Informatica

Informatica® Application Integration
July 2022

JDBC Connector Guide

Informatica Application Integration JDBC Connector Guide
July 2022

Publication Date: 2022-07-29

Table of Contents

Preface

Read the *JDBC Connector Guide* to learn how organization administrators and business users can use JDBC Connector to connect to databases such as IBM DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL by using a JDBC type 4 driver.

This guide assumes that you have an understanding of databases, and how to create connections and processes in Application Integration.

CHAPTER 1

Introduction to JDBC Connector

This chapter includes the following topics:

JDBC Connector

You can use JDBC connections to connect to databases with a JDBC Type 4 driver.

JDBC Connector uses the JDBC Specification 3.0 Java API and Type 4 Database Protocol Driver. JDBC supports multiple platforms.

You can use the JDBC Connector to connect to the following databases:

- IBM DB2
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

Note: JDBC Connector extends generic connectivity to databases that Informatica does not have access to through the Informatica Application Integration native connectors. When you use JDBC Connector, Informatica does not guarantee that the functionality or performance might be equivalent to what you expect when you use the Informatica Application Integration native connectors. The database that you want to connect to might not have been tested or certified for use with JDBC Connector and you might experience connection or execution issues.

Administration of JDBC Connector

Before you use JDBC objects as sources or targets in tasks, an administrator must install and configure JDBC connections.

As a user, you can use JDBC Connector after an administrator performs the following tasks:

- Install the JDBC_IC connector for your organization.

- Use the latest database driver version that your database supports. Install the JDBC driver jar files for the database and then specify the location of the JAR file directory in the JDBC connection properties.
- If you do not want to specify the JDBC JAR directory in the connection properties, install the JDBC JAR files in the following location: `<Secure Agent installation directory>/apps/process-engine/ext`. The Secure Agent picks up the JAR files from this directory.

CHAPTER 2

JDBC Connections

This chapter includes the following topics:

JDBC Connections Overview

Use a JDBC connection to connect to and import JDBC entities from a schema.

After you create a JDBC connection, you can validate, test, and save the connection.

You can then publish the JDBC connection and click the **Metadata** tab to view the generated process objects for the connection. To publish a JDBC connection successfully, you must have the connect and resource privileges, and other related permissions to perform CRUD operations on the schema.

Note: You cannot configure a JDBC connection to connect to multiple schemas.

Basic Connection Properties

The following table describes the basic properties that you can configure on the **Properties** tab of the connection creation page:

Property	Description
Name	Required. A unique name for the JDBC connection that identifies it in the Process Designer. The name must start with an alphabet and can contain only alphabets, numbers, or hyphens (-).
Location	Optional. The location of the project or folder where you want to save the connection. Click **Browse** to select a location. If the **Explore** page is currently active and a project or folder is selected, the default location for the connection is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.

Property	Description
Description	Optional. A description of the connection.
Type	Required. The type of connection that you want to create. Select **JDBC Generic Cloud Adapter**.
Run On	Required. The Secure Agent group or the Secure Agent machine where the connection must run.
Connection Test	Displays the result of the last connection test.
OData-Enabled	Optional. Specifies whether OData feeds are enabled for the connection. Select **Yes** to enable OData feeds. If you select **Yes**, you must specify either the allowed users or the allowed groups that can access the connection at design time. You can also enable or disable Cloud access to the OData endpoint URLs. Default is **No**. For more information about the OData endpoint URLs, see "Endpoint URLs for OData-Enabled Connections" on page 12.
OData Cloud Access Enabled	Optional. Specifies whether you can access data from the OData service by using the Cloud endpoint URLs or the Secure Agent endpoint URLs. If you enable OData and configure the connection to run on a Secure Agent machine or a Secure Agent group, you can choose to disable access to the Cloud endpoint URLs for security purposes. Select **No** to disable access to the Cloud endpoint URLs. If you select **No**, you cannot access data from the OData service by using the Cloud endpoint URLs. You can access data only by using the Secure Agent endpoint URLs. The following video shows you how to disable access to Cloud endpoint URLs for security purposes in OData-enabled JDBC connections: https://www.youtube.com/watch?v=XouJ5OjDAuY Select **Yes** to access data from the OData service by using both the Cloud endpoint URLs and the Secure Agent endpoint URLs. Default is **Yes**. **Note:** If you set the **OData-Enabled** option to **No**, the value that you set for the **OData Cloud Access Enabled** option does not apply because Application Integration does not generate the OData endpoint URLs. For more information about the OData endpoint URLs, see "Endpoint URLs for OData-Enabled Connections" on page 12.
Allowed Users for OData	Optional. The users that have access to the connection at design time. You can specify more than one user. After you specify the first value, press the Enter key or the Comma key, and then specify the next value.
Allowed Groups for OData	Optional. The user groups that have access to the connection at design time. You can specify more than one user group. After you specify the first value, press the Enter key or the Comma key, and then specify the next value.

After you configure the basic properties, you must also define the following properties:

- The properties applicable to the JDBC connection type
- The read and write attributes for the JDBC connection

JDBC Connection Properties

When you create a JDBC connection, you must configure the connection properties.

The following table describes the JDBC connection properties:

Connection Property	**Description**
JDBC Connection URL	Required. The URL schema for the database. Use the corresponding schema for the following databases: - IBM DB2. `jdbc:db2://<server>:<port>/<database>` - Microsoft SQL Server. `jdbc:sqlserver://<Host>\<Instance>:<Port>;databaseName=<Database>` - MySQL. `jdbc:mysql://<Host>:<Port>/<Database>` - Oracle. `jdbc:oracle:thin:@//<Host>:<Port>/<Service>` - PostgreSQL. `jdbc:postgresql://<Host>:<Port>/<Database>`
JDBC Jar Directory	The path to the JDBC driver .jar file. For example, you can enter the following directory: `C:/jdbc` If you do not specify a directory path, the Secure Agent gets the .jar file from the `process-engine/ext` directory. You must specify one of the following values for the JDBC connection to work successfully: - JDBC JAR directory. If you choose to specify the JDBC JAR directory, you can place the .jar file in any directory and specify the directory in the **JDBC Jar Directory** field. - JDBC driver class name. If you choose to specify the JDBC driver class name, you must place the JDBC driver .jar file in the following directory: `process-engine/ext`
JDBC Driver Class Name	The name of the JDBC driver class. Based on the database, you can specify one of the following driver class names: - IBM DB2: `jdbc.db2.DB2Driver` - Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver` - MySQL: `jdbc.mysql.MySQLDriver` - Oracle: `jdbc.oracle.OracleDriver` - PostgreSQL `org.postgresql.Driver` You must specify one of the following values for the JDBC connection to work successfully: - JDBC JAR directory. If you choose to specify the JDBC JAR directory, you can place the .jar file in any directory and specify the directory in the **JDBC Jar Directory** field. - JDBC driver class name. If you choose to specify the JDBC driver class name, you must place the JDBC driver .jar file in the following directory: `process-engine/ext`
Schema	Optional. The schema name, which varies by database. Use the following guidelines for the schema name: - IBM DB2. Use the schema name to specify the correct object. - Microsoft SQL Server. Use the schema name to specify the correct object. - MySQL. Optional. The schema name is the database name. - Oracle. Optional. The schema name is the user name. - PostgreSQL. Use the schema name to specify the correct object. If the JDBC connection URL does not provide enough context, you must enter a schema name to fetch the metadata.
User name	Required. User name to connect to the database.
Password	Required. Password to connect to the database.

Connection Property	**Description**
Object Filter	Optional. A comma-separated list of object names.
TimeZone	Optional. The time zone of the database. Specify the value of the database time zone when the machine hosting the database and the machine hosting the Secure Agent are in different time zones.

Paginating Data with a JDBC Connection

When you use a JDBC connection, use OData to paginate data and process data in chunks.

For example, use the following OData URL to use the `JDBC-MySQL-Connection` connection to skip 1000 entries and select the next 100 entries in the `ADDRESS` table:

```
<cloud-application-integration-home>/JDBC-MySQL-Connection/ADDRESS?$skip=1000&$top=100
```

You can use the OData `$skip` parameter with the following databases:

- Oracle
- Microsoft SQL Server
- DB2
- MySQL

To use the `$skip` parameter with a DB2 or a Microsoft SQL Server database, you must specify both the `$skip` and `$orderby` values.

For example, use the following OData URL to use the `JDBC-SQLSRVR-Connection` connection to skip 1000 entries, select the next 50 entries, and order data by the `ADDR_ID` column of the `ADDRESS` table in a Microsoft SQL Server database:

```
<cloud-application-integration-home>/JDBC-SQLSRVR-Connection/ADDRESS?$skip=1000&$top=50&
$orderby=ADDR_ID
```

You can use complex filter conditions that contain multiple logical operators and braces. For example, you can use the following filter condition:

```
$filter=AGE gt 20 and (SALARY gt 1500 or (SALARY gt 1000 and TITLE eq 'SE'))
```

You can also paginate data when you use complex filter conditions.

Setting the Database Time Zone for the JDBC Connection

When you enable OData for a JDBC connection and read data that contains the date, time, datetime, or timestamp datatypes, the OData request returns incorrect values from the database. This error occurs when the time zone of the machine that hosts the database is not the same as the machine that hosts the Secure Agent. To avoid this error, you must select the appropriate time zone value of the database in the connection properties to manage the time zone conversion and return correct values.

JDBC Connection Read Attributes

The following table describes the read attributes that you can specify for a JDBC connection:

Read Attributes	Description
Isolation Level	Optional. Specifies the isolation level while reading from source object. The value you specify determines the level of concurrency in transactions to avoid data inconsistency. Select from the following isolation levels: - None - TRANSACTION READ UNCOMMITTED - TRANSACTION READ COMMITTED - TRANSACTION REPEATABLE READ - TRANSACTION SERIALIZABLE Default is **None**.
Override Isolation Level	Optional. Overrides the isolation level if the database does not support it. If this option is checked, the task runs successfully with the default isolation level of the database. Default is **No**.

JDBC Connection Write Attributes

The following table describes the write attributes that you can specify for a JDBC connection:

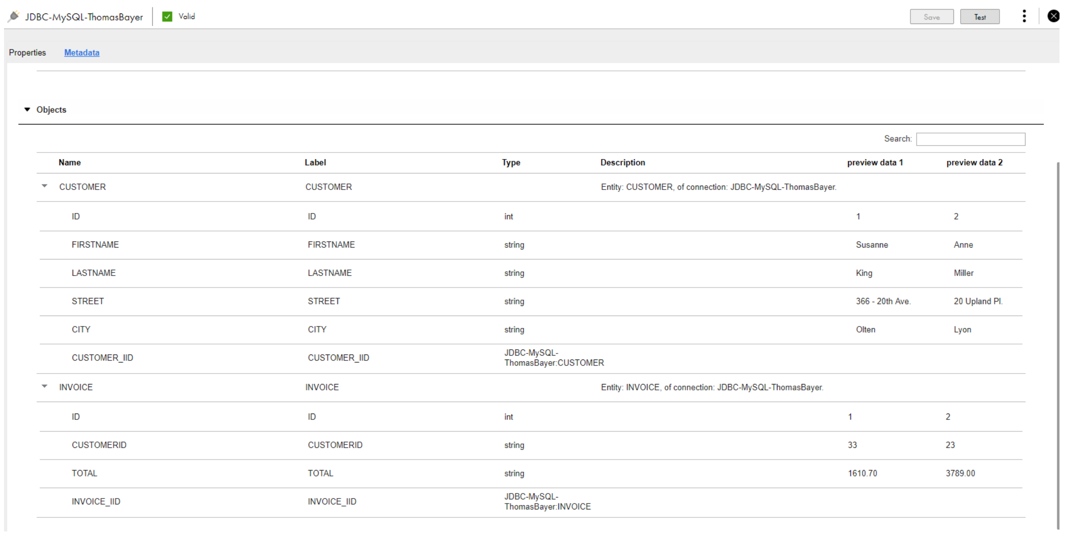
Write Attributes	Description
Number of retries	Optional. An integer value. Number of times to retry if the query execution fails due to a closed connection. Default is **5**.
Retry wait period	Optional. Interval in seconds between two retry attempts. Default is **3**.

JDBC Connection Metadata

After you create a JDBC connection, save the connection. You can then publish the JDBC connection and click the **Metadata** tab to view the generated process objects for the connection.

When you publish a JDBC connection, a list of **Objects** appear in the **Metadata** tab. The **Objects** section displays the entities along with the column metadata for the schema you specified in the connection. You can also preview the data.

The following image shows the **Metadata** tab for a published JDBC connection:



Name	Label	Type	Description	preview data 1	preview data 2
CUSTOMER	CUSTOMER		Entity: CUSTOMER, of connection: JDBC-MySQL-ThomasBayer.		
ID	ID	int		1	2
FIRSTNAME	FIRSTNAME	string		Susanne	Anne
LASTNAME	LASTNAME	string		King	Miller
STREET	STREET	string		366 - 20th Ave.	20 Upland Pl.
CITY	CITY	string		Olten	Lyon
CUSTOMER_IID	CUSTOMER_IID	JDBC-MySQL-ThomasBayer:CUSTOMER			
INVOICE	INVOICE		Entity: INVOICE, of connection: JDBC-MySQL-ThomasBayer.		
ID	ID	int		1	2
CUSTOMERID	CUSTOMERID	string		33	23
TOTAL	TOTAL	string		1610.70	3789.00
INVOICE_IID	INVOICE_IID	JDBC-MySQL-ThomasBayer:INVOICE			

When you publish a JDBC connection, the **Metadata** tab is refreshed by default. You can click **Publish - Skip Metadata Refresh** to skip the metadata refresh and reduce publishing time.

Best practices for metadata refresh

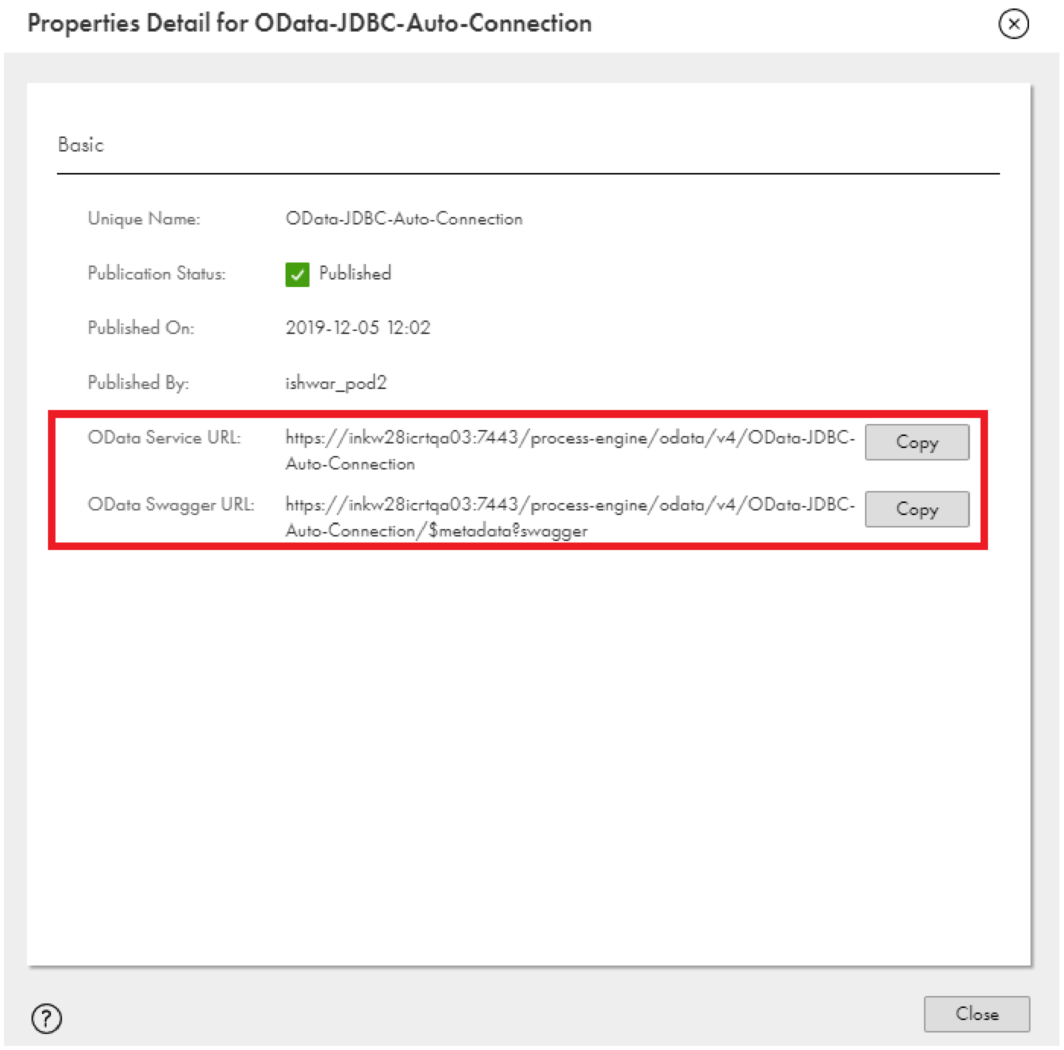
- When you create a JDBC connection for the first time, click **Publish** to download the metadata. Subsequently, you can click **Publish - Skip Metadata Refresh** if there are no metadata changes.
- If you want to change credentials in a JDBC connection but you know that the metadata has not changed, skip the metadata refresh while publishing the connection to apply just the credential changes.
- If you know that the metadata has changed, you must click **Publish** to retrieve the metadata changes.

Endpoint URLs for OData-Enabled Connections

When you publish an OData-enabled connection, Application Integration generates an OData Service URL and an OData Swagger URL.

To view the URLs, click **Actions** > **Properties Detail**. You can use the OData Service URL to view the details of the REST endpoint. You can use the OData Swagger URL to view the details of the endpoint and metadata of the OData API.

The following image shows the generated OData Service URL and OData Swagger URL for an Informatica Cloud endpoint:



Properties Detail for OData-JDBC-Auto-Connection

Basic

Unique Name:	OData-JDBC-Auto-Connection
Publication Status:	✓ Published
Published On:	2019-12-05 12:02
Published By:	ishwar_pod2
OData Service URL:	https://inkw28icrtqa03:7443/process-engine/odata/v4/OData-JDBC-Auto-Connection
OData Swagger URL:	https://inkw28icrtqa03:7443/process-engine/odata/v4/OData-JDBC-Auto-Connection/$metadata?swagger

If you enable Cloud access to the OData endpoint URLs, by default, the URLs are generated for an Informatica Cloud endpoint. You can also use the equivalent Secure Agent endpoint URLs to access data from the OData service.

The following table describes how to convert the Cloud endpoint URLs into the Secure Agent endpoint URLs by changing their format:

URL Type	Default Cloud Endpoint URL Format	URL Format to be Used for the Agent
OData Service URL	`https://<Informatica Intelligent Cloud Services URL>/active-bpel/odata/v4/<connection_name>`	`https://<host>:<port>/process-engine/odata/v4/<connection_name>/<schema_name>`
OData Swagger URL	`https://<Informatica Intelligent Cloud Services URL>/active-bpel/odata/v4/<connection_name>/$metadata?swagger`	`https://<host>:<port>/process-engine/odata/v4/<connection_name>/$metadata?swagger`

To view a video about generating a Swagger endpoint URL for an OData-enabled JDBC connection and viewing the metadata of the OData API, see the following community article:

https://knowledge.informatica.com/s/article/DOC-18549

CHAPTER 3

JDBC Connector Processes

This chapter includes the following topic:

JDBC Connector Processes Overview

You can use Process Designer in Application Integration to read, create, update, or delete data in databases with a JDBC type 4 driver.

In an Application Integration process, you can configure a JDBC connection to connect to databases with a JDBC type 4 driver. Use the connection in a process to perform multiple activities. You can specify a JDBC connection in step types such as Assignment and Create.

In a process that uses a JDBC connection, when you assign a value to a field, you can select the **Use Database-specific WHERE Clause Syntax** option to use nested queries and other database-specific querying options.

Example

As an HR administrator in an organization, you want to update the employee details when new employees join your organization. You can create an JDBC connection to the Employees table in an Oracle database. You can then design and run a process to insert the employee details from a source file object to the Employee table in the database.

CHAPTER 4

JDBC Data Type Reference

This chapter includes the following topic:

JDBC and Transformation Data Types

Application Integration uses the JDBC type 4 driver to read data. The Secure Agent converts the JDBC data type to the transformation data type, and uses the transformation data type to move data across platforms.

When Application Integration writes to a JDBC target, the Secure Agent converts the transformation data type to the corresponding JDBC data type.

The following table compares the JDBC data types that Application Integration supports to the transformation data types:

JDBC Data Type	Transformation Data Type	Range
BIGINT	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
BINARY	Binary	1 to 104,857,600 bytes
BIT	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
BLOB	Binary	1 to 104,857,600 bytes
BOOLEAN	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
CHAR	String	1 to 104,857,600 characters
CLOB	Text	1 to 104,857,600 characters
DATE	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
DECIMAL	Decimal	Precision 1 to 28, scale 0 to 28

JDBC Data Type	Transformation Data Type	Range
DOUBLE	Double	Precision 15
FLOAT	Double	Precision 15
INTEGER	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
LONGVARBINARY	Binary	1 to 104,857,600 bytes
LONGVARCHAR	Text	1 to 104,857,600 characters
NUMERIC	Decimal	Precision 1 to 28, scale 0 to 28
REAL	Double	Precision 15
SMALLINT	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
TIME	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
TIMESTAMP	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
TINYINT	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
VARBINARY	Binary	1 to 104,857,600 bytes
VARCHAR	String	1 to 104,857,600 characters

INDEX