blueprism®

6.7 User Guide

Data Gateways

Document Revision: 1.1

Trademarks and copyrights

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom.

Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

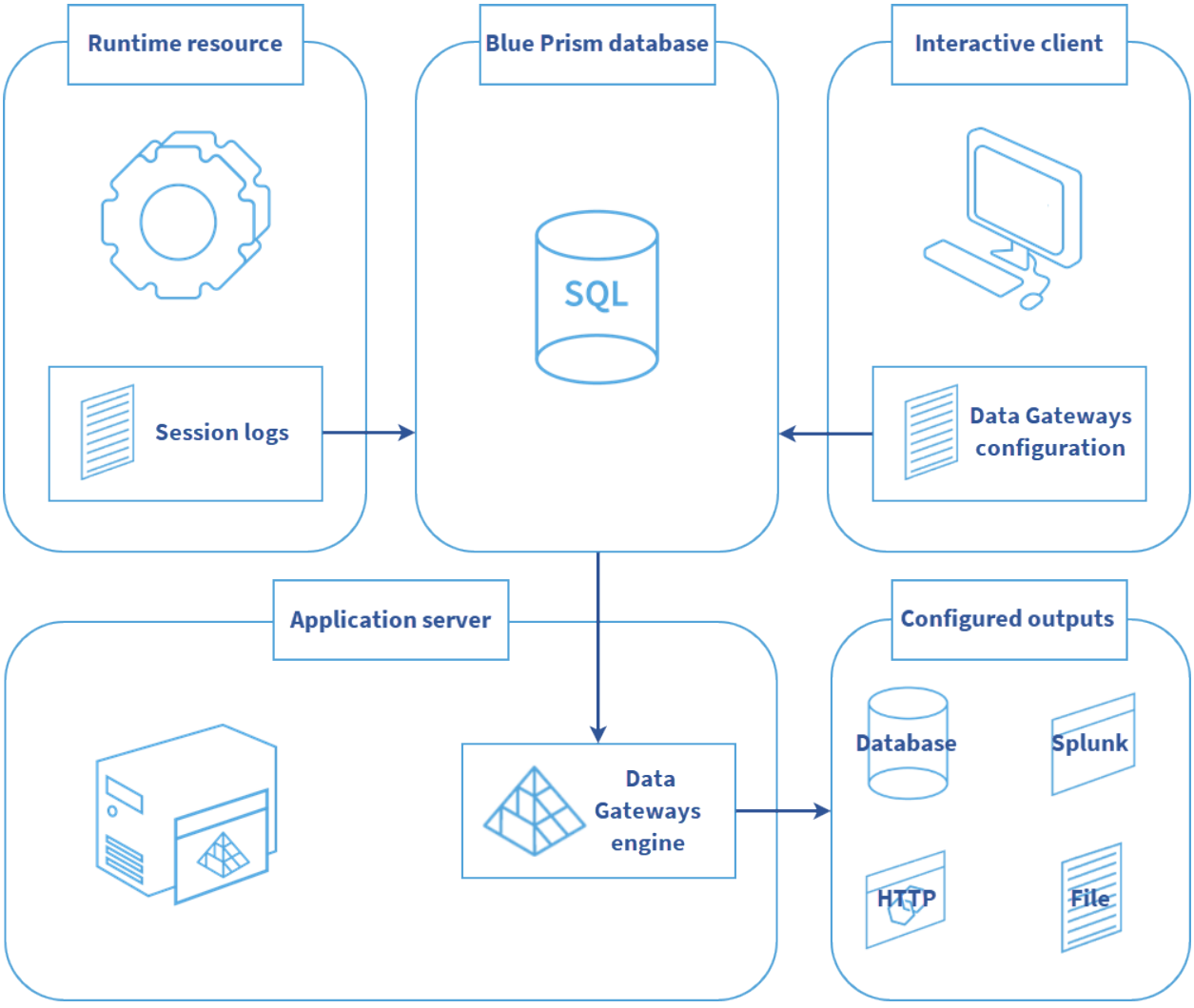
About Data Gateways

Data Gateways provides an easy-to-use, centralized method to push data out of Blue Prism for use in external systems for monitoring and reporting, long-term data storage, and to feed machine learning models. Advanced configuration methods allow data to be directed to any required target. The data can be visualized and analyzed to provide valuable insight about Blue Prism environments without having to manually build similar capabilities into each relevant process automation.

By providing the ability to store data outside the Blue Prism database, organizations can use Data Gateways to support flexible data storage requirements. For example, an organization might want to save all their session data outside the Blue Prism database, or they might choose to store data in the database for a shorter period of time, pushing out a copy of that data for longer term data storage.

Settings are applied to determine what data will be processed by the Data Gateways engine, and a configuration defines the outputs to which data will be pushed. Data from session logs, published dashboards, process stages, and work queue analysis can be sent to a variety of external outputs - HTTP endpoints, external databases, third-party analysis tools, and flat files - providing flexibility and control over data analytics and storage.

The Data Gateways engine utilizes a number of Logstash components to send session log data to configured outputs.



Prerequisites

The following conditions must be met before configuring Data Gateways:

- The Blue Prism version must be version 6.6 Enterprise Edition or above.
- The SQL server instance on which the Blue Prism database is located must be configured to allow the Data Gateways engine to connect to it. This requires:
  - The TCP/IP protocol to be enabled and set to *Listen All* on a static port.
  - Server authentication to be configured to accept both SQL Server and Windows authentication modes. Data Gateways requires a login, created in SQL Server, to access the Blue Prism database.
- Blue Prism must be configured to use an encryption scheme which is stored on the server. It is therefore recommended that the encryption scheme configured in the Server Configuration utility (BPServer.exe) is used on each application server that is required to run Data Gateways.
- The user account that runs the Blue Prism application server process must have read and write permissions so that the Logstash service can write to the bpconf.conf file (found in the same directory as Automate.config).

Setup overview

Configuring Data Gateways requires the following steps:

1. Install the Data Gateways engine components

   The Data Gateways engine required a number of components to be installed in the environment that all application servers running a Data Gateways process are in.

2. Enable the Data Gateways process

   The Data Gateways process must be enabled and the port for communication with the Data Gateways process selected in the Server Configuration Details for each application server that is required to run Data Gateways.

3. Create a Data Gateways SQL login and user

   Data Gateways requires a SQL login and user to be created against the server instance on which the Blue Prism databases are located.

4. Add a Data Gateways credential that holds the SQL server user credentials

   A Blue Prism credential is required to allow secure access to the Blue Prism database from the Data Gateways engine.

5. Configure the Blue Prism Data Gateways settings

   These settings determine what data will be processed by the Data Gateways engine.

6. Create a Data Gateways configuration

   Outputs in the Data Gateways configuration define where data will be sent. This could be a flat file, HTTP endpoint, Splunk instance, or database.

Environment setup

Install Data Gateways engine components

Data Gateways requires a number of components to be installed in the environment in which all application servers running a Data Gateways process will be used. If an application server hosts multiple environments, the components must be installed in each environment that will be running the Logstash service.

Version 1.1 of the Data Gateways engine must be installed for use with Blue Prism version 6.7. The components are included in a separate installer, *Blue Prism Data Gateways Components*, available from the Blue Prism portal, see https://portal.blueprism.com/product/extras.

Enable the Data Gateways process

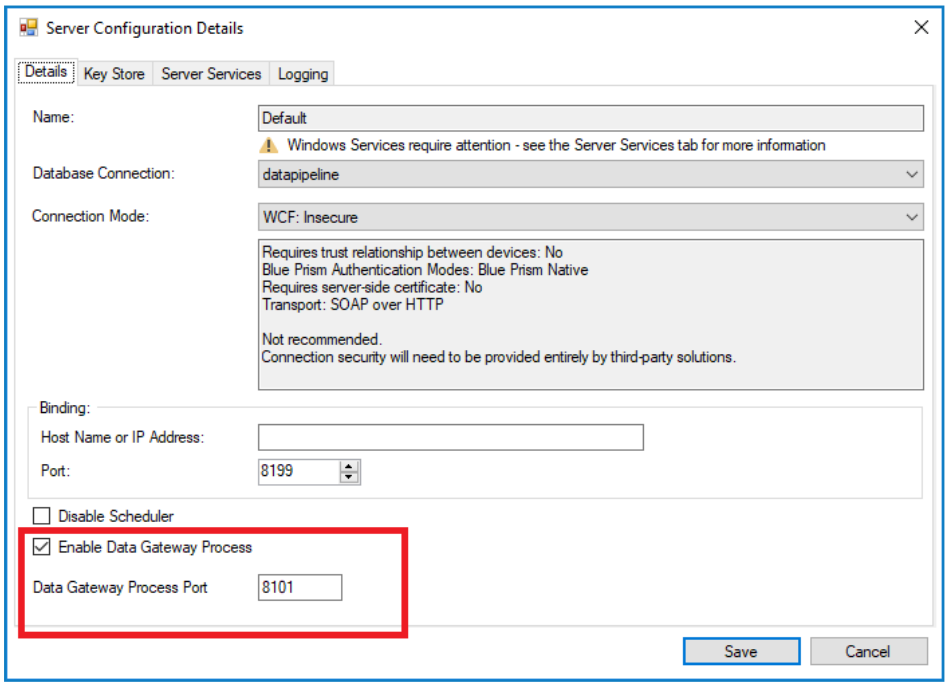
The Data Gateways process must be enabled and the associated port defined on all application servers that are required to run Data Gateways. The Data Gateways process can only run on one instance on a single device. Where multiple application servers are configured on a single device, the Data Gateways process must be enabled on each one.

Ensure all prerequisites have been met before enabling the Data Gateways process. See Prerequisites for details.

1. Access the Server Configuration Details screen and enter the port that the application server will listen on for commands to start or stop the Data Gateways process.

   This can be left as the default port (8101) unless there is another process on the application server that is already using that port.

   The application server must be stopped before changing the port number.



2. Click **Save** to apply the settings.
3. Start the Blue Prism Server Service from the Services menu.

To confirm that the service has started successfully, check the event viewer.

Create a Data Gateways SQL login and user

Run the following SQL query against the server instance on which your Blue Prism databases are located. This query creates a login and user that allows Data Gateways to access the server.

> Only users with ALTER ANY LOGIN permission on the server or membership to the securityadmin fixed server role can create logins.

Before running the query, select the required Blue Prism database and replace *password* with a complex password for the Data Gateways login.

```
CREATE LOGIN [Data Gateways BP Database SQL User] WITH PASSWORD = 'password';
CREATE USER [Data Gateways BP Database SQL User] FOR LOGIN [Data Gateways BP Database SQL User];
GO
sp_addrolemember 'BPA_DataGatewaysEngine', 'Data Gateways BP Database SQL User'
GO
```

Blue Prism setup

Data Gateways credentials

Data Gateways credentials can be used in any Data Gateways configuration that requires authenticated access to a database or HTTP endpoint. In addition to this, a Data Gateways credential must be created to store the SQL server user account that provides the Data Gateways engine with access to the database.

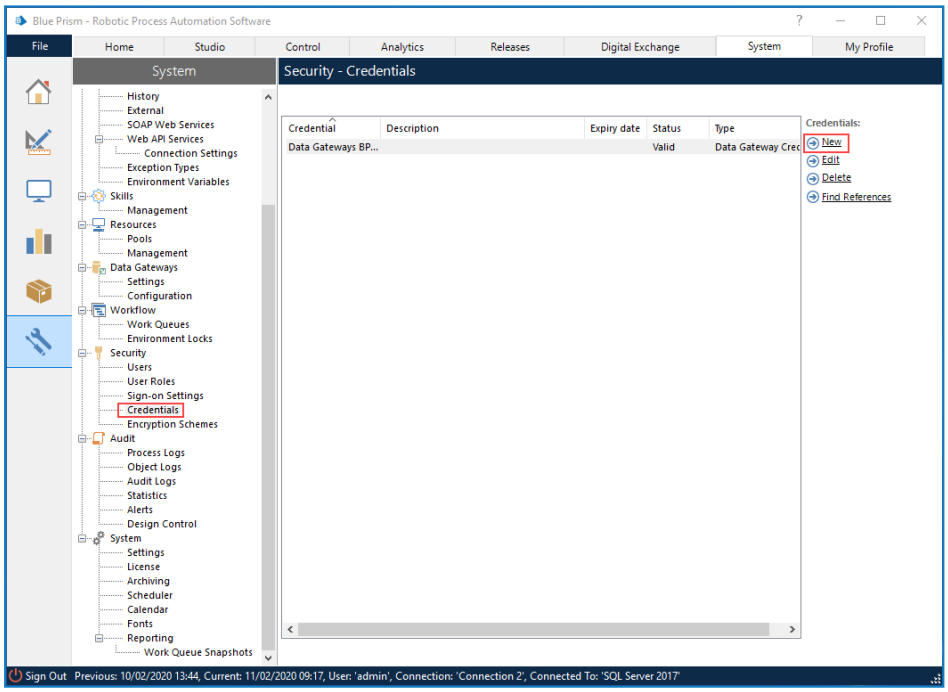
Credentials that use the Data Gateways credential type are only accessible by the application server when generating Data Gateway configurations - they cannot be used in standard Blue Prism processes or other Blue Prism elements such as web API definitions.

Add a credential for the SQL server user

Create a credential to store the SQL Server user account required for secure access to the Blue Prism database.

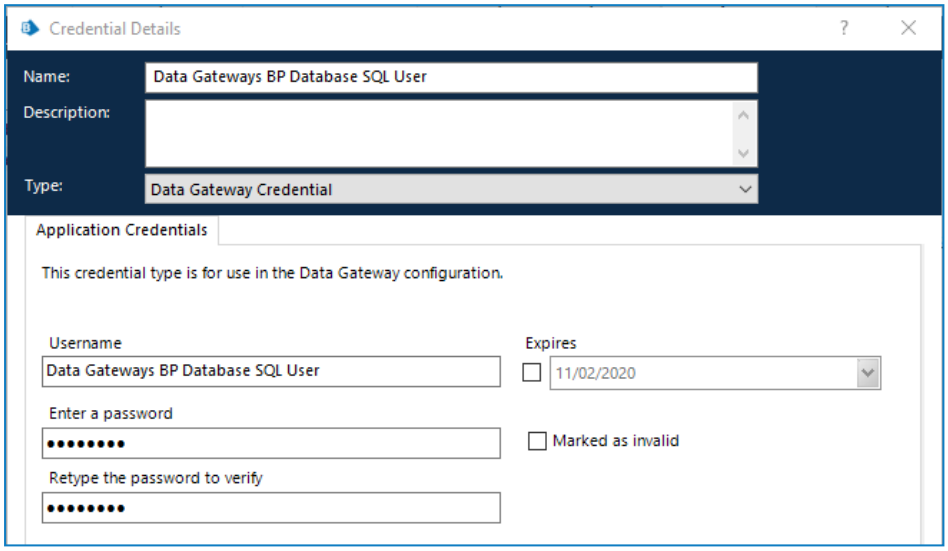
The credential name, username, and password must exactly match those specified below, mirroring the details entered for the SQL Server login.

1. Select the **System** tab.
2. Select **Security** > **Credentials**.



3. Click **New**. The Credential Details window displays.

4. Configure the following credential:
   - **Name** – *Data Gateways BP Database SQL User*
   - **Type** – Data Gateway Credential
   - **Username** – *Data Gateways BP Database SQL User*
   - **Password** – The password that is specified for the SQL Server user.

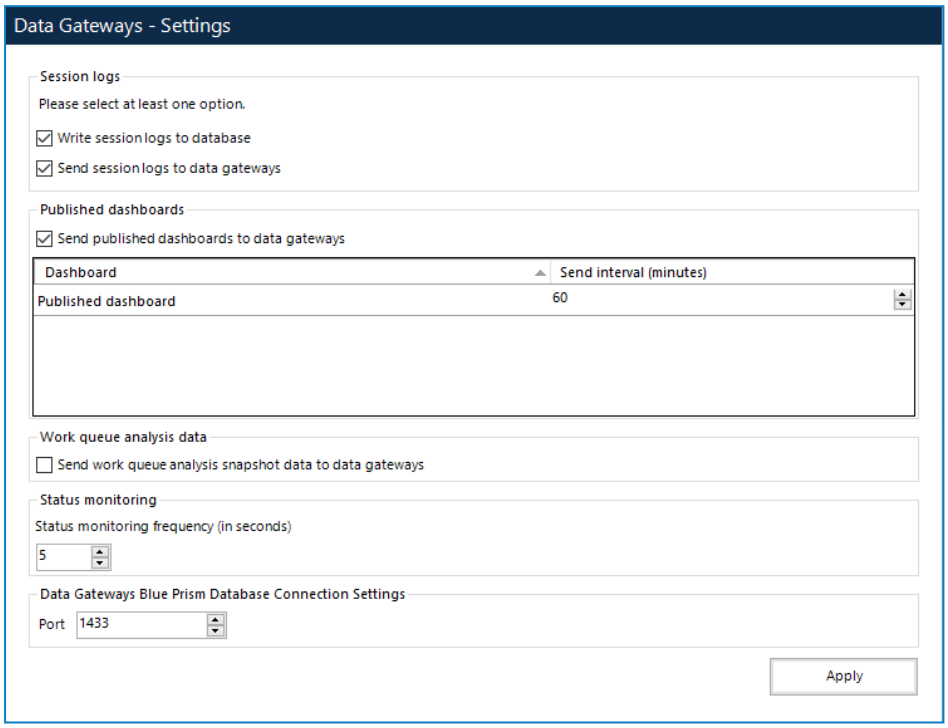


5. Click **OK** to save the credential.

Configure Data Gateways settings

Configure the Data Gateways engine by determining what data will be processed and where relevant, the frequency at which data is sent.

1. Select the **System** tab.
2. Select **Data Gateways** > **Settings**.



3. Select the required options to determine where session log data will be stored - at least one option must be selected:
   - **Write session logs to database** - Session logs will be sent to the session log table in the Blue Prism database. If not selected, the functionality to view new session logs in the Blue Prism client will be unavailable.
   - **Send session logs to data gateways** - Session logs will be sent to a temporary storage table on the Blue Prism database where they are accessed by the Data Gateways engine for use in the configured outputs.

     If the specified HTTP, Splunk, or Database Data Gateway endpoint is not reachable when the data is being processed, the data is stored temporarily until the endpoint becomes available. See Data Storage when endpoint unavailable for details.
4. Select **Send published dashboards to data gateways** to send data from published dashboards to a database table in the Data Gateways system and set the frequency that data is sent for each dashboard.
5. Select **Send work queue analysis snapshot data to data gateways** to send work queue analysis data to the database.

6. Set the **Status monitoring frequency** to a value between 5 and 3600 seconds. This determines how often the Data Gateways screen in Control Room screen is refreshed.

   For more information about Blue Prism dashboards, see the *Dashboards Overview* topic in the in-product help.

7. Update the default Port used for the Data Gateways database connection settings, if required. By default, the Data Gateways process uses port 1433.
8. Click **Apply** to save the settings.

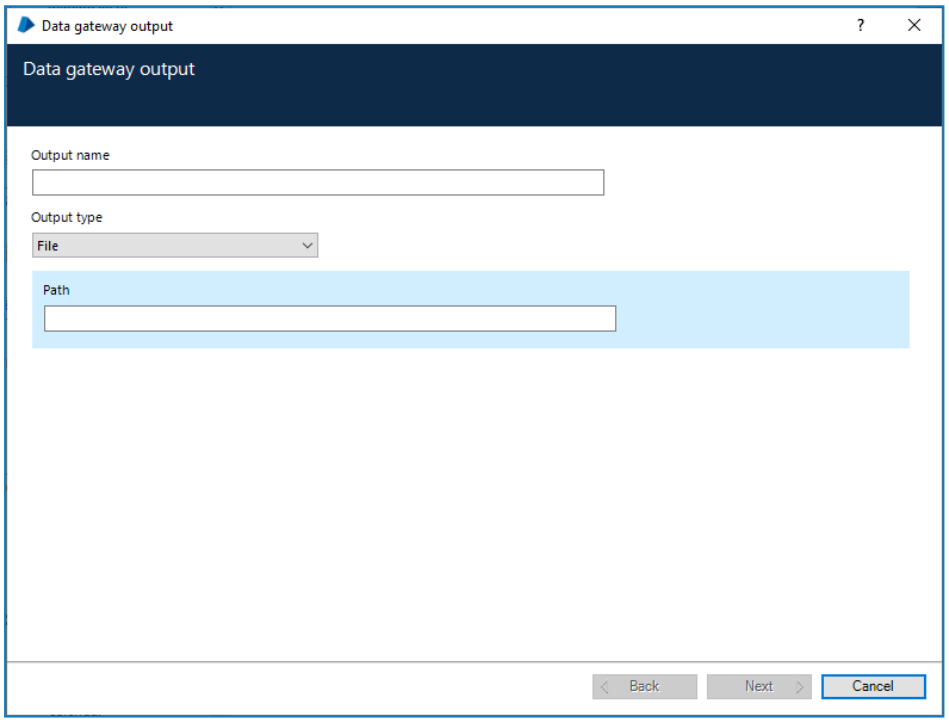
Data Gateway configurations

A Data Gateways configuration is a collection of outputs that define where data from session logs, published dashboards, and/or custom objects is sent. For each output, data can be sent to a file, HTTP endpoint, Splunk instance, or a database.

Add an output to a Data Gateways configuration

A Data Gateways configuration can consist of any number of individual outputs. The data from each of the outputs is added to a single configuration file.

1. Click the **System** tab and select **Data Gateways** > **Configuration**.
2. Click **Add new gateway output**. The Data gateway output wizard displays.

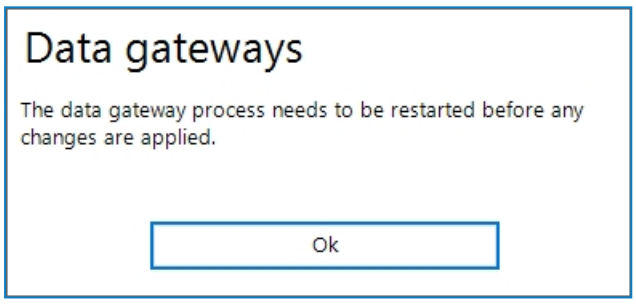


3. Enter a unique name for the output.
4. Select an output type, File, HTTP endpoint, Splunk, or Database and complete the fields for that output type.
5. Click **Next**.
6. Select the data to send to the configuration file for the specified output type:
   - **Session logs** – Data Gateways will process session log data for the specified output type.
   - **Published dashboards** – Data Gateways will process data from the configured published dashboards for the specified output type.
   - **Work queue analysis snapshot data** – Data Gateways will process the work queue analysis data for the specified output type.
   - **Custom object data** – Data Gateways will process data from any Blue Prism action configured to use the Data Gateways internal business object.

7. Click **Next**. A preview of the output data displays.

The data in the produced output can be edited directly by clicking **Edit advanced output**. For more information about advanced configurations, see Advanced outputs.

8. Click **Finish** to save the output to the configuration. A Data gateways message displays.



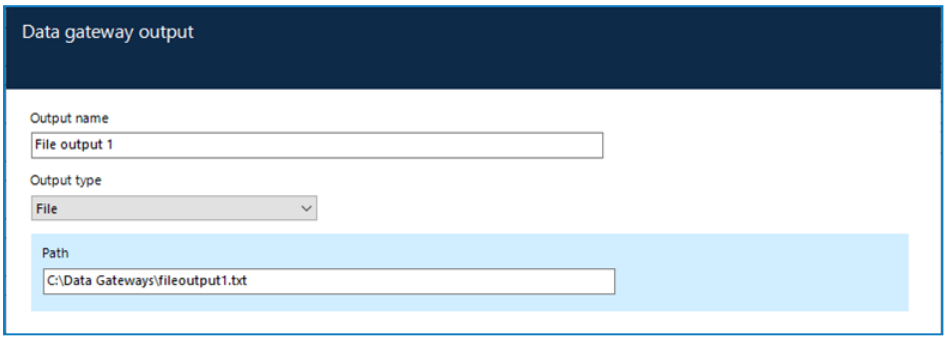
9. Click **OK** to close the message. For more information on restarting the data gateway process see Start and stop the Data Gateways engine.

Output types

File

Export the selected data to the specified file. Although creating outputs in the Data Gateways Configuration wizard only supports sending data to .txt files, advanced outputs and custom configurations can be edited to support other file formats, such as .csv.

For details about how to achieve this, see Configuration file structure.



Dynamic file paths can be used to include the creation date in filenames using variables for year, month, day, and time.

For example: C:\Session Logs\session_logs_%{+YYYY-MM-dd}.txt

This will create a new text file each day in the specified location, that includes the current date in the filename: session_logs_2019-04-30.txt

The variables can also be used in file paths to create date-based folder names. For example:

C:\Session Logs\%{+YYYY}\%{+MM}\%{+dd}\session_logs_%{+YYYY-MM-dd-hh}.txt

This will create folders based on the current date and append the filename. For example:

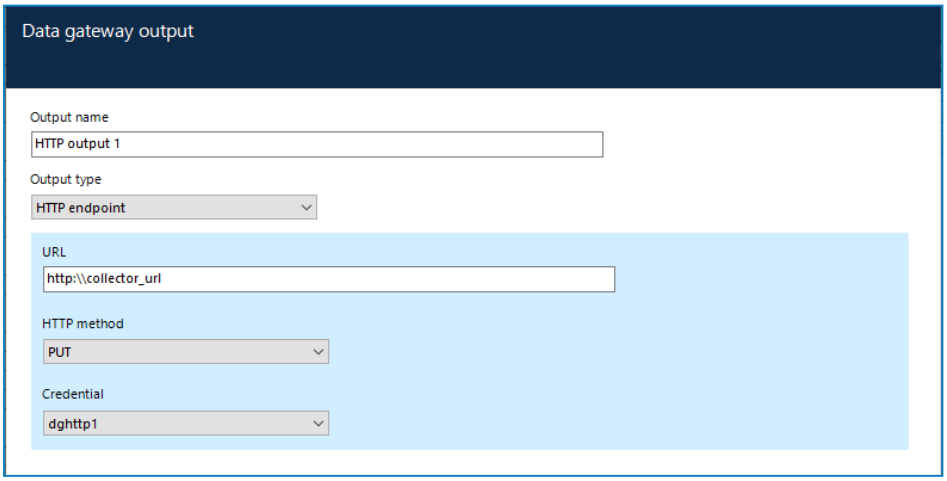
C:\Session Logs\2019\04\30\session_logs_2019-04-30-11.txt

The file/folder timestamp generated when using dynamic file paths is in UTC format.

Dynamic file paths can be used in configurations created using the data gateways wizard and advanced configuration.

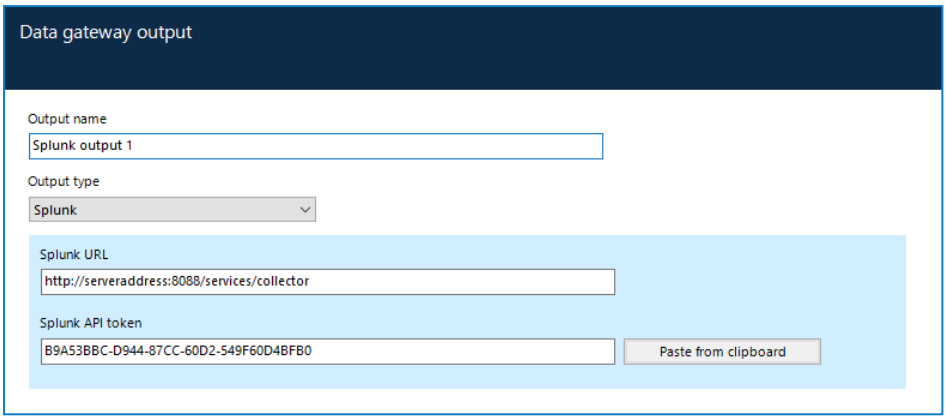
HTTP endpoint

Route data through the specified HTTP endpoint using the selected method. Select a Data Gateways credential if authentication is required to access to the endpoint.



Splunk

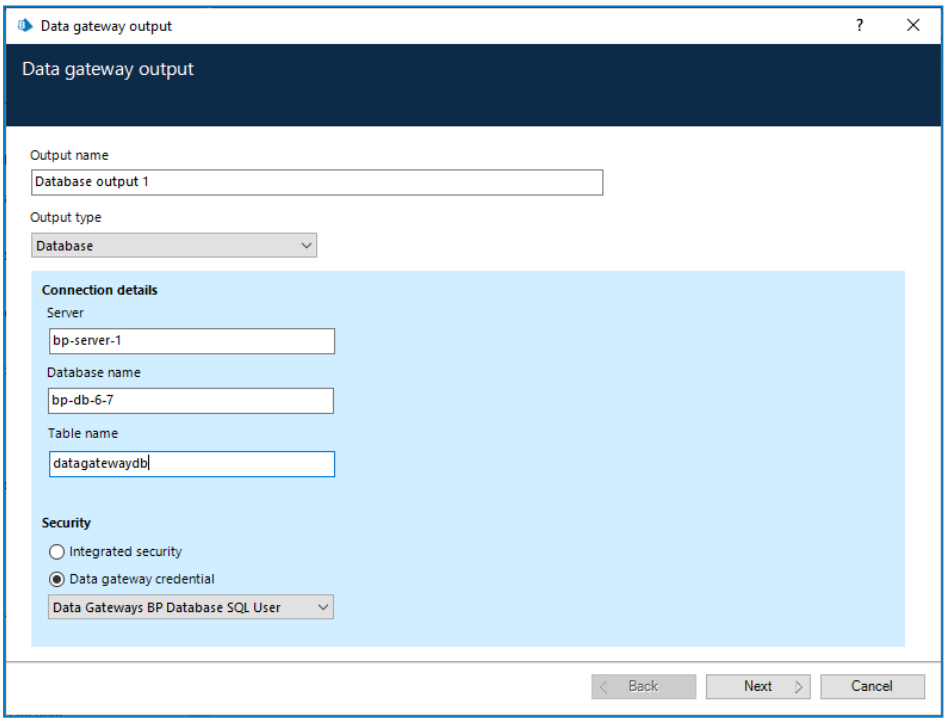
Send the selected data to the specified Splunk URL using the associated API token.



To successfully configure a Splunk output, the HTTP event collector must be enabled in your Splunk configuration. For further details, see https://docs.splunk.com/Documentation/Splunk/7.3.1/Data/UsetheHTTPEventCollector.

Database

Send the selected data to the specified SQL server database table. Select a Data Gateways credential if authentication is required to access to the database.

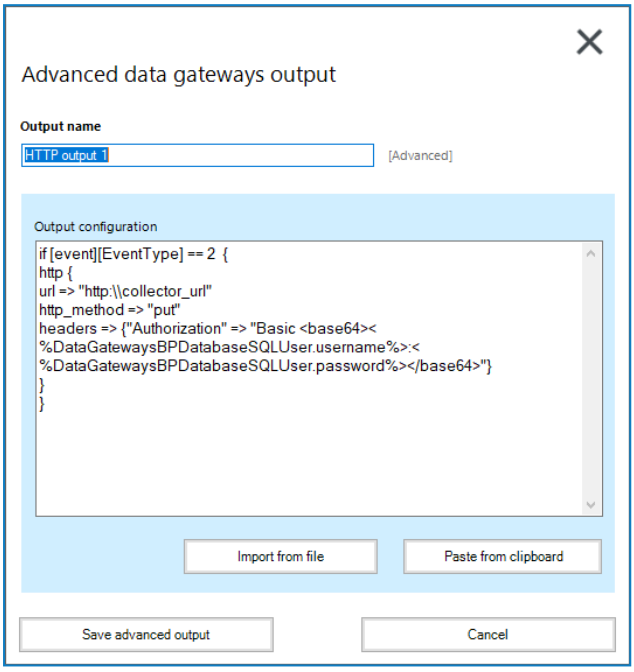


To use a port, other than the default port, the server name can be appended with an alternative port number, separated by a colon. For example, *bp-server-1:8001*.

Advanced outputs

When creating or editing an output, users with the *Data Gateways – Advanced Configuration* permission can edit the code in the output directly to create an advanced configuration.

From the output configuration preview screen of the output wizard, click **Edit advanced output** to open the advanced editor.

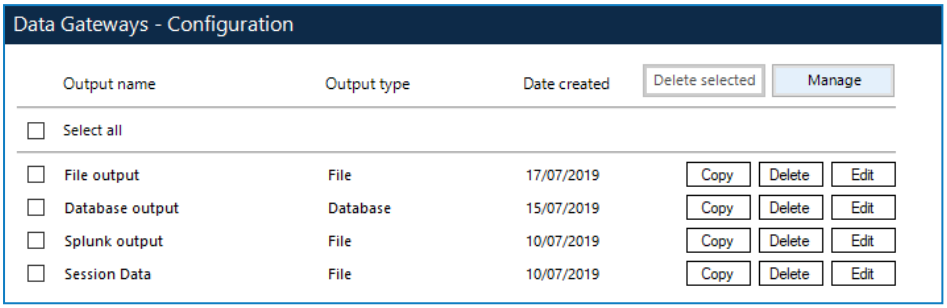


The output can be edited by entering data or pasting it from the clipboard. Files up to 10k in size can also be imported into an output, replacing all current data. When the required changes have been made, click **Save advanced output**.

Once an output has been saved as an advanced output, even if no changes have been made, it can no longer be configured in the Data Gateway output wizard – only in the advanced editor.

Manage a Data Gateways configuration

The outputs for a Data Gateways configuration are listed in the **Data Gateways** > **Configuration** screen. Click **Manage** to display the options to copy, edit, or delete one or more configurations.



Every time a Data Gateways configuration is changed, whether that is new, updated, or deleted output, the Data Gateways engine must be restarted. For more information, see Start and stop the Data Gateways engine.

Copy an output

Click **Copy** for the output you want to duplicate. Depending on the output type, the Data Gateways wizard or advanced editor will open enabling the duplicated output to be updated as required. When the output has been saved, restart the Data Gateways process.

Delete an output

Click **Delete** for the required output to remove it from the configuration and restart the Data Gateways process.

Edit an output

Click **Edit** for the required output. Depending on the output the output wizard or advanced editor displays allowing the output to be updated. When the output has been saved, restart the Data Gateways process.

Delete selected output

Click the **Select all** check box in the Output name column to select all of the outputs for group deletion. Select/deselect the check box for a specific output, as required. Click **Delete selected** to delete all selected output(s).

Custom configurations

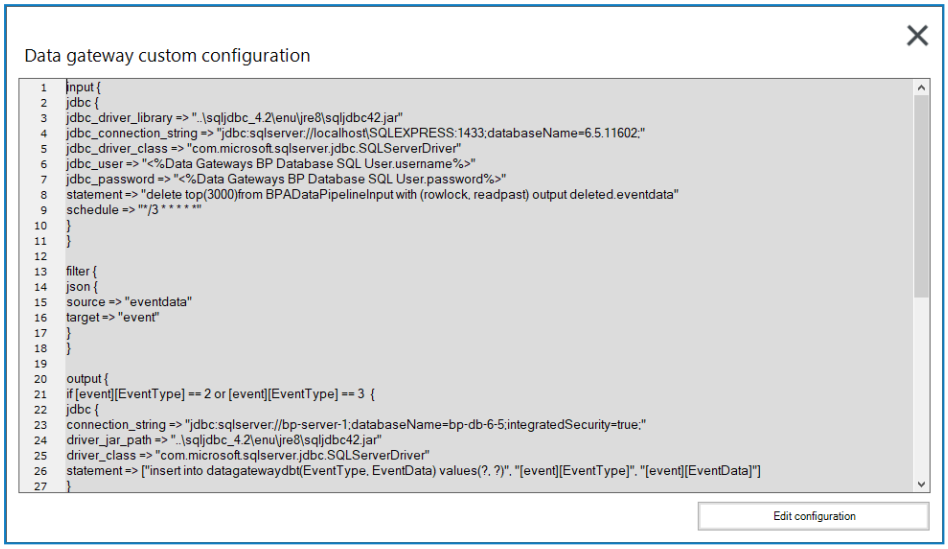
Users with the *Data Gateways – Advanced Configuration* permission can create a custom configuration by directly editing the underlying code. This allows users to create a configuration that has functionality beyond what is available through the wizard such as the ability to only send specific event fields to an output or customizing the format of the data that is sent to an output.

For further information about the composition of configuration files, see Configuration file structure.

Create a custom configuration

1. From the *Data Gateways – Advanced Configuration* click **View configuration**

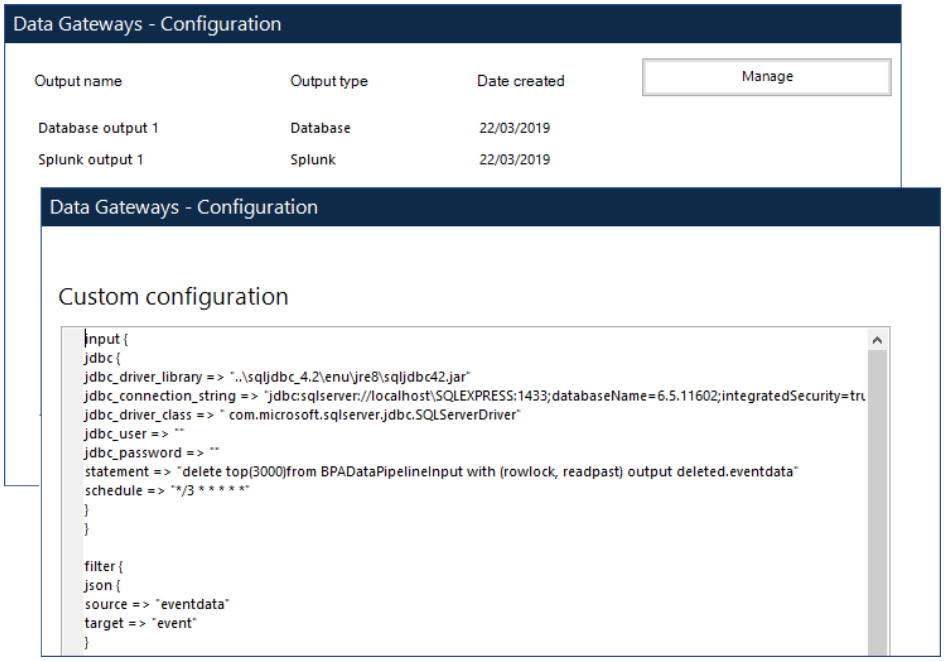
   The configuration code displays.



2. Click **Edit configuration** to make the configuration code editable.
3. When the required updates have been made, click **Save** and confirm.

> The Data Gateways engine must be restarted for the changes to take effect. For more information, see Start and stop the Data Gateways engine.

The custom configuration overwrites the outputs in the existing one. This is reflected in the Data Gateways screen, where the advanced configuration code replaces the list of outputs.



Click **Delete** in a custom configuration at any time to revert the configuration to its original composition and display the original outputs.

Manage Data Gateways processes in control room

Data Gateways processes are listed in the control room from where the status can be viewed and the process restarted.

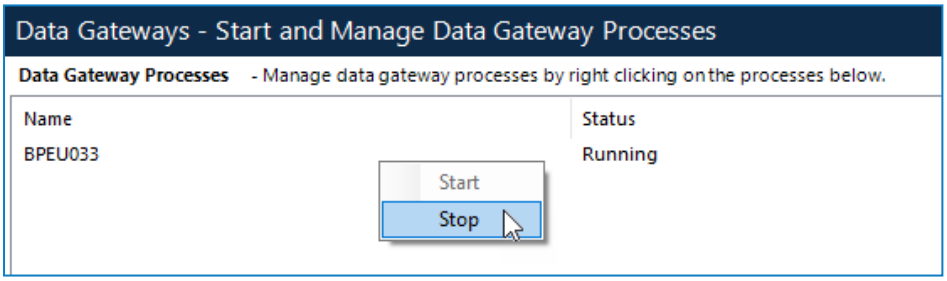
Select the **Control** tab and click **Data Gateways** to display all the Data Gateways processes in the environment.

Processes can be in the following states:

- **Online** - The application server hosting the Data Gateways process is online.
- **Offline** - The application server hosting the Data Gateways process is offline.
- **Starting** - The Data Gateways process is starting up.
- **Running** - The Data Gateways process is processing events.
- **Error** - The Data Gateways process has encountered an error. It will attempt to restart periodically to rectify the error.
- **Unrecoverable Error** - The Data Gateways process has encountered an error that restarting will not fix. It will not attempt to restart.

Start and stop the Data Gateways engine

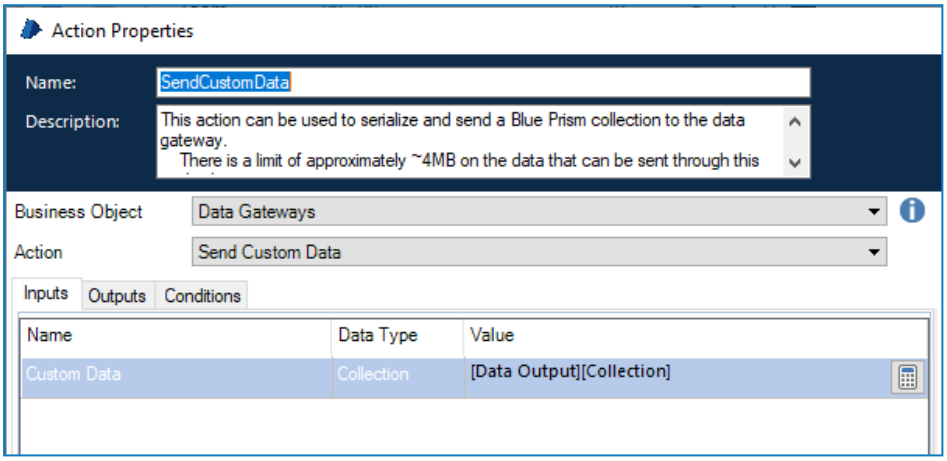
Right-click on a process to open the context menu. Depending on the current state of the process, select **Start** or **Stop** as required.



Data Gateways VBO

The Data Gateways internal business object can be used in action stages in any process and configured to send data from collections to the Data Gateways engine as the process runs. Any data that can be put into a collection can be sent to the Data Gateways engine using the VBO.

The VBO has one action, *Send Custom Data* that accepts collections as inputs and sends the data from those collections to the Data Gateways engine. The data from all actions that use the VBO will be used by any output that is configured to send custom object data.



Configure user role permissions

There are three Data Gateways user permissions that can be granted to the appropriate administrator user roles:

- **Data Gateways – Configuration** – The user can configure Data Gateways settings and add and manage gateway outputs. They cannot create or edit advanced outputs or custom configurations.
- **Data Gateways – Advanced Configuration** – The user can configure Data Gateways settings and add and manage gateway outputs. They can create and edit advanced outputs and custom configurations. This permission should only be granted to expert users who have the knowledge to edit output and configuration code.
- **Data Gateways – Control Room** – The user has access to the Data Gateways node in the Control Room to check the status of the Data Gateways engine and start and stop the process as required.

These permissions are granted to the System Administrator for upgrades and new installations - they are not enabled for any other built-in user role.

Advanced setup

This section provides details of configuration and editing techniques that should only be performed by experienced users.

Configuration file structure

Configuration files are created from the configured Data Gateways settings and outputs. They are composed of three main sections, Input, Output, and Filter, as outlined below.

Input

This section is automatically generated by Blue Prism based on the Blue Prism database settings. It determines how events are pulled into the Data Gateways engine for processing. In the example below, they are retrieved from the BPADataPipelineInput table in the Blue Prism database.

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string => "jdbc:sqlserver://SQL_SERVER_INSTANCE:1433;databaseName=BP_
DATABASE;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3* * * * *"
}
}
```

If required, the following areas of the input can be edited to suit the required preferences.

`schedule => "*/3* * * * *"`

This determines how often the SQL query to request data from the BPADataPipelineInput table is executed. The default value of every three seconds can be updated by replacing `3` with the required value.

`statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast) output deleted.eventdata"`

This is the SQL statement that is executed against the Blue Prism database to pull events out of the BPADataPipelineInput table. The value controls the maximum number of rows pulled from the BPADatapipelineInput table every interval. The default value of 3000 can be changed if required.

Filter

Filters can be used to perform intermediary processing on an event. These could be actions such as adding, removing, or modifying certain fields of an event before they are sent to the outputs, for example, removing the AttributeXML field of a session log.

A list of all filter plug-ins available are listed here: https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

By default, the configuration generated by Blue Prism will contains a single filter:

```
filter{
json {
source => "eventdata"
target => "event"
}
}
```

By default, configurations contain a single filter for JSON that is used to parse and expand the JSON string that contains the configured data types (session logs, dashboards etc.) so the contents are accessible in the configuration file.

This section can be edited to add and remove filters but the default JSON filter should not be removed or amended.

Output

Outputs determine where events are sent. If outputs have been configured using the Data Gateways wizard, they will be included here. Every event processed will be sent to every output listed in the configuration.

```
output {
file {
path => "C:\data.txt"
}

csv {
path => "C:\data.csv"
}
}
```

In the example above, a .txt file and a .csv output are specified. Every event sent to the Data Gateway system will be written into a text file at C:\data.txt and also a csv file at C:\data.csv

For a list of outputs available see here: https://www.elastic.co/guide/en/logstash/current/output-plugins.html

Event structure

This section details the structure of events in Logstash after they are received from Blue Prism. This information can be used to construct conditional statements in the Logstash configuration to divert events to outputs based on their content, or for creating custom message formats for your outputs.

The event (either session log or published dashboard) is stored in the Blue Prism database as a JSON string. In order to turn this JSON string into a set of fields which can be used in Logstash the JSON filter is added to the configuration:

```
filter{
json{source => "eventdata"
target => "event"}
}
```

This adds the session log / published dashboard as fields nested under the “event” field.

For example:

`[event][eventType]` contains the type of event (session log, published dashboard or custom object data).

`[event][EventData]` contains the data for the event as nested fields.

`[event][EventData][SessionNumber]` contains the Session number if this is a session log event.

To send only session logs from a process named “ProcessA” to a text file you a conditional statement around your output can be used:

```
output{
If [event][eventType] == 1 and [event][EventData][ProcessName] == “ProcessA” {
file {
path => “C:\log.txt”
}
}
}
```

For a full listing of the available fields see the following tables.

General

Event	Description
`[event][eventType]`	The number that represents the event type: 1 = Session Log 2 = Published Dashboard 3 = Custom 4 = Work Queue Analysis
`[event][EventData]`	The data for the event. The structure of this data will differ depending on the event type.

Event type - session logs

Event	Description
[event][EventData][StartDate]	The start date of the process stage formatted in ISO 8601 notation. For example: "2019-02-11T07:59:54.829674+00:00"
[event][EventData][SessionNumber]	The session number for the session this session log belongs to.
[event][EventData][ResultType]	The result type of the process stage.
[event][EventData][Result]	The result of the process stage.
[event][EventData][AttributeXML]	The input and output parameters of the stage serialized to XML.
[event][EventData][ProcessName]	The name of the process this stage belongs to. This will be empty if the session log is logged from a business object.
[event][EventData][ObjectName]	The name of the business object this stage belongs to. This will be empty if the session log is logged from a process.
[event][EventData][ActionName]	If this log is from an Action stage, this is the name of the Action. Otherwise it will be empty.
[event][EventData][PageName]	The name of the page this stage which created this session log belongs to.
[event][EventData][StageType]	The type of stage which created this session log.
[event][EventData][StageId]	The ID of the stage which created this session log.

Event type - published dashboards

Event	Description
[event][EventData][Source]	The name of the published dashboard.
[event][EventData][Subject]	The name of the dashboard tile which generated the data.
[event][EventData][Values]	The data from the dashboard tile.

Event type - custom object data

Event	Description
[event][EventData][CustomDataCollection]	The custom data from the process that will be sent.
[event][EventData][SessionNumber]	The session number of the process the data is coming from.
[event][EventData][StageID]	The ID of the stage that this action is called from.
[event][EventData][StageName]	Name of the Send Custom Data action stage.
[event][EventData][StageType]	The type of the Send Custom Data action stage.
[event][EventData][StartDate]	The start date of the session that the Send Custom Data action is running on.
[event][EventData][ProcessName]	The name of the process that the action is being called from.
[event][EventData][PageName]	The name of the page of the process the custom data action is on.
[event][EventData][ObjectName]	The object that the data is coming from - will always be "Data gateways".
[event][EventData][actionName]	The action that the data is coming from - will always be "Send Custom Data".

Directing data to outputs based on content

When session logs and dashboard data are sent to separate text files, conditional statements can be applied to the outputs that will only pass events to an output if it meets one or more conditions of those conditions. This allows outputs, customized in the advanced editor or created in an external text editor, to support Logstash functionality not provided in the Data Gateway Configuration wizard. For example, outputs can be edited to only send data for specified processes or dashboard tiles.

In this example the conditional statements around the file outputs check for a certain EventType value. Session Logs have an event type of 1, and dashboards have an even type of 2.

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string =>
"jdbc:sqlserver://localhost\sqlexpress:1433;databaseName=a;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3 * * * * *"
}
}
filter {
json {
source => "eventdata"
target => "event"
}
}
output {
if [event][EventType] == 2 and [event][EventData][Source] == "Dashboard 1" {
file {
path => "C:\dashboardlogs.txt"
codec => line { format => "%{event}"}
}
}
if [event][EventType] == 1 {
file {
path => "C:\sessionlogs.txt"
codec => line { format => "%{event}"}
}
}
}
```

For information about the structure of events pulled from the Blue Prism database see Event structure.

Advanced configuration for database outputs

Database outputs, configured in the wizard, must adhere to an expected format:

- There must be an eventType column of type integer - this stores the type of the event.
- There must be an eventData column of type nvarchar(max) - this stores the events serialized to a JSON string.

In advanced configurations, the columns of the table and the data inserted into the table can be customized.

In this example, certain fields from the session log events are sent to the tableabc table in a database.

The jdbc database output inserts the session number, process name, and attributexml fields from the session log into the appropriate columns of the tableabc table.

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string =>
"jdbc:sqlserver://localhost\sqlexpress:1433;databaseName=a;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3 * * * * *"
}
}
filter {
json {
source => "eventdata"
target => "event"
}
}
output {
if [event][EventType] == 1 {
bpjdbc {
connection_string =>
"jdbc:sqlserver://TheServer;databaseName=MyDB;integratedSecurity=true;"
driver_jar_path => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
statement => ["insert into tableabc(EventType, EventData) values(?, ?)", "[event]
[EventType]", "[event][EventData]"]
}
}
}
```

Custom configuration examples

For a full list of all events and more information about the event structure, see Event structure.

Filter events and divert outputs using conditional statements

In this example the [event][EventType] field is used to send event types to separate files depending on whether event type is a session log (EventType == 1) or a published dashboard (EventType == 2).

The event type for custom object data (EventType == 3) is not specified and so any data of this type in the Data Gateways engine is not included in discarded.

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string =>
"jdbc:sqlserver://localhost\sqlexpress:1433;databaseName=a;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3 * * * * *"
}
}
filter {
json {
source => "eventdata"
target => "event"
}
}
output {
if [event][EventType] == 2 and [event][EventData][Source] == "Dashboard 1" {
file {
path => "C:\dashboardlogs.txt"
codec => line { format => "%{event}"}
}
}
if [event][EventType] == 1 {
file {
path => "C:\sessionlogs.txt"
codec => line { format => "%{event}"}
}
}
}
```

Send events based on session log process names

In this example events to a particular output based on a process name from a session log. There are two outputs:

- All events get sent to the C:\allevents.txt text file
- Session log events from the *Process123* process are additionally sent to the specified HTTP endpoint.

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string =>
"jdbc:sqlserver://localhost\sqlexpress:1433;databaseName=a;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3 * * * * *"
}
}
filter {
json {
source => "eventdata"
target => "event"
}
}
output {
if [event][EventType] == 1 {
file {
path => "c:\allevents.txt"
codec => line { format => "%{event}"}
}
}
if [event][EventType] == 1 and [event][EventData][ProcessName] ==
"Process123" { bphttp {
url => "localhost:8080/api/post"
http_method => "post"
headers => {"Authorization" => "Basic <base64><%SQL Serv.username%>:<%SQL
Serv.password%></base64>"}
}
}
}
```

Credentials in custom configurations

When credentials or other sensitive data is required in the configuration, they should be added to a Blue Prism credential and then referenced in the configuration by the credential name.

When creating a Blue Prism credential for use in Data Gateways configurations, the credential type must be *Data Gateway Credential*. These credentials are accessible only by the Data Gateway system and are not accessible to Blue Prism processes.

Credentials can be referenced in the configuration using the syntax `<%{credentialname}.{property}%>`, where `{credentialname}` is the name of the credential and `{property}` is the name of the property in the credential.

For example, to use the username of a credential named *cred1* the configuration code would be `<%cred1.username%>` and `<%cred1.password%>` respectively.

Credential custom properties can be accessed using the property name.

Change the port that Data Gateways uses to connect to the SQL server

By default, the Data Gateways process uses port 1433. However, an alternative port can by specified by editing a custom configuration.

The port is defined in the Input section of the configuration file structure, as demonstrated in the example below. Edit a custom configuration and replace the highlighted value with the required port number:

```
input {
jdbc {
jdbc_driver_library => "..\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
jdbc_connection_string => "jdbc:sqlserver://SQL_SERVER_INSTANCE:1433;databaseName=BP_
DATABASE;"
jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbc_user => "<%Data Gateways BP Database SQL User.username%>"
jdbc_password => "<%Data Gateways BP Database SQL User.password%>"
statement => "delete top(3000)from BPADataPipelineInput with (rowlock, readpast)
output deleted.eventdata"
schedule => "*/3* * * * *"
}}
```

Data Storage when endpoint unavailable

If a connection to an HTTP, Splunk, or Database endpoint can't be made when session log data is sent from the Blue Prism database to the Data Gateways engine, the data for the associated output is stored temporarily in a dedicated file within the Data Gateways engine. When the endpoint becomes available the data is resent to the specified endpoint and deleted from the file.

The session log data is held indefinitely until the endpoint becomes available and will continue to store additional session log data as subsequent events are written to it.

The undelivered session log data is written to the same file, regardless of output type, which is stored by default in the following directory: logstash\data\dead_letter_queue\main on the application server that runs the Data Gateways process. The data is stored in a non-human readable format. The default maximum size for a data storage file in the Data Gateways engine is 1024 MB (1 GB). If a file reaches this limit, a new file is automatically created to store the excess data.

Blue Prism output extensions

By default, the output extension *bphttp* is used for HTTP and Splunk, and *bpjdbc* is used for Database output types. These Blue Prism extensions are integral to the automatic storage of session log data when an endpoint is unavailable.

Upgrading from 6.5

In Blue Prism version 6.6, *bphttp* and *bpjdbc* output extensions replaced *http* and *jdbc*. If upgrading Blue Prism from 6.5, the output types must be updated to use these output extensions to utilize the endpoint unavailable data storage functionality. This can be done by either creating new Data Gateways configuration, or manually editing the existing configuration via advanced or custom configuration and replacing *http* with *bphttp*, and *jdbc* with *bpjdbc*.

File and custom output types

This functionality does not apply to File or custom configured output types. When using these output types, we recommend enabling Write session logs to database to avoid data loss.

Troubleshooting

How do I check that the Data Gateways system is functioning as expected?

There are a number of ways to check Data Gateways for issues:

Check the status of the Data Gateways process in control room

If it is in an error state, the status message will provide information about what the problem is.

Check the event logs on the application server

Any Data Gateways process errors will be output into the event log from the Logstash Process Manager source. The standard output stream from the Logstash process is also written to the event log in the event of an error, which can help identify Logstash specific errors.

Check the Logstash logs on the application server

The logs are written to C:\Logstash\logstash\logs\.

Should I use a dedicated application server for Data Gateways?

For the majority of installations, existing application servers can be used for Data Gateways. However, where Data Gateways is being used to process large sets of data, or to direct data to a number of targets, it may be appropriate to deploy Data Gateways onto dedicated Blue Prism application servers which are used only for this purpose.

How do I set up Data Gateways Splunk outputs to use HTTPS?

For details about how to achieve this, see:

http://portal.blueprism.com/customer-support/support-center#/path/1357434942.