codes	208
■ Suppressing the Logging of Errors	209
■ Adapter for JDBC Error Codes	210

Overview

The following sections describe Adapter for JDBC message logging, exception handling, and customizing and overriding the 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 261.

Adapter for JDBC Logging Levels

Adapter for JDBC uses the Integration Server logging mechanism to log messages. You can configure and view the Integration Server logs to monitor and troubleshoot Adapter for JDBC. For detailed information about logging in Integration Server, including instructions for configuring and viewing the different kinds of logs supported by the server, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

Accessing Adapter Logging Levels

With Integration Server, you can configure different logging levels for Adapter for JDBC.

To access the adapter's logging information

1. From the Integration Server Administrator screen, select **Settings > Logging**.
The **Logging Settings** screen appears. The **Loggers** section has **Adapters** included in the **Facility** section.
2. Expand the **Adapters** tree to see a list of all installed adapters with their code number and adapter description, along with the logging level.

Changing Logging Levels

You use Integration Server to change the logging levels.

To change logging settings for the adapter

1. Click **Edit Logging Settings**. Select the required **Level of Logging** for Adapter for JDBC.
2. After making your changes, click **Save Changes**.
3. For complete information about specifying the amount and type of information to include in the log, see the webMethods logging guide for your release. See “About this Guide” for specific document titles.

Adapter for JDBC Message Logging

Integration Server maintains several types of logs; however, Adapter for JDBC only logs messages to the audit, error and server logs. Because Adapter for JDBC works in conjunction with the WmART package, the adapter's messages and exceptions typically appear within log messages for the WmART package.

The logging levels for Adapter for JDBC are given in the following table.

Integration Server	Log	Description
Integration Server 9.0 and higher	Audit Log	You can monitor individual adapter services using the audit log as you would audit any service in Integration Server. The audit properties for an adapter service are available in each Adapter for JDBC service template on the Audit tab.
	Error Log	Adapter for JDBC automatically posts fatal-level and error-level log messages to the server's error log. These log messages will appear as adapter run-time messages.
	Server Log	Adapter for JDBC posts messages to the server log, depending on how the server log is configured. Fatal-level through debug-level log messages appear as adapter run-time log messages. Trace-level log messages appear as Adapter for JDBC log messages.

Adapter for JDBC's log messages appear in either of the following formats:

- ADA.1.nnnnc
- ADA.0001.nnnnc

where the facility code ADA indicates that the message is from an adapter, 0001 or 1 indicate that it is Adapter for JDBC, *nnnn* represents the error's minor code, and (optionally) *c* represents the message's severity level. For detailed descriptions of Adapter for JDBC's minor codes, see [“Adapter for JDBC Error Codes” on page 210](#).

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

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

Adapter for JDBC Exception Handling

Adapter for JDBC issues 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

Adapter for JDBC 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 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

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

In this case, WmART resets the entire connection pool. WmART then wraps the exception in `com.wm.pkg.art.error.DetailedSystemException` and throws it to 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`. `SQLException`s include a `SQL STATE`, a `SQLCODE`, and an error message.

Adapter for JDBC 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 Adapter for JDBC to automatically refresh its 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 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 either of the following `watt` parameters:
 - `watt.adapter.JDBC.database driver .fatalErrors=+ErrorCode_1, ErrorCode_2, ErrorCode_n`

This `watt` property is database driver specific. Note that there is no space after the `,` (comma).

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

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

Note: If the error code of a database starts with a zero, then remove the zero from the error code and append the remaining code to the list. For example, for Oracle database, if you want to add the error code, 01401, then type the `watt` parameter as follows:

```
watt.adapter.JDBC.Oracle.fatalErrors=+1401
```

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

Driver	Setting
Microsoft SQL Server	<code>watt.adapter.JDBC.MsMssql.fatalErrors</code>
Oracle JDBC	<code>watt.adapter.JDBC.Oracle.fatalErrors</code>
JTOpen	<code>watt.adapter.JDBC.DB2JTOPEN.fatalErrors</code>
DataDirect Connect for JDBC driver for DB2	<code>watt.adapter.JDBC.CJDBCDB2.fatalErrors</code>
Teradata Type 4	<code>watt.adapter.JDBC.TeraData.fatalErrors</code>
JDBC 2.21 type 4 for Informix	<code>watt.adapter.JDBC.INFORMIX.fatalErrors</code>
jCONNECT 5.5 and 6.05 type 4 for Sybase	<code>watt.adapter.JDBC.SYBASE.fatalErrors</code>
DB2 Universal type 2 and type 4	<code>watt.adapter.JDBC.DB2UNIVERSAL.fatalErrors</code>
Other driver types	<code>watt.adapter.JDBC.Generic.fatalErrors</code>

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

This `watt` property is not database driver specific, instead applies to all drivers. Note that there is no space after the , (comma).

Example: To allow Adapter for JDBC to refresh connections when encountering error codes 12535, type:

```
watt.adapter.JDBC.database.fatalErrors=+12535
```

It is recommended that you use this `watt` property to add error codes to the adapter's list of error codes, instead of `watt.adapter.JDBC.database.driver.fatalErrors`.

4. Click **Save Changes**.
5. Restart Integration Server.

Overriding the Adapter's List of Fatal Error Codes

You can override the existing list of fatal error codes of Adapter for JDBC with a new list of error codes.

To override the existing fatal error list with a new fatal error list

1. Start 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
```

For example, to override the list of fatal error codes for Adapter for JDBC with a list of error codes 17002, 17003, and 16702 for Oracle driver, type the following:

```
watt.adapter.JDBC.Oracle.fatalErrors=17002,17003,16702
```

Note that there is no space after the , (comma). For a list of other supported driver settings, see [“Customizing the Adapter's List of Fatal Error Codes” on page 207](#).

Note: If the error code of a database starts with a zero, then remove the zero from the error code and append the remaining code to the list. For example, for Oracle database, if you want to override the error code, 01401, then type the watt parameter as follows:

```
watt.adapter.JDBC.Oracle.fatalErrors=1401
```

By default, Adapter for JDBC's fatal error codes are 17001, 17002, 17416, 1092, 28, 1012, 17410, 12571, 3114, 1089, 1033, 08S01, 40003, -30081, -99999, HY000, -601, JZ0C0, -79716. This list is a combination of common fatal error codes from different databases.

4. Click **Save Changes**.
5. Restart Integration Server.

Suppressing the Logging of Errors

When a database is down or is unreachable, errors are logged repeatedly causing the Server and Error logs to overflow. You can avoid this by suppressing the error after which only the first five consecutive occurrences of the error are logged and then the error is suppressed. The first five occurrences of the error are logged to avoid any potential dangers that may result due to the suppressing of the error. However, it is recommended to use this option of suppressing the logging of errors, only when you know about an activity that can lead to the continuous logging of errors, for example, a scheduled database shut down, and you want to suppress only those errors.

To suppress the logging of errors

1. Start 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.SuppressErrorCodes=ErrorCode_1, ErrorCode_2, ErrorCode_n, Error_Description
```

where *ErrorCode_1*, *ErrorCode_2*, and *ErrorCode_n* are the error codes of the errors that you want to suppress, and *Error_Description* is the description of the error that you want to suppress. For databases that do not have error codes, you can provide the error description.

Example: To allow Adapter for JDBC to suppress the error codes ORA-12500, ORA-01033, 17002, and the error with description `User account is locked`, type the following:

```
watt.adapter.JDBC.SuppressErrorCodes=ORA-12500,ORA-01033,17002,User account is locked
```

Note that there is no space after the , (comma).

4. Click **Save Changes**.
5. Reload the WmJDBCAdapter package.

Adapter for JDBC Error Codes

The following table lists Adapter for JDBC'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.

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

Response: Check the spelling and make sure the JDBC driver file is in the CLASSPATH or in the *packages_directory* /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: For supported drivers and their settings, see [“Using JDBC Drivers to Connect to Databases” on page 19](#).

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: For supported drivers and their settings, see [“Using JDBC Drivers to Connect to Databases” on page 19](#).

204 Cannot connect to the database with DataSource class *ClassName*.

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

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

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: For supported drivers and DataSource class names, see your Adapter for JDBC documentation.

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*. The connection between the adapter and database cannot be closed.

Cause:

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*. I/O error occurred when trying to read byte sequence data

Cause:

Response: Call Software AG Global Support 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*. SQL statements failed to execute.

Cause:

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: For complete instructions for configuring notifications, see [“Adapter Notifications” on page 151](#).

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 [“JDBC Data Type to Java Data Type Mappings” on page 218](#).

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*. The AS/400 environment may not be correct.

Cause:

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.

A Data Type Mapping

■ JDBC Data Type to Java Data Type Mappings	218
■ SQL Data Type to JDBC Data Type Mappings	224
■ Advanced Server Type to JDBC Data Type Mappings	224

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. Adapter for JDBC 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.

For a list of data types for which Integration Server has some constraints, see [“JDBC Data Type to Java Data Type Mapping Constraints” on page 223](#).

- Note:** Adapter for JDBC does not support the DATALINK DB2 data type when using the adapter with DB2 for AS/400 or DB2 for OS/390.
- Note:** Adapter for JDBC does not support the TIMESTAMP WITH TIME ZONE and TIMESTAMP WITH LOCAL TIME ZONE data types in Oracle 10g.
- Note:** Adapter for JDBC does not support user-defined data types, Oracle PL/SQL collections, or Oracle PL/SQL records.

JDBC Data Type	Java Data Type
ARRAY	java.sql.Array java.lang.Object
BIT	java.lang.Boolean java.lang.String java.lang.Object
TINYINT	java.lang.Byte java.lang.Integer java.lang.String java.lang.Object SetAsString
SMALLINT	java.lang.Short java.lang.Integer

JDBC Data Type	Java Data Type
	java.lang.String java.lang.Object
INTEGER	java.lang.Integer java.lang.String java.lang.Object
BIGINT	java.lang.Long java.lang.String java.lang.Object
FLOAT	java.lang.Double java.lang.String java.lang.Object java.math.BigDecimal SetAsString
REAL	java.lang.Float java.lang.String java.lang.Object java.math.BigDecimal
BOOLEAN	java.lang.Boolean java.lang.String java.lang.Object
DOUBLE	java.lang.Double java.lang.String java.lang.Object java.math.BigDecimal SetAsString
NUMERIC	java.math.BigDecimal

JDBC Data Type	Java Data Type
	java.lang.String java.lang.Object
DECIMAL	java.math.BigDecimal java.lang.String java.lang.Object
CHAR	java.lang.String java.lang.Character java.lang.Object
VARCHAR	java.lang.String java.lang.Object
LONGVARCHAR	java.lang.String java.lang.Object
DATE	java.sql.Date java.util.Date java.lang.String java.lang.Object *SetAsString
TIME	java.sql.Time java.util.Date java.lang.String java.lang.Object SetAsString
TIMESTAMP	java.sql.Timestamp java.util.Date java.lang.String java.lang.Object

JDBC Data Type	Java Data Type
	SetAsString
TIMESTAMP WITH TIME ZONE	
TIMESTAMP WITH LOCAL TIME ZONE	
BINARY	byte array (byte []) java.lang.Object
VARBINARY	byte array (byte[]) java.lang.Object
LONGVARBINARY	byte array (byte[]) java.lang.Object
LONGNVARCHAR	java.lang.String java.lang.Object
NCHAR	java.lang.String java.lang.Object
NULL	java.lang.String java.lang.Object
NVARCHAR	java.lang.String java.lang.Object
CLOB	java.sql.Clob java.lang.String java.io.Reader java.lang.Object
BLOB	java.sql.Blob byte array java.io.InputStream

JDBC Data Type	Java Data Type
	java.lang.Object
ORACLECURSOR	java.lang.Object
ORACLEFIXED_CHAR	java.lang.String
STRUCT	java.sql.Struct java.lang.Object
OTHER	java.lang.Object java.lang.String java.sql.Struct java.sql.Array

Important Considerations When Using BLOB and CLOB Data Types

- When passing large CLOB or BLOB data, use the Java data types `java.io.Reader` for CLOB and `java.io.InputStream` for BLOB to prevent Integration Server from running out of memory. When using these data types, Adapter for JDBC streams the data into bytes thus allowing to pass large data. The data types `java.io.Reader` and `java.io.InputStream` are supported only for the Oracle database using the Oracle driver.
- When using the CLOB data with `java.io.Reader` as input data type, it is recommended that you use the `InputStreamReader` implementation of `java.io.Reader` with the correct encoding parameter.
- When Designer executes a Adapter for JDBC SELECT service that has its output type set to `java.sql.Blob` for a BLOB data type, Designer issues a `java.io.NotSerializableException` error. To work around this issue, perform one of the following:
 - Use another valid Output Type for BLOB data types.
 - Execute the service by navigating through Integration Server Administrator instead of saving or viewing the BLOB data through Designer.

Important Considerations When Using the Array and Struct Database Specific Data Types

- In an adapter service, when using the `java.lang.Object` as the output field type for a database column of type array or struct, Adapter for JDBC returns the data as a

java.lang.Object array, provided that the array or struct data in the database table is composed of primitive data types.

- When using the java.sql.Array or java.sql.Struct as the output field type for a database column of type array or struct, Adapter for JDBC returns the java.sql.Array and the java.sql.Struct objects, respectively, as returned by the driver. However, when serializing the data across the JVMs, this returned data may not be serializable and may result into a java.io.NotSerializableException. Therefore, before serializing the data across the JVMs, it is important that you use a Java or a flow service to process the java.sql.Struct and java.sql.Array objects as required, and then drop them from the pipeline.

Note: The java.sql.Struct and java.sql.Array data types are available only for Adapter for JDBC services.

Using the SetAsString Data Type in Adapter for JDBC

The SetAsString data type is a dummy string data type. When using this data type, Adapter for JDBC does not try to convert the input data into the equivalent JDBC data type, but passes the data to the underlying database driver as a string data type. Thus, you have the flexibility to specify the format of the equivalent JDBC data type by using a database specific function.

For example, you can specify the format for date, time, or timestamp using the to_date function or a similar database function for Oracle database. Adapter for JDBC treats the input data as a string data type and does not convert it to the equivalent JDBC data type. The to_date function then uses the string data to provide the required format of the date, time or timestamp.

If your database has native database specific functions that can convert string data type to any other data type, you may use the SetAsString data type.

Note: The SetAsString data type is available only for Adapter for JDBC services.

JDBC Data Type to Java Data Type Mapping Constraints

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 Designer:

- 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.

Advanced Server Type to JDBC Data Type Mappings

Adapter for JDBC supports only the basic data types supported by the Postgres Plus Advanced Server 9.0 JDBC connector and listed in the following table:

JDBC Data Type	Advanced Server Type
INTEGER	INT4
TINYINT, SMALLINT	INT2
BIGINT	INT8
REAL, FLOAT	FLOAT4
DOUBLE	FLOAT8
DECIMAL, NUMERIC	NUMERIC

JDBC Data Type	Advanced Server Type
CHAR	BPCHAR
VARCHAR, LONGVARCHAR	VARCHAR
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
BINARY	BYTEA
BIT	BOOL

Advanced Server Type Constraints

When creating tables with the NUMERIC, VARCHAR, or BYTEA data types, you must specify a length for the table.

B Built-In Services

■ Overview	228
■ pub.jdbcAdapter:updateNotificationSchema	228
■ pub.jdbcAdapter:updateServiceSchema	233
■ pub.jdbcAdapter:updateConnectionPassword	237
■ pub.jdbcAdapter:createConnectionNodes	237
■ pub.pollingNotificationUtils:dropDatabaseObjects	240
■ pub.pollingNotificationUtils:getDatabaseObjectsForNotification	240

Overview

This appendix provides information on the built-in services provided by webMethods Adapter for JDBC. These services are located in the WmJDBCAdapter package.

pub.jdbcAdapter:updateNotificationSchema

The `pub.jdbcAdapter:updateNotificationSchema` Java service changes the schema names configured in the following types of notifications: `InsertNotification`, `DeleteNotification`, `UpdateNotification`, `OrderedNotification`, and `BasicNotification`.

This service always validates the new schemas against the notification settings, and sets the state of the notification to the state it is in before it is updated, regardless of whether the update is successful.

The service does the following:

Step	Description
1	The service checks the current status of the notification.
2	If the notification is enabled, the service checks the <i>forceDisable</i> parameter. If <i>forceDisable</i> is false, the service reports the error. If <i>forceDisable</i> is true, the service disables the notification.
3	The service validates the schema against the notification's settings. The validation requires that the schema be in the same catalog that is configured with the notification and that the table, synonym, view, or alias, configured in the notification be in the schema. If the schema is not valid, the service throws an exception.
4	The service updates the notification property with the new schemas, and throws an exception if the input fields for the new schemas are not valid.
5	If the original notification state is enabled, the service enables the notification.

Note: If an exception is thrown when the service attempts to re-enable a polling notification, you will receive the message "The schema is updated but the notification could not be enabled." This is because, by design, the metadata has already been updated. You might need to roll back to the previous state, as needed.

Input Parameters

<i>nodeName</i>	String. Required. Sets the name of the notification to be updated.
<i>forceDisable</i>	Boolean. Required. If the value of this field is set to true, the service disables the notification that is in the enabled state.
<i>allSchemaChange</i>	Record. Optional. Indicates that the update is to occur on every schema name in the notification.
<i>allSchemaChange.useCurrentSchema</i>	Boolean. Optional. It sets all schema names to <current schema>.
<i>allSchemaChange.schemaName</i>	String. Optional. Sets all schema names to the value of this field.
<i>schemaChanges</i>	Record List. Optional. Makes individual schema changes, replacing the schema identified by the <i>existingSchema</i> field with the new name set in the <i>newSchema</i> field.
<i>schemaChanges.existingSchema</i>	Record. Required. Identifies the schema name to change.
<i>schemaChanges.existingSchema.useCurrentSchema</i>	Boolean. Optional. Identifies the <current schema>.
<i>schemaChanges.existingSchema.schemaName</i>	String. Optional. Specifies an existing schema name for the <i>schemaName</i> field.
<i>schemaChanges.newSchema</i>	Record. Required. Identifies an existing schema name that will replace all occurrences of existing schemas identified in the <i>existingSchema</i> field.

<i>schemaChanges.newSchema.useCurrentSchema</i>	Boolean. Optional. Sets new schema names to <current schema>.
<i>schemaChanges.newSchema.schemaName</i>	String. Optional. Sets new schema names to this name.

Output Parameters

None.

Configuring the UpdateNotificationSchema Service

Keep the following points in mind when configuring the `pub.jdbcAdapter:updateNotificationSchema` service:

- The *allSchemaChange* and *schemaChanges* fields are mutually exclusive. This means that if *allSchemaChange* is set, then *schemaChanges* will be ignored.
- The *useCurrentSchema* and *schemaName* fields are mutually exclusive everywhere they occur in the input. This means that if *useCurrentSchema* is set to true, then the value in *schemaName* will be ignored.
- A localized string is used for the <current schema>.
- The service does not generate output. It throws an `AdapterServiceException` and wraps exceptions from ART and Adapter for JDBC.

Setting Input Fields

The following tables list the input fields to be set for certain use cases.

Note: Fields that do not list an input value in the tables below should be left empty.

Setting All Schemas to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>forceDisable</i>	true
<i>allSchemaChange</i>	

Input Field	Setting
<i>allSchemaChange.useCurrentSchema</i>	true

Setting All Schemas to Schema A

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>forceDisable</i>	true
<i>allSchemaChange</i>	
<i>allSchemaChange.schemaName</i>	A

Changing Occurrences of the <current schema> to Schema A

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>forceDisable</i>	true
<i>schemaChanges</i>	
<i>schemaChanges[0]</i>	
<i>schemaChanges[0].existingSchema</i>	
<i>schemaChanges[0].existingSchema.useCurrentSchema</i>	true
<i>schemaChanges[0].newSchema</i>	
<i>schemaChanges[0].newSchema.schemaName</i>	A

Changing Occurrences of Schema A to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>forceDisable</i>	true
<i>schemaChanges</i>	
<i>schemaChanges.schemaChanges[0]</i>	
<i>schemaChanges.schemaChanges[0].existingSchema</i>	
<i>schemaChanges.schemaChanges[0].existingSchema.schemaName</i>	A
<i>schemaChanges.schemaChanges[0].newSchema</i>	
<i>schemaChanges.schemaChanges[0].newSchema.useCurrentSchema</i>	true

Changing Occurrences of Schema A to Schema A1, and Occurrences of Schema B to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>forceDisable</i>	true
<i>schemaChanges</i>	
<i>schemaChanges.schemaChanges[0]</i>	
<i>schemaChanges.schemaChanges[0].existingSchema</i>	
<i>schemaChanges.schemaChanges[0].existingSchema.schemaName</i>	A

Input Field	Setting
<i>schemaChanges.schemaChanges[0].newSchema</i>	
<i>schemaChanges.schemaChanges[0].newSchema.schemaName</i>	A1
<i>schemaChanges.schemaChanges[1]</i>	
<i>schemaChanges.schemaChanges[1].existingSchema</i>	
<i>schemaChanges.schemaChanges[1].existingSchema.schemaName</i>	B
<i>schemaChanges.schemaChanges[1].newSchema</i>	
<i>schemaChanges.schemaChanges[1].newSchema.useCurrentSchema</i>	true

pub.jdbcAdapter:updateServiceSchema

The `pub.jdbcAdapter:updateServiceSchema` Java service enables you to change the schema settings associated with an adapter service without having to manually update the service from the Developer Adapter Service Editor.

This service supports the following types of services: InsertSQL, DeleteSQL, UpdateSQL, SelectSQL, StoredProcedure, and StoredProcedureWithSignature.

Unlike the [pub.jdbcAdapter:updateNotificationSchema](#) service, the `pub.jdbcAdapter:updateServiceSchema` service does not validate the new schemas against the service settings.

Input Parameters

<i>nodeName</i>	String. Required. Sets the name of the service to be updated.
<i>allSchemaChange</i>	Record. Optional. Indicates that the update is to occur on every schema name in the adapter service.
<i>allSchemaChange.useCurrentSchema</i>	Boolean. Optional. It sets all schema names to <current schema>.
<i>allSchemaChange.schemaName</i>	String. Optional. Sets all schema names to the value of this field.

<i>schemaChanges</i>	Record List. Optional. Makes individual schema changes, replacing the schema identified by the <i>existingSchema</i> field with the new name set in the <i>newSchema</i> field.
<i>schemaChanges.existingSchema</i>	Record. Required. Identifies the schema name to change.
<i>schemaChanges.existingSchema.useCurrentSchema</i>	Boolean. Optional. Identifies the <current schema>.
<i>schemaChanges.existingSchema.SchemaName</i>	String. Optional. Specifies an existing schema name for the <i>SchemaName</i> field.

Output Parameters

None.

Configuring the updateServiceSchema Service

Keep the following points in mind when configuring the [pub.jdbcAdapter:updateServiceSchema](#) service:

- The *allSchemaChange* and *schemaChanges* fields are mutually exclusive. This means that if *allSchemaChange* is set, then *schemaChanges* will be ignored.
- The *useCurrentSchema* and *schemaName* fields are mutually exclusive everywhere they occur in the input. This means that if *useCurrentSchema* is set to true, then the value in *schemaName* will be ignored.
- A localized string is used for the <current schema>.
- The service does not generate output. It throws an AdapterServiceException and wraps exceptions from ART and Adapter for JDBC.

Setting Input Fields

The following tables list the input fields to be set for certain use cases.

Note: Fields that do not list an input value in the tables below should be left empty.

Setting All Schemas to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>allSchemaChange</i>	
<i>allSchemaChange.useCurrentSchema</i>	true

Setting All Schemas to Schema A

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>allSchemaChange</i>	
<i>allSchemaChange.schemaName</i>	A

Changing Occurrences of the <current schema> to Schema A

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>schemaChanges</i>	
<i>schemaChanges[0]</i>	
<i>schemaChanges[0].existingSchema</i>	
<i>schemaChanges[0].existingSchema.useCurrentSchema</i>	true
<i>schemaChanges[0].newSchema</i>	
<i>schemaChanges[0].newSchema.schemaName</i>	A

Changing Occurrences of Schema A to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>schemaChanges</i>	
<i>schemaChanges.schemaChanges[0]</i>	
<i>schemaChanges.schemaChanges[0].existingSchema</i>	
<i>schemaChanges.schemaChanges[0].existingSchema.schemaName</i>	A
<i>schemaChanges.schemaChanges[0].newSchema</i>	
<i>schemaChanges.schemaChanges[0].newSchema.useCurrentSchema</i>	true

Changing Occurrences of Schema A to Schema A1, and Occurrences of Schema B to the <current schema>

Use the following fields and settings:

Input Field	Setting
<i>nodeName</i>	folder1:notification1
<i>schemaChanges</i>	
<i>schemaChanges.schemaChanges[0]</i>	
<i>schemaChanges.schemaChanges[0].existingSchema</i>	
<i>schemaChanges.schemaChanges[0].existingSchema.schemaName</i>	A
<i>schemaChanges.schemaChanges[0].newSchema</i>	
<i>schemaChanges.schemaChanges[0].newSchema.schemaName</i>	A1

Input Field	Setting
<i>schemaChanges.schemaChanges[1]</i>	
<i>schemaChanges.schemaChanges[1].existingSchema</i>	
<i>schemaChanges.schemaChanges[1].existingSchema.schemaName</i>	B
<i>schemaChanges.schemaChanges[1].newSchema</i>	
<i>schemaChanges.schemaChanges[1].newSchema.useCurrentSchema</i>	true

pub.jdbcAdapter:updateConnectionPassword

The `pub.jdbcAdapter:updateConnectionPassword` service updates the existing password of an existing connection with a new password without requiring you to manually change the connection's password in the **Connection Properties** screen.

You need to disable the connection before updating the password.

Input Parameters

<i>connectionAlias</i>	String. Required. The name of the connection for which the password has to be updated.
<i>oldPassword</i>	String. Required. The existing password of the connection.
<i>newPassword</i>	String. Required. The new password for the connection.
<i>confirmNewPassword</i>	String. Required. The new password for the connection. This is required to confirm the new password.

Output Parameters

None.

pub.jdbcAdapter:createConnectionNodes

The `pub.jdbcAdapter:createConnectionNodes` service automatically configures the JDBC connections without requiring you to use the Administrative screens.

The `pub.jdbcAdapter:createConnectionNodes` service is useful when you need to configure a large number of connections. The input for this service is an XML file that contains the connection configuration properties that are required to configure the connections.

A sample properties file, `SampleConnectionProperties.xml`, is available in the `pub` directory of the `WmJDBCAdapter` package. You can use this file as a template to provide the connection configuration properties and create a new XML file to configure connections. You can place the XML file in any folder you want to. For security reasons, it is not recommended to place the XML file having user names and passwords in clear text, in the `pub` directory of the `WmJDBCAdapter` package.

Note: When using the encoding attribute in the XML file, ensure that it matches the encoding used when the XML file is saved, and also supports the characters specified in the file. This ensures the correct interpretation of any foreign characters in the XML file.

In the XML file, provide the Connection properties and the Connection Manager properties as name-value pairs. Based on the number of connections that need to be configured, provide the same number of connection configuration property sets in the XML file. For example, in the XML file, if you provide two connection configuration property sets, the `pub.jdbcAdapter:createConnectionNodes` service configures only two connections.

Note: If the Connection Manager properties for the connection are not provided in the XML file, the connection is configured using the default Connection Manager properties.

Note: If the `SampleConnectionProperties.xml` file is deleted, it is automatically re-generated the next time the `WmJDBCAdapter` package is reloaded.

After executing the service, the Results panel displays the status (success or failure) for each connection. The Results panel also displays an error message for the connection that has failed to configure.

Input Parameters

fileName **String.** Required. The path of the XML file that contains the connection configuration properties.

Output Parameters

None.

Usage Notes

Using the `pub.jdbcAdapter:createConnectionNodes` service, you can configure a deleted connection that was configured using this service, but you cannot configure a connection that already exists. An error will be displayed in the Results panel indicating that

there was a failure in configuring the connection. For example, consider the following scenario:

1. Invoke the `pub.jdbcAdapter:createConnectionNodes` service.
2. In the `fileName` field, type the path of the XML file containing the connection properties. The input XML file has properties for configuring connections `conn_local1` and `conn_local2`.
3. Click **OK**. The connections `conn_local1` and `conn_local2` are configured.
4. Delete connection `conn_local1`. Now, the only existing connection is `conn_local2`.
5. Repeat step 1 and step 2 and then click **OK**.
6. The deleted connection `conn_local1` is configured again, but the existing connection `conn_local2` is not configured. No error message is thrown, but the Results panel shows a message indicating that there was a failure in configuring connection `conn_local2`.

Sample XML file with connection properties

```
<Connections>
  <Connection>
    <packageName>MyJDBC</packageName>
    <connectionAlias>Connections:con_local</connectionAlias>
    <connectionSettings>
      <transactionType>LOCAL_TRANSACTION</transactionType>
<datasourceClass>oracle.jdbc.pool.OracleConnectionPoolDataSource</datasourceClass>
      <serverName>localhost</serverName>
      <user>user1</user>
      <password>abc123</password>
      <databaseName>ORCL1</databaseName>
      <portNumber>1521</portNumber>
      <otherProperties>driverType=thin</otherProperties>
      <networkProtocol>TCP</networkProtocol>
    </connectionSettings>
    <connectionManagerSettings>
      <poolable>true</poolable>
      <minimumPoolSize>1</minimumPoolSize>
      <maximumPoolSize>10</maximumPoolSize>
      <poolIncrementSize>1</poolIncrementSize>
      <blockingTimeout>1000</blockingTimeout>
      <expireTimeout>1000</expireTimeout>
      <startupRetryCount>1</startupRetryCount>
      <startupBackoffSecs>20</startupBackoffSecs>
    </connectionManagerSettings>
  </Connection>
  <Connection>
    <packageName>MyJDBC</packageName>
    <connectionAlias>Connections:con_local2</connectionAlias>
    <connectionSettings>
      <transactionType>NO_TRANSACTION</transactionType>
<datasourceClass>oracle.jdbc.pool.OracleConnectionPoolDataSource</datasourceClass>
      <serverName>localhost</serverName>
      <user>user2</user>
      <password>xyz321</password>
```

```

    <databaseName>ORCL1</databaseName>
    <portNumber>1521</portNumber>
    <otherProperties>driverType=thin</otherProperties>
    <networkProtocol>TCP</networkProtocol>
  </connectionSettings>
  <connectionManagerSettings>
    <poolable>true</poolable>
    <minimumPoolSize>5</minimumPoolSize>
    <maximumPoolSize>10</maximumPoolSize>
    <poolIncrementSize>1</poolIncrementSize>
    <blockingTimeout>1000</blockingTimeout>
    <expireTimeout>1000</expireTimeout>
    <startupRetryCount>1</startupRetryCount>
    <startupBackoffSecs>20</startupBackoffSecs>
  </connectionManagerSettings>
</Connection>
</Connections>

```

pub.pollingNotificationUtils:dropDatabaseObjects

The service `pub.pollingNoificationUtils:dropDatabaseObjects` allows you to automatically clean up the existing database objects of a notification.

Input Parameters

notificationName **String.** Required. The name of the notification from which the database objects need to be deleted.

Output Parameters

None.

Usage Notes

Disable the target notification before invoking this service. Invoking this service for an enabled or a suspended notification, throws an error.

pub.pollingNotificationUtils:getDatabaseObjectsForNotification

The service `pub.pollingNotificationUtils:getDatabaseObjectsForNotification` allows you to list the existing database objects associated with a notification.

The service returns null against the database objects that do not exist.

Input Parameters

notificationName **String.** Required. The name of the notification from which to list the database objects.

jdbcConnectionSchemaName **String** The name of the schema in the connection for the notification.

cliOrSysSchemaName **String** The name of the alternative search schema.

Output Parameters

schemaName The name of the schema associated with the notification.

bufferTableName The buffer table name of the notification.

triggerName The trigger name of the notification.

sequenceName The sequence name of the notification.

C Built-In Transaction Management Services

■ Transaction Management Overview	244
■ Built-In Transaction Management Services	252
■ pub.art.transaction:commitTransaction	252
■ pub.art.transaction:rollbackTransaction	253
■ pub.art.transaction:setTransactionTimeout	254
■ pub.art.transaction:startTransaction	254
■ Changing the Integration Server Transaction Timeout Interval	255

Transaction Management Overview

This appendix provides an overview and examples of using transactions. It describes how Integration Server supports the built-in services used to manage explicit transactions for your Adapter for JDBC services in the WmART package. For descriptions of each of the specific built-in transaction management services that can be used with the WmART package, see [“Built-In Transaction Management Services” on page 252](#).

For information about other built-in services available with Adapter for JDBC, see the Integration Server built-in services reference guide for your release. See “About this Guide” for specific document titles.

Transactions

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

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

Integration Server can automatically manage both kinds of transactions, without requiring the adapter user to do anything. Integration Server uses a container-managed (implicit) transaction management approach based on the Connector Architecture standard, 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 245](#).

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 Transaction Examples” on page 247](#).

To support transactions, Integration Server relies on an Oracle 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.

The transaction manager only manages operations performed by adapter services, a transacted JMS trigger, or a built-in JMS service that uses a transacted JMS connection alias.

Important: You cannot step or trace a flow that contains a transacted adapter service or a transacted JMS 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 Integration Server to manage XA transactions, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.

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, Integration Server automatically manages both local and XAResource transactions without requiring you to explicitly do anything. That is, 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 you choose

to provide dynamic user credentials at run time, then all the adapter services using the LOCAL_TRANSACTION connection within a single transaction must use the same user credentials. For example, if you have two adapter services *s1* and *s2* configured using the LOCAL_TRANSACTION connection *c1* in a single transaction context, then both *s1* and *s2* must either use the same dynamic credentials at run time or the default configured credentials provided at design time. For more information on dynamic user credentials for a service's associated connection, see [“Changing the User Credentials of a Service's Associated Connection at Run Time” on page 27](#).

For implicit transaction examples, see [“Implicit and Explicit Transaction Examples” on page 247](#).

For more information about designing and using flows, see the Designer Service Development online help for your release. See [“About this Guide”](#) for specific document titles.

For more information about transaction types, see [“Transaction Management of Adapter Connections” on page 19](#).

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. If you choose to provide dynamic user credentials at run time, then all the adapter services using the LOCAL_TRANSACTION connection within a single transaction must use the same user credentials. For example, if you have two adapter services *s1* and *s2* configured using the LOCAL_TRANSACTION connection *c1* in a single transaction context, then both *s1* and *s2* must either use the same dynamic credentials at run time or the default configured credentials provided at design time.

For more information on dynamic user credentials for a service's associated connection, see [“Changing the User Credentials of a Service's Associated Connection at Run Time” on page 27](#).

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 Integration Server. You must also ensure that the `startTransaction` is outside the SEQUENCE.

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, the following is a *valid* construct:

```
pub.art.transaction:startTransaction
  pub.art.transaction:startTransaction
    pub.art.transaction:startTransaction
      pub.art.transaction:commitTransaction
        pub.art.transaction:commitTransaction
          pub.art.transaction:commitTransaction
```

The following example shows an *invalid* construct:

```
pub.art.transaction:startTransaction
  pub.art.transaction:startTransaction
pub.art.transaction:commitTransaction
  pub.art.transaction:commitTransaction
```

For explicit transaction examples, see [“Implicit and Explicit Transaction Examples”](#) on page 247.

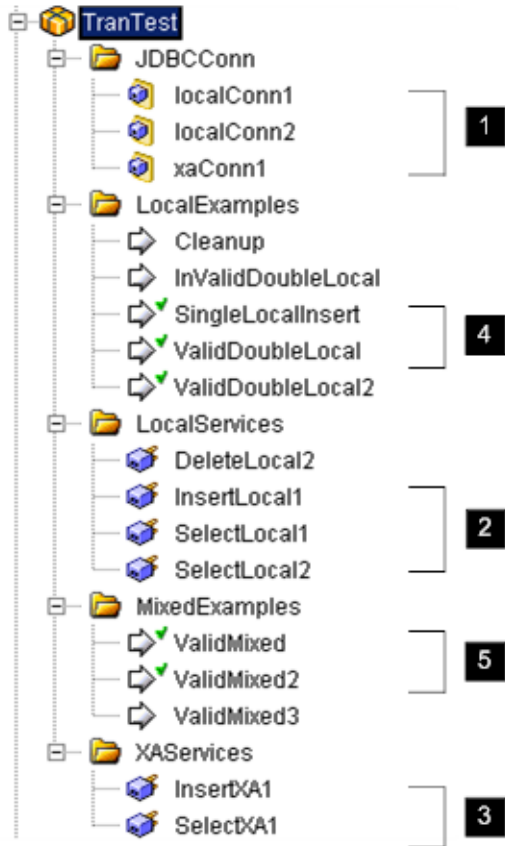
Note: You can use the `pub.flow:getLastError` service in the SEQUENCE, to retrieve the error information when a sequence fails. For more information on using the `pub.flow:getLastError` service, see the Designer Service Development online help for your release. See [“About this Guide”](#) for specific document titles.

For more information about designing and using flows, see the Designer Service Development online help for your release. See [“About this Guide”](#) for specific document titles.

For more information about transaction types, see [“Transaction Management of Adapter Connections”](#) on page 19.

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.

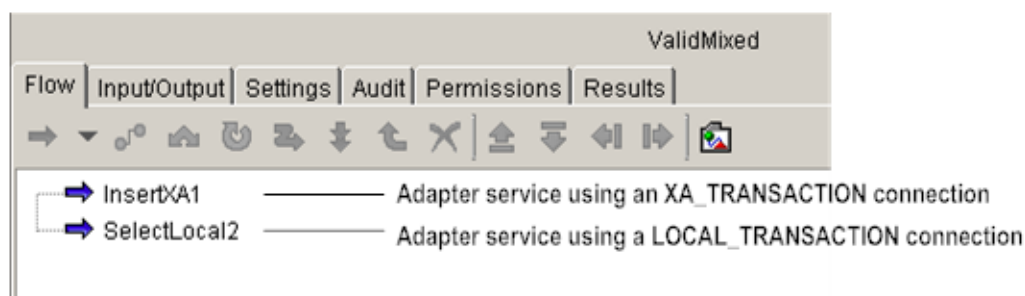


Step	Description
1	<p>You configured three connections:</p> <ul style="list-style-type: none"> ■ localConn1: LOCAL_TRANSACTION type ■ localConn2: LOCAL_TRANSACTION type ■ xaConn1: XA_TRANSACTION type
2	<p>You configured the following adapter services which use the LOCAL_TRANSACTION connections listed in step 1 above.</p> <ul style="list-style-type: none"> ■ InsertLocal1: configured to use localConn1 connection ■ SelectLocal1: configured to use localConn1 connection ■ SelectLocal2: configured to use localConn2 connection
3	<p>You configured the following adapter services which use the XA_TRANSACTION connection listed in step 1 above.</p> <ul style="list-style-type: none"> ■ InsertXA1: uses xaConn1 connection

Step	Description
	<ul style="list-style-type: none"> ■ SelectXA1: uses xaConn1 connection
4	<p>You create the following flow examples (described in this section) using LOCAL_TRANSACTIONs:</p> <ul style="list-style-type: none"> ■ SingleLocalInsert (explicit transaction). See “Flow Example: SingleLocalInsert” on page 249. ■ ValidDoubleLocal (explicit transaction). See “Flow Example: ValidDoubleLocal” on page 251.
5	<p>You create the following flow examples (described in this section) using both XA_TRANSACTIONs and LOCAL_TRANSACTIONs:</p> <ul style="list-style-type: none"> ■ ValidMixed (implicit transaction). See “Flow Example: ValidMixed” on page 249. ■ ValidMixed2 (implicit/explicit transaction). See “Flow Example: ValidMixed2” on page 250.

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)

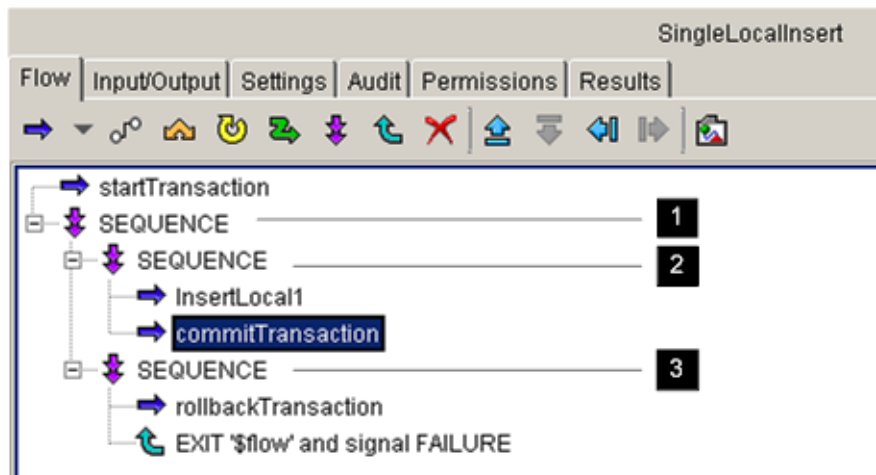


Flow Example: SingleLocalInsert

- 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. Note that the start transaction is outside the SEQUENCE.
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.

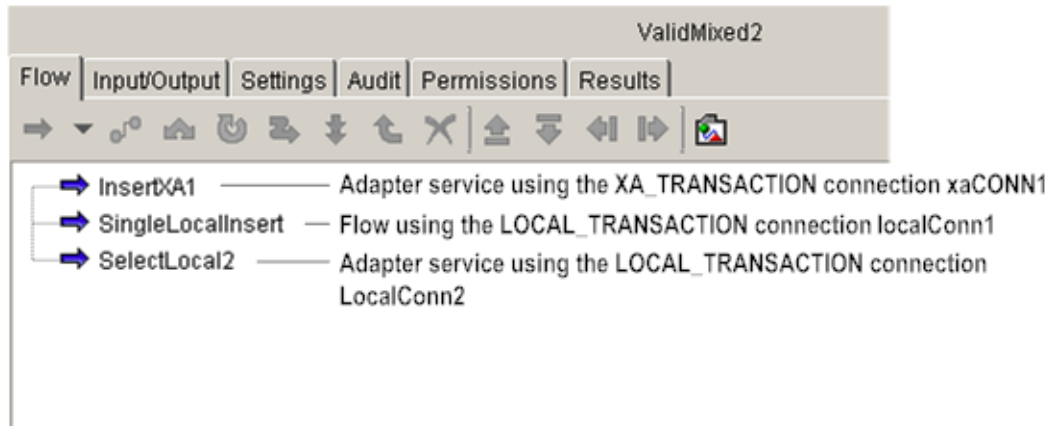
Note that with this construct, you will not get trigger retries or a retryable exception. The EXIT statement will result in generating a Flow exception which is not retryable. To get retries, you will need to use a REPEAT step statement in your flow. For information about using the REPEAT statement, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles.

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 [“Flow Example: SingleLocalInsert”](#) on page 249) which contains its own explicit transactions and using a LOCAL_TRANSACTION connection (localConn1)

- 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).

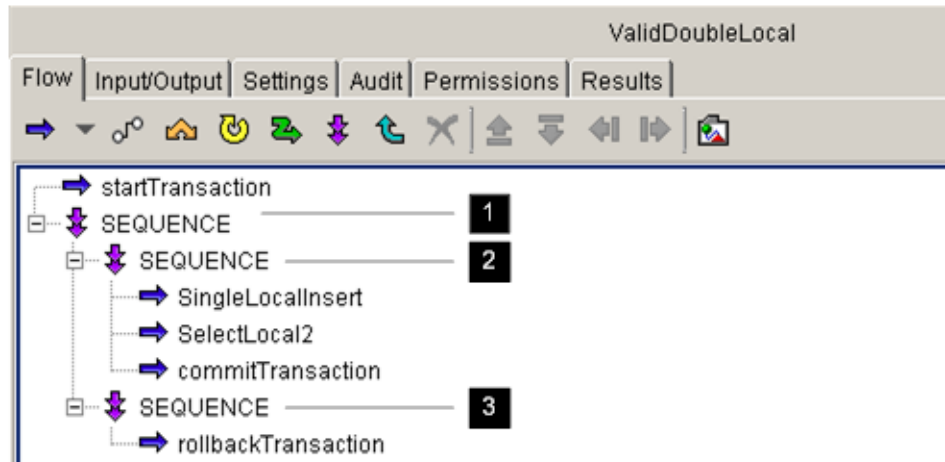


Flow Example: ValidDoubleLocal

- 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 shown in [“Flow Example: SingleLocalInsert” on page 249](#), 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. Note that the start transaction is outside the SEQUENCE.
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 run time error. For more information about implicit and explicit transactions, see [“Implicit and Explicit Transactions” on page 245](#).

Input Parameters

<i>commitTransactionInput</i>	Document. A document that contains the variable <i>transactionName</i> , described below.
<i>transactionName</i>	String. Used to associate a name with an explicit transaction. The <i>transactionName</i> must correspond to the <i>transactionName</i> in any <code>pub.art.transaction:startTransaction</code> or <code>pub.art.transaction:rollbackTransaction</code> services associated with the explicit transaction. This value must be mapped from the most recent <code>pub.art.transaction:startTransaction</code> 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 run time error. For more information about implicit and explicit transactions, see [“Implicit and Explicit Transactions” on page 245](#).

Input Parameters

<i>rollbackTransactionInput</i>	Document. A document that contains the variable <i>transactionName</i> , described below.
<i>transactionName</i>	String. Used to associate a name with an explicit transaction. The <i>transactionName</i> must correspond to the <i>transactionName</i> in any <code>pub.art.transaction:startTransaction</code> or <code>pub.art.transaction:commitTransaction</code> services associated with the explicit transaction. This value must be mapped from the most recent <code>pub.art.transaction:startTransaction</code> 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. For information on changing the server's default transaction timeout, see [“Changing the Integration Server Transaction Timeout Interval” on page 255](#).

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

<i>timeoutSeconds</i>	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 run time error.

For more information about implicit and explicit transactions, see [“Implicit and Explicit Transactions” on page 245](#).

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, 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 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.

For more information about adding parameters to the `server.cnf` file, see the Integration Server administration guide for your release. See "About this Guide" for specific document titles.

D Adapter Configuration Parameters Appendix

■ Overview	258
■ watt.adapter.JDBC.AutomaticNotification.Joincolumn.BufferTable	258
■ watt.adapter.JDBC.DateWithTimestamp	258
■ watt.adapter.JDBC.DateWithTimestampAndMilliseconds	258
■ watt.adapter.JDBC.DisableEmptyResult	259
■ watt.adapter.JDBC.StoredProcedure.customRSColNames	259
■ watt.adapter.JDBC.UsePlainString	259

Overview

This appendix contains a description of Adapter for JDBC parameters you can specify in the server configuration file (`server.cnf`), which is located in the `Integration Server_directory\config` directory. Typically you use the Settings > Extended screen in Integration Server Administrator to update this file, but there might be times when you need to edit the file directly using a text editor. If you edit the file directly, you should first shut down Integration Server before updating the file. After you make the changes, restart the server. If you are using the Settings > Extended screen to update the server configuration file (`server.cnf`), a server restart is not required unless otherwise specified. The server uses default values for the parameters. If a parameter has a default, it is listed with the description of the parameter.

watt.adapter.JDBC.AutomaticNotification.Joincolumn.BufferTable

Specifies whether Adapter for JDBC displays the columns of the buffer table on the **Joins** tab for an Insert, Delete, or Update polling notification, when the notification has an Output Field name different from the column name in the source table, and the same column is selected on the **Joins** tab. If the parameter is set to `true`, the adapter displays the columns of the buffer table on the **Joins** tab for the notification. If the parameter is set to `false`, the adapter displays the columns of the source table on the **Joins** tab for the notification.

watt.adapter.JDBC.DateWithTimestamp

Appends the timestamp to the output of an Adapter for JDBC service when the adapter service retrieves data from a database table with a Date column, the JDBC type is set to DATE, and the **Output Field Type** parameter is set to `java.lang.String`. When the parameter is set to `true`, the adapter services append the timestamp to the output. When the parameter is set to `false`, the default, the adapter services do not append the timestamp to the output.

watt.adapter.JDBC.DateWithTimestampAndMilliseconds

Appends the timestamp with milliseconds to the output of an Adapter for JDBC service when the adapter service retrieves data from a database table with a Date column, the JDBC type is set to DATE, and the **Output Field Type** parameter is set to `java.lang.String` or `SetAsString`. When the parameter is set to `true`, the adapter services append the timestamp with milliseconds to the output. When the parameter is set to `false`, the default, the adapter services append the timestamp to the output without milliseconds.

watt.adapter.JDBC.DisableEmptyResult

Specifies whether Adapter for JDBC Custom SQL services and Dynamic SQL services return a document when the result set returned by the database is empty. When the parameter is set to `true`, the adapter services do not return an empty results document. When the parameter is set to `false`, the default, the adapter services return an empty results document.

watt.adapter.JDBC.StoredProcedure.customRSColNames

Specifies whether Adapter for JDBC supports the mapping of custom column names in the **Column Name** field to the result set in the Output signature of `StoredProcedure` and `StoredProcedureWithSignature` services. When the parameter is set to `true`, the default, the adapter supports the mapping of custom column names in the **Column Name** field. If the parameter is set to `false`, the adapter uses the standard values that match the result set in the Output signature of the `StoredProcedure` and `StoredProcedureWithSignature` service.

watt.adapter.JDBC.UsePlainString

Specifies whether Adapter for JDBC services return an exponential form of the column value retrieved from a database when using Java version 1.5 and higher. When the adapter services retrieve a column value of `Decimal` or `Numeric` JDBC Data Type and the **Output Field Type** is defined as `java.lang.String`, the output returned may sometimes be in exponential form. The output returned is the `String` obtained by invoking the `BigDecimal.toString()` method. Due to changes in the behavior of `BigDecimal.toString()` in Java 5, `BigDecimal.toString()` now returns exponential values in some cases. When the `watt.adapter.JDBC.UsePlainString` parameter is set to `true`, Adapter for JDBC invokes the `toPlainString()` method and returns the non-exponential form in all cases. When the parameter is set to `false`, the default, the adapter invokes the `toString()` method, and returns the exponential form if needed.

E Database Driver Known Limitations

- Driver Limitations 262

Driver Limitations

This appendix provides a high-level list of limitations and issues. For additional details, refer to your vendor documentation.

General Constraints

- When you reload the adapter values after modifying an existing `StoredProcedureWithSignature` service at the backend, the service parameters are updated, but the input fields are not updated. To work around this limitation:
 - Add new arguments to the end of the argument list in the stored procedure definition. Do not add new arguments in-between existing arguments in the list.
 - When you change the order of the service parameters, you must re-edit the input fields for the parameters manually.
- The `TEXT`, `IMAGE`, and `UNITEXT` data types are not supported for all types of adapter notifications in Sybase.

Driver Specific Constraints

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
DataDirect Connect for JDBC 3.2	UDB 7.2	<p>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:</p> <pre>[DataDirect][DB2 JDBC Driver][DB2]AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION 2 ON A PREPARE OR DESCRIBE OPERATION</pre> <p>Note:This driver does support BLOB data types using OS/390 or DB2 iSeries V5R2.</p> <hr/> <p>Driver does not support <code>XA_TRANSACTIONS</code> using Java Transaction API (JTA). Instead, use UDB 8.1.</p> <hr/> <p>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).</p>

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
	UDB 7.2 and UDB 8.1	<p>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.</p> <p>The CallableStatement.getClob() does not work; instead, use the IBM driver versions (DB2 app type 2 or DB2 net type 3).</p>
		<p>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.</p>
DB2 JDBC app (type 2)	AIX5.1	Cannot enable XA_TRANSACTION connections.
	Linux	Cannot enable XA_TRANSACTION connections.
	UDB DB2 8.1/ Oracle 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	<p>If a Stored Procedure Notification has been enabled for long periods of time, the following message is posted:</p> <pre>[IBM][CLI Driver][DB2/] SQL1131N DARI (Stored Procedure) process has been terminated abnormally is posted. SQLSTATE=38503</pre>
	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).

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
	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.
	UDB 7.2 and 8.1	StoredProcedure and StoredProcedureWithSignature adapter services, and StoredProcedure and StoredProcdeureNotificationWithSignature adapter notifications do not display functions in the Procedure Name field. The workaround is to use the StoredProcedure service or the StoredProcedureNotification and type the function name in the Procedure Name field.
	UDB 9.1	The support for DB2 JDBC app (type 2) driver is deprecated.
DB2 Universal type 4	UDB 8.x	<p>This driver does not support XA transactions for versions earlier than UDB 8.2. Instead, use the Universal type 2 driver if you need XA_TRANSACTION support in versions earlier than UDB 8.2.</p> <p>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.</p>
	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.
DB2 JDBC net (Type3)	DB2 7 on OS/390	<p>If you attempt to insert 20k or more records, either the system will deadlock or you will receive a timeout error.</p> <hr/> <p>StoredProcedureWithSignature services and StoredProcedureNotificationWithSignature</p>

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
		notifications are supported only on DB2 for OS/390 V6. Instead, you can use the StoredProcedure service or StoredProcedure Notification.
	DB2 7.2 on OS/390	When configuring a Adapter for JDBC notification in Software AG Designer (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.
DB2 JDBC net (Type3)	UDB 7.2	<p>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 Integration Server Administrator and select Settings > Extended > Edit Extended Settings and type:</p> <pre>watt.adapter.JDBC.JDBCLogFile= c:\log.txt</pre>
		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).
	UDB 7.2 and 8.1	StoredProcedure and StoredProcedureWithSignature adapter services, and StoredProcedure and StoredProcdeureNotificationWithSignature adapter notifications do not display functions in the Procedure Name field. The workaround

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
		is to use the StoredProcedure service or the StoredProcedureNotification and type the function name in the Procedure Name field.
	UDB 9.1	The DB2 database system does not support the type 3 driver.
Informix Driver for JDBC Version 2.21 type 4	Informix 7.31 and 9.x	<p>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.</p> <hr/> <p>With Informix 9.3 and 9.4 using XA_TRANSACTION, you cannot update LONGVARCHAR data type columns with a null value.</p> <hr/> <p>With Informix 9.3 and 9.4 using XA_TRANSACTION, you cannot update BOOLEAN data type columns with a NOT NULL value.</p>
JT400 versions lower than 6.0	DB2 for AS/400 (all versions)	<p>StoredProcedureWithSignature adapter services cannot operate on a DB2 for AS/400 database when:</p> <ul style="list-style-type: none"> ■ a stored procedure has a stored procedure name that is different from its specific name. ■ there are two stored procedures with the same procedure name but with different specific names.
jCONNECT 5.5 and 6.05 type 4	Sybase 11.x, 12.x, and 15.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.

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
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 8i, 9i, 10g, and 11g Thin Driver and OCI JDBC classes12 Driver	All supported Oracle databases	StoredProcedureWithSignature services and StoredProcedureNotificationWithSignature notifications do not work with Stored Procedures containing a parameter of type Oracle Cursor, when the ref cursor is declared as a cursor type that is defined as a strong type with the %ROWTYPE attribute. The workaround is define the ref cursor as a weak type.

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
Oracle JDBC OCI Driver	All supported databases	You must set an environment variable before you can configure connections. For details, see “Setting the Environment Variable for Oracle JDBC OCI Drivers” on page 79.
SQL Server 2000 Driver for JDBC Version 2.2.0019	Microsoft SQL Server 2000	<ul style="list-style-type: none"> ■ 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 Oracle'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 manager javax/sql/DataSource."
SQL Server 2005 Driver for JDBC Version 1.0.809.102	Microsoft SQL Server 2005	<ul style="list-style-type: none"> ■ 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	<p>This driver does not support the following services and notifications:</p> <ul style="list-style-type: none"> ■ BatchInsertSQL or BatchUpdateSQL services ■ InsertNotifications, UpdateNotifications, DeleteNotifications, and OrderedNotifications

If you use the @ character in a table or column name, you will receive the following syntax error:

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
		Expected something between the word 'SP\$CHAR#TABLE' and '@'." while using in Insert service on Teradata.
Teradata V2R6, R12.0, R13.0, R13.10, and R14.0 (Type 4)	All supported databases	This driver does not support the following services and notifications: <ul style="list-style-type: none"> ■ BatchInsertSQL or BatchUpdateSQL services ■ InsertNotifications, UpdateNotifications, DeleteNotifications, and OrderedNotifications
Advanced Server JDBC Connector (edb- jdbc16.jar)	Postgres Plus Advanced Server 9.0/ Any supported operating system	Query timeout is not supported.
Apache Hive	For all supported databases	To enable LOCAL_TRANSACTION, add transactionmode=ignore in Other Properties field. This driver does not support the following features: <ul style="list-style-type: none"> ■ StoredProcedure services, UpdateSQL services, BatchUpdateSQL services, DeleteSQL services ■ Polling Notifications ■ Support for XA_TRANSACTION
MongoDB		To enable LOCAL_TRANSACTION, add transactionmode=ignore in Other Properties field. REFRESH SCHEMA SQL statement adds newly discovered objects to your relational view of native data type. To work with a different operating system other than Windows, add SchemaDefinition=<Valid file path> in Other Properties field.

Driver	Database/Adapter IS Operating System/ Platform Affected	Limitation Description
		<p>This driver does not support the following features:</p> <ul style="list-style-type: none"> ■ StoredProcedure services ■ Polling Notifications ■ Support for XA_TRANSACTION
Apache Cassandra	For all supported databases	<p>To enable LOCAL_TRANSACTION, add <code>transactionmode=ignore</code> in Other Properties field .</p> <p><code>REFRESH SCHEMA Keyspace Name SQL</code> statement adds newly discovered objects to your relational view of native data type.</p> <p>To work with a different operating system other than Windows, add <code>SchemaMap=<Valid file path></code> in Other Properties field.</p> <p>This driver does not support the following features:</p> <ul style="list-style-type: none"> ■ StoredProcedure services ■ StoredProcedureWithSignature services ■ Polling Notifications ■ Support for XA_TRANSACTION
Apache Impala	For all supported databases	<p>This driver does not support the following:</p> <ul style="list-style-type: none"> ■ UpdateSQL, DeleteSQL, BatchInsertSQL, BatchUpdateSQL, StoredProcedure, and StoredProcedureWithSignature services ■ InsertNotifications, UpdateNotifications, DeleteNotifications, StoredProcedureNotifications, StoredProcedureNotificationsWithSignature and OrderedNotifications ■ Support for XA_TRANSACTION ■ Support for LOCAL_TRANSACTION

For an updated list of database driver limitations, see *webMethods Adapters System Requirements*.

Index

A

- adapter notifications. 152
- AdapterException 206
- Automatic polling of adapter metadata option 198

B

- buffer tables
 - retrieving data from 163

C

- clustering
 - requirements 73

D

- documentation
 - using effectively 11

E

- exceptions for JDBC Adapter
 - AdapterException 206

F

- flat files, specifying the input field types 116

N

- notifications
 - StoredProcedureNotification 175

P

- pub.jdbcAdapter.createConnectionNodes
 - example of the input XML file 239
 - sample properties file 238
 - usage notes 238

S

- Service accessing a connection object
 - forbidden methods 145
- Specifying the maximum number of result sets for a StoredProcedure service
 - watt property 137
- StoredProcedureNotification
 - configuring 175

T

- testing services 146

U

- user-defined data types 218

W

- Watt property to suppress error codes 209