Database

Database Connector

Overview

The database connector is accessed from the **Connectivity** tab of the design component palette:

This connector is used to first configure a database connection to establish access with a specific database, and then to configure one or more database activities associated with that connection to use as a source or target within an operation or script:

- **Query:** Queries data from a database connection and is used as a source in an operation or called in a script.
- **Insert:** Inserts new data in a database connection and is used as a target in an operation or called in a script.
- **Update:** Updates existing data in a database connection and is used as a target in an operation or called in a script.
- **Upsert:** Both updates existing data and inserts new data in a database connection and is used as a target in an operation or called in a script. (Jitterbit Harmony supports upsert activities for databases by using a combination of query, insert, and update.)

TIP: As there is no database delete activity, to delete data from a database connection, use the `DBExecute()` function in a Jitterbit Script by specifying the SQL command to be executed against the database.

Together, a specific database connection and its activities are referred to as a database endpoint. Once a connection is configured, activities associated with the endpoint are available from the **Endpoints** filter:

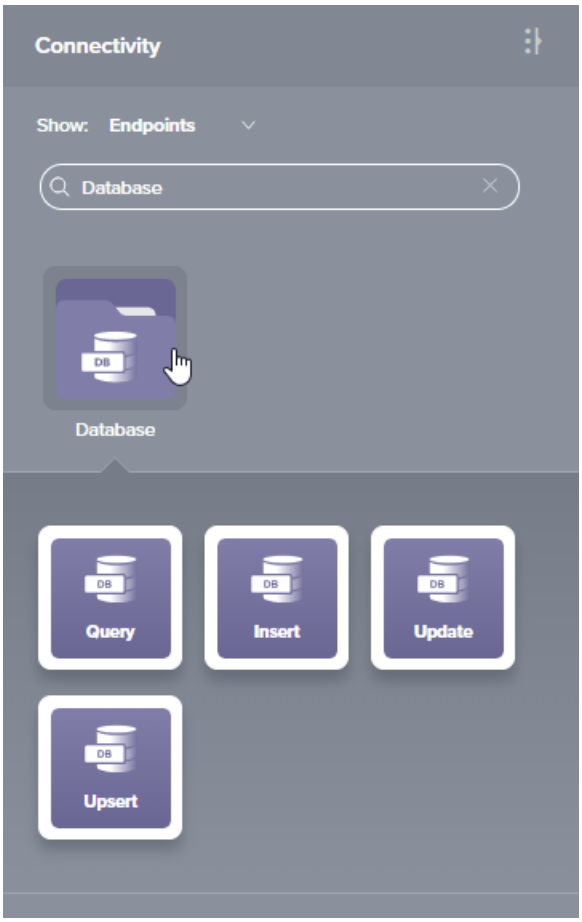
On This Page

Pages in This Topic

Search in This Topic

Related Topics

Last updated: Jan 14, 2020



Most JDBC- and ODBC-compliant databases are supported. For a complete list of supported databases, see Supported Endpoints and Protocols.

> **CAUTION:** In databases, Jitterbit does not support data with these encoding types or object names:
>
> - Binary data
> - Unicode/UTF data
> - Tables or views with spaces in the names

Installing Additional JDBC or ODBC Drivers

Database drivers are automatically detected from the Harmony Agent. For Cloud Agents, managed by Jitterbit, a set of commonly used JDBC drivers is supported and is already provided. For Private Agents, installed drivers are detected from the operating system where each Private Agent is installed. In addition, you can install other database drivers on Private Agents as needed. If you require the use of an ODBC driver, you can use a Windows Private Agent.

For general information and instructions on installing additional drivers on Private Agents, refer to Installing Additional ODBC or JDBC Drivers.

Special Characters in Database Table/Column Names

If using a Private Agent, you can specify characters used to define delimiters within database table /column names within the Private Agent configuration file under the `[DbDrivers]` section.

If using an ODBC database driver, note that some special characters in database table/column names are unable to be handled by the driver. For example, database fields that have an at sign (@) are not compliant with SQL-based specifications and may not be supported. If the database uses such special characters in table/column names, as a workaround we recommend creating a view on the physical table that does not use the special character in column names and using that instead.

Database-Specific Information

See Database-Specific Information for reference information on configuring these databases:

- IBM DB2 (AS400)
- Microsoft Access
- Microsoft Excel
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Progress

Database Functions

A number of database functions can be used within scripts to provide access to basic database interactions, including these:

- `CacheLookup`
- `CallStoredProcedure`
- `DBCloseConnection`
- `DBExecute`
- `DBLoad`
- `DBLookup`
- `DBLookupAll`
- `DBRollbackTransaction`
- `DBWrite`
- `SetDBInsert`
- `SetDBUpdate`
- `SQLEscape`
- `Unmap`
- `<SEQUENCE>`
- `<SQLIDENTITY>`
- `<UDF>`

For details on using these functions, see Database Functions.