

Data – PostgreSQL V12

user guide V0.1



Contents

[1. Introduction 3](#_Toc26192121)

[2. Prerequisites 3](#_Toc26192122)

[3. Configuration 4](#_Toc26192123)

[4. Environment Variables 4](#_Toc26192124)

[5. Using the Business Object 5](#_Toc26192125)

[5.1. Configure 5](#_Toc26192126)

[5.2. Set Connection 5](#_Toc26192127)

[5.3. Begin Transaction 5](#_Toc26192128)

[5.4. Commit Transaction 6](#_Toc26192129)

[5.5. Rollback Transaction 6](#_Toc26192130)

[5.6. Execute 6](#_Toc26192131)

[5.7. Get Number 7](#_Toc26192132)

[5.8. Get Text 7](#_Toc26192133)

[5.9. Get Collection 8](#_Toc26192134)

[5.10. Get CSV File 8](#_Toc26192135)

[5.11. Get CSV 9](#_Toc26192136)

[5.12. Delete Rows - Where 9](#_Toc26192137)

[5.13. Drop Table 10](#_Toc26192138)

[5.14. Truncate Table 10](#_Toc26192139)

[6. Support 11](#_Toc26192140)

[7. Functional Tests 11](#_Toc26192141)

[8. Troubleshooting Guidelines 11](#_Toc26192142)

[9. Frequently Asked Questions 11](#_Toc26192143)

The information contained in this document is the proprietary and confidential information of Blue Prism Limited and should not be disclosed to a third party without the written consent of an authorised Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Limited.

**© Blue Prism Limited, 2001 – 2019**®Blue Prism is a registered trademark of Blue Prism Limited

All trademarks are hereby acknowledged and are used to the benefit of their respective owners.  
Blue Prism is not responsible for the content of external websites referenced by this document.

Blue Prism Limited, Centrix House, Crow Lane East, Newton-le-Willows, WA12 9UY, United Kingdom  
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: [www.blueprism.com](file:///C:\Users\adutton\Documents\Rebranding\Templates\www.blueprism.com)

# Introduction

This asset is designed to provide connectivity from Blue Prism to the PostgreSQL V12 RDBMS using the npgsql managed data access library. This VBO is a code wrapper around this library and this means that the stages have C# code that undertakes the actual work. Unless you wish to modify or add functionality to the VBO C# coding experience is not needed to make use of this VBO. Please note that we are using version 4.0.9 of the library. This is deliberate. The more current versions have issues with dependencies and will not work correctly with Blue Prism. We hope to be able to update this in due course.

# Prerequisites

To use this asset, you require a licenced Blue Prism installation or trial. More information on Blue Prism can be found here.

<https://www.blueprism.com>

You will also require the Managed Data Access provider mentioned previously. The file that is required is **npgsql.dll**

This is available from our GitHub repository at <https://github.com/blue-prism/PostgreSQL-V12>. Here you will find the release file and documentation. Please note that the npgsql.dll, is a .net library and experience with coding in .net is not a prerequisite for the usage of this VBO, however understanding of SQL and knowledge of your database schema will be needed.

The asset is created as a *bprelease* file, so that it may be reviewed and edited as you find necessary. It is maintained on a best endeavours’ basis by *Blue Prism* however, user feedback is appreciated, and this can be in relation to issues or errors found and feature requests.

# Configuration

To use this asset, import the ***.bprelease*** file which contains the following:

* Data – PostgreSQL 12
* Blue Prism – PostgreSQL 12 Test

Now the npgsql.dll file works best if it is installed in the *GAC* (Global Assembly Cache) this is a location where applications that are based on the .NET framework can store dependent modules so that all applications can make use of them. To do this, we need to install from an msi installer file. This file can be found here <https://github.com/npgsql/npgsql/releases/tag/v4.0.9>

It will also be necessary to ensure that the Blue Prism development environment has access to this file for the execution of the stages the VBO contains. These areas can be found under the properties for the note block in the Initialize page.

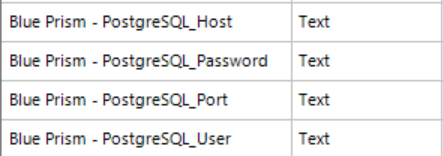


You will need to add a reference and navigate to the path shown above. Please note, your windows installation may be on another drive.

# Environment Variables

In order to simplify the usage of the asset, we have made available some environment variables that can be used in your processes.

These can be found in the System Tab, in the Processes/Objects group.



There is a further variable, Database, that is a requirement for a *connectionstring*, this has not been set as an environment variable as it may be a requirement that multiple databases need to be addressed from a process.

# Using the Business Object

The Visual Business Object contains the following actions:

## Configure

This action will allow a flag to be set to control the reporting of exceptions. Useful during development of a process.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| RaiseExceptions | This is a flag that sets whether exceptions are reported. | Flag |

## Set Connection

Create the connection string that is necessary to connect to the PostgreSQL database.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Server | This is the IP address of the Postgres server. | Text |
| Port | This is the listening port that the client is expecting to connect to. (Default 5432) | Text |
| User | The username by which you will access the Postgres database system | Text |
| Password | The supplied password by which you will access the Postgres database system. | Password |

This method sets the ConnectionString in the actual process. This connectionstring is used by other actions.

## Begin Transaction

This stage allows a set of database interactions to be encapsulated in a transaction. This allows a set of interactions to be completed or aborted if conditions are not met. Begin Transaction is followed by a Commit Transaction or a Rollback Transaction.

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The flag indicating if the transaction was started correctly. | Flag |
| Message | The text message that holds any error information that may have occurred. | Text |

## Commit Transaction

This stage compliments the Begin Transaction stage and makes all the changes made to the database since the transaction began permanent.

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The flag indicating if the transaction was committed correctly | Flag |
| Message | The text message that holds any error information that may have occurred. | Text |

## Rollback Transaction

This stage compliments the Begin/Commit Transaction as it allows the database to be returned to the stage prior to the commencement of the Begin Transaction. This would be the case should appropriate conditions or an exception occur.

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The flag indicating if the transaction was rolled back correctly | Flag |
| Message | The text message that holds any error information that may have occurred | Message |

## Execute

This stage takes an SQL command and passes it to the database.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Query | A valid PostgreSQL command script. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The flag indicating if the command was successful. | Flag |
| Message | The text message that holds any error information that may have occurred | Text |

## Get Number

This function executes an SQL script and returns a single numeric value.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Query | A valid Postgres command script. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Result | The value returned from the supplied SQL query. | Number |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Test |

## Get Text

This function executes an SQL script and returns a text result.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Query | A valid PostgreSQL command script. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Result | The value returned from the supplied SQL query. | Text |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Get Collection

This function executes a SQL *Select* statement and returns the resulting records in a Blue Prism collection data item.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| SelectQuery | A valid Postgres select command script | Number |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Result | The resulting collection data item containing the results of the select query. | Collection |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Get CSV File

This function takes a valid SQL Select query and returns the results to a CSV (Comma Separated Variable) file.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| SQL | A valid PostgreSQL select command script. | Text |
| File | The destination filename for where the results will be written to. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Get CSV

This function takes a valid SQL Select query and returns the results to a string in comma separated format.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| SQL | A valid PostgreSQL select command script. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| CSV | The text value containing the resulting data. | Text |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Delete Rows - Where

This function will delete a set of records from a table where the where clause is satisfied. This function requires some database schema knowledge.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| TableToDeleteFrom | The table where the rows are to be removed from. | Text |
| WhereClause | The Boolean clause that must be satisfied. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Drop Table

This function takes table name and executes an SQL statement to drop the table. This is a destructive action. This function requires some database schema knowledge.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| TableToDrop | The table that is to be removed from the database schema. | Number |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

## Truncate Table

This removes the complete set of rows in a table. This function requires some database schema knowledge.

Inputs:

| Name | Description | Data Type |
| --- | --- | --- |
| TableToTruncate | The table that is to have all its rows removed. | Text |

Outputs:

| Name | Description | Data Type |
| --- | --- | --- |
| Success | The value to explain if the query was successful. | Flag |
| Message | The exception message, if the query failed. | Text |

# Support

Support for this skill is provided via the Blue Prism Digital Exchange Community Forum. Post your questions here:

[Digital Exchange Community Forum](https://community.blueprism.com/communities/community-home?communitykey=1e516cfe-4d1f-4de9-a9eb-58d15bf38c81&tab=groupdetails)

# Functional Tests

There is a simple test process for each of the functions however the user should satisfy themselves that the results are both correct and satisfactory for their needs. Usage of these functions reflects that satisfaction. To make use of these tests, locate the PostgreSQL Test process and open it in process studio. Enter your values in the appropriate areas or make use of the provided environment variables. Your parameter areas are in a yellow block. Click on the start stage and then right click your mouse, from the drop-down choose *Set Next Stage,* the from the toolbar click the green triangle to run it.

# Troubleshooting Guidelines

There are no known commonly encountered issues at this stage, or corresponding resolutions for them. If users begin encountering issues, then this section will be updated with known resolutions.

It should be noted that this asset was built against the standard PostgreSQL download. It has not been tested against any other version neither hosted or cloud.

# Frequently Asked Questions

There are no frequently asked questions at this stage.