



TIBCO Flogo[®] Connector for PostgreSQL User's Guide

*Software Release 2.1
August 2019*

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, TIBCO Cloud Integration, TIBCO Flogo Enterprise, TIBCO Flogo, and TIBCO Flogo® Connector for PostgreSQL are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2018-2019. TIBCO Software Inc. All Rights Reserved.

Contents

- TIBCO Documentation and Support Services5**
- Overview 6**
- Creating a PostgreSQL Connection7**
 - PostgreSQL Connection Details7
- PostgreSQL Query 8**
- PostgreSQL Insert 11**
- PostgreSQL Delete14**
- PostgreSQL Update 16**

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Documentation for TIBCO Flogo[®] Connector for PostgreSQL is available on the TIBCO Flogo[®] Connector for PostgreSQL Product Documentation page.

Product-Specific Documentation

The following documents for this product can be found on the TIBCO Documentation site:

- *TIBCO Flogo[®] Connector for PostgreSQL Installation*
- *TIBCO Flogo[®] Connector for PostgreSQL User's Guide*
- *TIBCO Flogo[®] Connector for PostgreSQL Release Notes*

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](https://community.tibco.com). For a free registration, go to <https://community.tibco.com>.

Overview

This connector enables you to run SQL queries on a PostgreSQL or a Greenplum database instance. A PostgreSQL connection is used to create a PostgreSQL or Greenplum query activity.

For details about using PostgreSQL or Greenplum Database, see their respective product documentation.

Creating a PostgreSQL Connection

You must create a PostgreSQL connection before you can use this connector. The PostgreSQL connection contains all the parameters required to connect to the PostgreSQL database. This connection is used by all the activities in the PostgreSQL category.



By default, the PostgreSQL database listens for connections on port 5432, although that is configurable as is the connection property on the PostgreSQL Connector.

To create a connection, click the **Connections** tab on the Flogo Enterprise page.

If this is the first connection you are adding, do the following:

1. Click the **PostgreSQL Connector** tile.
2. Enter the values for the fields in the **PostgreSQL Connector** dialog. See [PostgreSQL Connection Details](#) topic for a description of the fields.
3. Click **Connect**.

If you already have existing connections for any connector in Flogo Enterprise, the connections will be displayed on the **Connections** page.

1. Click **Add Connection**.
2. Click the **PostgreSQL Connector** tile.
3. Enter the values for the fields in the **PostgreSQL Connector** dialog. See [PostgreSQL Connection Details](#) topic for a description of the fields.
4. Click **Connect**.

PostgreSQL Connection Details

Provide the information of the PostgreSQL server that this connection will connect to.

The **PostgreSQL Connector** dialog contains the following fields:

Field	Description
Connection Name	A name for the PostgreSQL Connector connection that you are creating
Description	A short string describing the connection
Host	URL of the server that hosts the PostgreSQL Connector database
Port	Port number on which the PostgreSQL Connector database listens <div> By default PostgreSQL Connector cluster is configured with port 5432. For custom configurations, the port range should be within 1024 - 32767. </div>
Database Name	Name of the PostgreSQL Database
User	User name of the PostgreSQL Database user
Password	Password for the PostgreSQL Database account

PostgreSQL Query

Use this activity to execute a simple or a complex SQL Query on a database. The **PostgreSQL Query** activity returns information in the form of rows.


Settings

The **Settings** tab has the following fields:

Field	Description
Connection	Name of the PostgreSQL database connection from which to retrieve information. You can select the connection from the drop-down list.

Input Settings

The **Input Settings** tab has the following fields:

Field	Description
Query	<p>An SQL statement used to query the database. The query can be a simple query or a complex query. A complex query has nested SQL statements. You can construct prepared SQL queries by using substitution variables (or parameters) of the form ?<fieldname> in the query statement. For example, <code>select * from student where name = ?name;</code></p> <p>Each substitution variable identifies an input parameter whose mapped value will be substituted into the substitution variable at runtime. You can reuse the substitution variable for the same input parameter elsewhere in the query. Input parameters used in the WHERE clause, and output parameters used in the SELECT clause, and their corresponding type information is automatically fetched from the database using the selected connection for the entered query. Input and output fields in the Input and Output tabs of the activity are also automatically generated.</p> <div><p>Be sure to include the semicolon (;) at the end of the query. This activity expects a query to end with a semicolon to indicate the end of the query. A missing semicolon at the end of the query results in the query hanging.</p></div> <p>The following examples represent simple and complex queries:</p> <ul style="list-style-type: none">Simple query example: <code>SELECT * FROM student;</code> For the above query, the output fields are generated from the table student's column information. <pre>SELECT name, dept_name, tot_cred FROM student WHERE dept_name = ?dept_name and tot_cred > ?tot_cred ORDER BY dept_name;</pre> For the above query, output fields are generated for name, dept_name and tot_cred and input fields are generated for dept_name (varchar) and tot_cred (numeric). Also the mapped values for the fields, dept_name and tot_cred are substituted into the substitution variables ?dept_name and ?tot_cred at runtime.Nested query example: <pre>SELECT * FROM (SELECT dept_name, SUM(tot_cred) AS total_credit FROM student GROUP BY dept_name) SUBS, department WHERE SUBS.dept_name = department.dept_name AND total_credit > 8000;</pre>
Fields	The grid is provided for informational purposes only.

Input

This tab contains the input schema. The fields that were selected in the **Input Settings** tab will be available in the schema. You can either hard code their values or map them to a field from the output of a preceding activity in the flow using the Mapper.

Output

This tab displays the activity output schema in a tree structure format. The output of an activity is displayed for informational purposes only and cannot be modified or altered. The information in this schema varies depending on the fields that you selected in the **Input Settings** tab.

The properties that are displayed in the **Output** tab schema correspond to the output of this activity and can be used as input by subsequent activities in the flow.

PostgreSQL Insert

Use this activity to execute an SQL Insert to insert the records into the database and return the information based on the returning clause specified in the insert query.



Settings

The **Settings** tab has the following fields:

Field	Description
Connection	Name of the PostgreSQL database connection from which to retrieve information. You can select the connection from the drop-down list.

Input Settings

The **Input Settings** tab has the following fields:

Field	Description
Insert	<p>An SQL statement used to insert a record in the table. You can construct prepared SQL queries by using substitution variables (or parameters) of the form ?<fieldname> in the query statement.</p> <p>Each substitution variable identifies an input parameter whose mapped value will be substituted into the substitution variable at runtime. You can reuse the substitution variable for the same input parameter elsewhere in the insert query. The type information for the input parameters used in the VALUES and RETURNING clause is fetched from the database using the selected connection for the entered insert query. Also based on the output parameters used in the RETURNING clause, the corresponding type information is fetched from the database using the selected connection for the entered insert query. Similarly, input and output fields in the Input and Output tabs of the activity are also populated based on the SQL Insert statement.</p> <div>  <p>Be sure to include the semicolon (;) at the end of a query. This activity expects an insert query to end with a semicolon to indicate the end of the query. A missing semicolon at the end of the query results in the query hanging.</p> </div> <div>  <p>When substitution variables used in the VALUES clause are identical to the column names in the table, those variables are displayed under the Values array in Input tab.</p> </div> <p>The following examples represent insert queries:</p> <ul style="list-style-type: none"> <pre>INSERT INTO products (product_no, name, price) VALUES (1, 'Cheese', ?price), (2, 'Juice', ?price), (3, 'Milk', ?price) returning (select name from instructor where name = ?name);</pre> <p>For the above insert query, output field is generated for name and input field is generated for price (NUMERIC) under Values[] node as its part of values clause, and name(VARCHAR) under parameters node as it is part of the parameter select sub-query. Also, the mapped value for the field price and name is substituted into the substitution variable ?price and ?name.</p> <pre>INSERT INTO products (product_no, name, price) VALUES (?product_no, ?name, ?price) returning price;</pre> <p>For the above insert query, output field is generated for price and input fields are generated for product_no (INTEGER), name(TEXT), and price(NUMERIC). Also, the mapped value for the field product_no, name, price is substituted into the substitution variables ?product_no, ?name, ?price. The parameters node in the input tab will not have mappings as there is no parameter in the insert query statement.</p>
Fields	The grid is provided for informational purposes only.

Input

This tab displays the input schema of the activity as a tree structure. The information in the schema varies based on the insert query provided. The fields that were selected in the **Input Settings** tab will be available in the schema. You can either hard code the values or map them to a field from the output of a preceding activity in the flow using the Mapper.

Fields from RETURNING clause are displayed under Parameters node and fields from VALUES clause are displayed under VALUES node in input schema.

Output

This tab displays the output schema of the activity as a tree structure. The output is read-only. The information in the schema varies based on the fields selected on the **Settings** tab. The properties that are displayed in the schema correspond to the output of this activity and can be used as input by subsequent activities in the flow.

PostgreSQL Delete

Use this activity to execute an SQL Delete to delete the record and return the information based on the returning clause specified in the delete query.


Settings

The **Settings** tab has the following fields:

Field	Description
Connection	Name of the PostgreSQL database connection from which to retrieve information. You can select the connection from the drop-down list.

Input Settings

The **Input Settings** tab has the following fields:

Field	Description
Delete	<p>An SQL statement used to delete the record from the table. You can construct prepared SQL queries by using substitution variables (or parameters) of the form ?<fieldname> in the query statement.</p> <p>Each substitution variable identifies an input parameter whose mapped value will be substituted into the substitution variable at runtime. You can reuse the substitution variable for the same input parameter elsewhere in the delete query. The type information for the input parameters used in the WHERE and RETURNING clause, is fetched from the database using the selected connection for the entered delete query. Also, output parameters used in the RETURNING clause and their corresponding type information is automatically fetched from the database using the selected connection for the entered delete query. Similarly, input and output fields in the Input and Output tabs of the activity are also automatically generated.</p> <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-top: 10px;">  Be sure to include the semicolon (;) at the end of a query. This activity expects a delete query to end with a semicolon to indicate the end of the query. A missing semicolon at the end of the query results in the query hanging. </div> <p>The following example represents a typical delete query:</p> <pre>Delete from student where name = ?name returning id;</pre> <p>For the above delete query, output field is generated for id from the returning clause and input field is generated for name (VARCHAR). Also the mapped value for the field name is substituted into the substitution variable ?name.</p>
Fields	The grid is provided for informational purposes only.

Input

This tab displays the input schema of the activity as a tree structure. The information in the schema varies based on the delete query provided. The fields that were selected in the **Input Settings** tab will be available in the schema. You can either hard code the values or map them to a field from the output of a preceding activity in the flow using the Mapper.

Output

This tab displays the output schema of the activity as a tree structure. The output is read-only. The information in the schema varies based on the fields selected on the **Settings** tab. The properties that are displayed in the schema correspond to the output of this activity and can be used as input by subsequent activities in the flow.

PostgreSQL Update

Use this activity to execute an update statement on one or more rows of an PostgreSQL table.


Configuration

The **Configuration** tab has the following fields.

Field	Description
Connection	Name of the PostgreSQL database connection from which to retrieve information. You can select a connection from the drop-down list.

Input Settings

The **Input Settings** tab has the following fields:

Field	Description
Update Statement	<p>An SQL statement used to update one or more rows in a table. You can construct prepared SQL queries by using substitution variables (or parameters) of the form ?<fieldname> in the query statement.</p> <p>Each substitution variable identifies an input parameter whose mapped value will be substituted at runtime. The type information for the input parameters is fetched from the database using the selected connection for the entered query. Similarly, input fields in the Input tab of the activity are also populated based on the query.</p> <div style="border-left: 1px solid #0070c0; padding-left: 10px; margin-top: 10px;">  Be sure to include the semicolon (;) at the end of the query. This activity expects an update query to end with a semicolon to indicate the end of the query. The metadata for the tables in the query are not fetched until the statement is completed with a semicolon. </div> <p>The following example represents a typical update query:</p> <pre>UPDATE pet SET Name=?Name , Color = ?Color WHERE Species = 'cat';</pre> <p>For the above query all rows in the 'pet' table will have their Name and Color columns set to the values provided on the Input tab where the species is cat. As usual all rows satisfying the where clause will be updated. If the where clause is omitted then ALL rows are updated. The limit clause can also be used to control this behavior.</p> <p>It is also possible to use a subquery to derive the values to be used in the update. The subquery format follows the general form:</p> <pre>UPDATE table_name SET column1 = ?column1, column2 = ?column2..., columnN = ?columnN WHERE [condition];</pre> <p>In this case the parameters supplied are used in the select query that provides replacement values in the outer update statement. The where clause could also be parameterized.</p>
Fields	The grid is provided for informational purpose only and documents the fields that will be made available as parameters on the input tab.

Input

This tab displays the input schema of the activity in a tree structure format. The information in the schema varies based on the update query provided. The fields that were selected in the **Input Settings** tab will be available in the schema. You can either hard code the values or map them to a field from the output of a preceding activity in the flow using the Mapper.

Substitution parameters from the query statement will be provided in the Parameters node.

Output

The Update activity returns the number of rows affected by the query.