OMRON

Data Flow Edge Device

DX-series

SpeeDBee Synapse

User's Manual

DX100-0010

CPU Unit



V243-E1-02

NOTE

1. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
2. No patent liability is assumed with respect to the use of the information contained herein.
   Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice.
3. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions.
   Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Microsoft, Windows, Excel, Visual Basic, and Microsoft Edge are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- SpeeDBee Synapse is a trademark of SALTYSTER Co., Ltd.
- Grafana is a trademark of Grafana Labs.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

- Microsoft product screen shots used with permission from Microsoft.
- This product incorporates certain third party software. The license and copyright information associated with this software is available at https://www.fa.omron.co.jp/product/tool/dx-info/index_en.html.

Table of Contents

1 Index

2 Introduction

3 Overview of SpeeDBee Synapse

4 Introduction Flow

- 4.1 Introduction Flow
- 4.2 Install (Linux)
- 4.3 Install (Windows)
- 4.4 Initial Setting
- 4.5 Setting of Components
- 4.6 Running the Collection Service

5 Explanation of Functions

- 5.1 Explanation of Functions
- 5.2 Overview of engine functions
- 5.3 Flow Editor
- 5.4 Collector
- 5.5 Serializer
- 5.6 Emitter
- 5.7 Logic (Event)
- 5.8 Logic (Real Time Analysis and Data Manipulation)
- 5.9 Action
- 5.10 Development Components
- 5.11 System Components
- 5.12 Component Default Settings
- 5.13 Status Timeline
- 5.14 Various Logs
- 5.15 Database Query (SQL)
- 5.16 Arithmetic and Conditional Expressions
- 5.17 Managing Configuration Information (Project)

6 Advanced Use

6.1 Advanced Use
6.2 Creating and Using Custom Components
6.2.1 Creating and Using Custom Components
6.2.2 Overview
6.2.3 Creating Python custom components
6.2.4 Creating C language custom components
6.2.5 Creating custom UI
6.2.6 Custom component development environment (SCCDE)
6.3 Linking with Control Programs (WebAPI)
6.4 Linking with Grafana
6.5 Global Variables
7 Troubleshooting
8 Security Guide
9 Backward Compatibility
10 Execution License
11 i-Trigger (Expansion Custom Package)
11.1 i-Trigger (Expansion Custom Package)
11.2 i-Trigger Overview
11.3 Cameras That Can Be Used
11.4 Installation
11.5 Video Viewer for Video Playback
11.6 Components
11.7 Tutorial
11.8 Recording Specifications for i-Trigger
12 Release Notes
13 Trademarks
14 Contact manufacturer
15 Appendix
15.1 Appendix
15.2 Error code list
15.3 Custom component Python-API reference
15.4 Custom component C-API reference
15.5 Custom UI JS-API reference

15.6 Open source used

15.7 Index

15.8 DB query service Web-API specification

1 SpeeDBee Synapse Document

- Introduction
- Overview of SpeeDBee Synapse
- Introduction Flow
- Explanation of Functions
- Advanced Use
- Troubleshooting
- Security Guide
- Backward Compatibility
- Execution License
- i-Trigger (Expansion Package)
- Release Notes
- Trademarks
- Contact manufacturer
- Appendix

2 Introduction

This manual describes how to operate "SpeeDBee Synapse" (abbreviated as SBS). By using the SpeeDBee Synapse, it is possible to collect data on manufacturing site facilities and link other systems. Of course, outside the factory floor, IoT systems can also be built.

2.1 Target Audience

This manual is intended for the following users:

- Designers of IoT systems and FA systems
- Those who manage IoT systems and FA systems
- Those who actually operate and run IoT and FA systems

You can also implement your own functions as "custom components" using programming languages. In that case, you need some programming knowledge. Details are described in the manual.

2.2 Agreements

Refer to the manual of the hardware unit on which SpeeDBee Synapse is installed, or the contents displayed at the executing the installer of SpeeDBee Synapse.

2.3 Safety Precautions

For the following functions, it is possible to change the register, variable, status, etc. of the other device, and the change result may affect the behavior of the equipment.
Before using these functions, be sure to carefully consider and verify the effects on equipment.

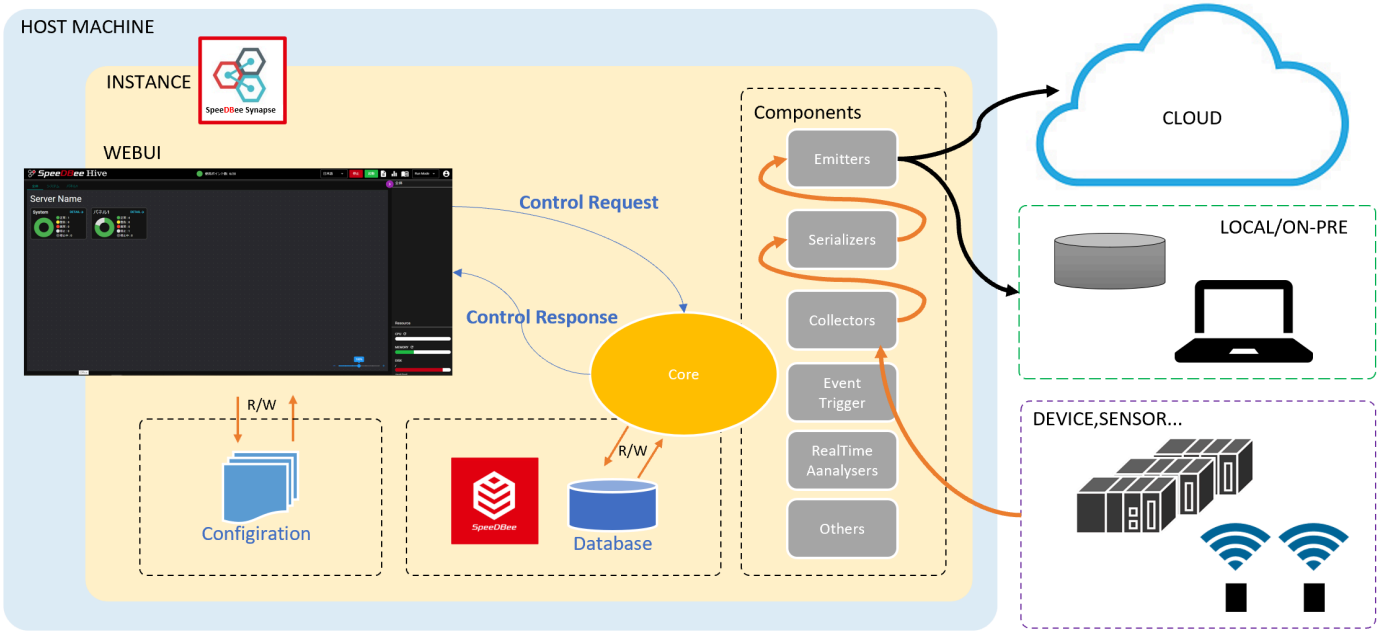
- PLC Writing
- Modbus Writing
- EtherNet/IP Writing

3 Overview of SpeeDBee Synapse

This chapter explains the features, configuration, and operational flow of SpeeDBee Synapse.

3.1 Features of SpeeDBee Synapse

SpeeDBee Synapse can collect data from various devices (PLC, etc.) that constitute FA and IoT. Collected data can be converted, analyzed, and linked with other systems by using various functions of SpeeDBee Synapse. This enables you to build a system that achieves everything from data collection to data utilization.



3.2 Configuration of SpeeDBee Synapse

SpeeDBee Synapse consists of four main functions.

Function name	Description
Core	This is the central part that controls the overall functionality of the collection engine.
Component	This is a group of modules that realize data collection, processing, analysis and transmission. Various functions can be realized by connecting components.
Time series database	This is a database that stores collected data in line with time.

Function name	Description
WEBUI	This is the user interface for setting various functions and visualizing data.

Components are organized into eight categories. The following table lists the components that belong to each category.

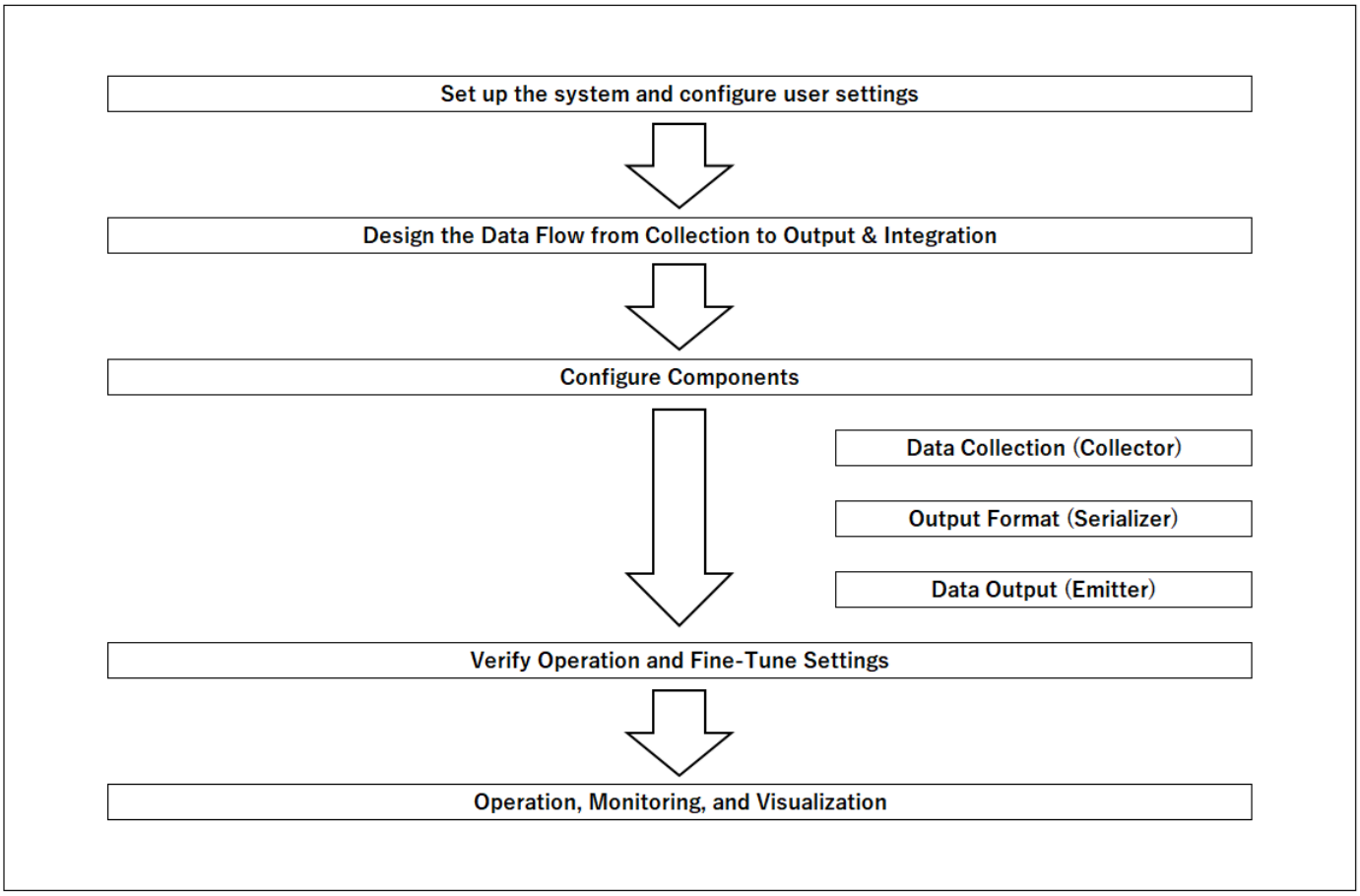
Category	Name	Description
Collector	-	Provides data collection functionality. This is the starting point for using various functions after data collection.
	EtherNet/IP	Collects data from EtherNet/IP-enabled devices.
	PLC	Collects data from PLC register/variable information.
	Modbus	Collects data from Modbus compatible devices.
	MQTT	Data collection is performed using the MQTT protocol (Subscribe).
	Resources	Collect system resources (CPU usage, memory consumption, etc.) in the operating machine.
	Compute Collector	Registers the result of a calculation that uses collected data as new separate data.
Emitter	-	Provides data output or data transmission functionality. You can use this information to link with external systems.
	Cloud for Azure	Sends data to Azure IoT Hub.
	MQTT	Sends data using the MQTT protocol (Publish).
	FTP/FTPS/SFTP	Transfers data via FTP, FTPS or SFTP.
	File	Saves the collected data as a file to storage.
	Cloud (for AWS)	Sends data to AWS IoT Core.
Serializer	-	Processes data into the specified format.

Category	Name	Description
	CSV	Processes data into CSV format.
	JSON	Processes the specified data into JSON format.
Deserializer	-	The specified format is processed into data.
	JSON	Processes the specified data in JSON format.
Action	-	Provides any action that uses the collected data.
	Shell Command	Execute any shell command that uses the collected data.
	Send email	Sends an email using the collected data.
	Modbus Writing	Writes a value to the specified address of a Modbus compatible device.
	PLC Writing	Writes a value to the specified address of the PLC.
	EtherNet/IP Writing	Writes a value to an EtherNet/IP device.
Logic	-	Provides real-time analysis and decision processing of collected data.
	Event data	The collected data is judged against the condition, and only when the conditions are met, it is newly registered as other data.
	Event trigger	Collected data is judged against the condition and the information on sections that meet the conditions is provided.
	Basic Stats	Registers the result of statistical processing that uses collected data as new separate data.
	Moving average	Registers the moving average of the collected data as data.
	FFT	Registers the result of FFT of collected data as data. (Frequency, Intensity)
System	-	Provides functionality related to the system.

Category	Name	Description
	DB query service	You can query the data collected in SpeeDBee Synapse. Also used for Grafana linkage.
	Core internal	This is the component that manages internal status within SpeeDBee Synapse.
	Error manager	This component can detect all errors that occur within SpeeDBee Synapse.
Custom (C)	-	This is a component composed of programs designed by the user in Python or C. You can add various functions that are not possible with standard components.

3.3 Flow up to operation

The basic flow up to the operation of SpeeDBee Synapse is as follows.



① Setting Up of a Setup User

Refer to this manual to set up SpeeDBee Synapse. In addition, it sets up the users according to their privileges.

② Study the flow from data collection to output and linkage

Depending on the purpose of the system, what data is to be actually collected and the equipment to be used are selected and examined. In addition, it designs the format in which the collected data will ultimately be outputted and utilized. Checks the supported features such as collectors, serializers, emitters, etc. to be used. At this stage, if you need an unsupported protocol, consider developing a custom component.

③Setting of Components

Configure the various components according to the flow examined. The basic flow is to set up in the order of data collection, processing and output.

- Data Collection Configuration (Collector)

Configures data collection settings. Make sure that the source device and its settings are correct. Then you actually configure the components.

- Setting the Output Format (Serializer)

Configures data output format settings. Set up either CSV or JSON. You can also set the units in which the output file is divided.

- Data Output Settings (Emitters)

Specifies how output data is to be processed in the final result. For example, since the File emitter saves the output file to storage, it sets up where to save it. To send data to an external device, check the information required for external connection and make settings.

④ Operation check and setting tuning

Verify that each configured component is operating correctly. Make sure that data has been collected normally, that data has been outputted in the specified format, and that the transmission has been successfully done. If there is an unintended operation, review the settings, change the settings, and check again.

⑤ Operation, monitoring and visualization

Actually operate the system. You can monitor data and visualize graphs using the features provided by the Data Exploitation Engine. If a problem occurs during operation, reconsider and perform configuration tuning again.

In addition, you can configure a system that utilizes SpeeDBee Synapse by setting up real time analysis and event linkage according to the purpose.

4 Introduction Flow

4.1 Introduction Flow

This chapter describes the procedures for deploying SpeeDBee Synapse.

As the deployment flow, the system is configured by the following flow.:

1. Install (Linux)
2. Install (Windows)
3. Initial Setting
4. Setting of Components
5. Running the Collection Service

Although the system described here is very simple, it is recommended that you read through it to understand the basics of SpeeDBee Synapse.

4.2 Install (Linux)

This section describes how to install/uninstall SpeeDBee Synapse (Linux).

4.2.1 Execution environment

The system requirements for this package are Ubuntu20 (AMD64)/Ubuntu22(AMD64, ARM64).
For environments other than those listed above, please refer to the manual for the relevant environment.

> In the case of using Data Flow Controller, Synapse is pre-installed. so you do not need to follow the instructions in this chapter.

4.2.2 Installation and Updates

Follow the instructions below to install SpeeDBee Synapse on your device. The same procedure is used when updating.

Place File

1. Place the file SpeeDBee Synapse in the /tmp directory on the device. The following is an example of deployment using the scp command from Windows PowerShell.

If the device OS is Ubuntu20 (amd64)

```
PS >scp speedbeesynapse_ubuntu20_x.x.x-r0_amd64.deb (account name)@(device IP address):/tmp
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22 (amd64)

```
PS >scp speedbeesynapse_ubuntu22_x.x.x-r0_amd64.deb (account name)@(device IP address):/tmp
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22 (arm64)

```
PS >scp speedbeesynapse_ubuntu22_x.x.x-r0_arm64.deb (account name)@(device IP address):/tmp
```

Note: Where x.x.x is the version number.

If the device OS is Raspberry Pi OS (bookworm 64 bit)

```
PS >scp speedbeesynapse_raspi_bookworm_x.x.x-r0_arm64.deb (account name)@(device IP
address):/tmp
```

Note: Where x.x.x is the version number.

If the device OS is debian12 (bookworm arm64)

```
PS >scp speedbeesynapse_debian12_x.x.x-r0_arm64.deb (account name)@(device IP address):/tmp
```

Note: Where x.x.x is the version number.

Installation and Startup

1. Log in to the device using ssh. The following is an example of using the ssh command to log in from Windows PowerShell.

   ```
   PS >ssh (account name)@(device IP address)
   ```

2. Run the following command to install SpeeDBee Synapse:

If the device OS is Ubuntu20 (amd64)

```
$ sudo apt install /tmp/speedbeesynapse_ubuntu20_x.x.x-r0_amd64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22 (amd64)

```
$ sudo apt install /tmp/speedbeesynapse_ubuntu22_x.x.x-r0_amd64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22 (arm64)

```
$ sudo apt install /tmp/speedbeesynapse_ubuntu22_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Raspberry Pi OS (bookworm 64 bit)

```
$ sudo apt install /tmp/speedbeesynapse_raspi_bookworm_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

If the device OS is debian12 (bookworm arm64)

```
$ sudo apt install /tmp/speedbeesynapse_debian12_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

Installation routed through a proxy

In environments where a proxy server (forward proxy) is deployed within the organization, proxy connection information must be explicitly configured to perform the installation. Please specify the environment variables `http_proxy` and `https_proxy` as shown in the command below.

```
$ sudo http_proxy=<url of proxy server> https_proxy=<url of proxy server> apt install ...
```

Note: To specify a username and password for Basic Authentication, include them in the URL as follows.

```
http://<username>:<password>@<host>:<port>
```

3. Run the following command to start SpeeDBee Synapse:

```
$ sudo systemctl start speedbeesynapse
```

Access from browser

1. You can open the SpeeDBee Synapse window by accessing the following URL from your browser.
   - `http://{hostname or device IP address}:8120`

     Example) `http://192.168.1.3:8120`

     > **About PCs accessing SpeeDBee Synapse**
     >
     > From a PC belonging to the same network as the installed device, use a browser (Google Chrome/Microsoft Edge) to access the URL of SpeeDBee Synapse.
     > To access by host name, you need to set up an environment that allows the PC to access by host name.
     > Note: The operation has been confirmed with some iPad terminals (using Safari), and basic operations are possible.

     > **After the Update**
     >
     > If SpeeDBee Synapse is updated while the screen is open and operations are continued without reloading, the system may not function correctly. Please reload the screen before performing any operations after the update.
     > If you experience issues such as newly added screen items not appearing after the update, please clear your browser's cache.

2. Because it is through http, a security warning may appear on the screen when you access it for the first time.

> **Access via https**
>
> You can also access SpeeDBee Synapse via https. For details, see "HTTP/HTTPS Communication".

4.2.3 Control operate for SpeeDBee Synapse

You can control SpeeDBee Synapse by logging in using ssh to the device where SpeeDBee Synapse is installed and running the following commands:

- Start

```
$ sudo systemctl start speedbeesynapse
```

- Restart

```
$ sudo systemctl restart speedbeesynapse
```

- Stop

```
$ sudo systemctl stop speedbeesynapse
```

- Enabling Autostart

  This setting enables SpeeDBee Synapse to start automatically when the device is restarted.

```
$ sudo systemctl enable speedbeesynapse
```

- Disabling Autostart

  When this setting is applied, the SpeeDBee Synapse service does not start when the device is restarted.

```
$ sudo systemctl disable speedbeesynapse
```

4.2.4 Uninstall

You can uninstall SpeeDBee Synapse from the device by following the instructions below.

1. Log in to the device using ssh. The following is an example of using the ssh command to log in from Windows PowerShell.

```
PS >ssh (account name)@(device IP address)
```

2. Run the following command to uninstall SpeeDBee Synapse:
   - Keeping logs and configuration files

```
$ sudo apt remove speedbeesynapse
```

   - To delete all logs and configuration files

```
$ sudo apt purge speedbeesynapse
```

To reinstall SpeeDBee Synapse, follow the procedure described in the Installation section.

If logs and configuration files were left during uninstallation, the settings will be retained after reinstallation.

4.3 Install (Windows)

This section describes how to install/uninstall SpeeDBee Synapse (Windows). The described operations are based on Windows 11. Depending on the OS versions, the method to start Windows applications may be varied. Perform the operation that matches your OS version.

> **About OS Version**
>
> Windows 10 support will end on 2025/10/14, so using Windows 11 is recommended.

4.3.1 Execution environment

The system requirements for this package are as follows.

- OS: Windows 10 (64bit) version 22 H2 or later, Windows 11 (64bit) version 23 H2 or later
- Architecture: amd64 (x86_64)

For environments other than those listed above, please refer to the manual for the relevant environment.

4.3.2 Installing and Updating

Follow the instructions below to install SpeeDBee Synapse on your PC. The same procedure is used when updating.

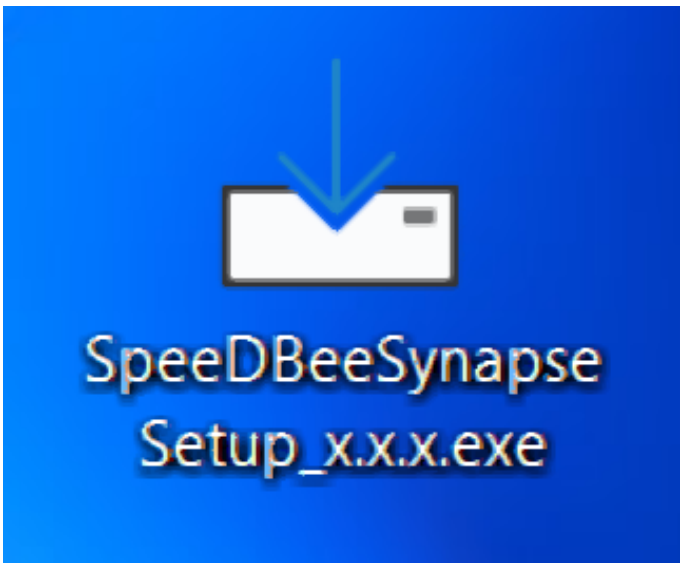
Preparing for Installation

Use the installer below. Place the file on the desktop of the target Windows PC beforehand.

- SpeeDBeeSynapseSetup_x.x.x.exe (The x.x.x part depends on the version.)

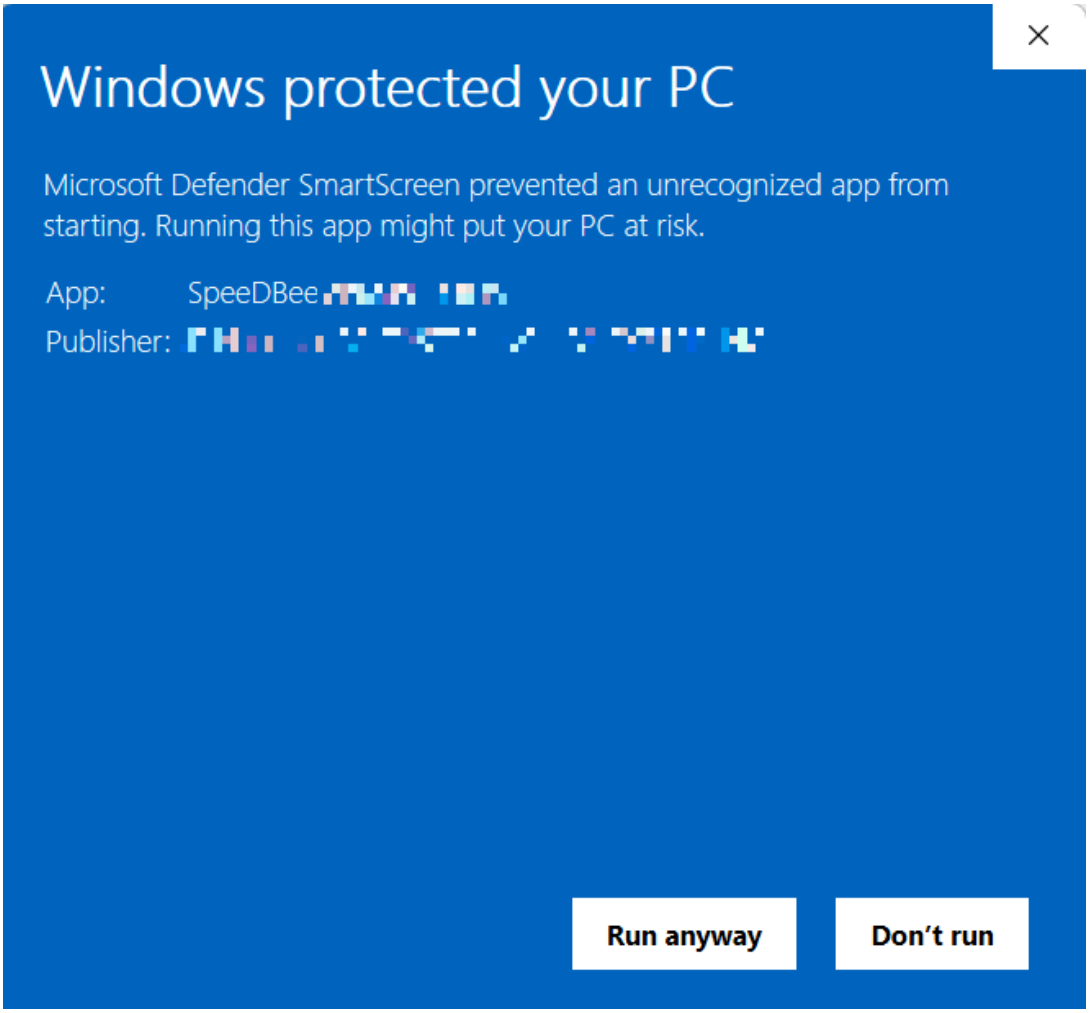
Installation and Startup

1. Run the installer of SpeeDBee Synapse.

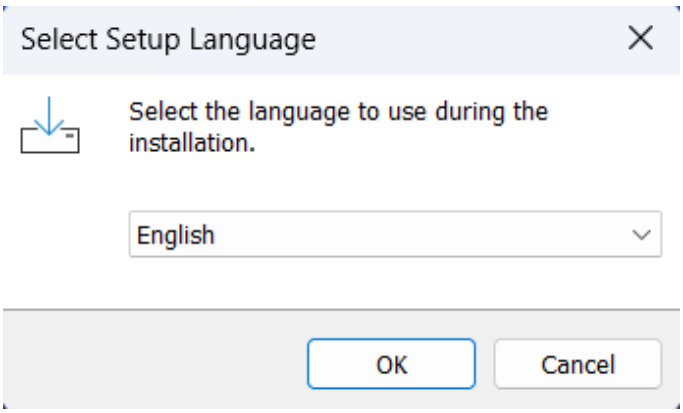


- You must have Windows administrator privileges to run the installer.

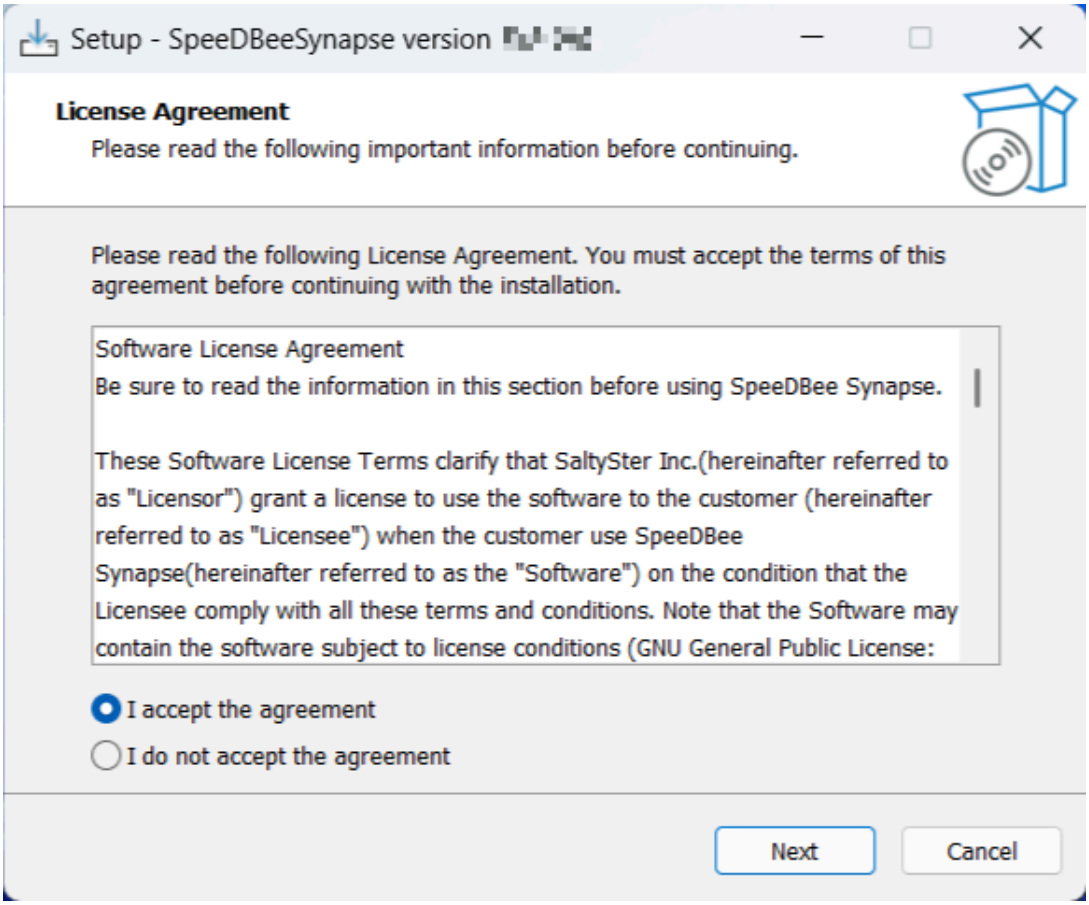
2. You may see a warning when Microsoft Defender is enabled. Make sure the publisher is "OMRON Corporation" and click Run anyway.



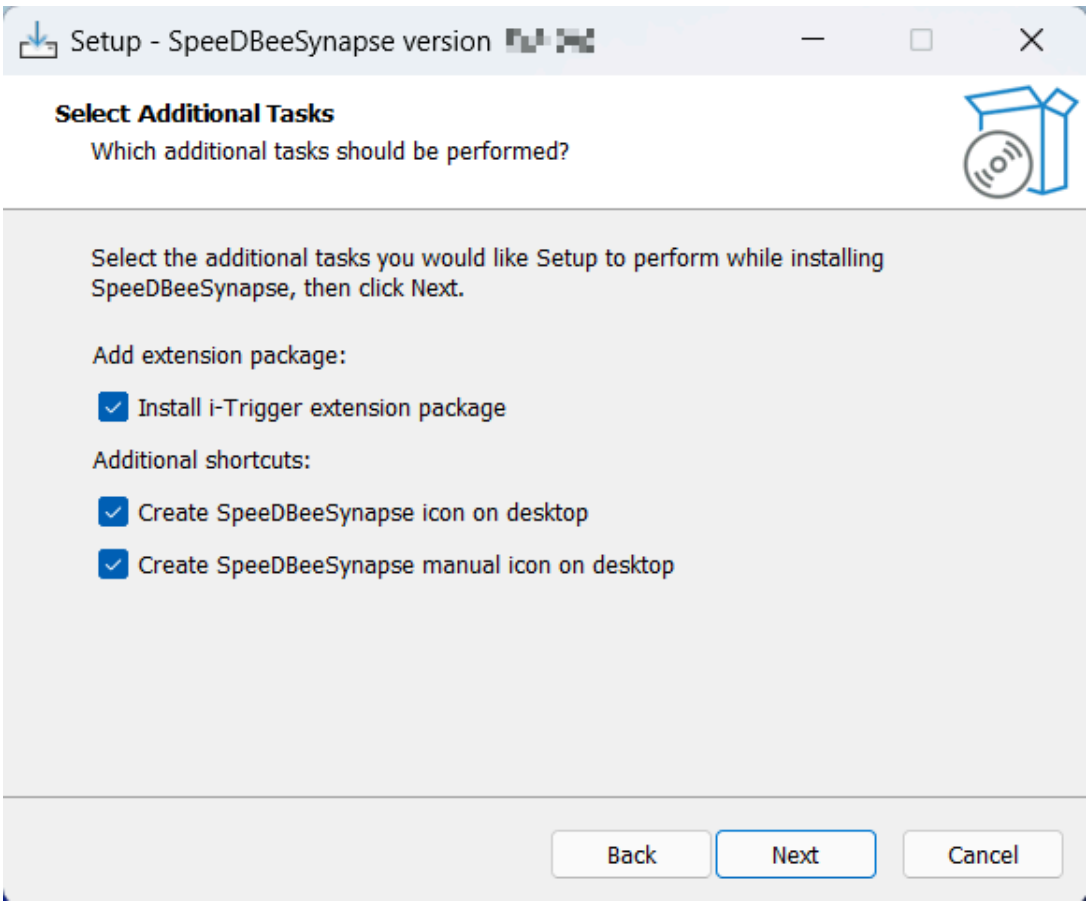
3. Select the language to use for setup, then click OK.



4. Upon reviewing the Software License Agreement, click "I accept the agreement" and Next.

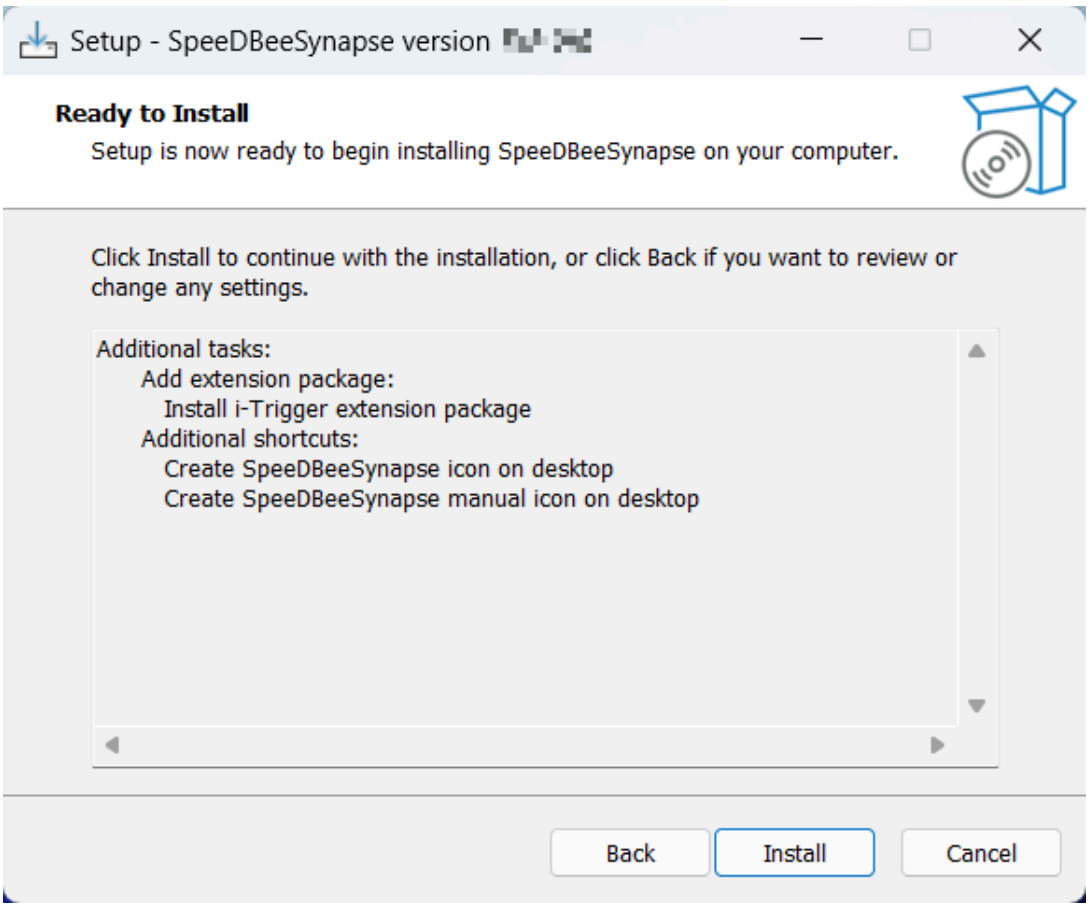


5. A selection of additional tasks appears. Click Next.

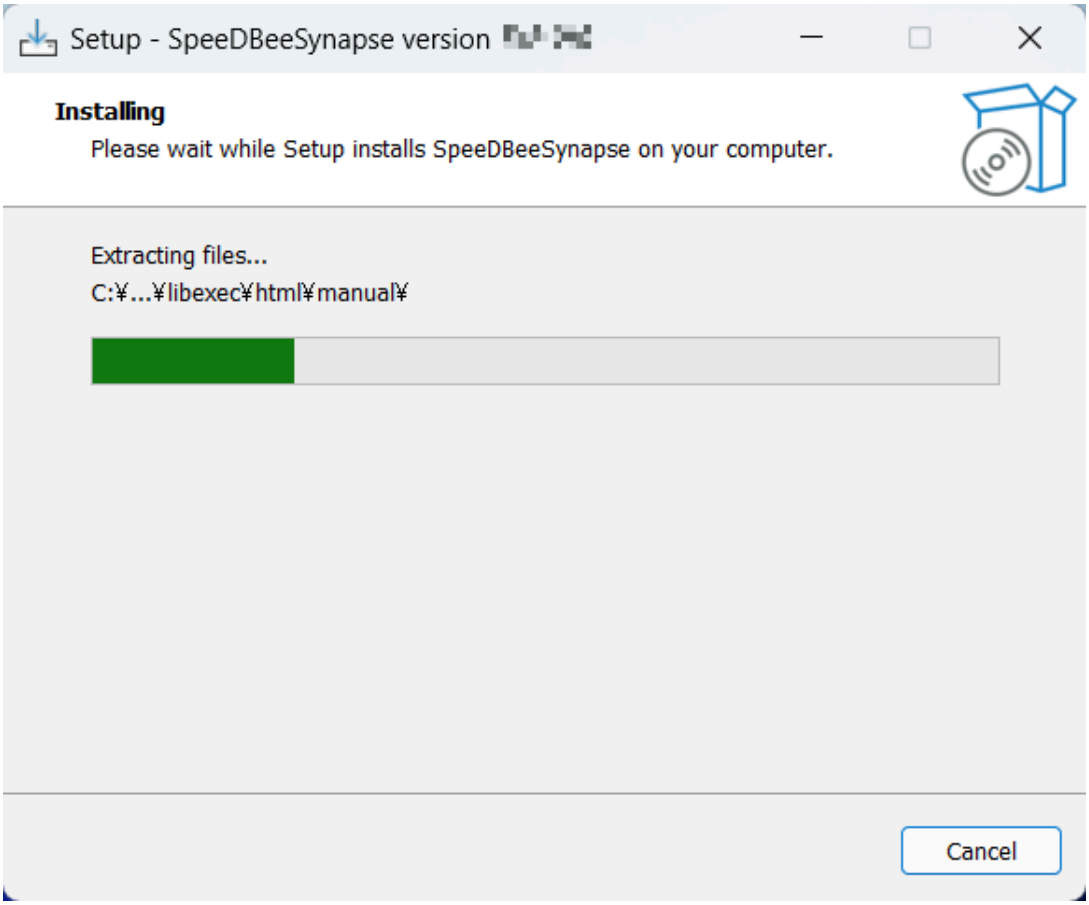


- If you do not need the i-Trigger function, uncheck the checkbox.
- Uncheck any unnecessary icons.

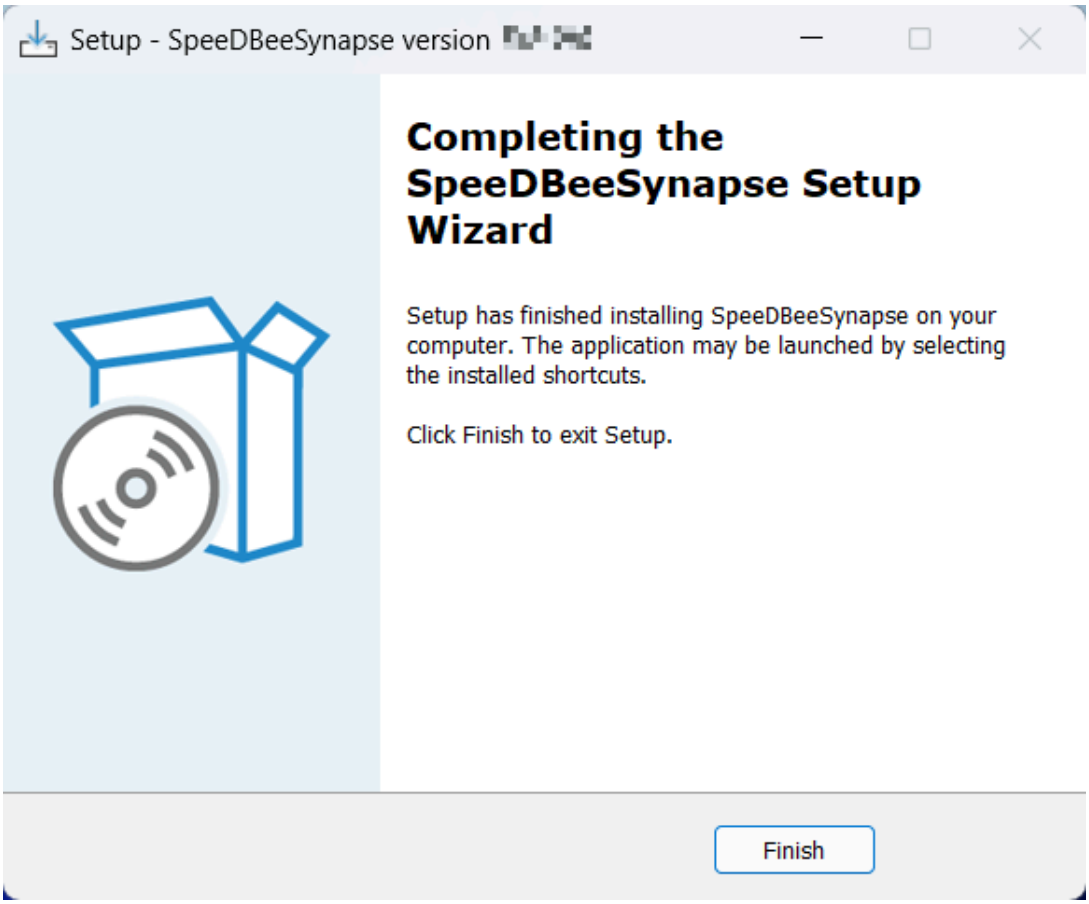
6. Confirm the displayed information, and then execute Install.



7. The installation proceeds.



8. When the installation is complete, click Finish.



> You may be prompted to restart your PC after the installation. In that case, please restart your computer.

Access from browser

1. You can open the SpeeDBee Synapse window by accessing the following URL from your browser.
   - `http://{hostname or device IP address}:8120`

     Example) `http://192.168.1.3:8120`

> **About PCs accessing SpeeDBee Synapse**
>
> From the PC on which it was installed or a PC that is part of the same network as the target PC, use a browser (Google Chrome/Microsoft Edge) to access the URL of SpeeDBee Synapse.
> To access by host name, you need to set up an environment that allows the PC to access by host name.
> Note: The operation has been confirmed with some iPad terminals (using Safari), and basic operations are possible.

> **After the Update**
>
> If SpeeDBee Synapse is updated while the screen is open and operations are continued without reloading, the system may not function correctly. Please reload the screen before performing any operations after the update. If you experience issues such as newly added screen items not appearing after the update, please clear your browser's cache.

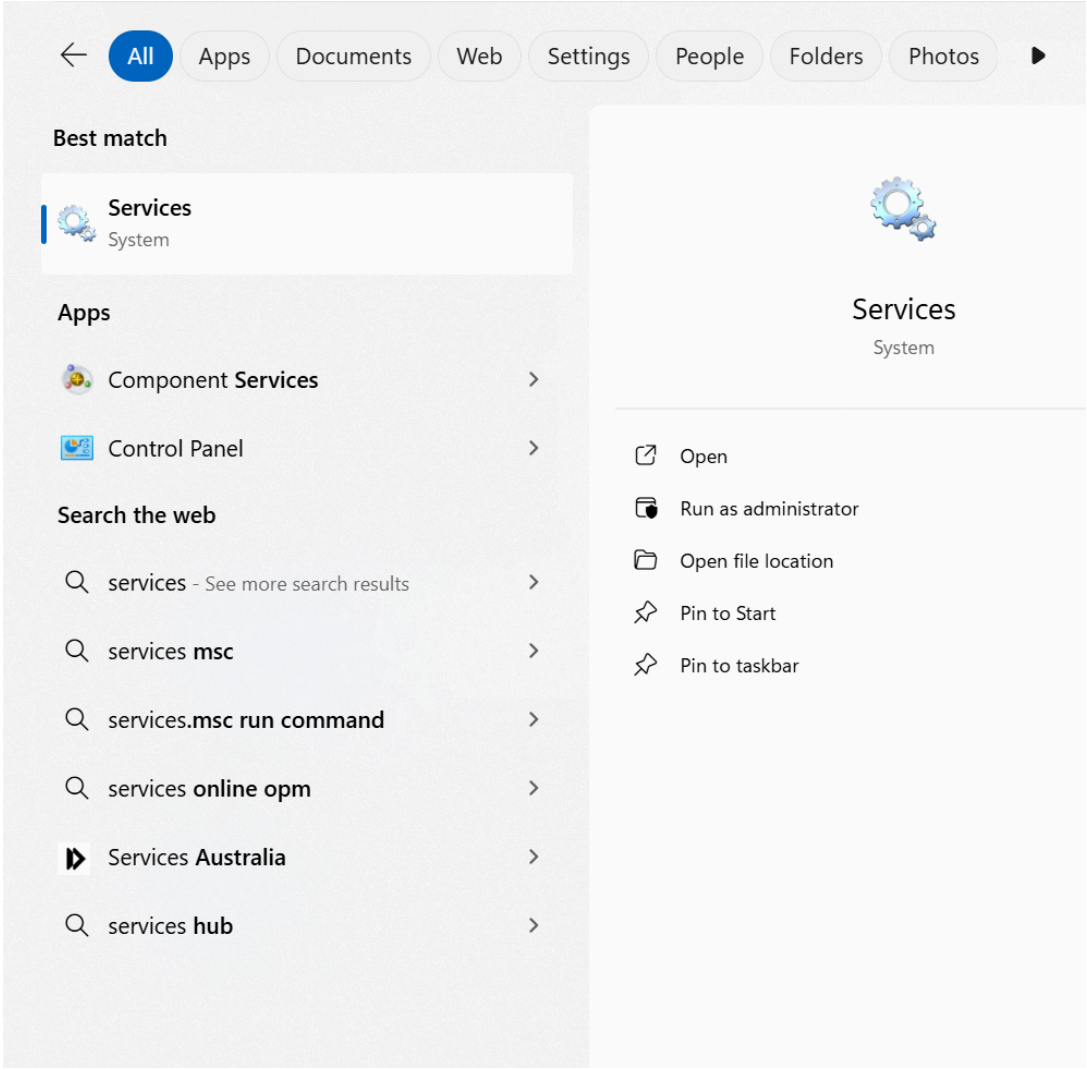
2. Because it is through http, a security warning may appear on the screen when you access it for the first time.

> **Access via https**
>
> You can also access SpeeDBee Synapse via https. For details, see "HTTP/HTTPS Communication".

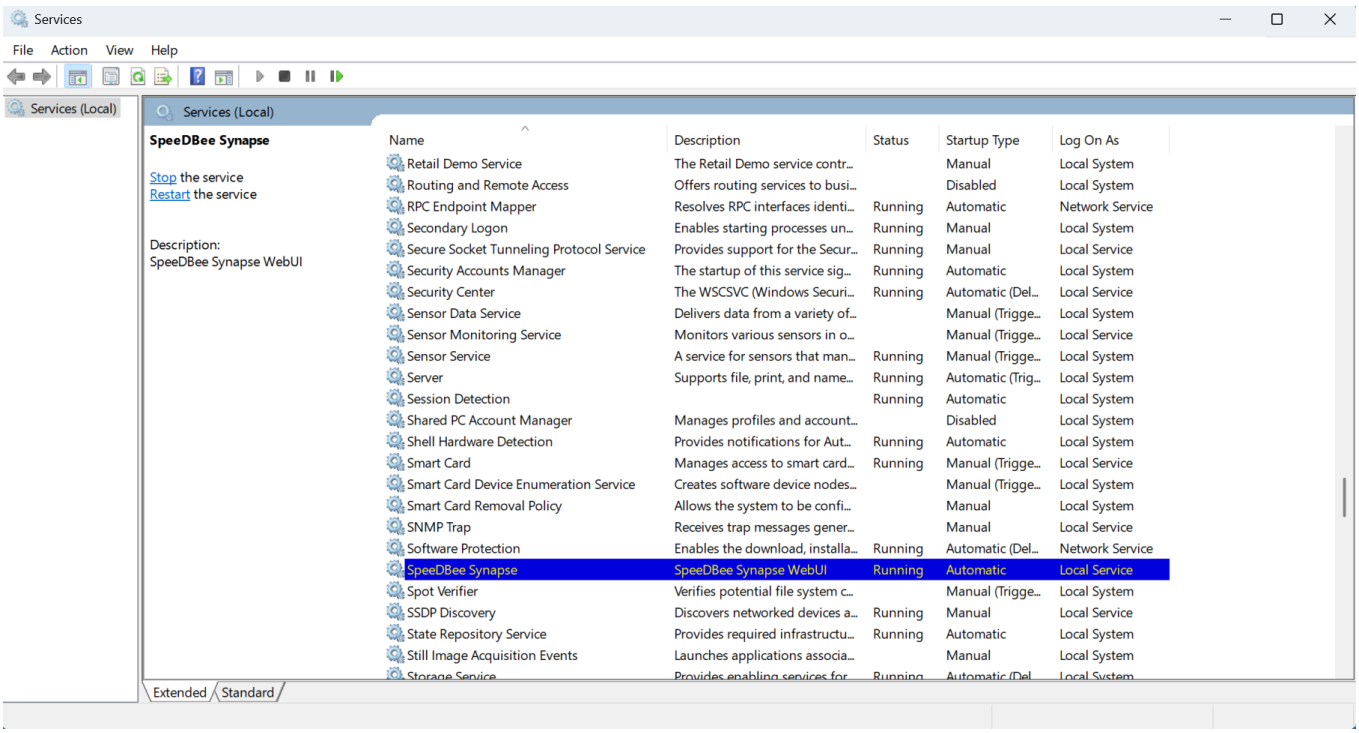
4.3.3 Control for SpeeDBee Synapse

SpeeDBee Synapse is registered as a Windows service when you install it. Therefore, each control operates from a Windows service.

1. Type "Services" into the search field at the bottom of the desktop and launch the service from the search results.

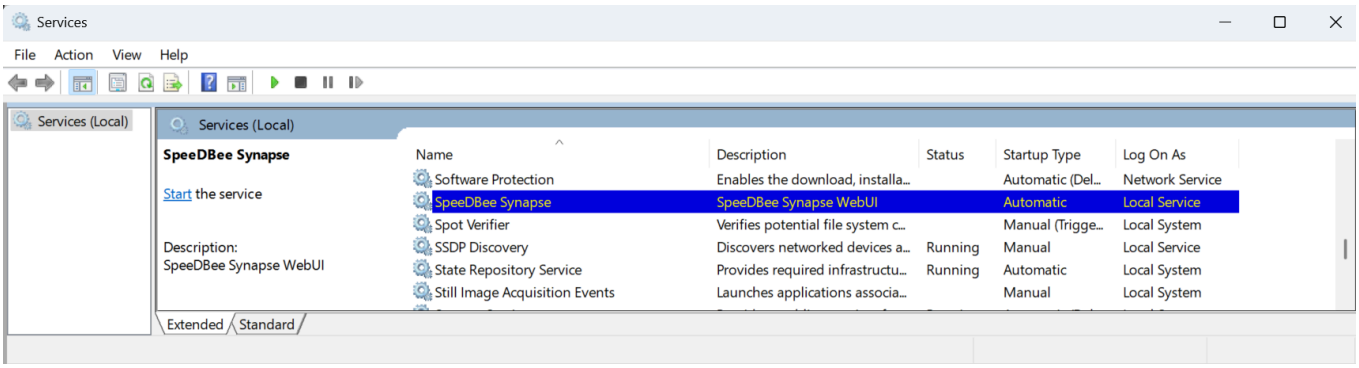


2. Select "SpeeDBee Synapse" to perform various controls.

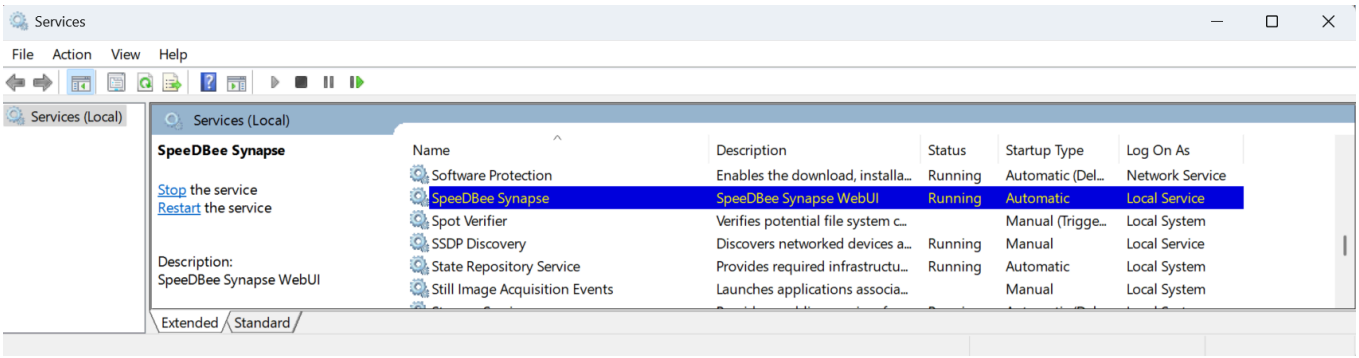


4.3.3.1 Start

1. Select SpeeDBee Synapse and click Start Service.

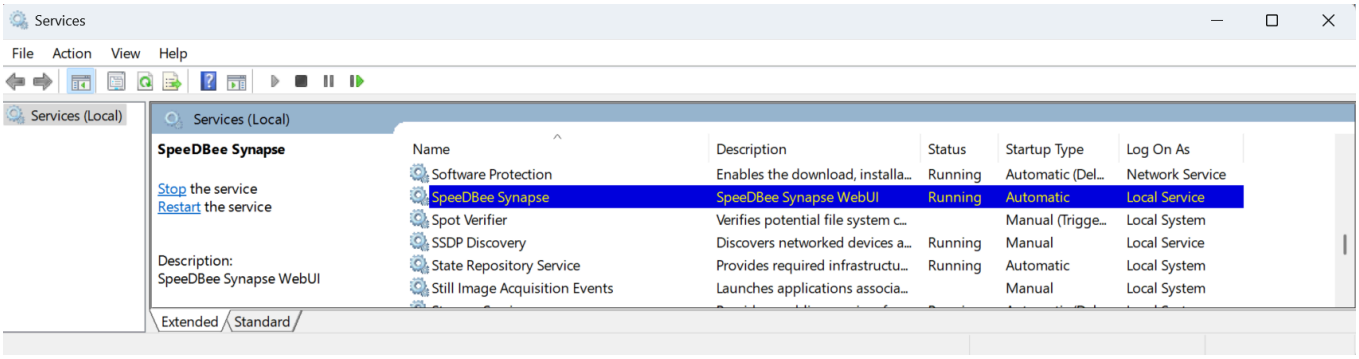


2. The service starts. If it has started successfully, the status shows Running.

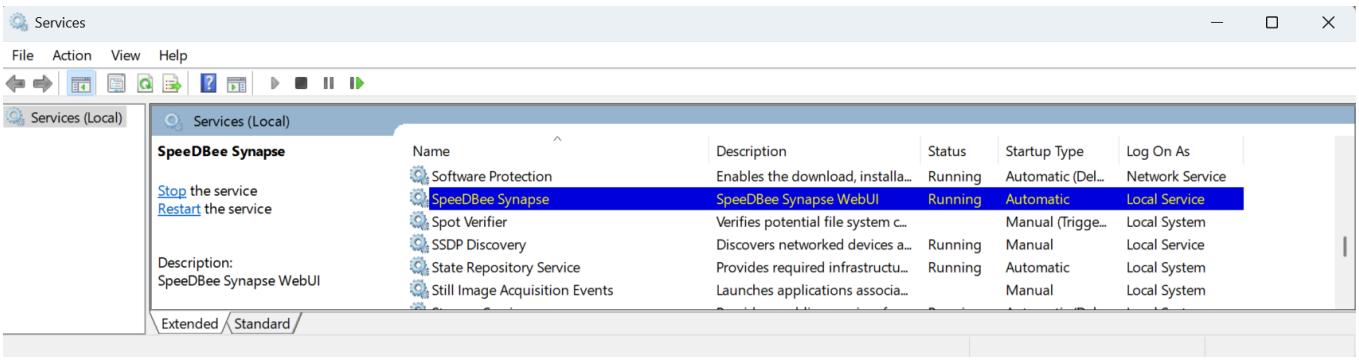


4.3.3.2 Restart

1. Select SpeeDBee Synapse and click Restart Service.

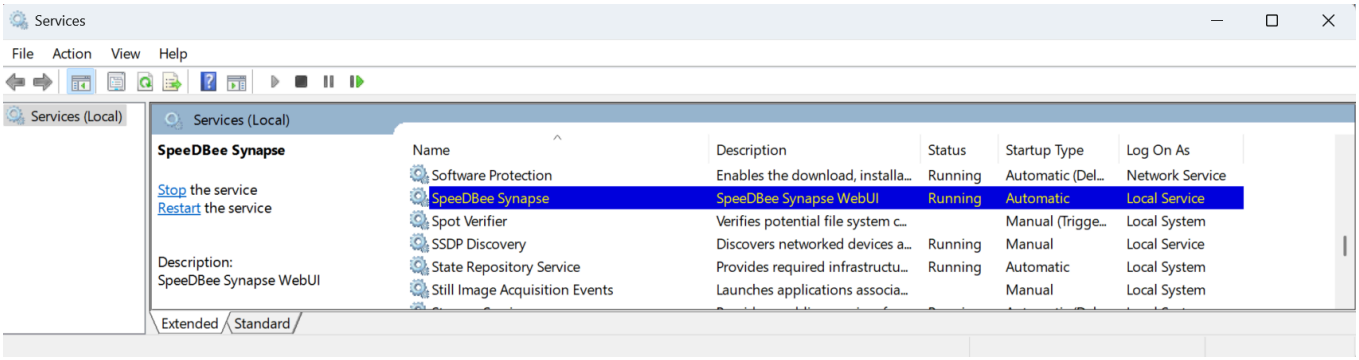


2. The service restarts.

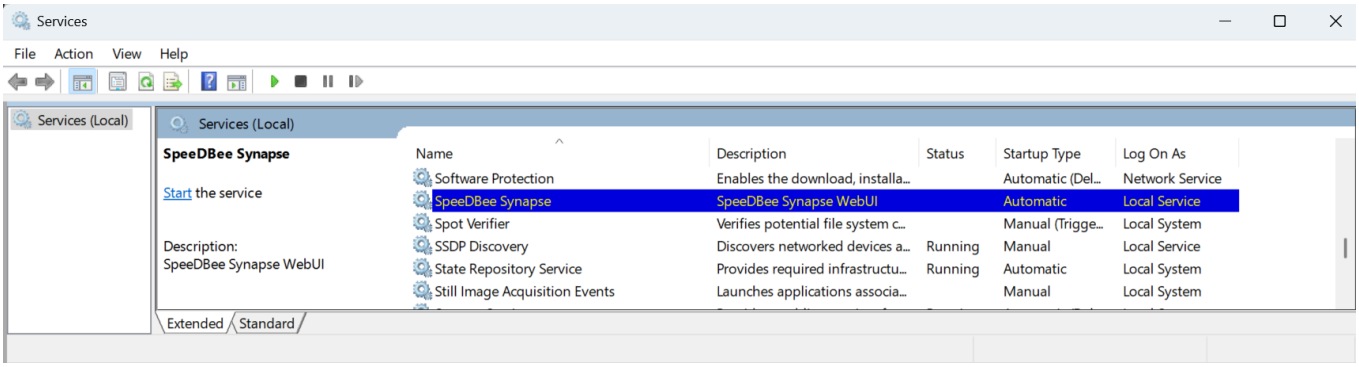


4.3.3.3 Stop

1. Select SpeeDBee Synapse and click Stop Service.

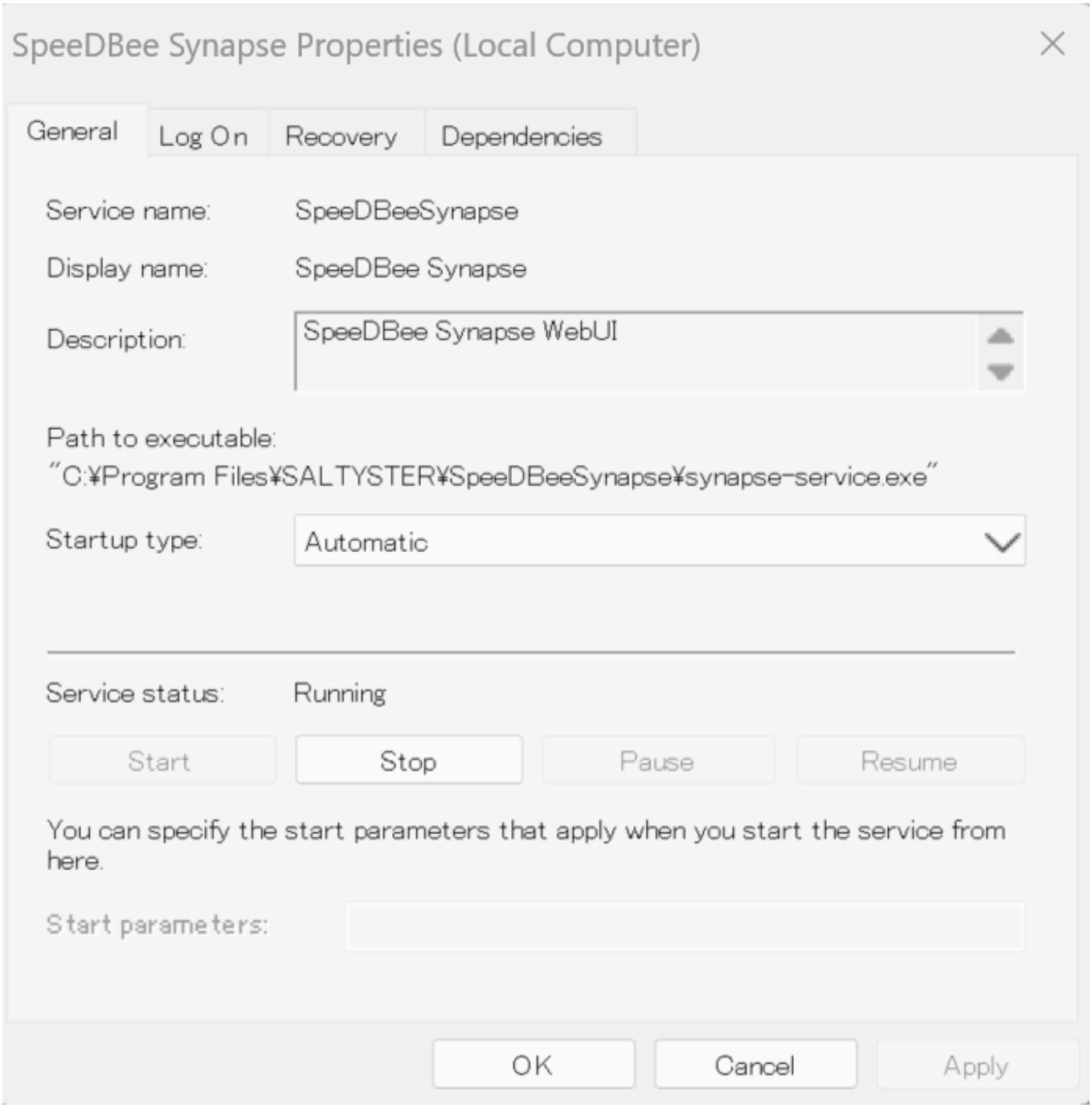


2. The service stops.



4.3.3.4 Enabling Autostart

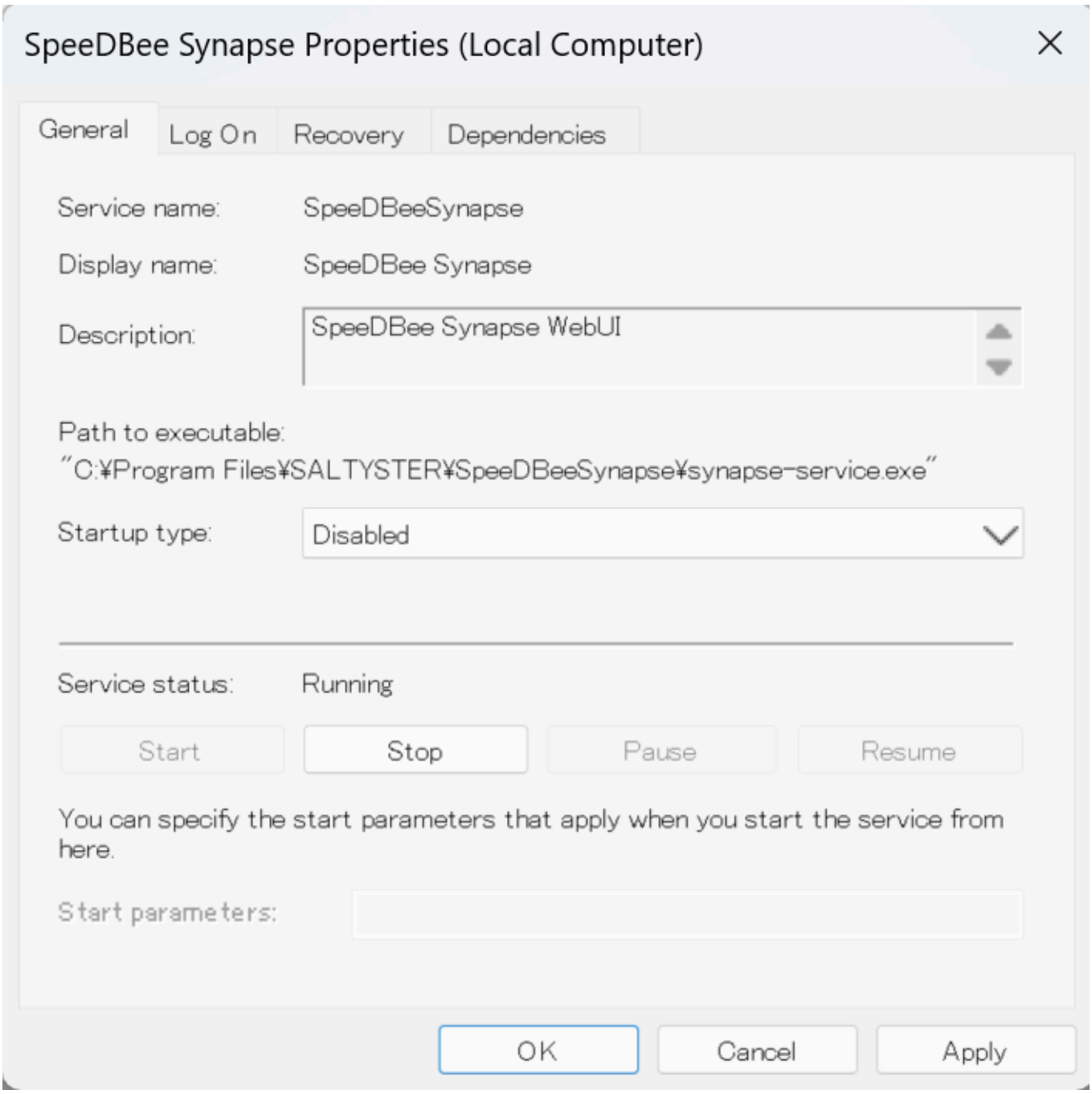
1. Select SpeeDBee Synapse, then click Properties from the right-click menu.
2. Set Startup Type to Automatic, and click OK.



3. With this setting, the SpeeDBee Synapse service will start automatically when the PC is started.

4.3.3.5 Disabling Automatic Startup

1. Select SpeeDBee Synapse, then click Properties from the right-click menu.
2. Set Startup Type to Disabled, and click OK.

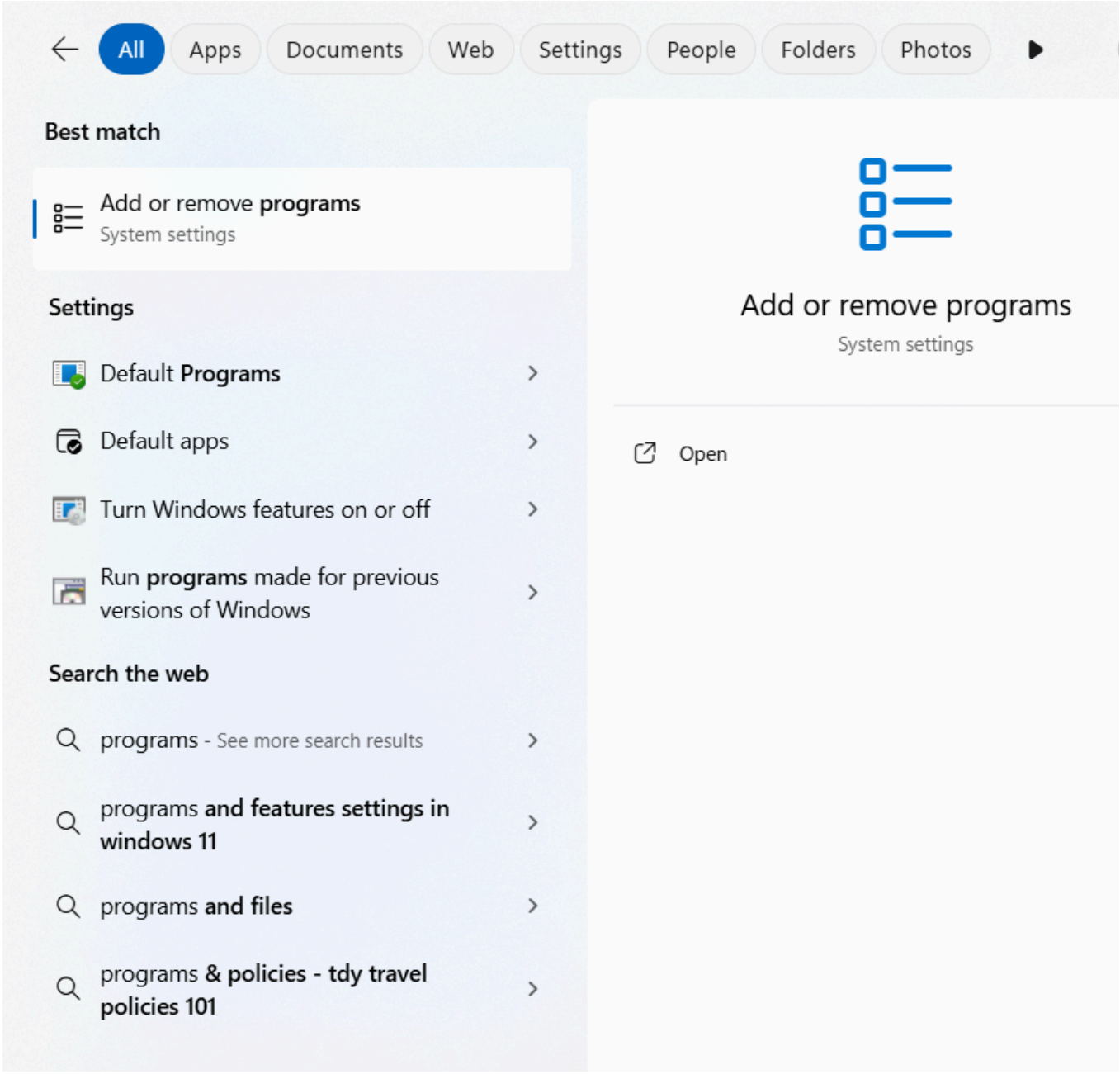


3. When this setting is applied, the SpeeDBee Synapse service does not start when the PC is started. If you want to start it, you need to start it manually.

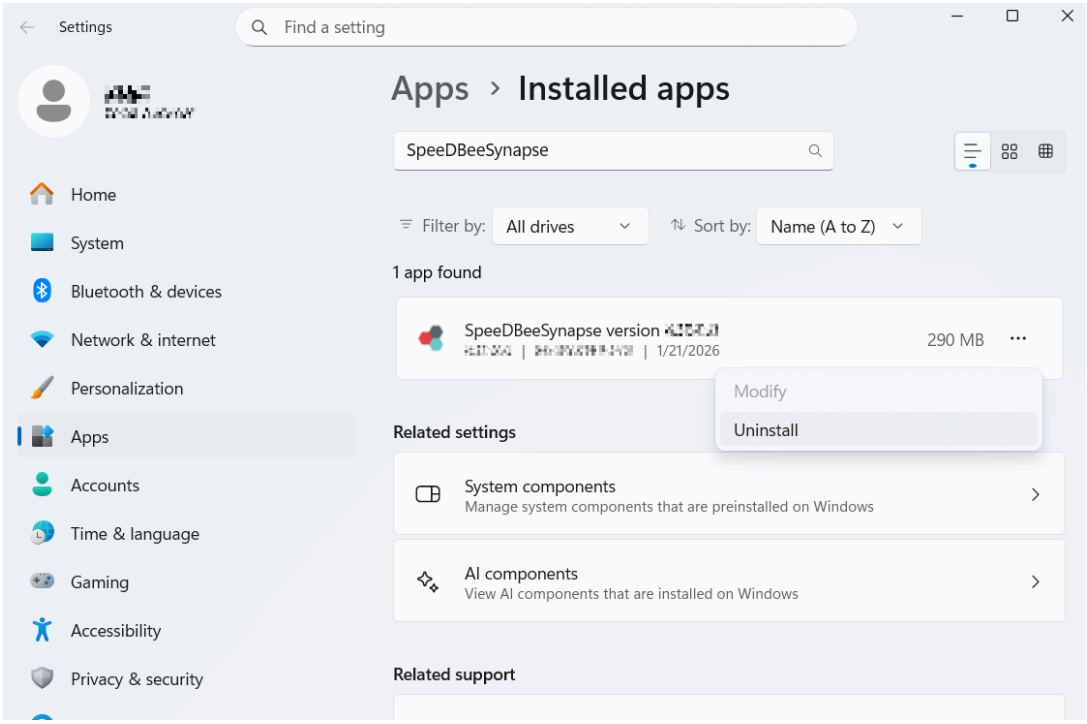
4.3.4 Uninstall

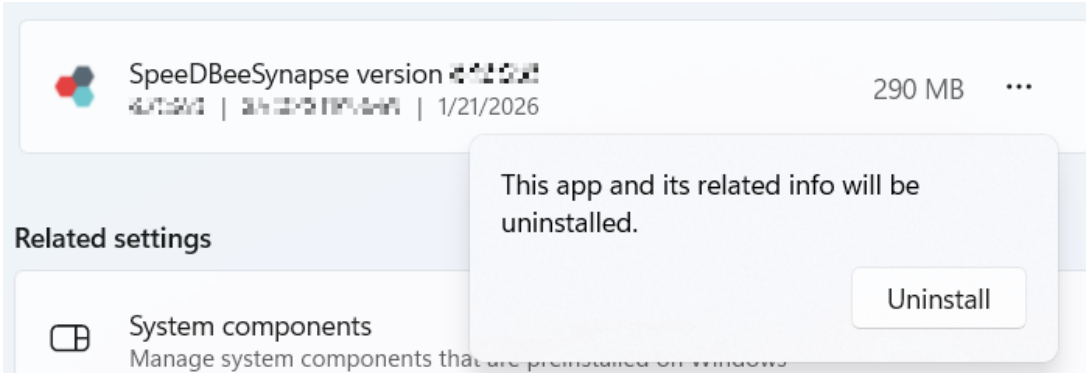
You can uninstall SpeeDBee Synapse from the device by following the instructions below.

1. Type “Add or Remove Programs” into the search field at the bottom of the desktop, and then launch “Add or Remove Programs” from the search results.



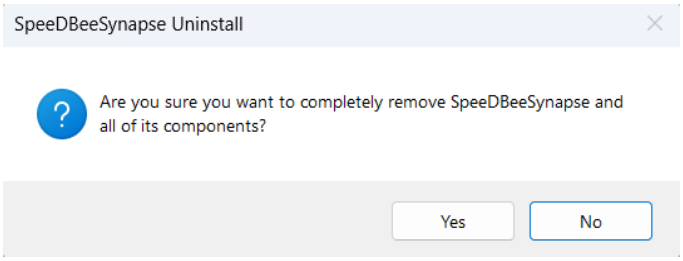
2. Enter "SpeeDBee Synapse" into the search field of installed applications, and click “Uninstall” from the menu of SpeeDBee Synapse from the search results.



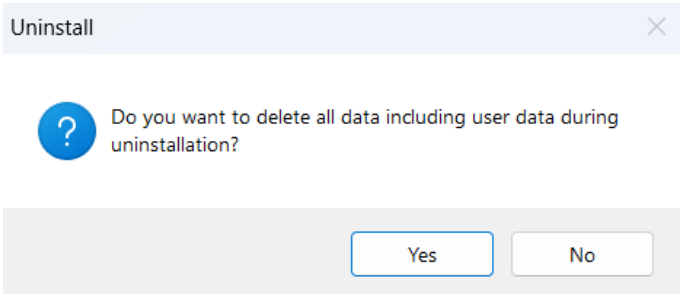


- You must have Windows administrator privileges to perform the uninstall.

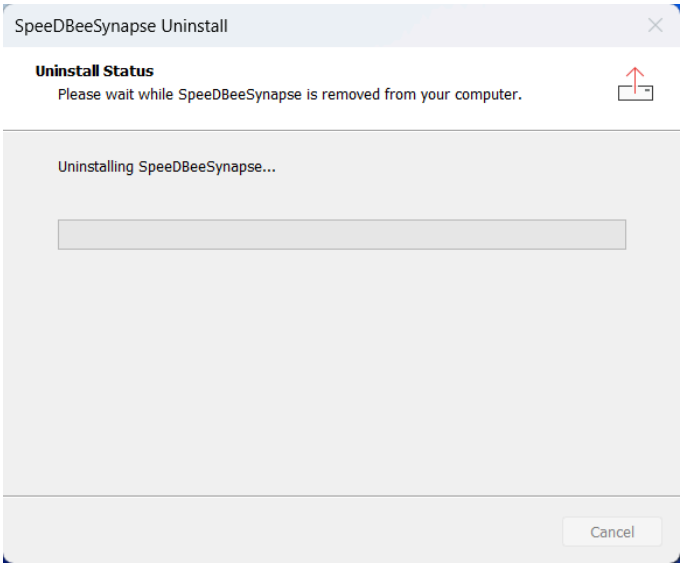
3. When the confirmation message is prompted, click Yes.



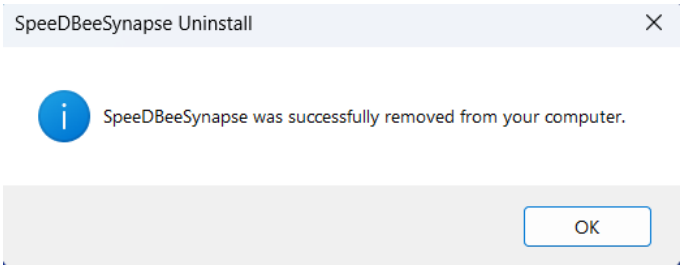
4. Then check for the following message. Click Yes to delete all data (information stored in ProgramData). To keep the data, click "No".



5. The uninstall process is performed.



6. Upon completion, click OK. This completes the uninstallation of SpeeDBee Synapse.



4.4 Initial Setting

This section describes how to perform initial settings (registering licenses, changing admin user password) and how to add users.

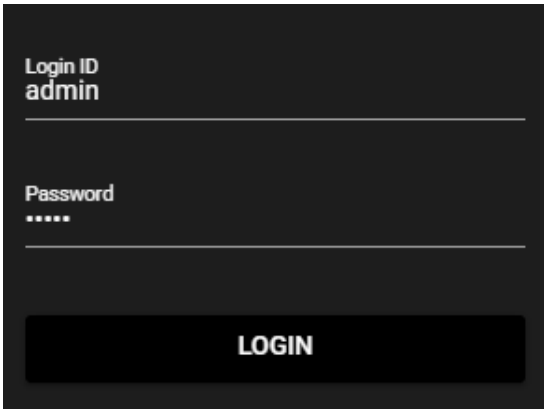
4.4.1 Changing the Administrator Password

As an initial setting, register the license and change the password for the admin user from the default settings. admin is the default registered user in SpeeDBee Synapse. The default settings for the admin user are:

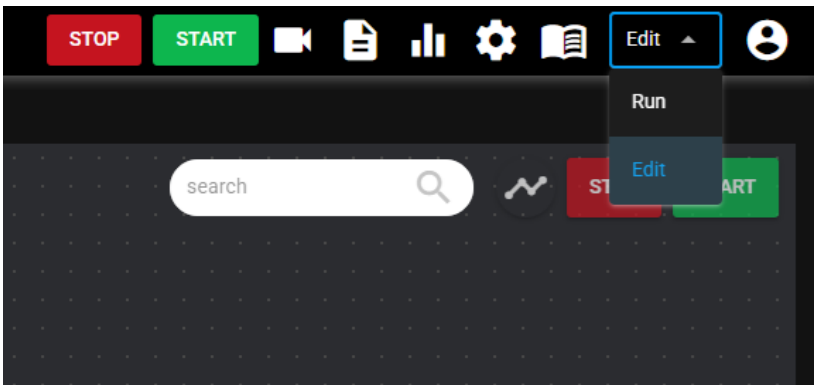
Item	Description
Login ID	admin
User name	admin
Privileges	System administrator
Password	admin
E-mail address	admin@example.com

> In the Data Flow Controller version and OMRON-SBS for Windows version, the license is already registered by default. In that case, please proceed from step 7.

1. On the login screen, enter “admin” for the login ID and password and press “Login”.



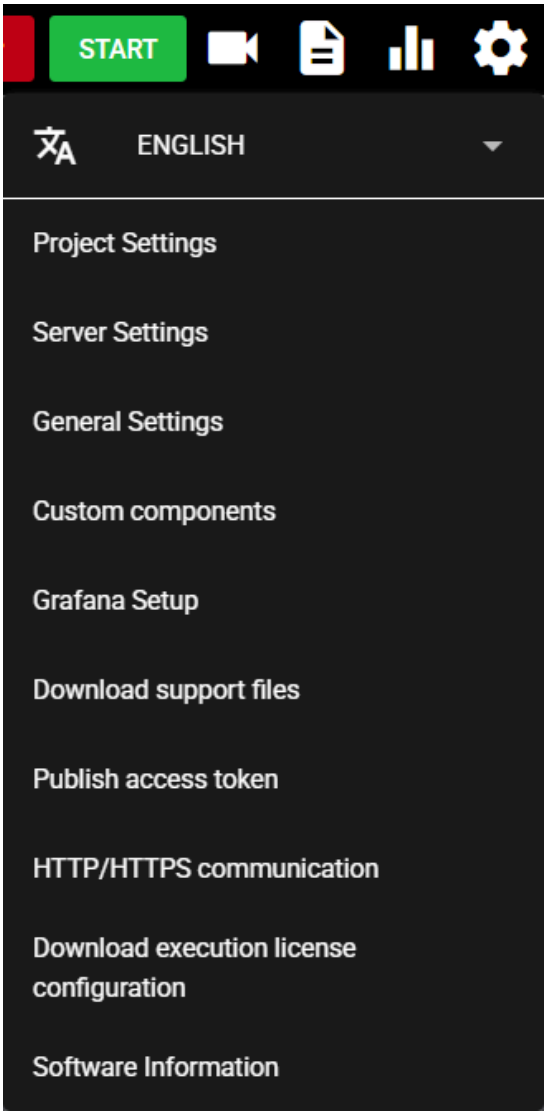
2. Select "Edit mode" in the header mode selection to open the Edit mode screen.



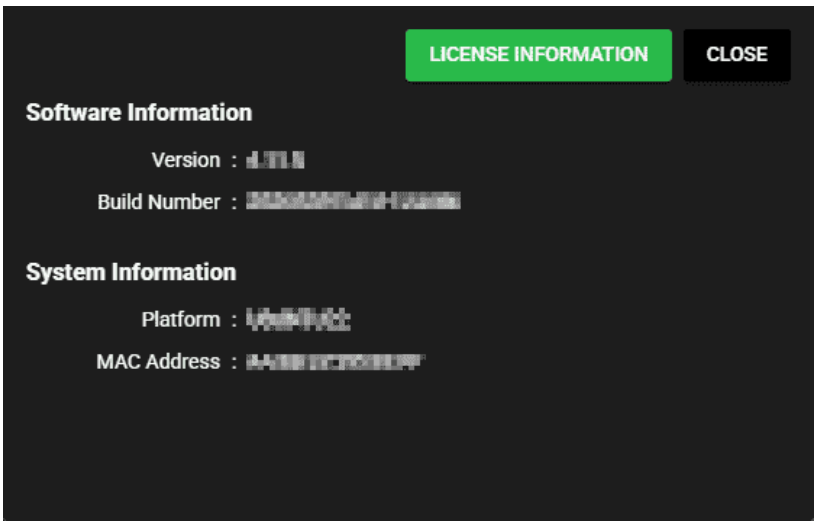
3. Press the Setting menu icon of the Header.



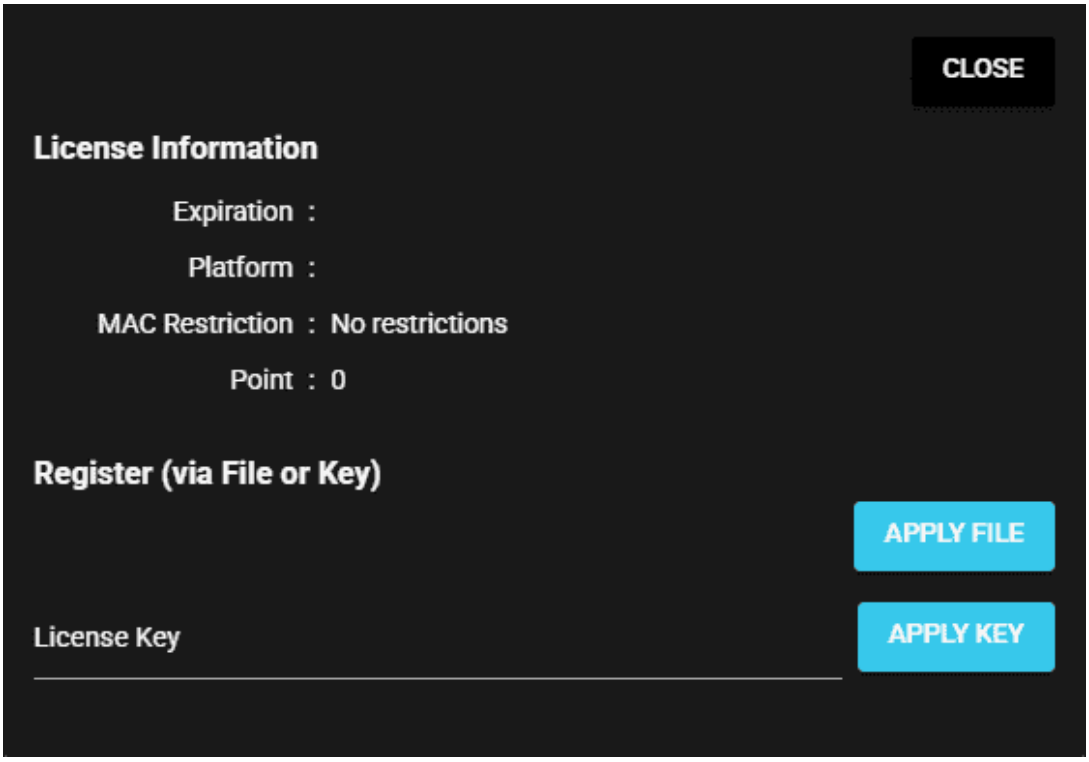
4. Select “Software Information”.



5. Press “License Information”.



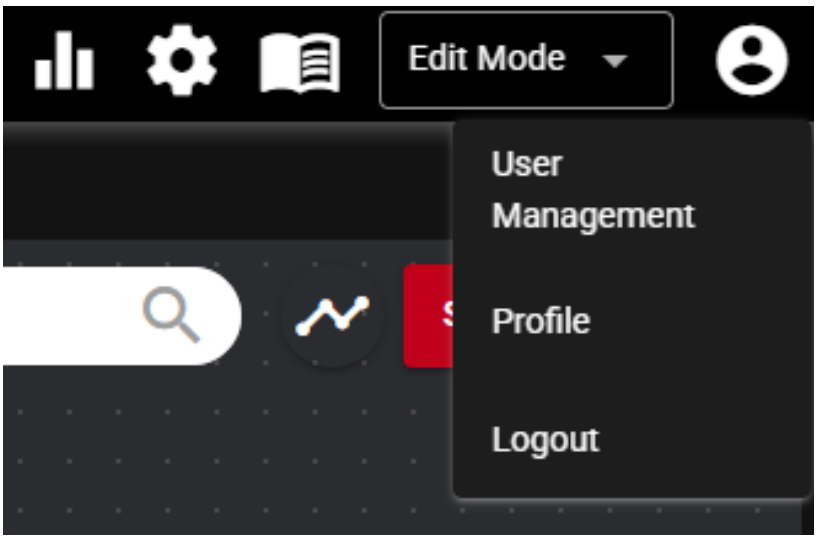
6. Click “Apply File” to select the license file, or enter a license key in the input field and click “Apply Key”.



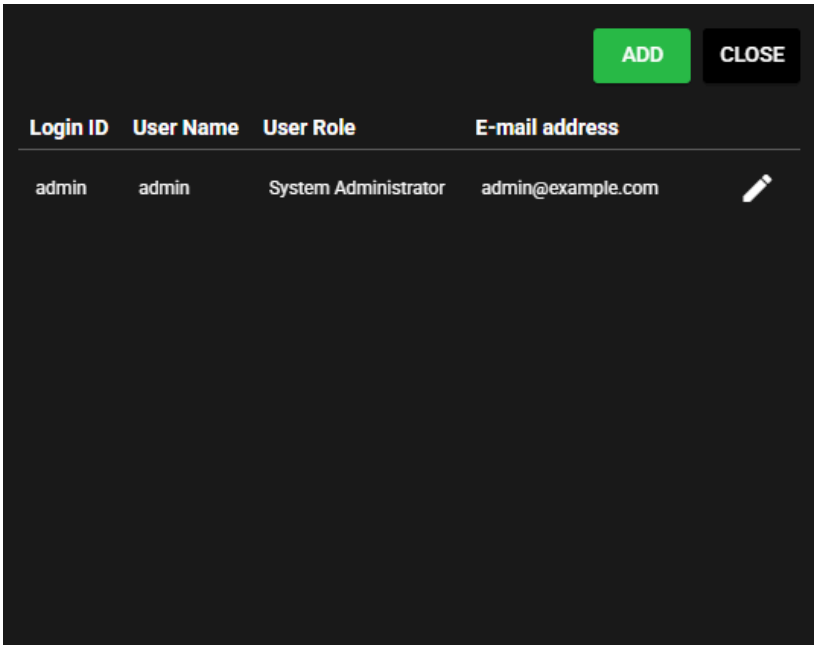
7. If the correct license is applied, the license details are displayed.



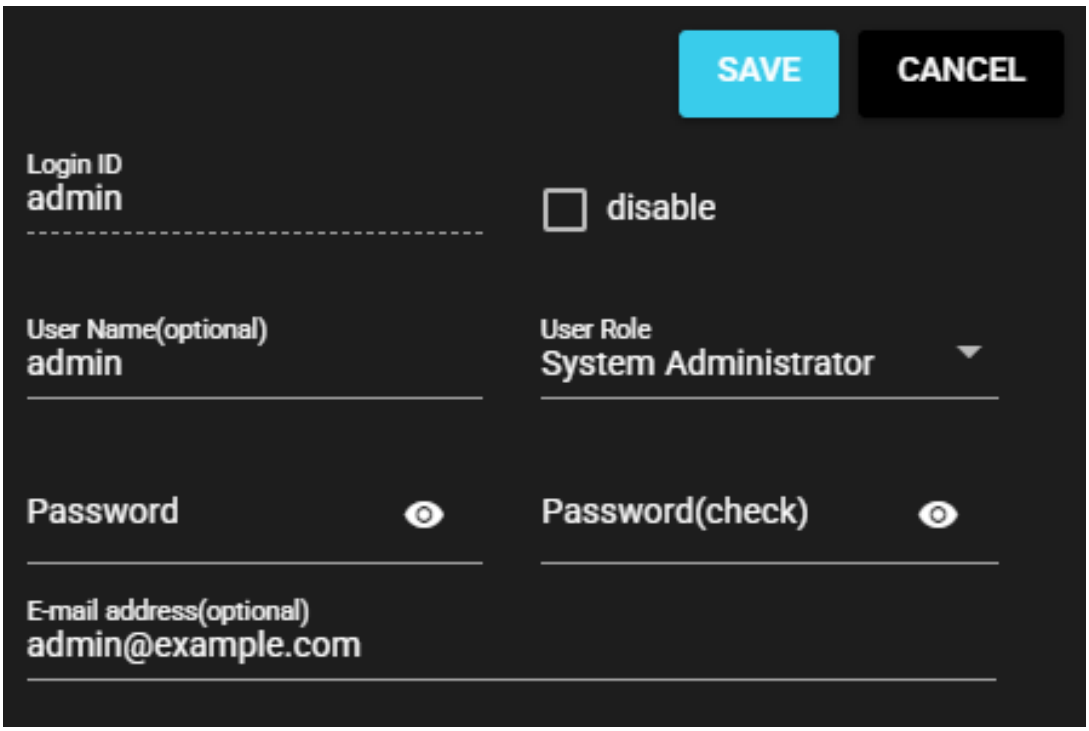
8. Then press the user icon and select “User Management” to open the User Management dialog.



9. Press the edit icon for the "admin" user.



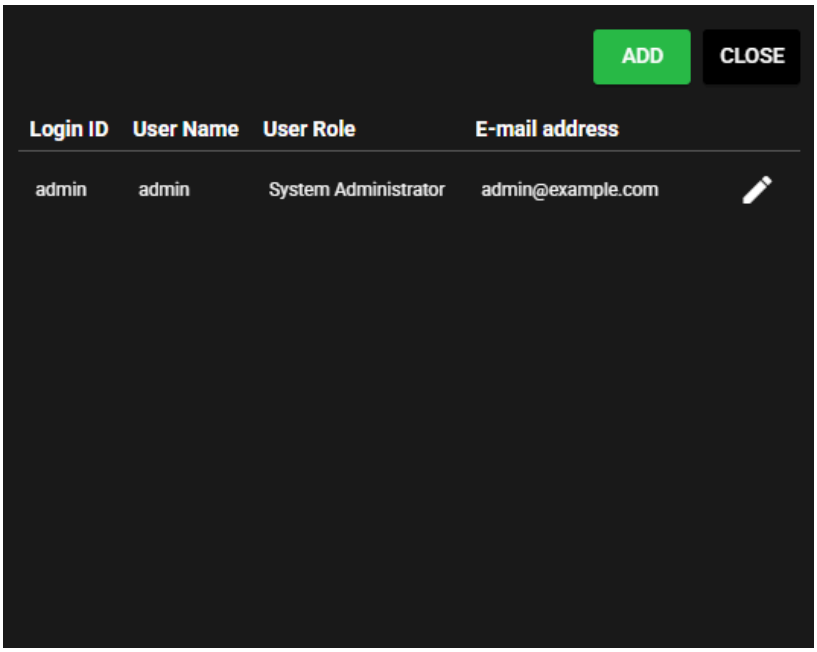
10. Enter any password other than "admin" in “Password” and “Password (Confirm)” and press “Save”.



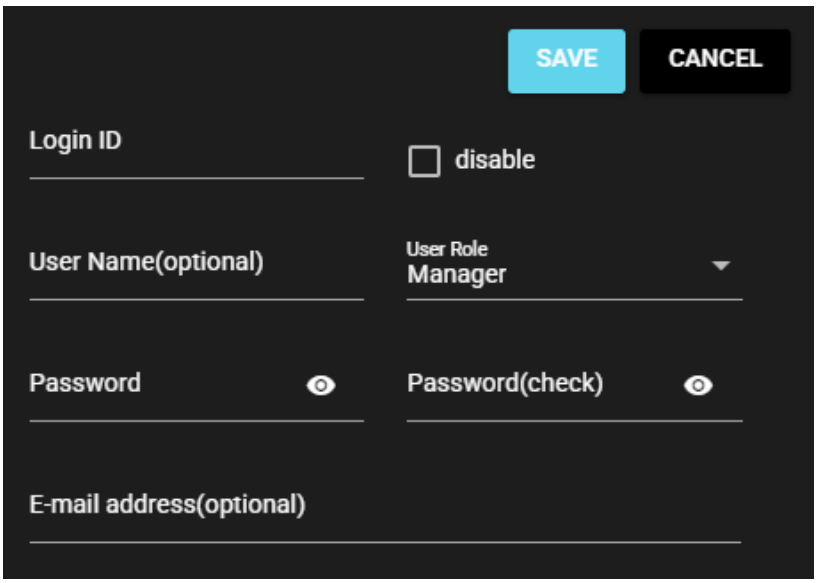
4.4.2 Add User

You can add users by the following procedure.

1. In the User Management dialog, click the “Add” button.



2. Enter the following items. For the privileges, select “Administrator”. When finished, press the “Save” button.

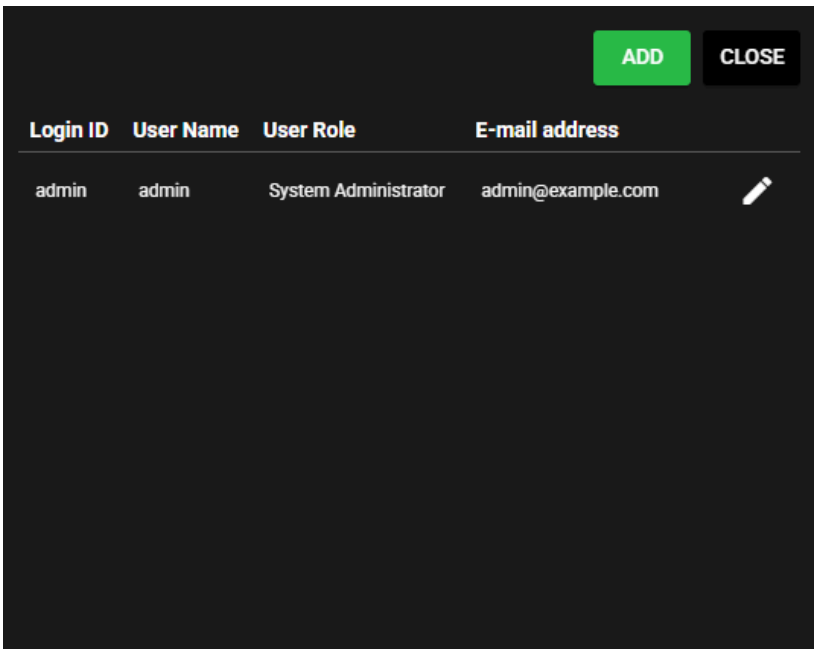


Item	Description
disable	If you select this check box, the user will be disabled and will not be able to log in.
Login ID	This is the ID you enter at login.
User name	This is the name of the user. Note: It is possible to omit the input.
Privileges	See table below.
Password	This is the password to enter at login.
E-mail address	The email address associated with the user. Note: It is possible to omit the input.

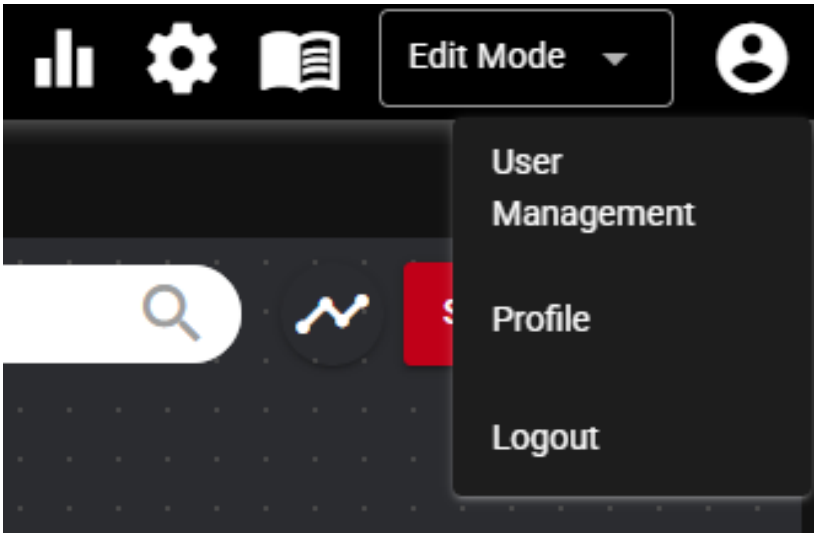
There are four types of user privileges:

Privileges	Description
System administrator	All operations are possible. Granted to initial user "admin" and requires at least one.
Administrator	All operations are possible. You cannot add, update, or delete system administrators.
Operator	This person can view the Run Mode and start and stop components.
Viewer	This person can view the Run Mode.

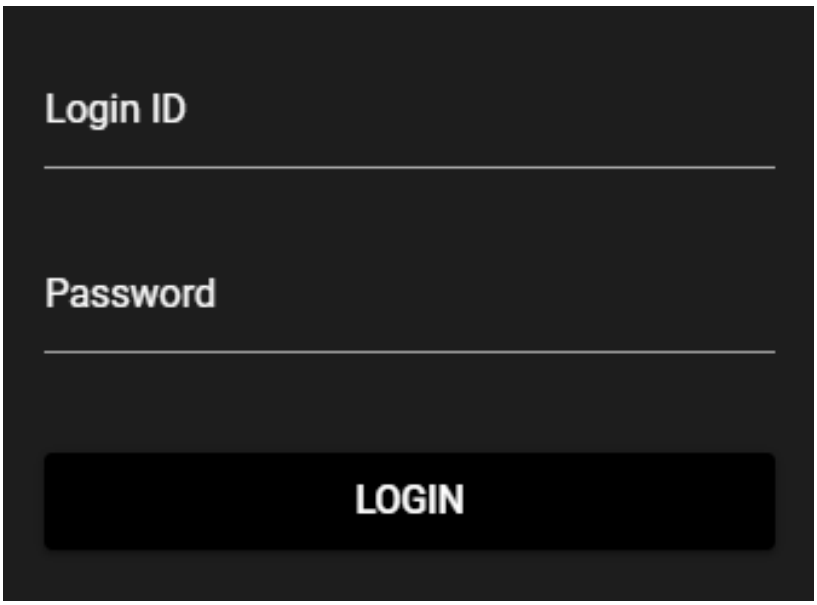
3. In the User Management dialog, click the “Close” button.



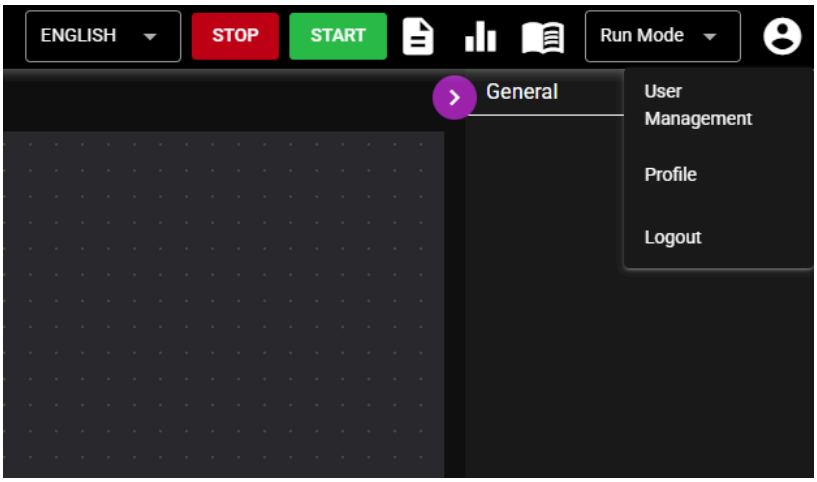
4. Press the user icon and select “Logout”.



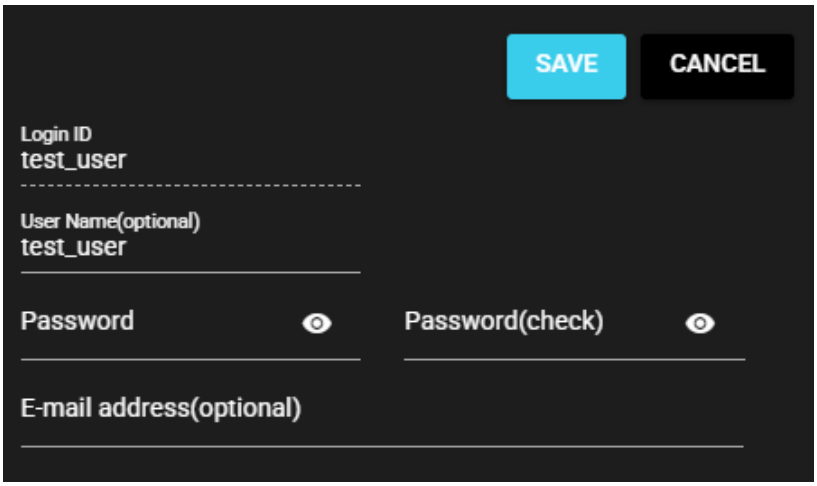
5. At the login screen, enter the login ID and password of the newly added user and press “Login”. This operation must be done by the relevant user him/her self.



6. Press the user icon and select “Profile”.

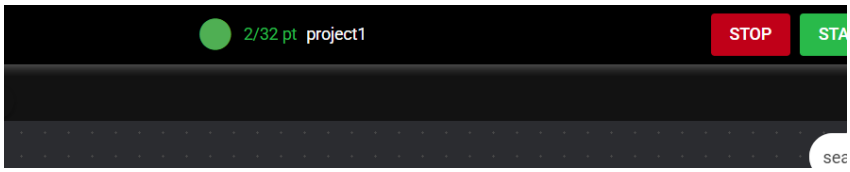


7. The information of the currently logged in user is displayed, and you know that you are logged in as the user you added above. You can change the settings of the currently logged in user by changing the settings and pressing "Save."



4.4.3 Setting of the Project Name

SpeeDBee Synapse manages the specified data flow in units called projects.
Set a project name to make it easier to identify the configured data flow.
(Example: Project name such as "Line A production information visualization," "Facility linkage system," etc. are arbitrarily set up.)
Initially, the name project1 is set, and the project name is displayed in the header.



Refer to the following for details on the setting procedure.

Managing Configuration Information (Project)

4.5 Setting of Components

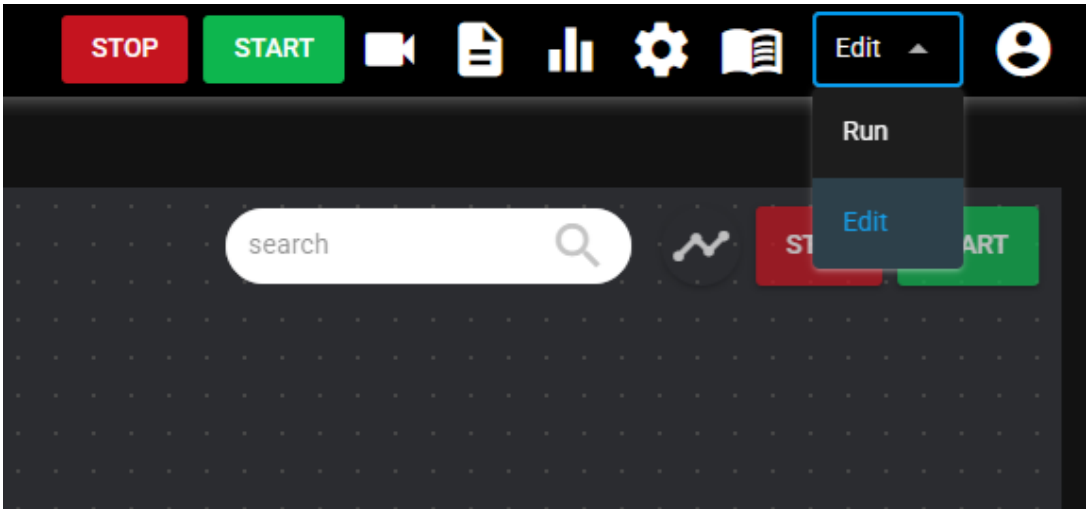
This section describes how to set up the collected data until it is saved to a file.

4.5.1 Configuring the Resource Collector

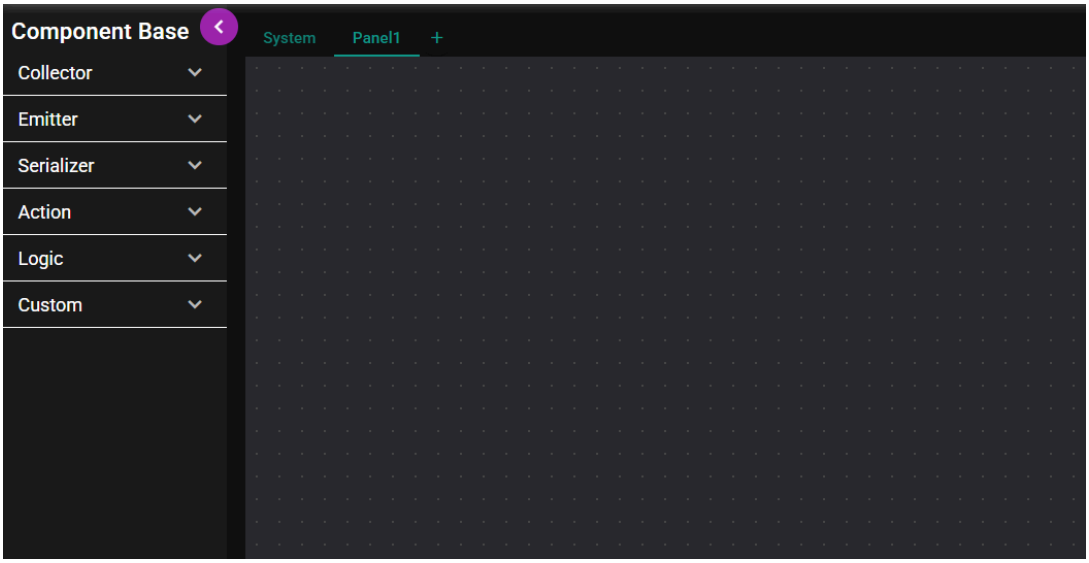
Describes how to configure the "Resource Collector" to collect CPU utilization rate of Operating machine. The "Resource Collector" is a component that collects system resources of Operating machine (such as CPU usage, memory consumption, etc.).

You can configure the resource collector as follows.:

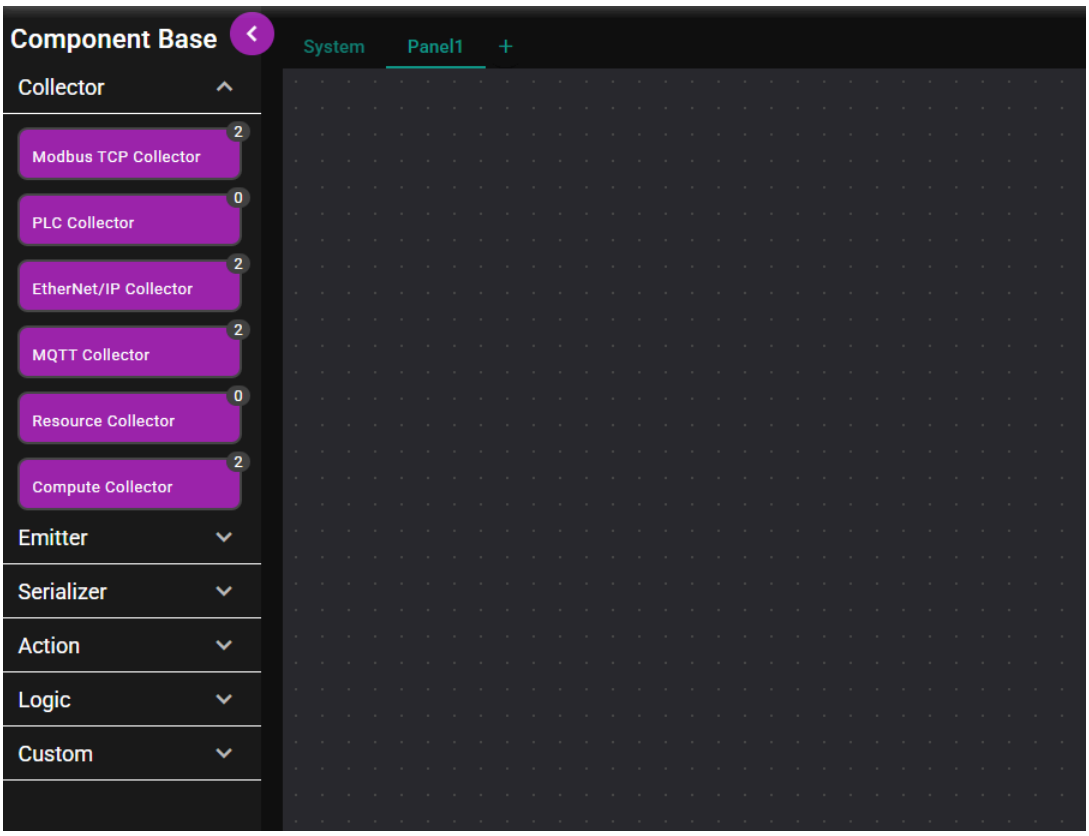
1. Select "Edit mode" in the header mode selection to open the Edit mode window.



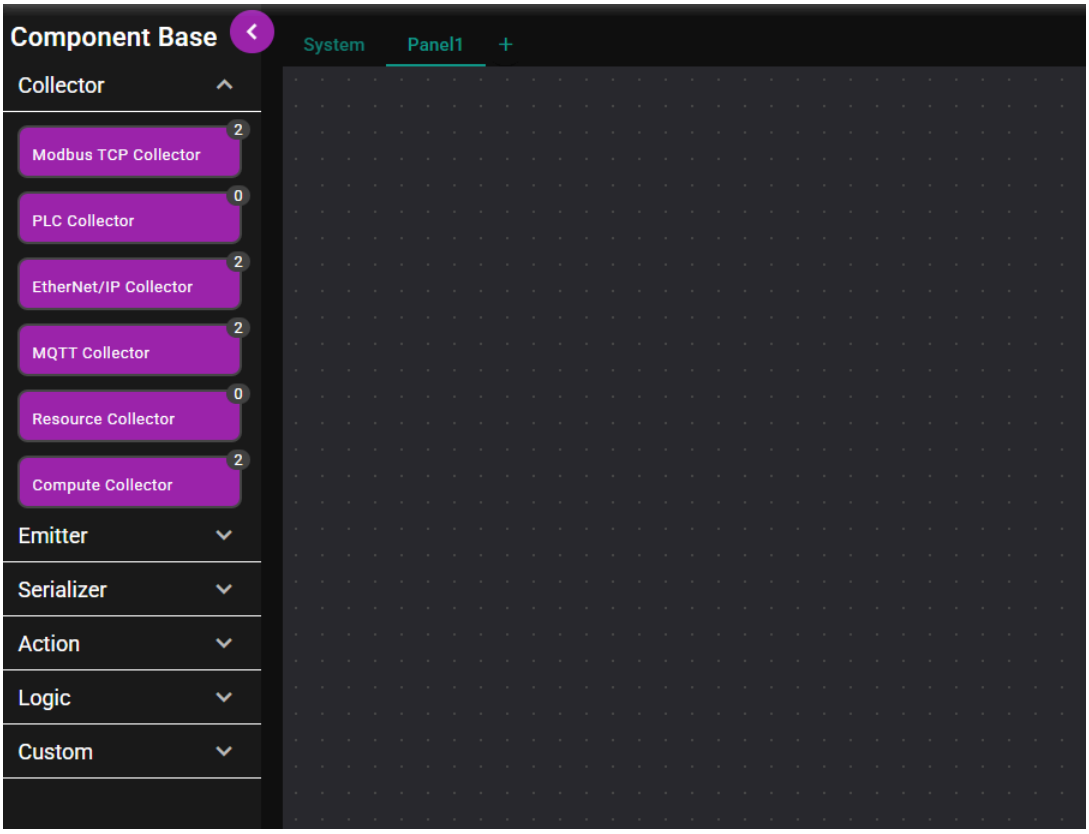
2. Displays the panel 1.



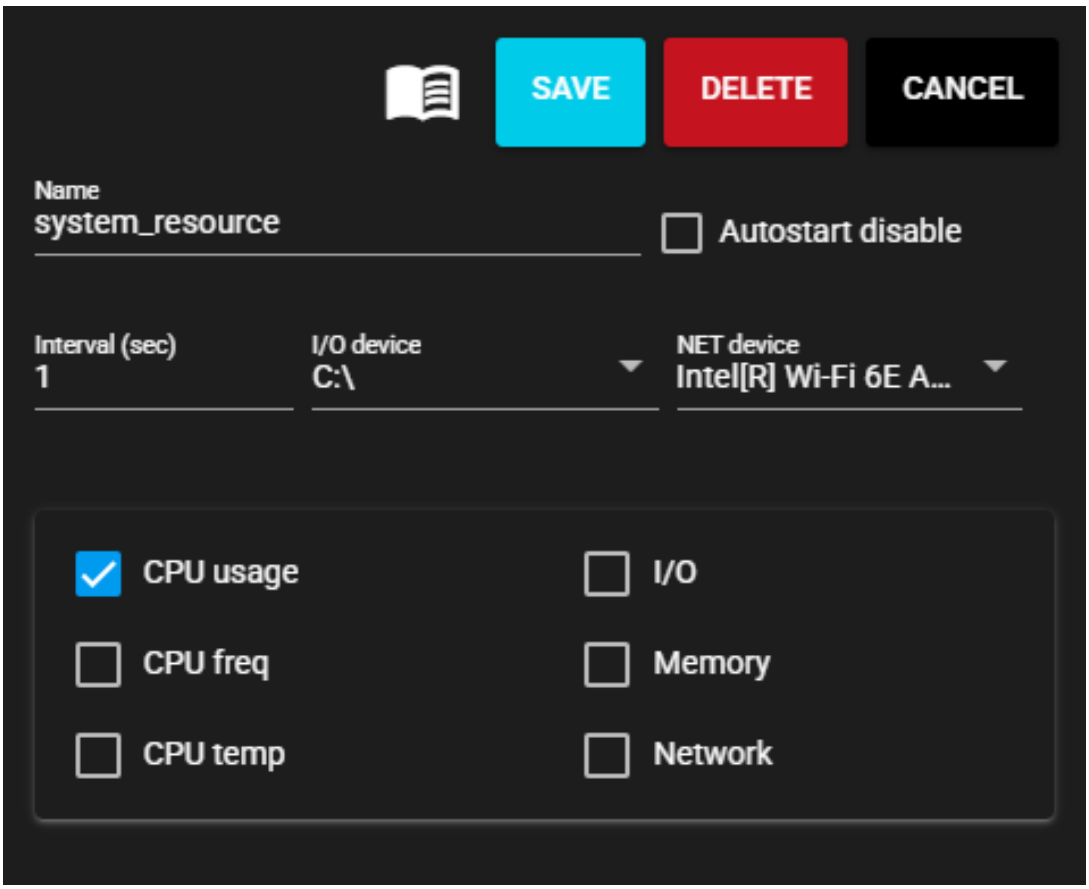
3. Press the "Collector" indicator in the left menu to display the collector list.



4. Drag "Resource Collector" and drop it on the panel.



5. The settings dialog will open automatically. "Enter "name" as desired, check "CPU usage" and click the "Save" button.
   Name example: System resource



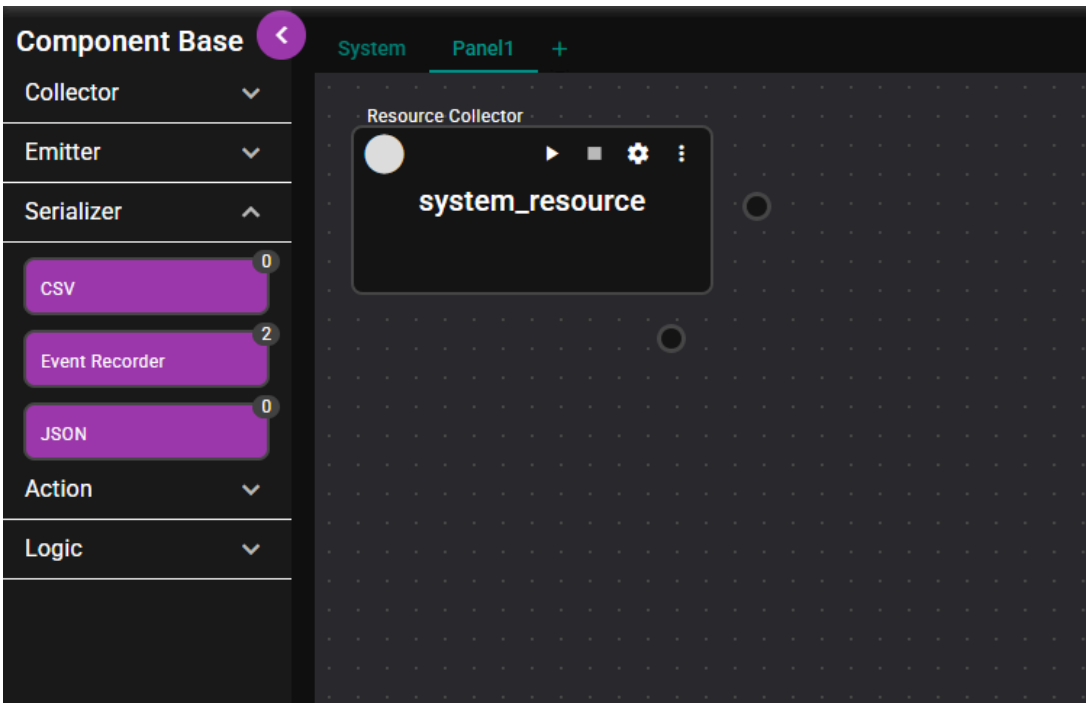
4.5.2 Configuring the CSV Serializer

This section describes how to configure the "CSV serializer" to output the CPU usage rate collected by the resource collector to a file in CSV format. "The "CSV serializer" is a component that takes arbitrary data, processes it into CSV format, and re-outputs it.

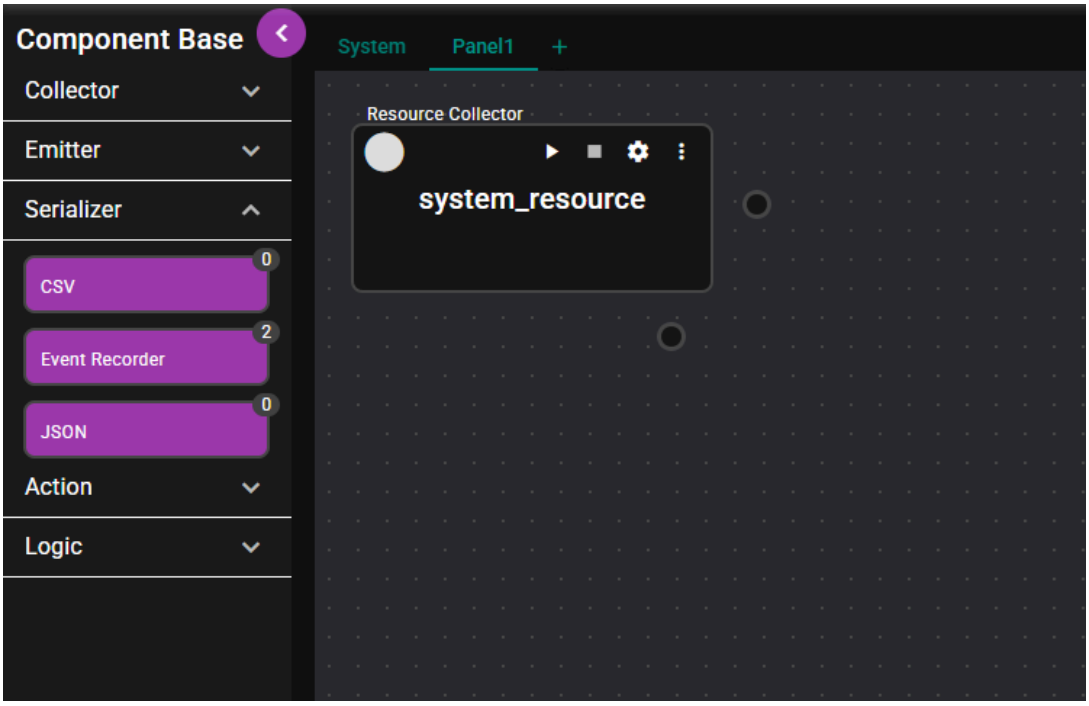
The flow of data between components is defined by Flow links. A flow link connects two components and flows data between the connected components.

You can configure the CSV Serializer by the following procedure.:

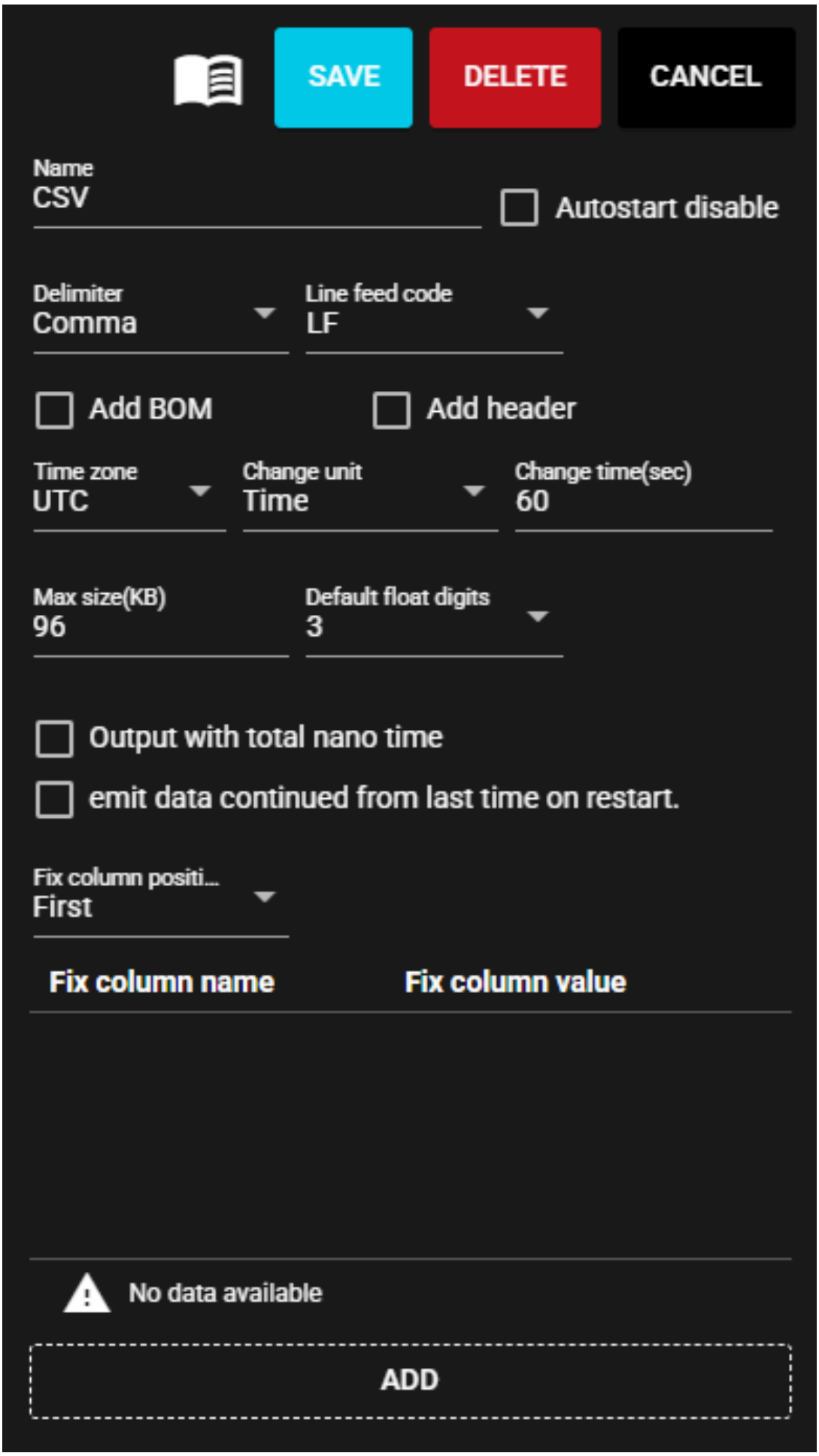
1. In the Edit mode window, press the "Serializer" indicator in the left menu to display the Serializer List.



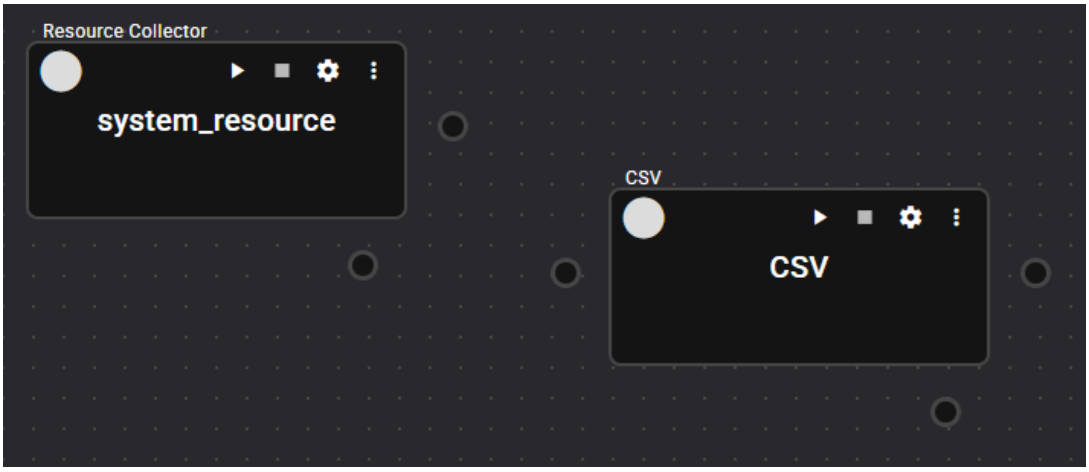
2. Drag the “CSV Serializer” and drop it on the panel.



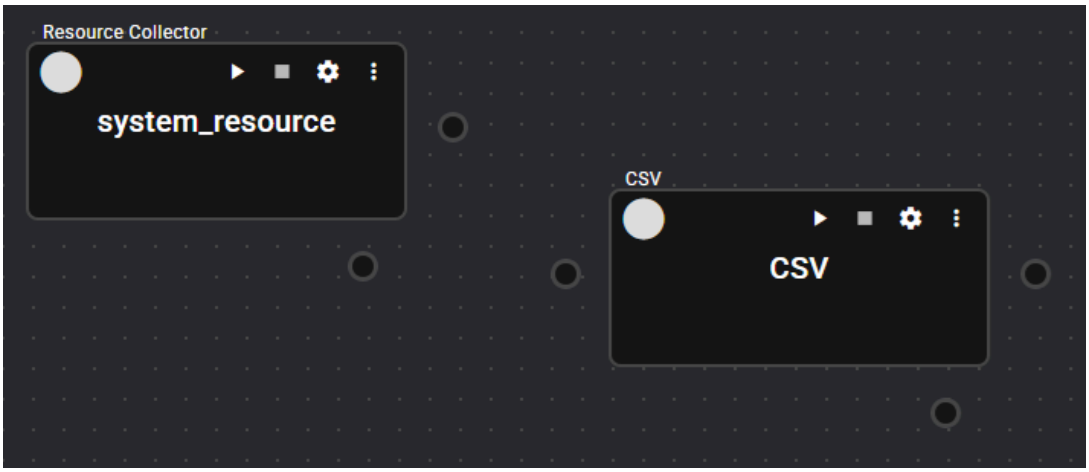
3. The settings dialog will open automatically. "Enter "name" as desired, and click the "Save" button.
   Name example: CSV format


SAVE
DELETE
CANCEL
Name
CSV
Autostart disable
Delimiter
Comma
Line feed code
LF
Add BOM
Add header
Time zone
UTC
Change unit
Time
Change time(sec)
60
Max size(KB)
96
Default float digits
3
Output with total nano time
emit data continued from last time on restart.
Fix column positi...
First
Fix column name
Fix column value
No data available
ADD

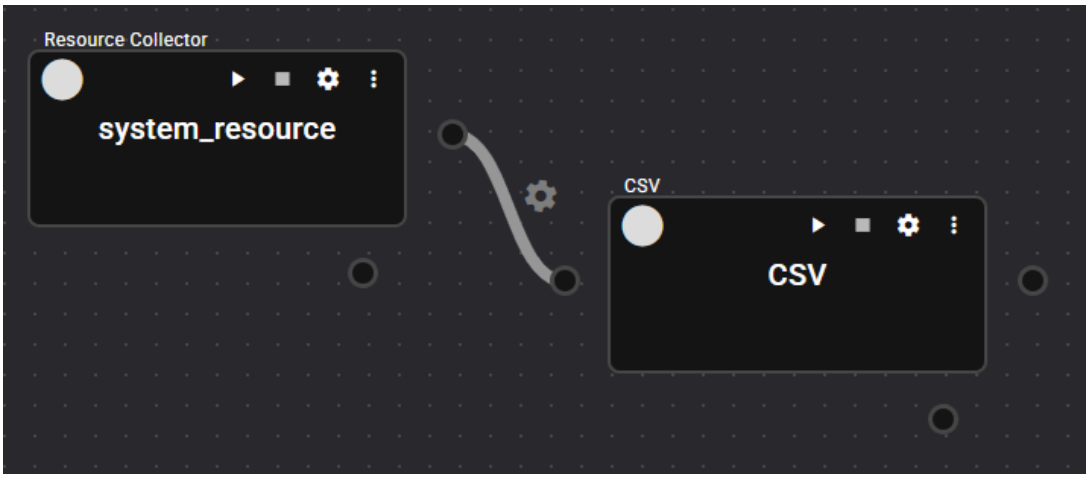
4. Drag the output port of the Resource Collector.



5. Release the mouse button at the input port of the CSV Serializer.



6. A flow link is added between the components.

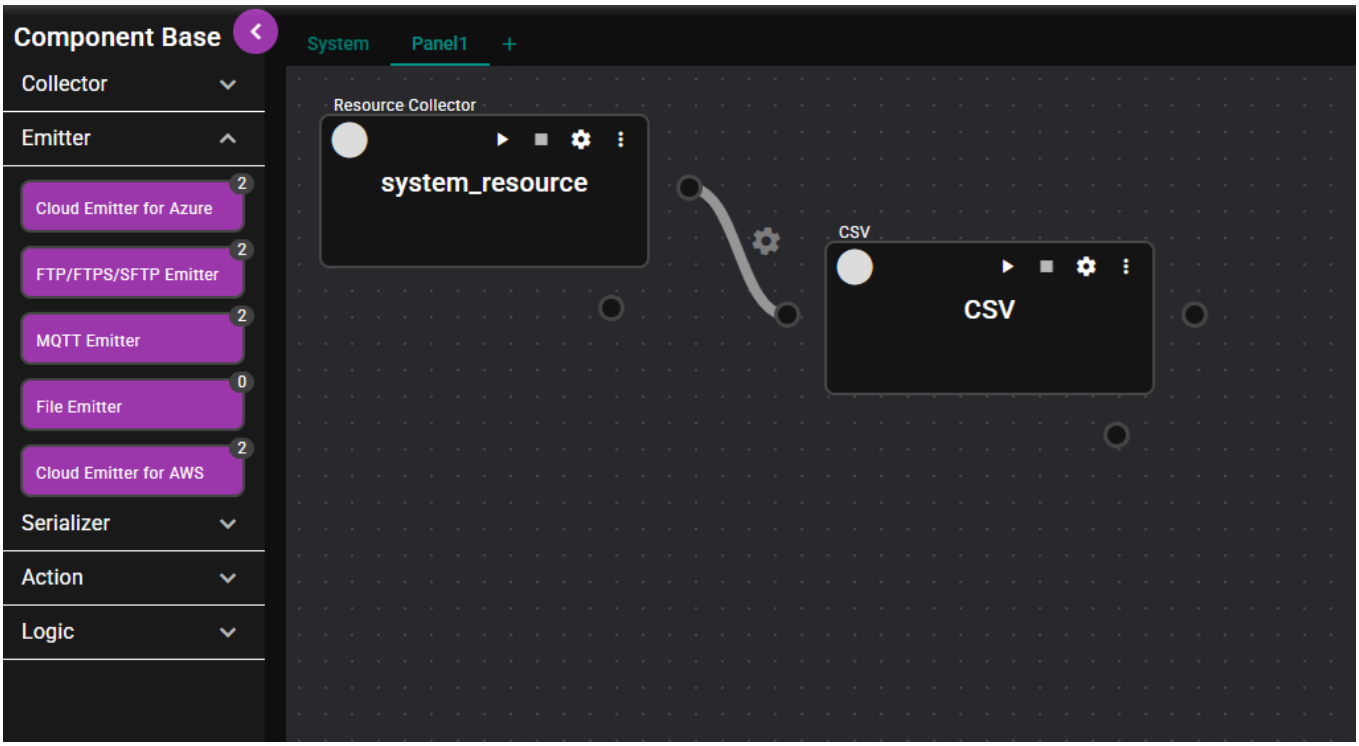


4.5.3 File Emitter Settings

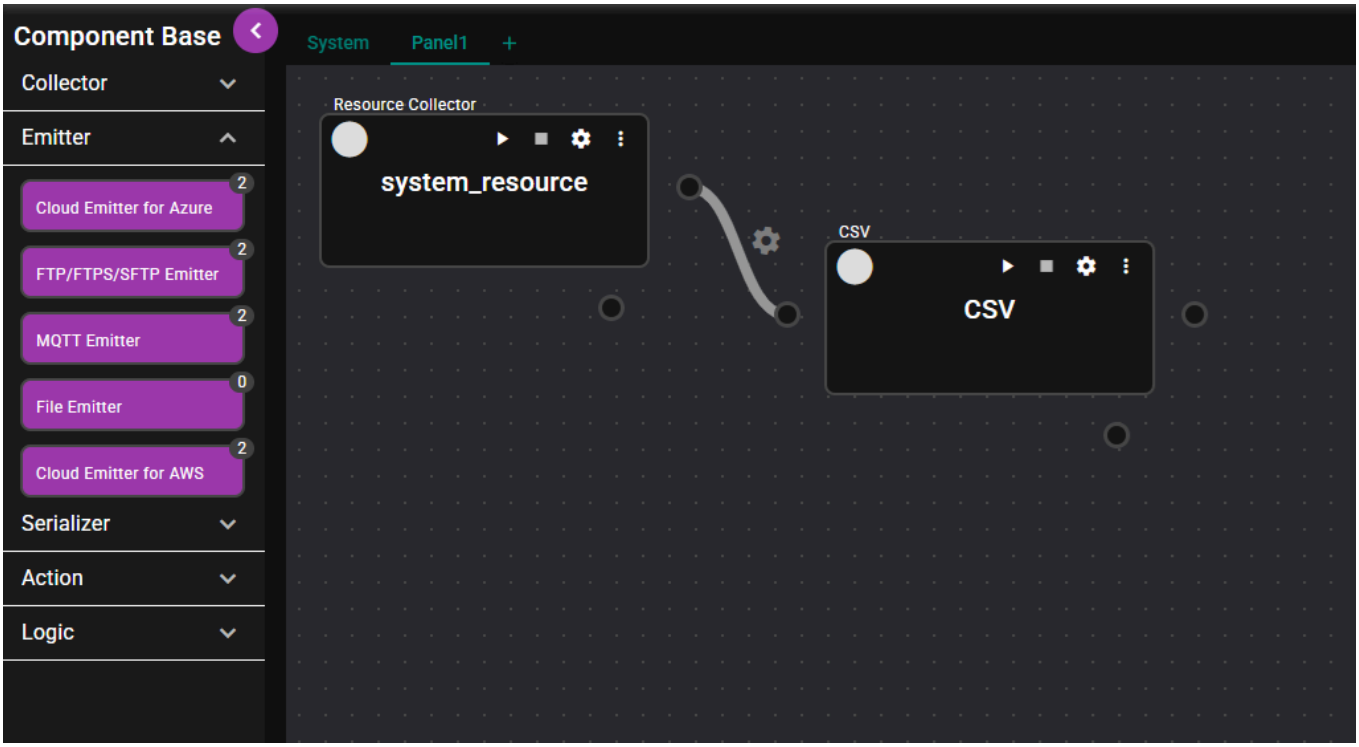
Next, we'll explain how to set up the "File Emitter". "The "File Emitter" is the component that receives the data processed by the serializer and outputs it to the local environment as a file.

You can configure the File Emitter by the following procedure.:

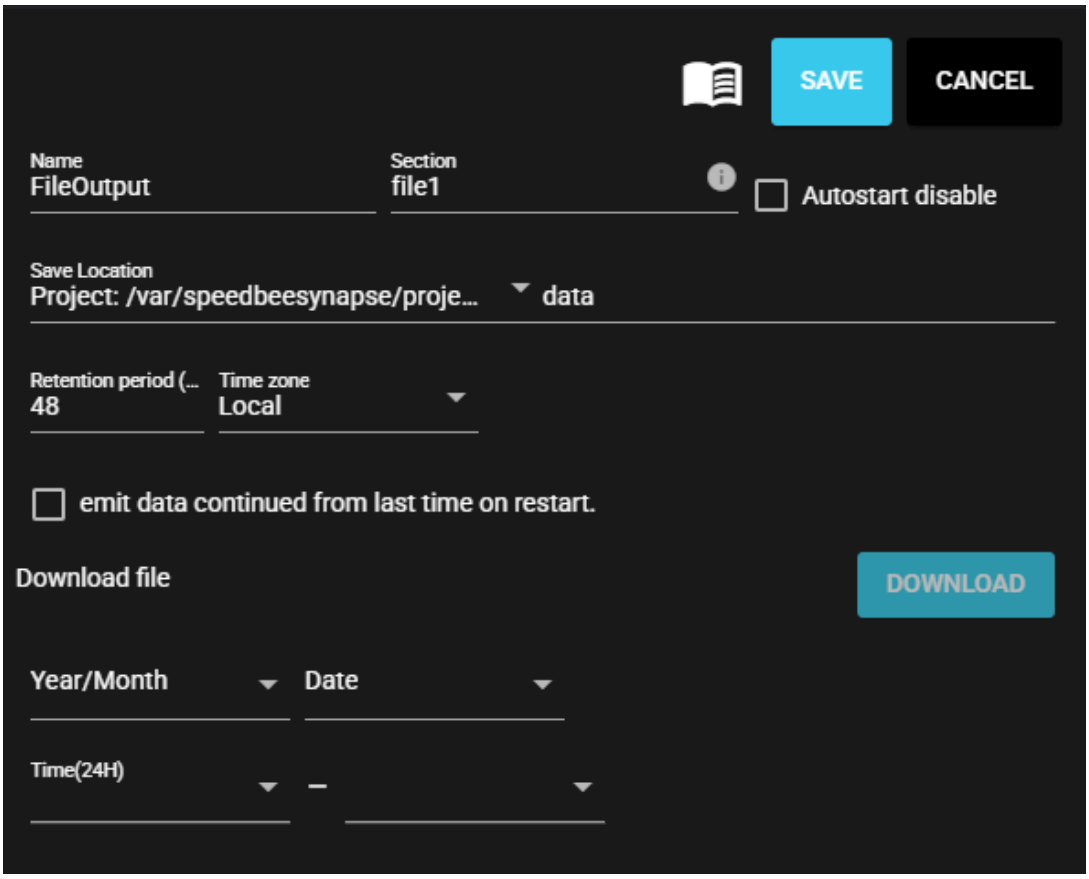
1. In the Edit mode window, press the "Emitter" indicator in the left menu to display the Emitter List.



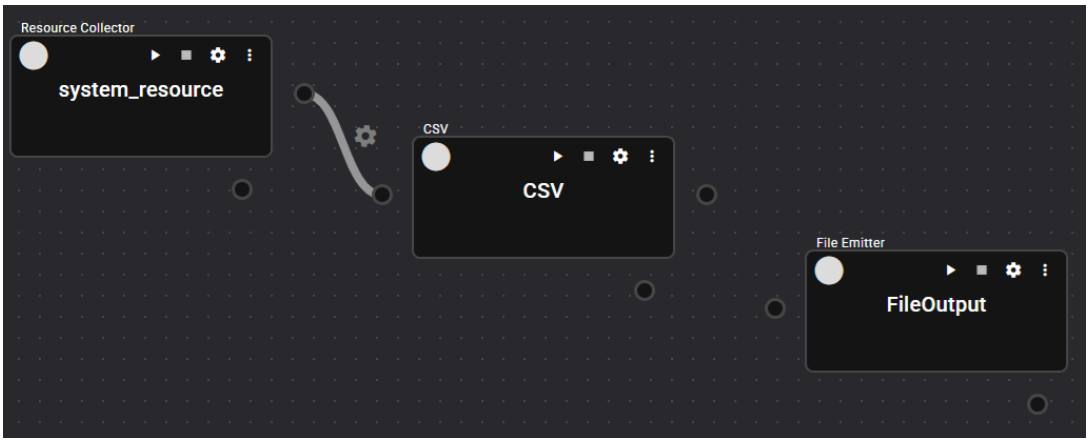
2. Drag “File Emitter” and drop it on the panel.



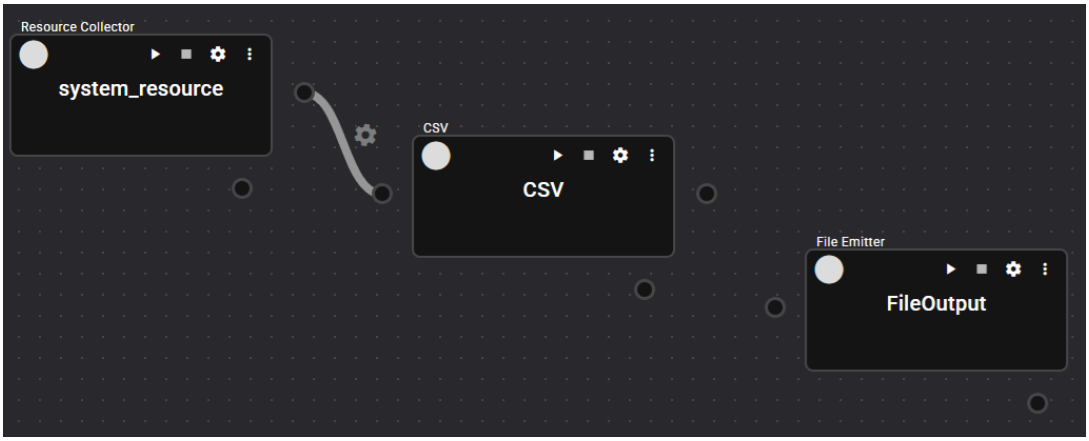
3. The settings dialog will open automatically. "Enter "Name" and “Category” as desired, and click the "Save" button.
   File name sample: File Output
   Category sample: file1
   Note: Only half-width characters can be entered for "Category".



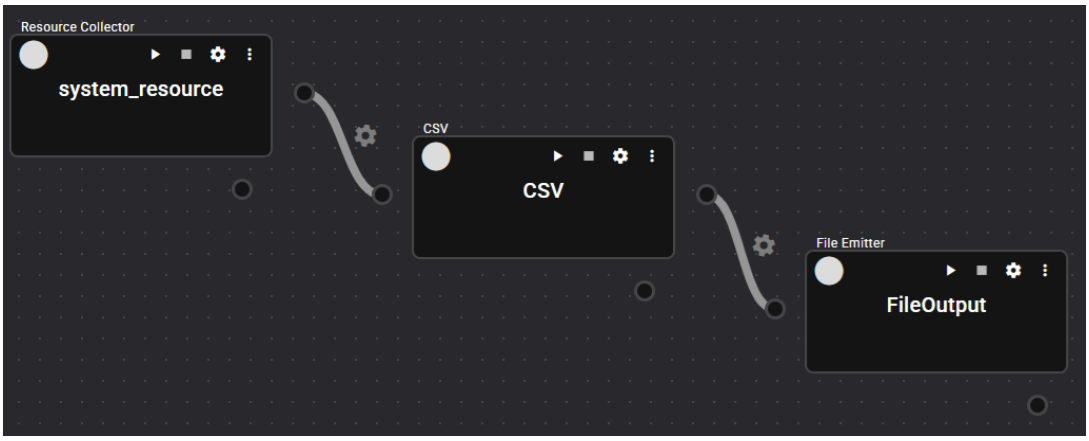
4. Drag the output port of the CSV Serializer.



5. Release the mouse button at the input port of the File Emitter.



6. A flow link is added between the components.



4.6 Running the Collection Service

This procedure describes how to start the collection service by starting the component.

It also describes how to verify that the collector, serializer and emitters configured in the previous sections are working properly after the collection service is started, and how to tune the settings.

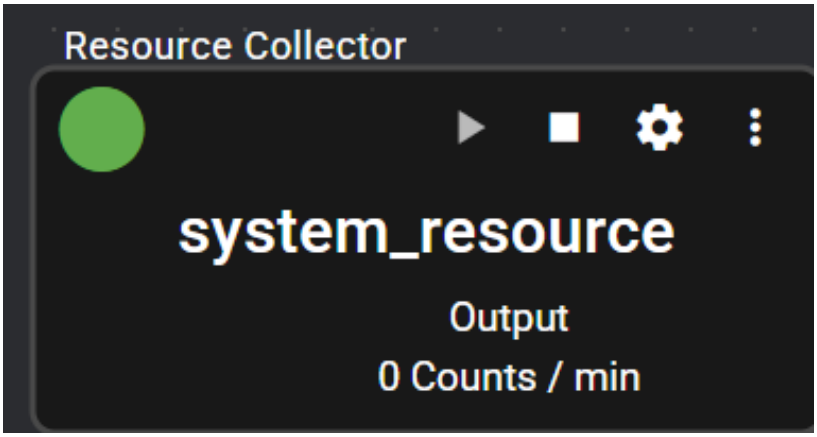
4.6.1 Start/Stop Collection Service

You can start the collection service by starting the component. Components can be activated in three areas: by component, by entire panel, and by entire SpeeDBee Synapse.

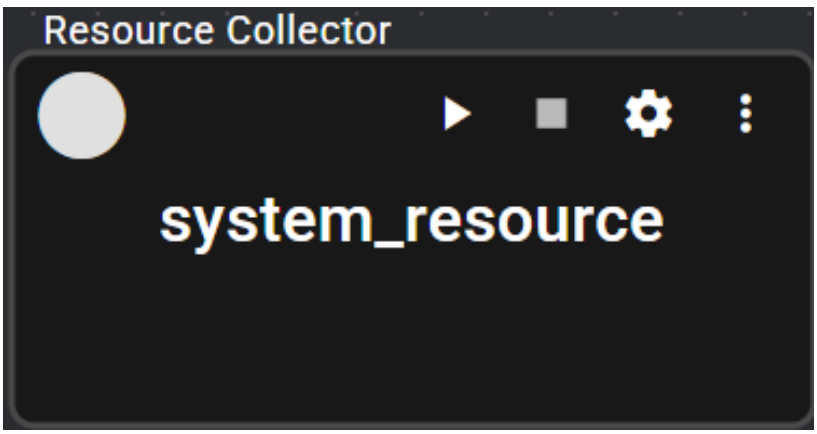
4.6.1.1 Starting/Stopping Components

You can use the following procedure to start or stop a single component.

1. You can start the component by pressing the start icon of the component. When the component starts, the status light turns green.



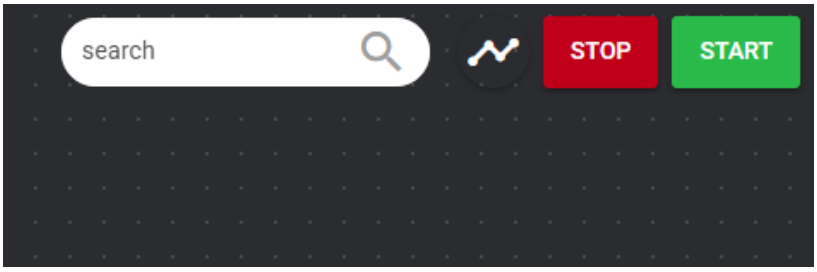
2. You can stop a component by pressing the stop icon of the component. When the component stops, the status light turns gray.



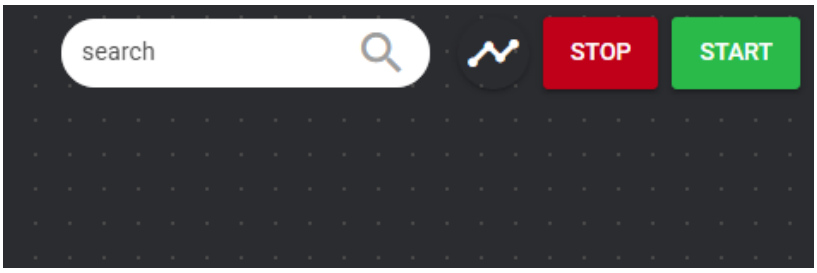
4.6.1.2 Start/Stop All Components on a Panel

You can use the following procedure to start or stop components on a single panel.

1. By pressing the "Start" button on the panel, you can start all the components on the panel for which "Autostart disable" is OFF.



2. All components on the panel can be stopped by pressing the "Stop" button on the panel.



4.6.1.3 Start/stop all components on SpeeDBee Synapse

You can start or stop components on SpeeDBee Synapse using the following procedure.

1. By pressing the "Start" button on the header, you can start all the components on the SpeeDBee Synapse for which "Autostart disable" is OFF.



2. You can stop all components on SpeeDBee Synapse by pressing the "Stop" button on the header.

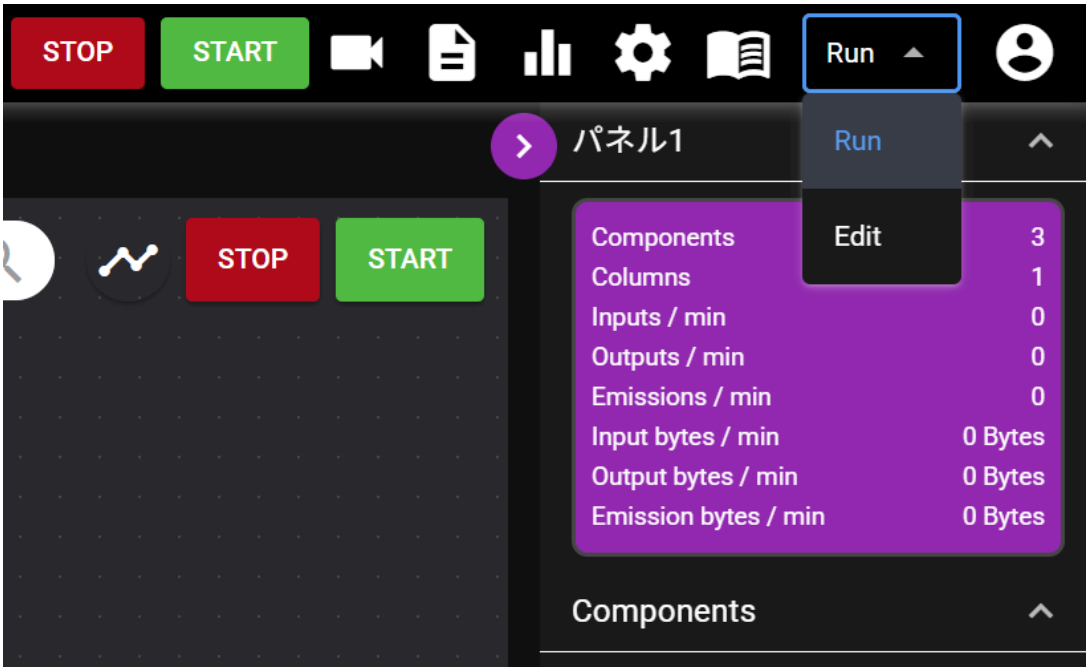


4.6.2 Test

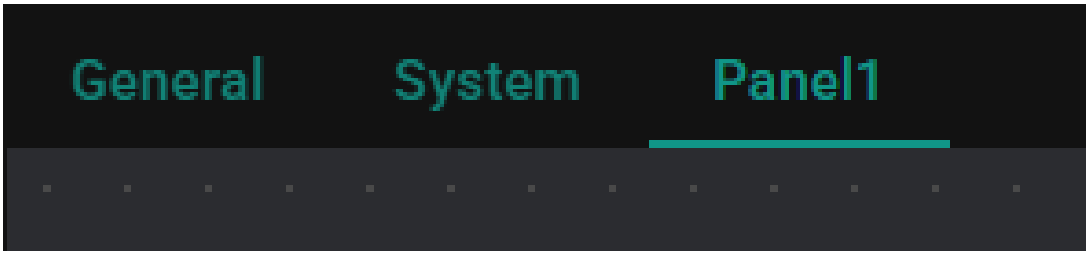
This section describes how to review the data collected by the Resource Collector and how to download the file outputted by the File emitter.

Verifying Resource Collector Operation

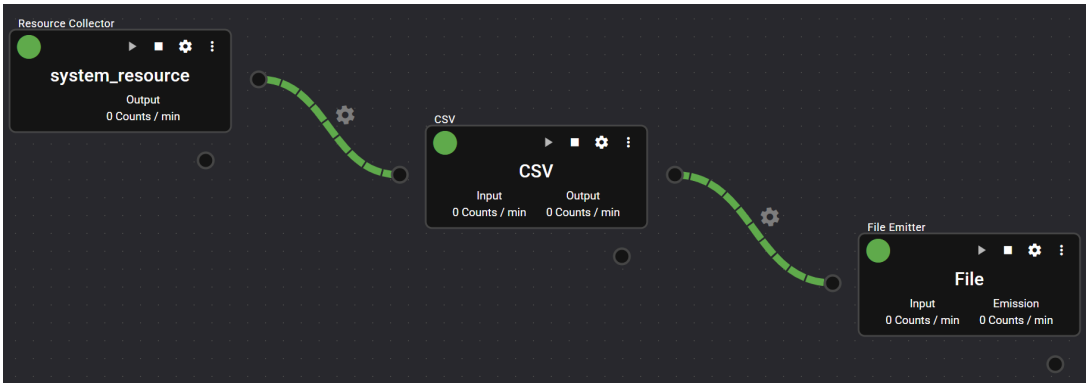
1. Select "Run" in the header mode selection to open the Run mode window.



2. Displays the panel 1.



3. Click the output port of the Resource Collector.

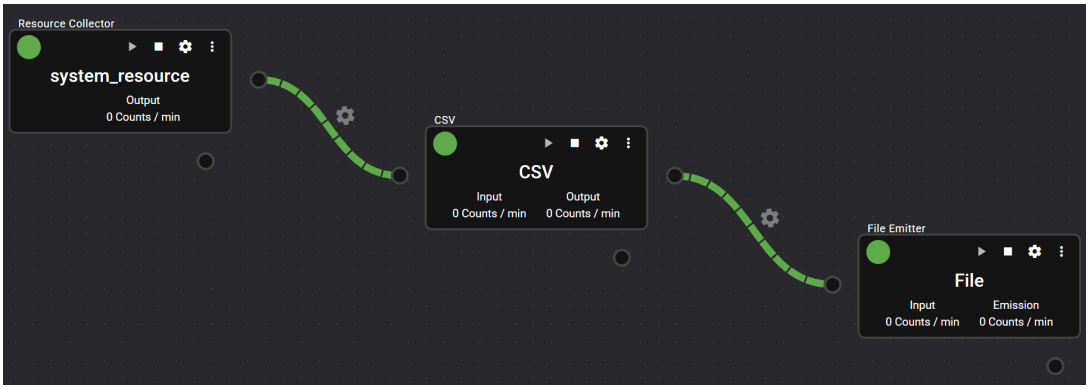


4. Click the indicator at the bottom of the screen to display the Data Monitor and see what data the Resource Collector is collecting.

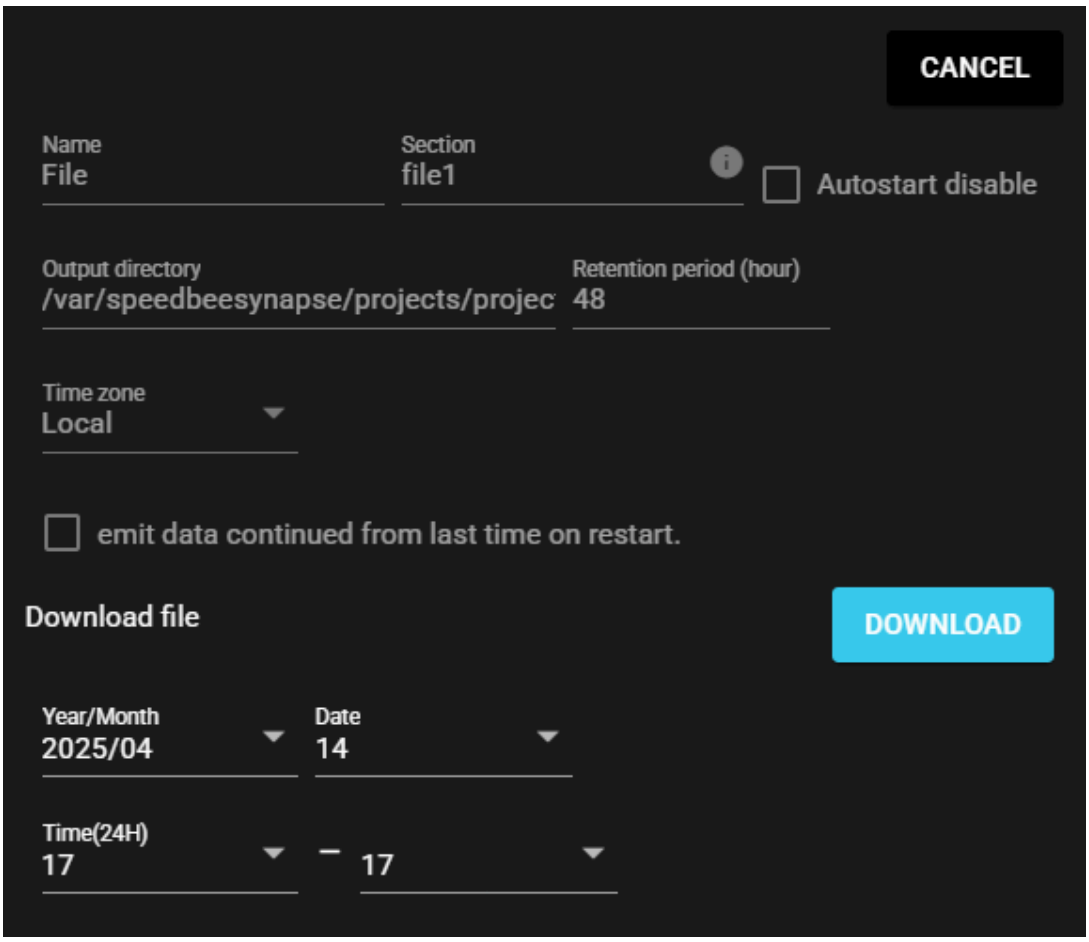


Verifying the File Emitter in Action

1. Press the File emitter's settings icon.



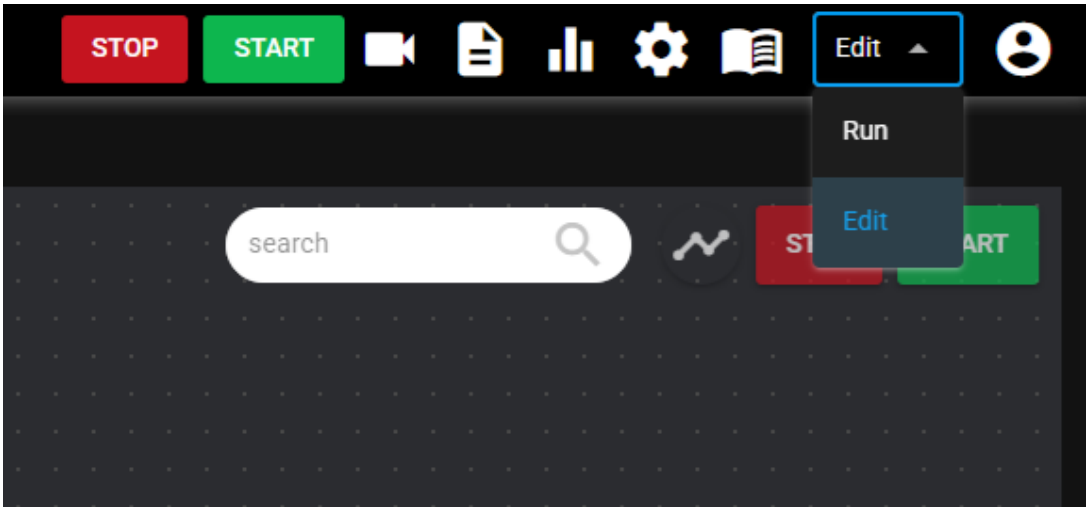
2. Enter a time range and press Download. The files outputted by the File emitter during the specified time range are downloaded in zip format.



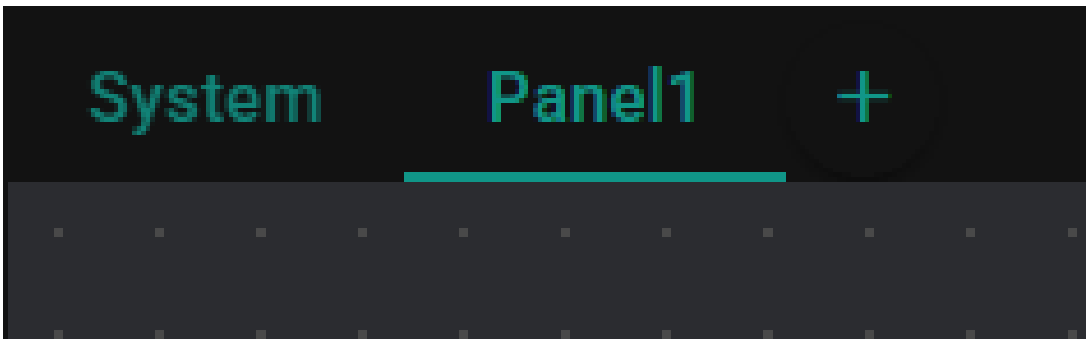
4.6.3 Configuration Tuning

This task explains how to edit component settings and delete components.

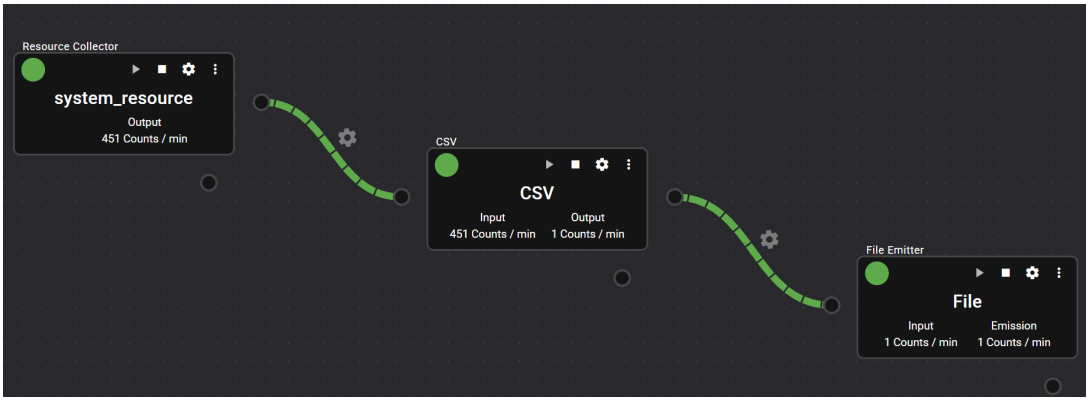
1. Select "Edit mode" in the header mode selection to open the Edit mode window.



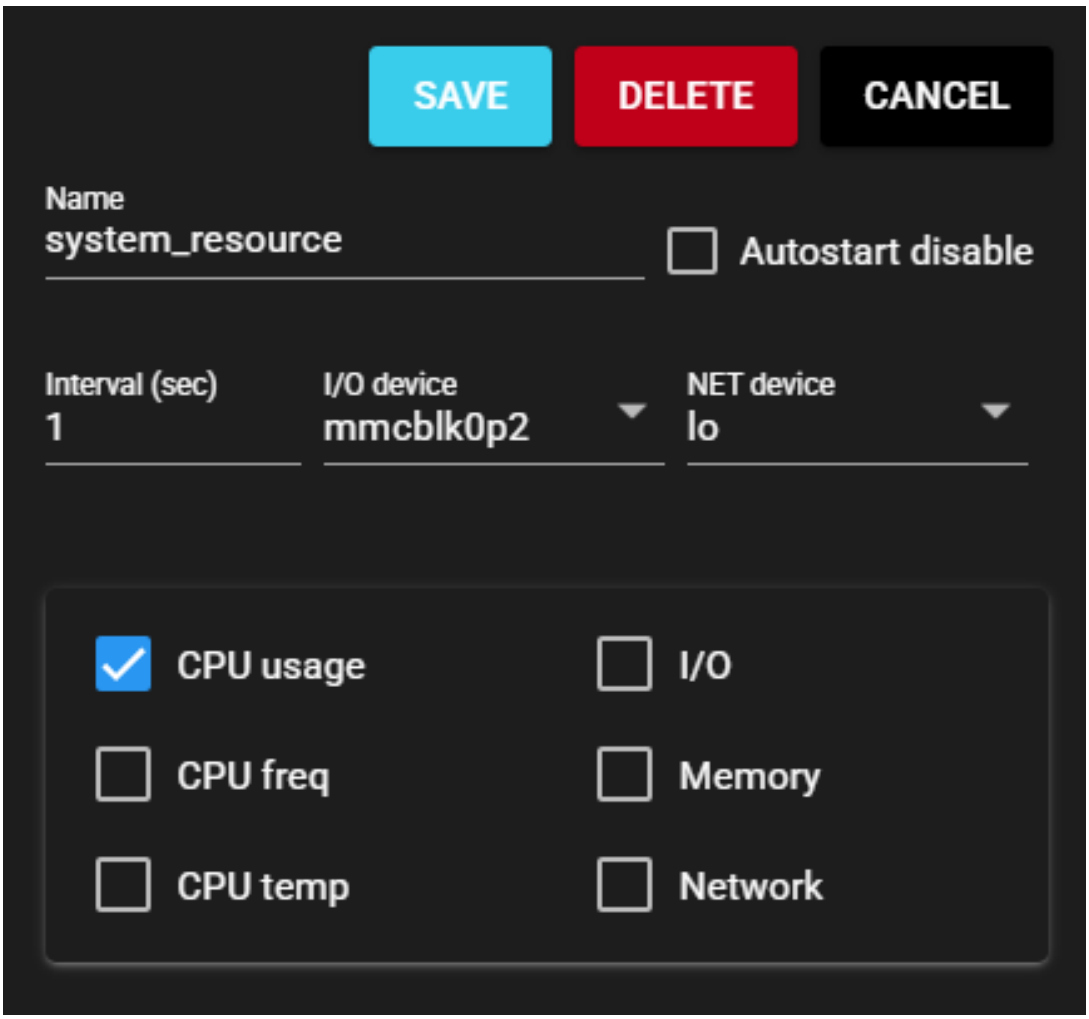
2. Open the panel 1.



3. Press the configuration icon for the component whose settings you want to change.



4. Change the settings in the settings dialog and press "Save." The Resource Collector image is used here. For example, if you want to collect information such as memory usage, select this check box. If you want to delete the component, press Delete and select Yes in the confirmation dialog box.



5. The configuration can be applied to the collection service by stopping and restarting the component whose configuration has been changed. For details about starting and stopping components, see “Starting and Stopping the Collection Service”.

4.6.4 Stop Collection Service

Stopping all components on the SpeeDBee Synapse puts the collection service in a standby state. For procedures, see "Start/stop all components on SpeeDBee Synapse."

5 Explanation of Functions

5.1 Explanation of Functions

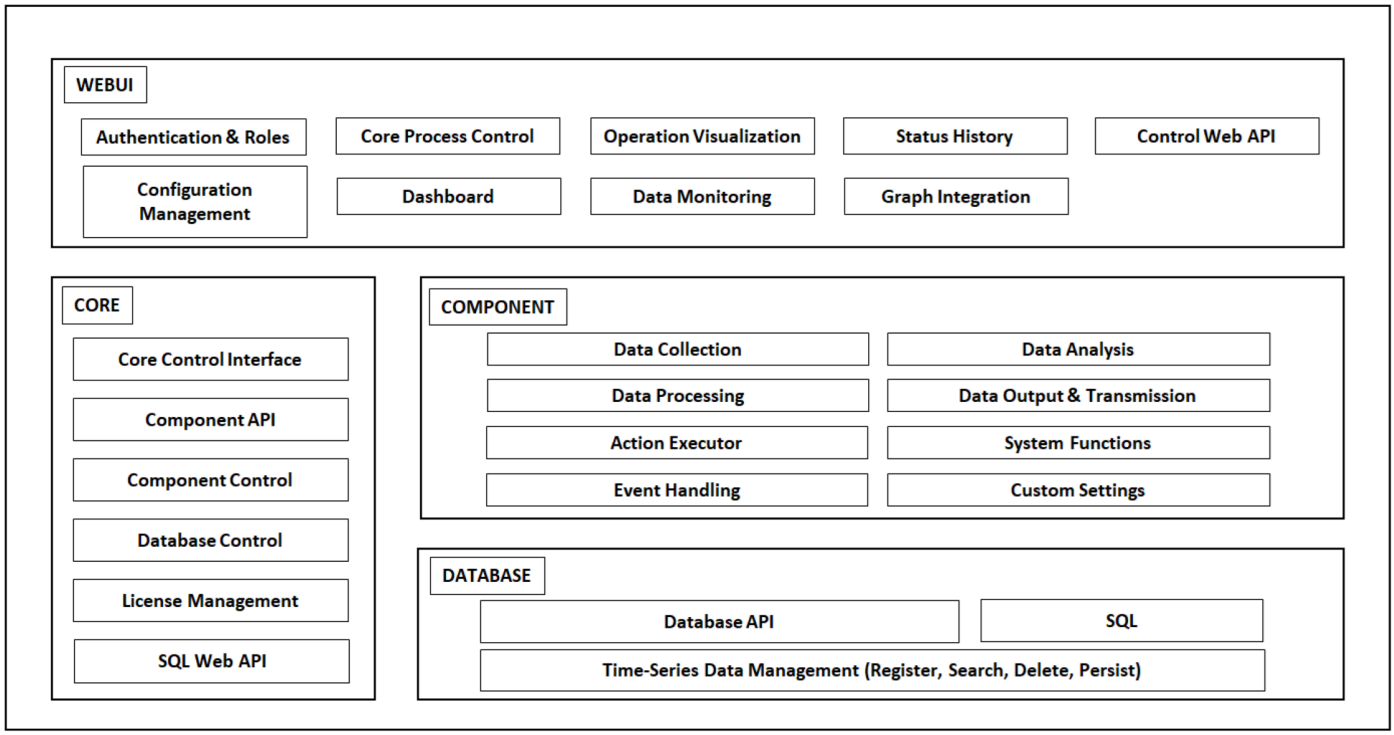
This chapter explains the functions.

The following topics are covered:

1. Overview of Engine Functions
2. Flow Editor
3. Collector
4. Serializer
5. Deserializer
6. Emitter
7. Logic (Event)
8. Logic (Real Time Analysis and Data Manipulation)
9. Action
10. Development Components
11. System Components
12. Component Default Settings
13. Status Timeline
14. Various Logs
15. Database Query (SQL)
16. Arithmetic and Conditional Expressions
17. Managing Configuration Information (Project)

5.2 Overview of engine functions

This section explains a general overview of engine functions. Please refer to this section if you would like to know more about the internal process. The following diagram illustrates the SpeeDBee Synapse configuration.



5.2.1 Feature List

This section describes the contents shown in the configuration diagram.

5.2.1.1 Core

The core is the center of SpeeDBee Synapse that controls each feature. The core provides the following features.

Item	Description
Core Control Interface	Controls each feature of SpeeDBee Synapse. Accepts and processes requests from the WEBUI. The user will not use it directly.
Component API	Provides the APIs used by the component. Component developers need to use these APIs for programming.
Component Control	Controls the processing of components (a group of modules that enable data collection, processing, analysis, emission, etc.). The core automatically handles this internally.

Item	Description
Database Control	Controls the time series database on the SpeeDBee Synapse. The core automatically controls this internally.
License Control	Controls licenses. Controls the availability of SpeeDBee Synapse under the correct license.
SQL Web-API	Provides a Web-API that enables you to execute SQL against the time series database on SpeeDBee Synapse. For more information, see DB Query (SQL) and DB Query Service.

5.2.1.2 Component

A component is a group of modules that performs arbitrary processing in SpeeDBee Synapse. Using the components allows for the following.

Item	Description
Data Collection	You can collect data from a specific target. For more information, see Collector.
Data Analysis	You can analyze the collected data. For more information, see Logic (Realtime Analysis/Data Manipulation).
Event	Processes events according to the collected data. For more information, see Logic (Event).
Data Processing	You can process data into a specific format. For more information, see Serializer.
Data Output/Emission	You can output and emit data to a specific target. For more information, see Emitter.
Action	Performs any action. For example, it can execute OS commands or send mail. For more information, see Action.
System	Provides system-related functions. It is initially deployed and does not allow the user to add new instances. For more information, see System Component.
Custom	The user can develop components on its own. For more information, see Overview in Creating and Using Custom Components.

5.2.1.3 WEBUI

WEBUI provides a SpeeDBee Synapse screen for the user to interact with. WEBUI provides the following features:

Item	Description
Authentication/Roles	Provides authentication features that allow SpeeDBee Synapse to be used by legitimate users. Roles can also be assigned according to User Roles. For more information, see User Management.
Core Process Control	Controls the core process. Enforces process control during user operations and core process errors.
Dashboard	Provides an interface that visually displays the data collection flow and allows for interactive customization. For more information, see Panel.
Configuration Information Management	You can manage configuration information on SpeeDBee Synapse. The information input by the user is maintained as a configuration file.
Operational Information Visualization	Displays operating status on SpeeDBee Synapse and components. For more information, see Status List.
Status History	Displays the component's operating status history. For more information, see Status Timeline.
Control Web-API	Provides a Web-API to control SpeeDBee Synapse. For more information, see Work with Control Program.
Data Monitor	You can view the collected data on the monitor. For more information, see Data Monitor.
Work with Graph	You can create a graph based on the collected data. For more information, see Work with Grafana.

5.2.1.4 Database

SpeeDBee Synapse includes a time series database. The time series database provides the following features.

Item	Description
Database API	Provides an API for working with the time series database. The user does not need to be aware of this API on SpeeDBee Synapse.

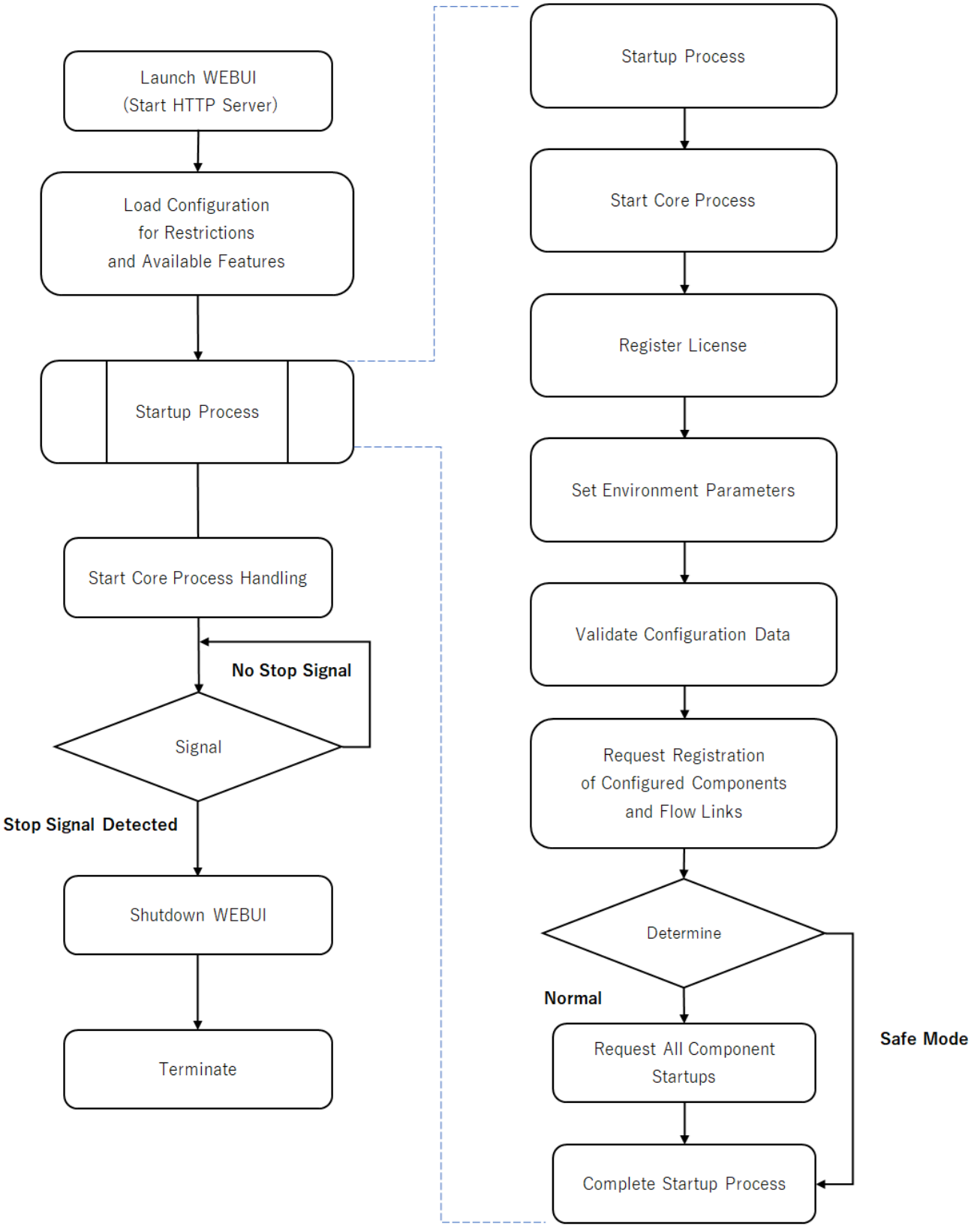
Item	Description
SQL	You can execute SQL against the time series database. For more information, see DB Query (SQL).
Time Series Data Processing (Register, Search, Delete, Storage)	You can register, search, delete, and perpetuate the collected data. It is controlled by the core.

5.2.2 Operating Specifications

This section explains the operating specifications for SpeeDBee Synapse.

5.2.2.1 Basic Operation

The basic operating specifications for SpeeDBee Synapse are as follows.



The table below shows the details of the processing executed in the above flow diagram.

- Basic Operation

Processing	Description
WEBUI Start	Starts the WEBUI for SpeeDBee Synapse. An HTTP server runs internally.
Loading the Configuration of Restrictions and Available Functions	Loads configuration information. This includes information on the functions that will be enabled and the values of various restrictions. Basically, this does not involve any preparation or configuration by the user.
Startup Processing	Performs processing to start the core process and enable SpeeDBee Synapse. See below for details.
Core Process Monitoring Startup	Starts core process monitoring. Monitors the core process status and restarts the core process when an error occurs. If the error persists, the system enters safe mode without starting the core process.
Signal Acceptance	Accepts signals from the OS. When a stop signal is received, the system transitions to the termination process.
WEBUI Stop	Stops the WEBUI. Terminates processes such as core process monitoring and performs cleanup operations for each process.

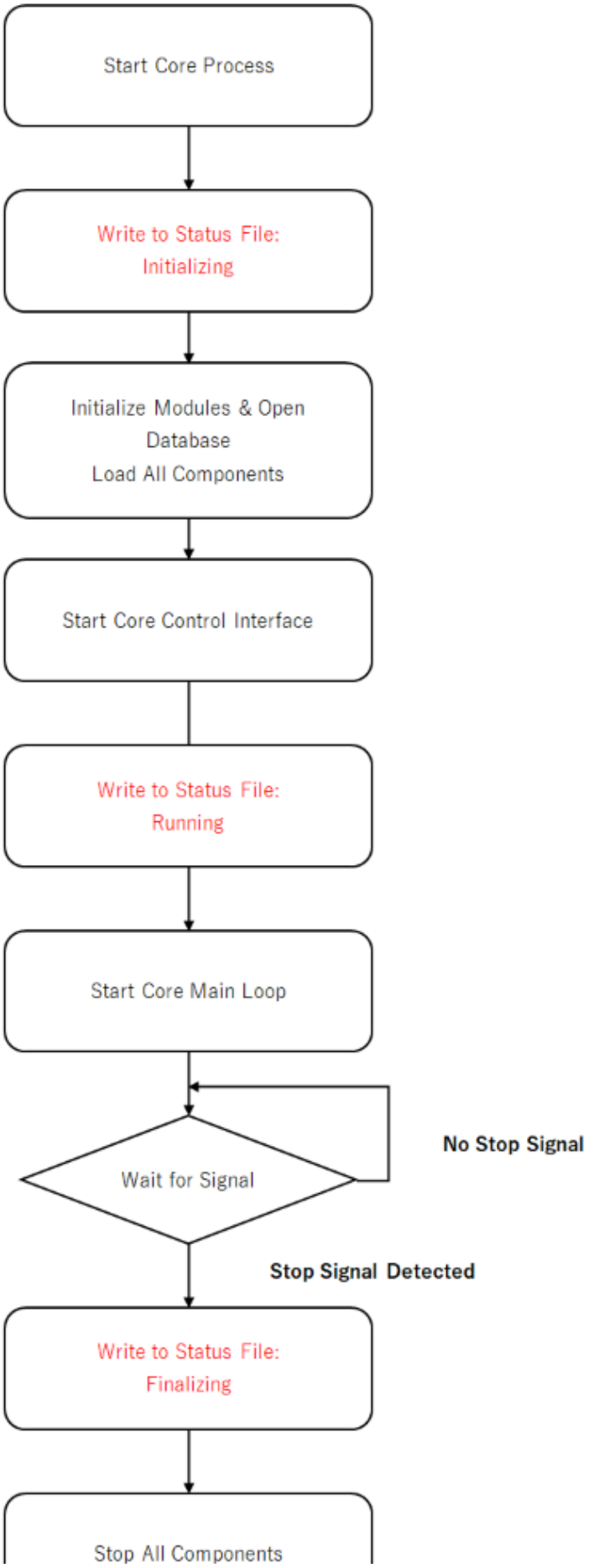
- Startup Processing (Details)

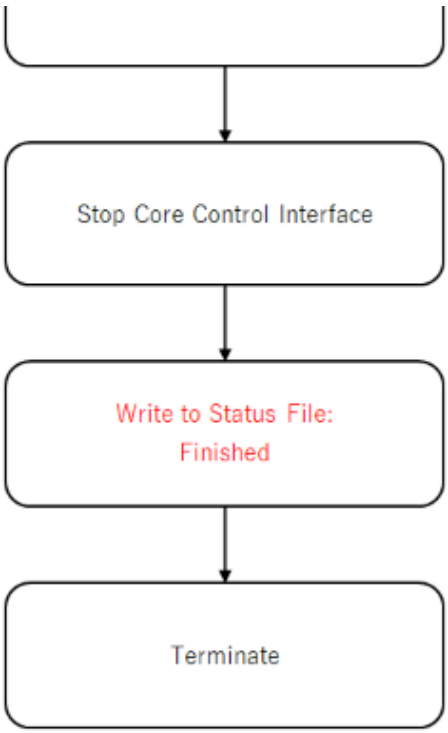
Processing	Description
Core Process Start	Starts the core process (background process) that will be the data engine for SpeeDBee Synapse. See below for details.
License Registration	Register a license for SpeeDBee Synapse. The license configured in the WEBUI will be applied, controlling the available features.
Environmental Parameter Configuration	Applies the database parameters configured in the WEBUI to the core process.
Configuration Information Verification	Verifies the components configured in the WEBUI and other information. Checks for configuration inconsistencies to ensure consistency is maintained.
Configured Component/Flowlink Register Request	Applies components and flowlinks configured by the user to the core process.

Processing	Description
Mode Judgment	Performs mode judgment. The modes include the following types. - Normal mode (This is the normal mode in which components are started in the startup process) - Safe mode (This mode is used when all components cannot be started during an error)
Start All Components Request	Starts all registered components (except those with autostart disabled). This will put the data collection, processing, output, emission, and other processes into execution state. This processing is skipped in safe mode.

The details of the core process start processing are as follows.

- Core Process Start (Details)


Start Core Process
Write to Status File:
Initializing
Initialize Modules & Open
Database
Load All Components
Start Core Control Interface
Write to Status File:
Running
Start Core Main Loop
Wait for Signal
No Stop Signal
Stop Signal Detected
Write to Status File:
Finalizing
Stop All Components

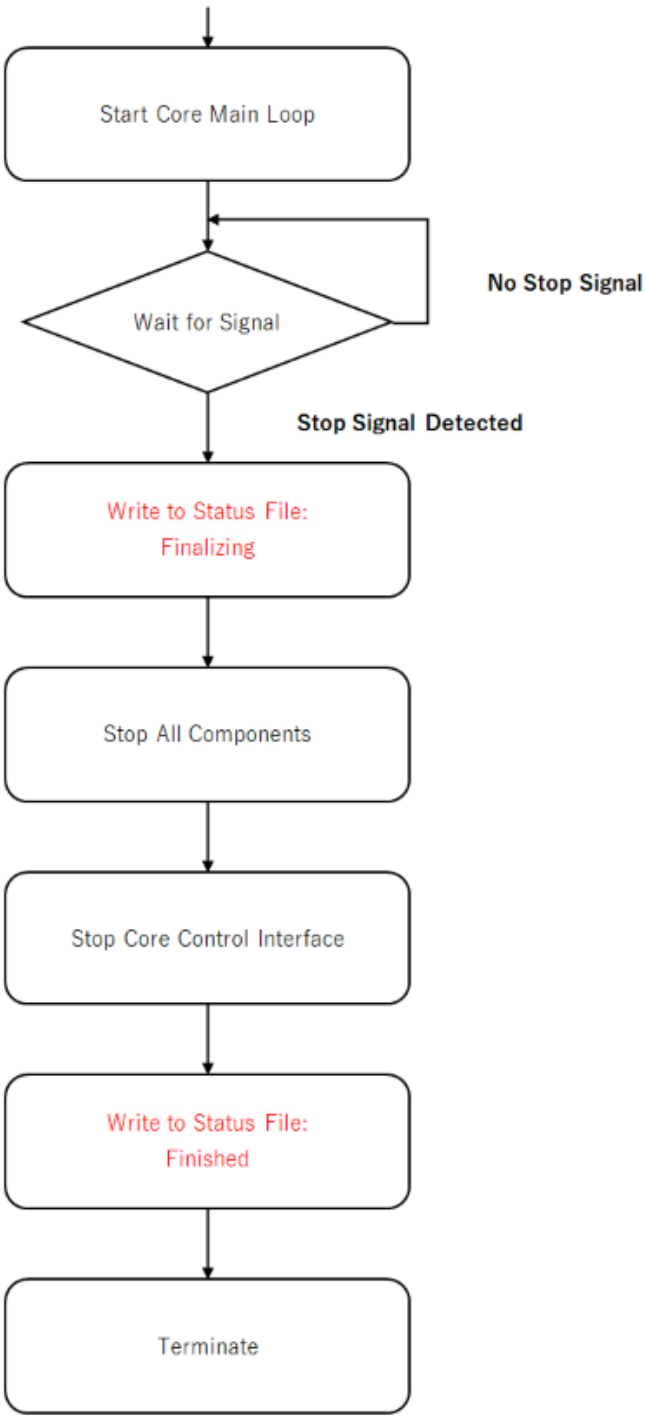


Processing	Description
Record to Status File	Writes records to the status file for recording the start and stop status of the core. The status at this stage is Initializing.
Module Initialization and DB Opening Loading All Components	Performs core module initialization. Within the initialization, the database is opened and all components are loaded. Problematic components are not loaded at this stage. Components that are deployed but cannot be initialized are excluded from being loaded.
Core Control IF Startup	Enables an interface for controlling the core process. The system enters a state in which it can receive requests from the WEBUI.
Record to Status File	Writes records to the status file for recording the start and stop status of the core. The status at this stage is Running. When this status is recorded, it can be determined that this processing has started normally.
Core Main Loop Startup	Starts the main loop of the core. Executes various processes according to requests to the core control IF.
Signal Acceptance	Accepts signals from the OS. When a stop signal is received, the system transitions to the termination process.

Processing	Description
Record to Status File	Writes records to the status file for recording the start and stop status of the core. The status at this stage is Finalizing. When this status is recorded, it can be determined that the termination process has been executed. However, since the next status is recorded after the termination process is completely completed, if the status remains in this state, it is determined that an error has occurred.
Stop All Components	Stops and destroys all components that are currently started.
Core Control IF Stop	Stops the core control IF. External requests will no longer be accepted.
Record to Status File	Writes records to the status file for recording the start and stop status of the core. The status at this stage is Finished. When this status is recorded, it can be determined that the termination process has completed normally. The next time the core process is started, this status is referenced to determine the error during the previous startup. When it detects that a status other than Finished has been recorded, the core internal component registers an error status. If there is an error, you can view information on the Status Timeline.

5.2.2.2 Normal Stop

This section describes the operation at a normal stop. The chart below is an excerpt from the Core Process Start (Details). The red boxed area indicates important processing.



When the stop process is executed normally, a correct record is made in the status file. When Finished is recorded in the status file, it indicates that the stop process has been executed successfully. For more information, see Core Process Start (Details).

5.2.2.3 Abnormal Stop (e.g., power interruption)

This section describes the operation at an abnormal stop. Abnormal situation means, for example, when the host machine stops due to power interruption while SpeeDBee Synapse is running.

- Determination of Core Process Abnormal Stop

As described in the explanation for normal stop operation, if SpeeDBee Synapse stops while it is running, a correct record will not be made in the status file. The next time the core process is started, the status file is determined and if a normal stop cannot be confirmed, the status is registered as an error status in the core internal. You can view error information on the Status Timeline.

- Determination of Database

If SpeeDBee Synapse experiences an abnormal stop while writing to the database, data loss in the database may occur depending on the timing of the stop. Data loss refers to the loss of some data while writing data stored in memory to storage because the data writing does not finish successfully due to a power interruption or other failure. The next time the core process is started, this determination will also be made internally. If an error is detected as a result of the determination, the status is registered as an error status in the core internal. You can view error information on the Status Timeline.

5.2.2.4 Operational Specifications Related to Data Collection

In SpeeDBee Synapse, data collection in the component is the starting point for processing. However, there are some operational specifications that must be taken into account depending on the setting and timing of the operation, which are described below.

Timing of application of component configuration changes

When a configuration change is made, some changes are applied immediately, while others are applied after the component is stopped and restarted. In addition, some configurations have special timing for applying changes.

1. Item to which the new configuration is applied after component restart
2. Specific configuration for each component

Configuration items specific to each component will be reflected after the restart.
For example, if you add an address and save the configuration with a PLC Collector or other device, the configuration will not be applied if you do nothing. Once you stop the PLC Collector and restart it, it will operate with the new configuration reflected.

1. Items to which the new configuration is applied immediately after a configuration change
2. Flowlink Filters
3. Flowlink Linking/Delete
4. Inport setting
5. Item to which the new configuration is applied at special timing after a configuration change
6. OUT Port

For OUT Ports, the new configuration is applied by creating a column after the configuration has been changed.
If a component creates columns at startup, the configuration will not be applied until restart. On the other hand, if a component is configured with regular expressions or other means and creates columns at a special timing, the configuration may be applied in the middle of the process.
For example, an MQTT Collector may not create columns until it receives data. In this case, if the OUT Port setting is specified by a regular expression or other means, the new configuration may be applied.
Note: If you want to ensure that the configuration is applied, restart the relevant component.

Data flow at the time of flowlink linking/delete

If you start a component that collects data and then link a flowlink to another component, the data before the link is made will not be collected.
In addition, if you delete a flowlink in the middle of the process, the relevant data will no longer flow to the destination of the link at the time of deletion, and the data processed by the relevant component will change. This is because even if the configuration is not changed, the data flowing through it will change, resulting in a change in the behavior of the operating components.

For example, assume that a PLC Collector and a Modbus Collector each have a flowlink to a JSON Serializer. In this case, if the flowlink between the PLC Collector and the JSON Serializer is deleted, the data referenced by the JSON Serializer will only be data from the Modbus Collector in the middle of the process.

Data Storage

For data acquired by the component that collects the data, storage is not created to the DB by default. Once the data is stored in the DB, you can use Grafana or the DB query service to view historical data for the storage period. If storage is not created, only the data currently kept in memory can be referenced.

If you want to create storage for these historical data for reference, please configure the settings in "storage setting" in "out port".

Delay time for data reference

The IN Port has a Delay Time setting.
This is the setting of how many seconds to delay the time range of the acquired data relative to the current time when referencing data.
This setting is intended to prevent the loss of data for which data collection has not been completed because data collection and data referencing are processed in parallel without synchronization.
Adjust this delay time when using data from components that require more time for data collection.
Depending on the component, it may be possible to guarantee data collection time and perform data referencing at intervals shorter than this delay. This is done automatically within SpeeDBee Synapse.

If you want to set the delay time, please refer to "in port".

Data when components stop

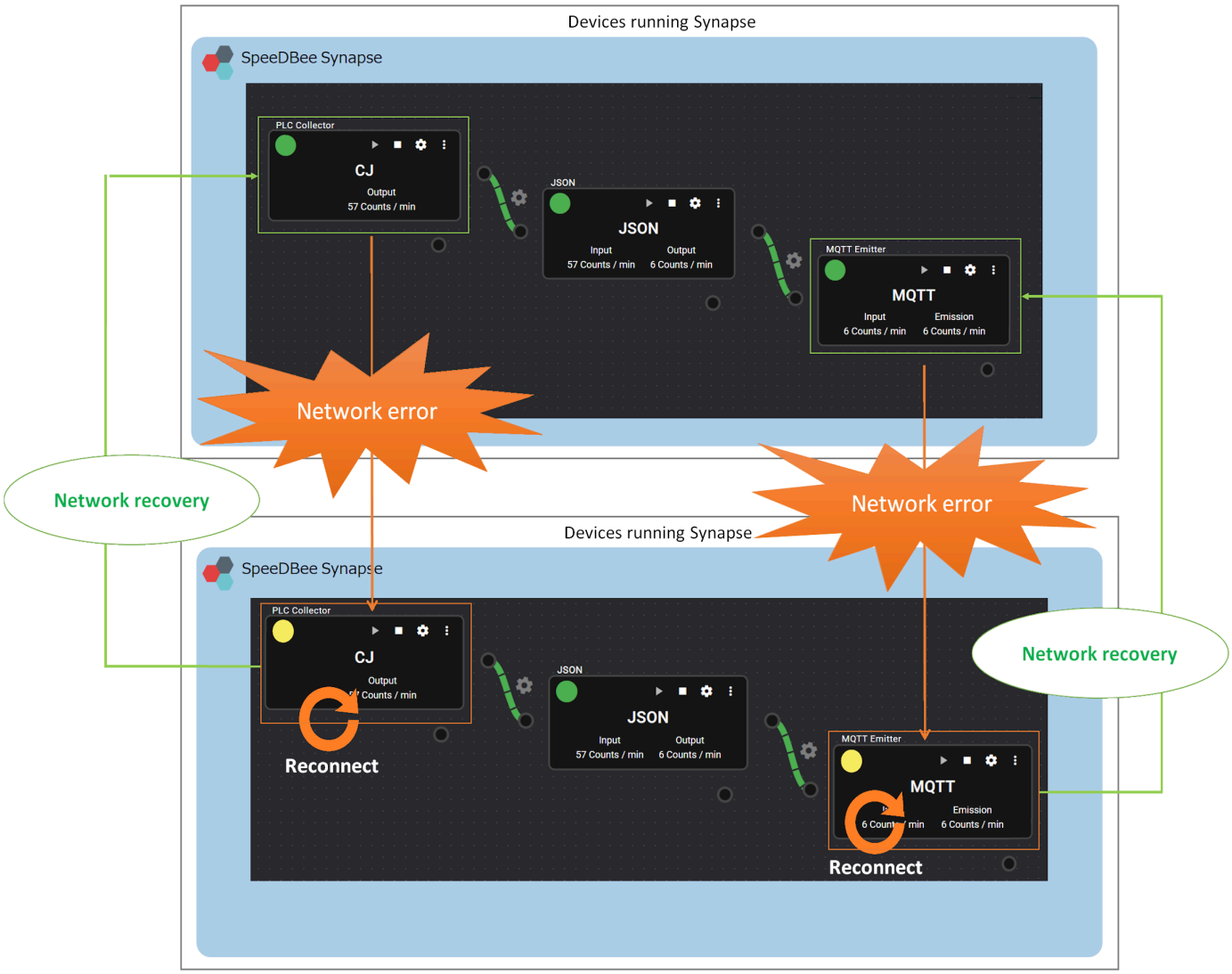
If the user stops a component, the most recent data from the operation in which the user stopped the component is not fully guaranteed.

For example, if a flowlink is linked from any collector to the serializer and the serializer is stopped, the serializer will operate to stop as quickly as possible. Therefore, while the serializer is stopped, data is not guaranteed since the data referencing process is not executed.
Do not inadvertently stop a component while the data you want to process is flowing.

5.2.2.5 Operational Specifications for Network Errors and Recovery

This section explains the operational specifications for the occurrence of a network error on SpeeDBee Synapse and recovery from a network error.

Basically, components that use communication will reconnect to communication when a network error occurs.



1. Components that collect data

   Collector components that use communication will attempt to reconnect to communication if a network error occurs while collecting data. While an error occurs in communication, the network is in an error state and the status lamp changes to a Warning state. Data cannot be collected while a network error occurs. After the network in Operating machine is restored and reconnected successfully, data collection will resume normally. The relevant components include:

   - PLC Collector
   - Modbus Collector
   - EtherNet/IP Collector
   - VIDEO Collector
   - MQTT Collector

2. Components that emit collected data
   Action components and emitter components that use communication will attempt to reconnect to communication if a network error occurs during operation. While an error occurs in communication, the network is in an error state and the status lamp changes to a Warning state.

   Action components
   Data cannot be emitted while a network error occurs.
   The relevant components include:

   - PLC Writing
   - Modbus Writing
   - EtherNet/IP Writing
   - Send mail

   Emitter components
   Data cannot be emitted while a network error occurs. However, it is also possible to set a retry mode to re-emit data that has not yet been emitted.
   The relevant components include:

   - Cloud Emitter (Azure)
   - Cloud Emitter (AWS)
   - FTP/FTPS/SFTP Emitter
   - MQTT Emitter

FTP/FTPS/SFTP Emitters always attempt to re-emit data that has not yet been emitted. For information on retry modes for other emitters, see the functional description of the relevant component.
When the communication process is carried out by a custom component or shell command generated by the user, the operation depends on the implementation.
If reconnection of communication is required when a network error occurs, the process to reconnect must be implemented in each implementation.

5.3 Flow Editor

This section describes the Flow Editor. Flow Editor is the operation screen of SpeeDBee Synapse.

5.3.1 About the mode

The Flow Editor has three types of modes:

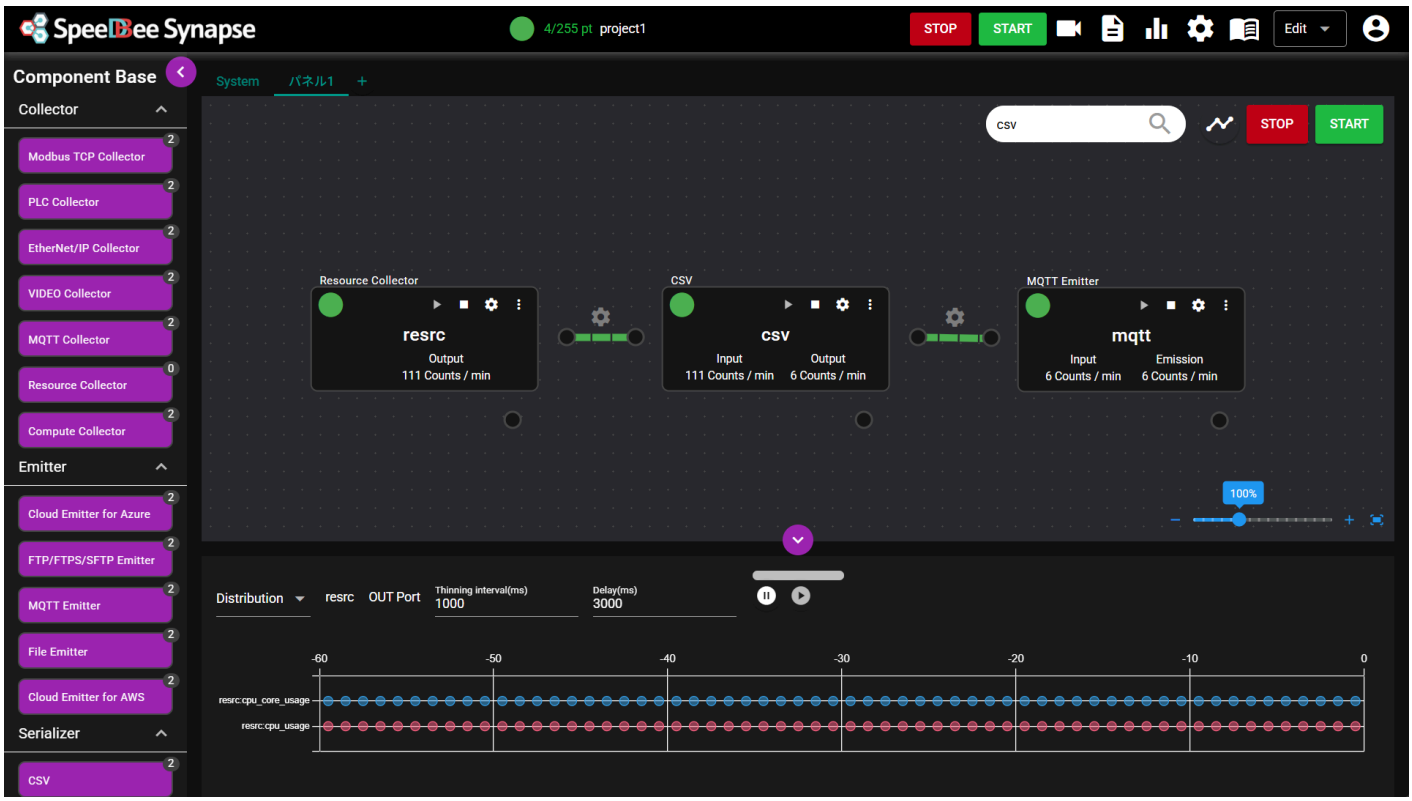
Mode	Description
Edit Mode	This is used to configure settings such as the data collection flow. Only users with the role of the system administrators or managers can use this function.
Run Mode	This operates the data collection flow. This function is available to users of all user roles.
List Mode	Settings for all components can be configured on the list screen. Only users with the role of the system administrators or managers can use this function.

5.3.2 Items on the screen

This section describes the items shown on the screen of Flow Editor.

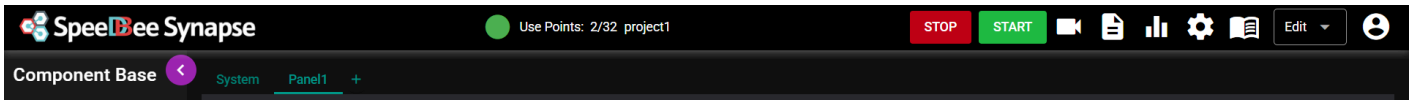
5.3.2.1 Edit Mode

The items shown on the screen of Edit Mode Flow Editor are as follows.



The Edit Mode screen consists of the following: ① Header, ② Componentbase, ③ Panel, and ④ Data monitor

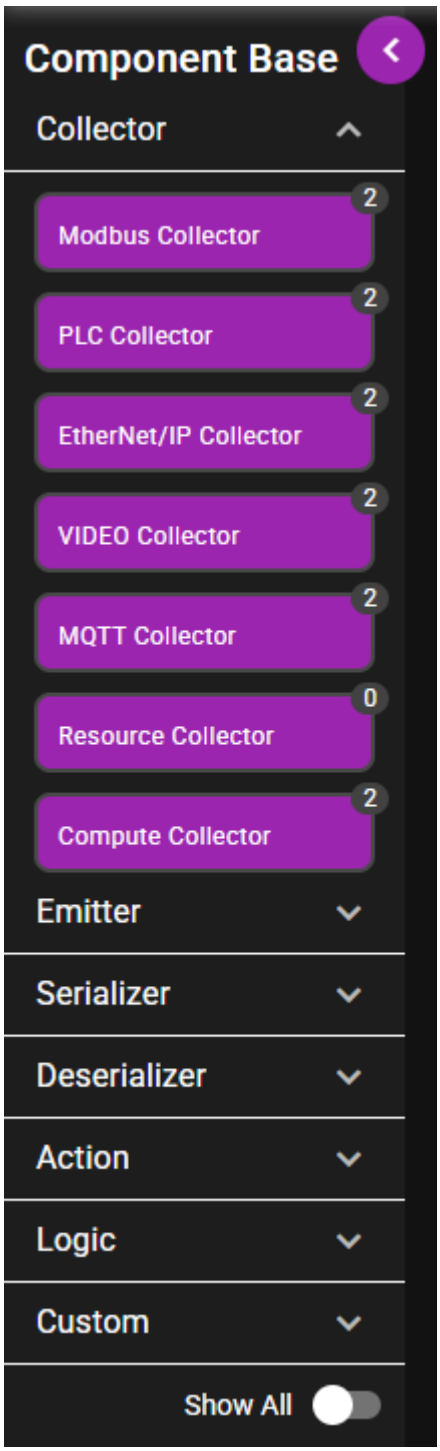
① Header



Number	Item	Description
①	Use Points	This displays the number of registered license points and the number of points used.
②	Current project	This displays the currently selected project. Note: For detail, see Management of Configuration Information (Project).
③	Stop button	Clicking the stop button stops all components registered in SpeeDBee Synapse.
④	Start button	Clicking the Start button starts all the components registered in SpeeDBee Synapse for which "Autostart disable" is set to OFF.
⑤	Video Viewer icon	Clicking the Video Viewer icon switches the display to the Viewer screen. Note: This is displayed only when the "i-Trigger" extension custom package is applied. For detail, see i-Trigger (extension custom package).

Number	Item	Description
⑥	Log icon	Clicking the Log icon displays the SpeeDBee Synapse operation log window. Note: For detail, see Logs.
⑦	Graph icon	Clicking the Graph icon displays the Grafana screen. Note: For detail, see Integration with Grafana.
⑧	Setting menu icon	Clicking this icon displays the Setting Menu. Note: For detail, see Setting Menu.
⑨	Manual	Clicking this displays the manual screen.
⑩	Mode selection	Flow Editor display mode can be selected from below. • Edit Mode • Run Mode • List Mode Note: This is displayed only when a user logs in with the user role of system Administrator or manager.
⑪	User management icon	Clicking this displays the User Management menu. Note: For detail, see User Management.

② Componentbase



Number	Item	Description
①	Open/close icon	By clicking this icon, the display/non-display of the componentbase is switched.
②	Indicator	By clicking the indicator, the display/non-display of the component base can be switched.

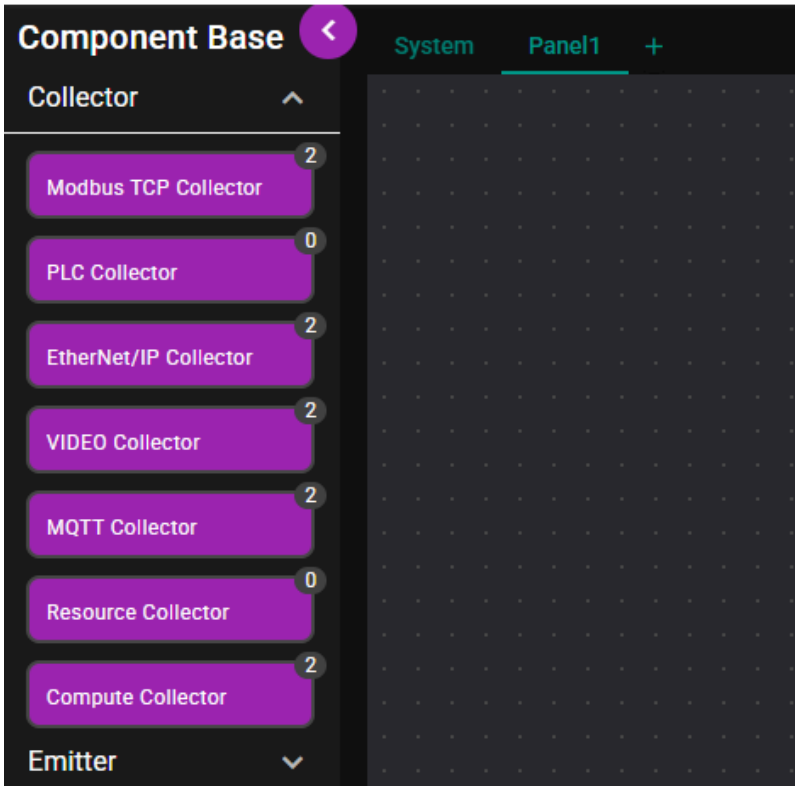
Number	Item	Description
③	Componentbase	This is a component that can be registered to the panel. The base name is displayed in the middle. The number of points used upon registration is displayed in the upper right. When the mouse pointer is placed over a component, the information of the component and the presence or absence of IN/OUT ports are additionally displayed.
④	Show All	By switching this ON, all hidden component bases are temporarily displayed.

- The componentbase is available in the following types:

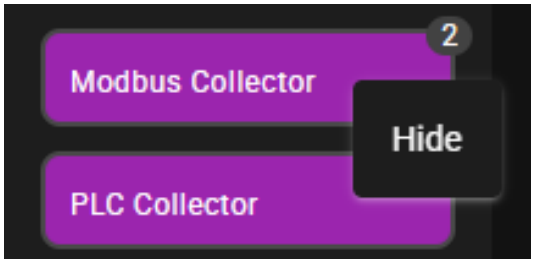
Type	Link
Collector	For detail, see Collector.
Serializer	For detail, see Serializer.
Deserializer	For detail, see Deserializer.
Emitter	For detail, see Emitter.
Logic	For detail, see Logic (event) and Logic (Real time analysis/data manipulation).
Action	For detail, see Action.
Custom	For detail, see "Overview" of "Creation and Use of Custom Components". Note: This is displayed only when Custom Component has been added.

- You can hide component bases as follows.

  a. Right-click a component base.



b. Select "Hide".



How to show a hidden component base

a. Turn on "Show All" to display hidden component bases.

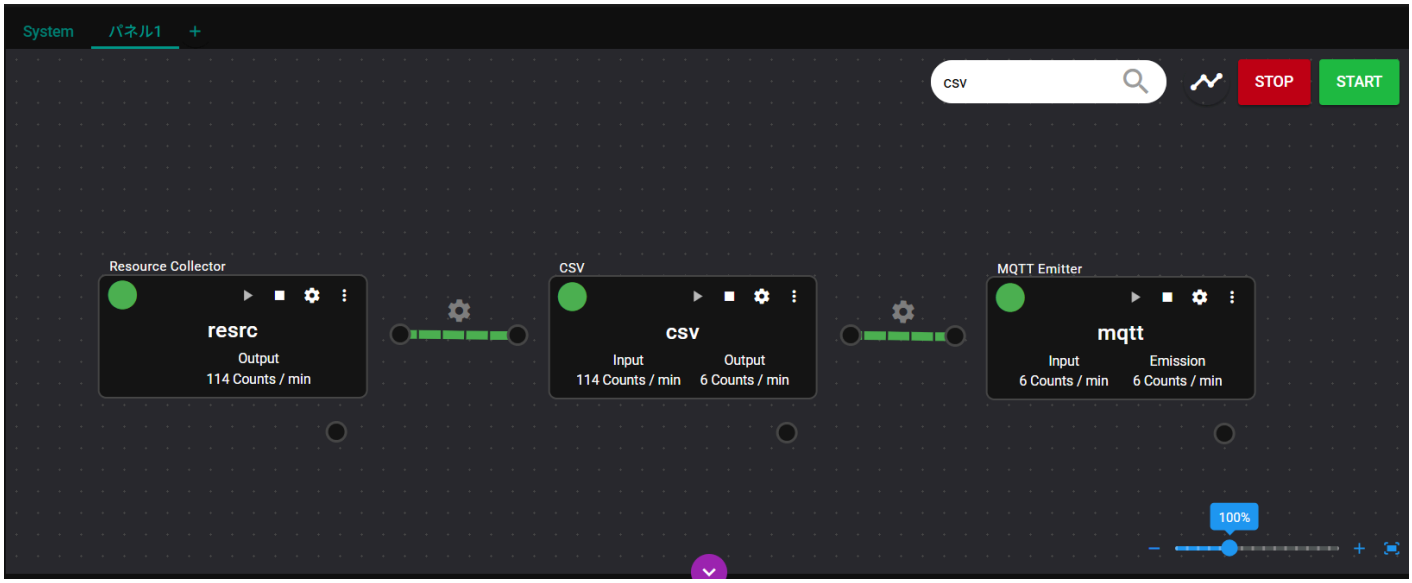
b. Right-click the displayed component base.

c. Select "Show".

Scope of hide settings

Hide settings for component bases are saved per project.
When you switch projects, the new project's settings apply, so components hidden in the previous project may become visible.

③ Panel

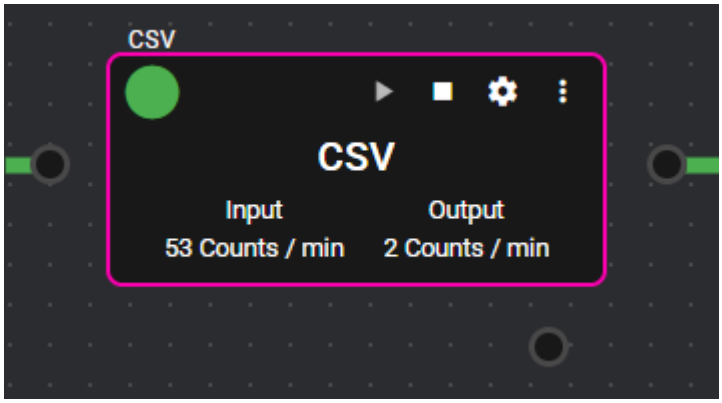


Number	Item	Description
①	Tab	This tab is used to switch the panel to display.
②	Add panel icon	By clicking this icon, a new panel can be added.
③	Component search box	The location of a component can be searched using this box.
④	Status Timeline	Clicking this displays a screen where the component status is organized in chronological order. Note: For detail, see Status Timeline.
⑤	Stop button	Clicking this stops all components on the panel.
⑥	Start button	Clicking this starts all components on the panel for which Autostart Disabled is OFF.
⑦	Component	This is the component registered to the panel.
⑧	Flowlink	This is a flowlink registered to the panel. Note: A flowlink is an object that connects two components and transfers data between them. For detail, see Add a Flowlink.
⑨	Display magnification	The panel display magnification can be adjusted.

- For detail, see Panel.

- The panel named "System" is exclusively used for system component.

③-① Component

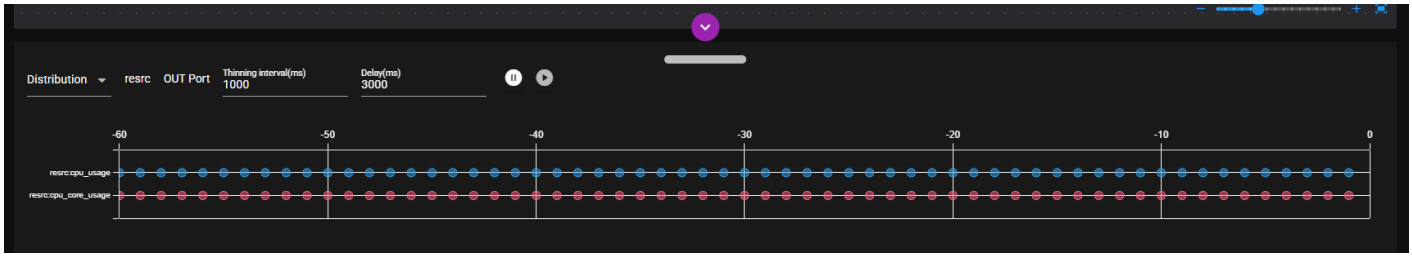


Number	Item	Description
①	IN Port	This port accepts data input. Note 1: Clicking this displays the data flowing to the port on the data monitor. For detail, see Data Monitor. Note 2: Right-clicking this allows users to configure settings. For detail, see IN Port.
②	Componentbase name	Componentbase name is displayed.
③	Status lamp	Colors corresponding to the component operation status are displayed. For detail, see Status List.
④	Start icon	Clicking this icon starts the component.
⑤	Stop icon	Clicking this icon stops the component.
⑥	Configuration icon	Clicking this icon displays the detail of component configuration.
⑦	Menu icon	Clicking this icon displays a menu that provides various options, such as port settings, logs, and the status timeline for the component.
⑧	Component name	Component name is displayed.
⑨	Data volume monitor	This displays the volume of data a component is processing.
⑩	OUT Port	This port outputs data. Note 1: Clicking this displays the data flowing to the port on the data

Number	Item	Description
		monitor. For detail, see Data Monitor. Note 2: Right-clicking this allows users to configure settings. For detail, see OUT Port.
⑪	SYSTEM Port	This port outputs system-related data. Note 1: Clicking this displays the data flowing to the port on the data monitor. For detail, see Data Monitor. Note 2: Right-clicking this allows users to configure settings. For detail, see SYSTEM Port.

- For detail, see Component.

④ Data monitor



Number	Item	Description
①	Open/close icon	By clicking the indicator, the display/non-display of the data monitor can be switched.
②	Display size adjustment grip	By dragging the cursor and moving the mouse up and down, the display size of the data monitor can be adjusted.
③	Display type selection form	The data monitor display method can be selected.
④	Selection port	A selected port is displayed.
⑤	Thinning interval	The thinning interval for the data to be displayed can be configured.
⑥	Delay time	The delay time for the data to be displayed can be configured.
⑦	Monitor display area	The selected type of monitor is displayed.

- For detail, see Data Monitor.

5.3.2.2 Run Mode

The items shown on the Run Mode Flow Editor screen are as follows:



The Run Mode screen consists of ① Header, ② Panel, ③ Panel status, ④ Data monitor, and ⑤ System resources.

① Header



Number	Item	Description
①	Use Points	This displays the number of registered license points and the number of points used.
②	Current project	This displays the currently selected project. Note: For detail, see Management of Configuration Information (Project).
③	Stop button	Clicking the stop button stops all components registered in SpeeDBee Synapse.
④	Start button	Clicking the Start button starts all the components registered in SpeeDBee Synapse for which "Autostart disable" is set to OFF.

Number	Item	Description
⑤	Video Viewer icon	Clicking the Video Viewer icon switches the display to the Viewer screen. Note: This is displayed only when the "i-Trigger" extension custom package is applied. For detail, see i-Trigger (extension custom package).
⑥	Log icon	Clicking the Log icon displays the SpeeDBee Synapse operation log window. Note: For detail, see Logs.
⑦	Graph icon	Clicking the Graph icon displays the Grafana screen. Note: For detail, see Integration with Grafana.
⑧	Setting menu icon	Clicking this icon displays the Setting Menu. Note: The language configuration is the only menu available for selection. For detail, see Setting Menu.
⑨	Manual	Clicking this displays the manual screen.
⑩	Mode selection	Flow Editor display mode can be selected from below. • Edit Mode • Run Mode • List Mode Note: This is displayed only when a user logs in with the user role of system Administrator or manager.
⑪	User management icon	Clicking this displays the User Management menu. Note: For detail, see User Management.

② Panel

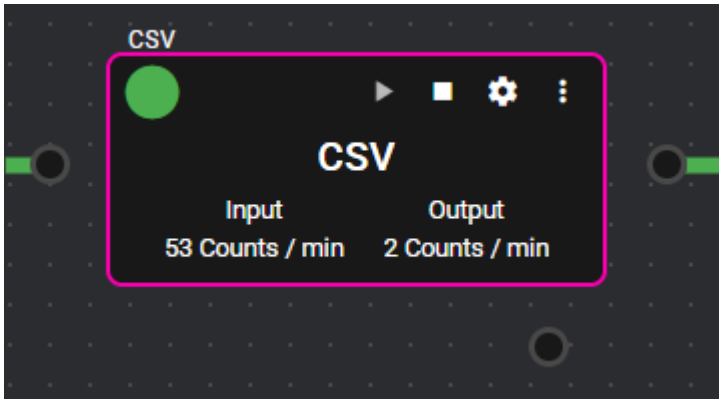


Number	Item	Description
①	Tab	This tab is used to switch the panel to display.
②	Component search box	The location of a component can be searched using this box.
③	Status Timeline	Clicking this displays a screen where the component status is organized in chronological order. Note: For detail, see Status Timeline.
④	Stop button	Clicking this stops all components on the panel.
⑤	Start button	Clicking this starts all components on the panel for which Autostart Disabled is OFF.
⑥	Component	This is the component registered to the panel. Note: The component is the module group that performs functions such as data collection, processing, analysis, and emission.
⑦	Flowlink	This is a flowlink registered to the panel.
⑧	Display magnification	The panel display magnification can be adjusted.

- For detail, see Panel.

- The panel named "System" is exclusively used for system component.

②-① Component

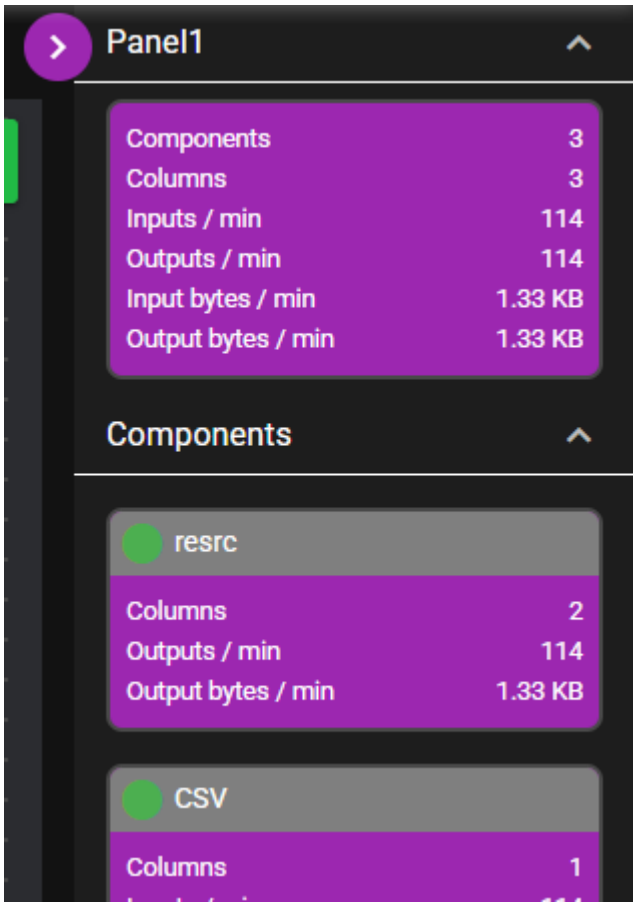


Number	Item	Description
①	IN Port	This port accepts data input. Note: Clicking this displays the data flowing to the port on the data monitor. For detail, see Data Monitor.
②	Componentbase name	Componentbase name is displayed.
③	Status lamp	Colors corresponding to the component operation status are displayed. For detail, see Status List.
④	Start icon	Clicking this icon starts the component.
⑤	Stop icon	Clicking this icon stops the component.
⑥	Configuration icon	Clicking this icon displays the detail of component configuration.
⑦	Menu icon	Clicking this icon displays a menu that provides various options, such as port settings, logs, and the status timeline for the component.
⑧	Component name	Component name is displayed.
⑨	Data volume monitor	This displays the volume of data a component is processing.
⑩	OUT Port	This port outputs data. Note: Clicking this displays the data flowing to the port on the data monitor. For detail, see Data Monitor.

Number	Item	Description
⑪	SYSTEM Port	This port outputs system-related data. Note: Clicking this displays the data flowing to the port on the data monitor. For detail, see Data Monitor.

- For detail, see Component.

③ Panel Status

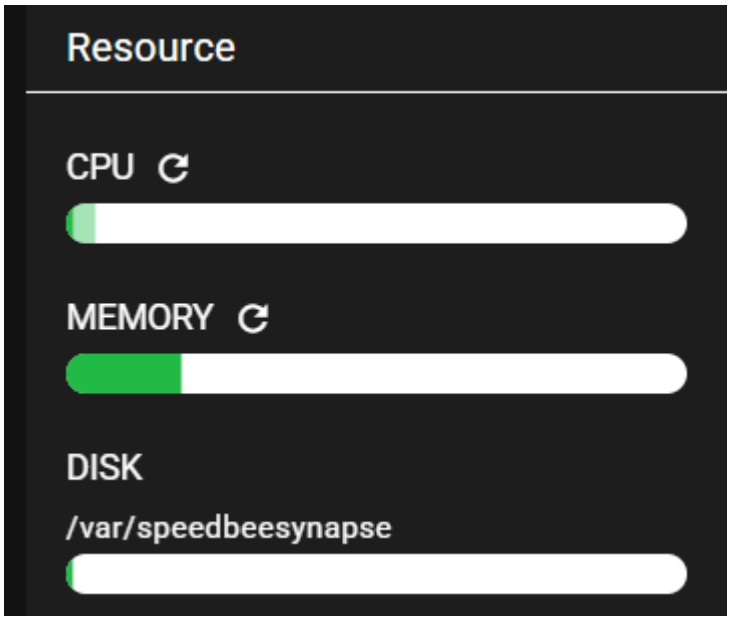


Number	Item	Description
①	Open/close icon	By clicking this icon, the display/non-display of the panel status can be switched.
②	Indicator	By clicking the indicator, the display/non-display of the component base can be switched.
③	Panel status	This displays the status of the panel.

Number	Item	Description
④	Component status	This displays the status of the component.

- For detail, see Panel Status.

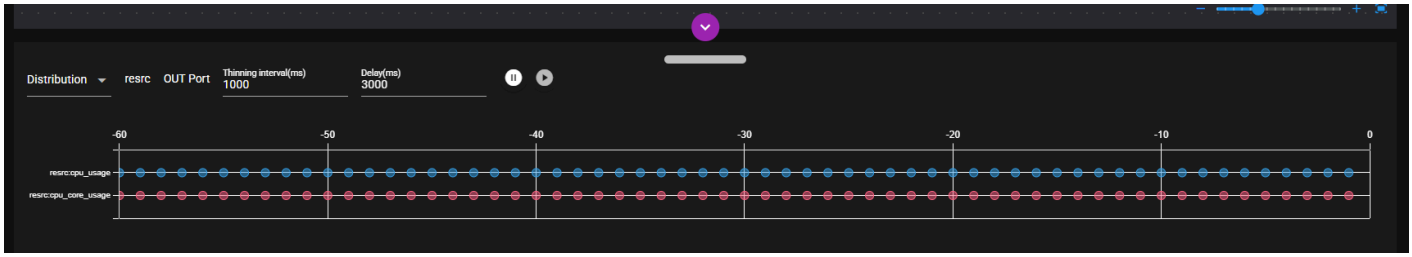
④ System resources



Number	Item	Description
①	Refresh icon	Clicking this icon updates the contents of the display.
②	Gauge	This displays the ratios of resources.

- For detail, see System Resources.

⑤ Data monitor

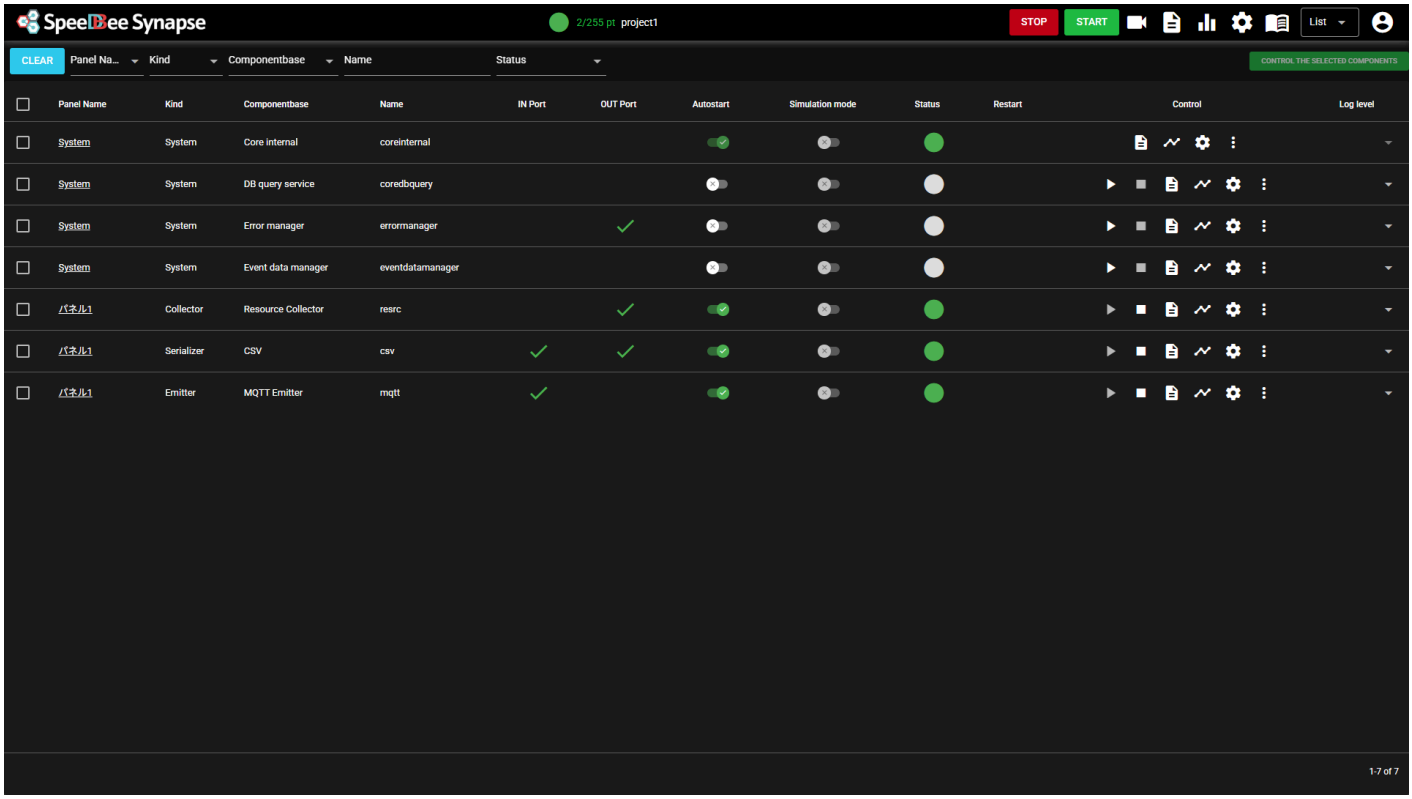


Number	Item	Description
①	Open/close icon	By clicking the indicator, the display/non-display of the data monitor can be switched.
②	Display size adjustment grip	By dragging the cursor and moving the mouse up and down, the display size of the data monitor can be adjusted.
③	Display type selection form	The data monitor display method can be selected.
④	Selection port	A selected port is displayed.
⑤	Thinning interval	The thinning interval for the data to be displayed can be configured.
⑥	Delay time	The delay time for the data to be displayed can be configured.
⑦	Monitor display area	The selected type of monitor is displayed.

- For detail, see Data Monitor.

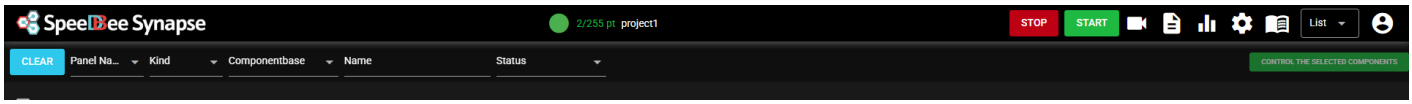
5.3.2.3 List Mode

The items shown on the screen of List Mode Flow Editor are as follows.



The List Mode screen consists of the following: ① Header, ② List header, and ③ Component list

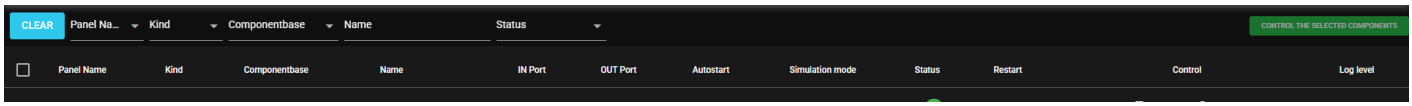
① Header



Number	Item	Description
①	Use Points	This displays the number of registered license points and the number of points used.
②	Current project	This displays the currently selected project. Note: For detail, see Management of Configuration Information (Project).
③	Stop button	Clicking the stop button stops all components registered in SpeeDBee Synapse.
④	Start button	Clicking the Start button starts all the components registered in SpeeDBee Synapse for which "Autostart disable" is set to OFF.
⑤	Video Viewer icon	Clicking the Video Viewer icon switches the display to the Viewer screen. Note: This is displayed only when the "i-Trigger" extension custom package is applied. For detail, see i-Trigger (extension custom package).

Number	Item	Description
⑥	Log icon	Clicking the Log icon displays the SpeeDBee Synapse operation log window. Note: For detail, see Logs.
⑦	Graph icon	Clicking the Graph icon displays the Grafana screen. Note: For detail, see Integration with Grafana.
⑧	Setting menu icon	Clicking this icon displays the Setting Menu. Note: For detail, see Setting Menu.
⑨	Manual	Clicking this displays the manual screen.
⑩	Mode selection	Flow Editor display mode can be selected from below. • Edit Mode • Run Mode • List Mode Note: This is displayed only when a user logs in with the user role of system Administrator or manager.
⑪	User management icon	Clicking this displays the User Management menu. Note: For detail, see User Management.

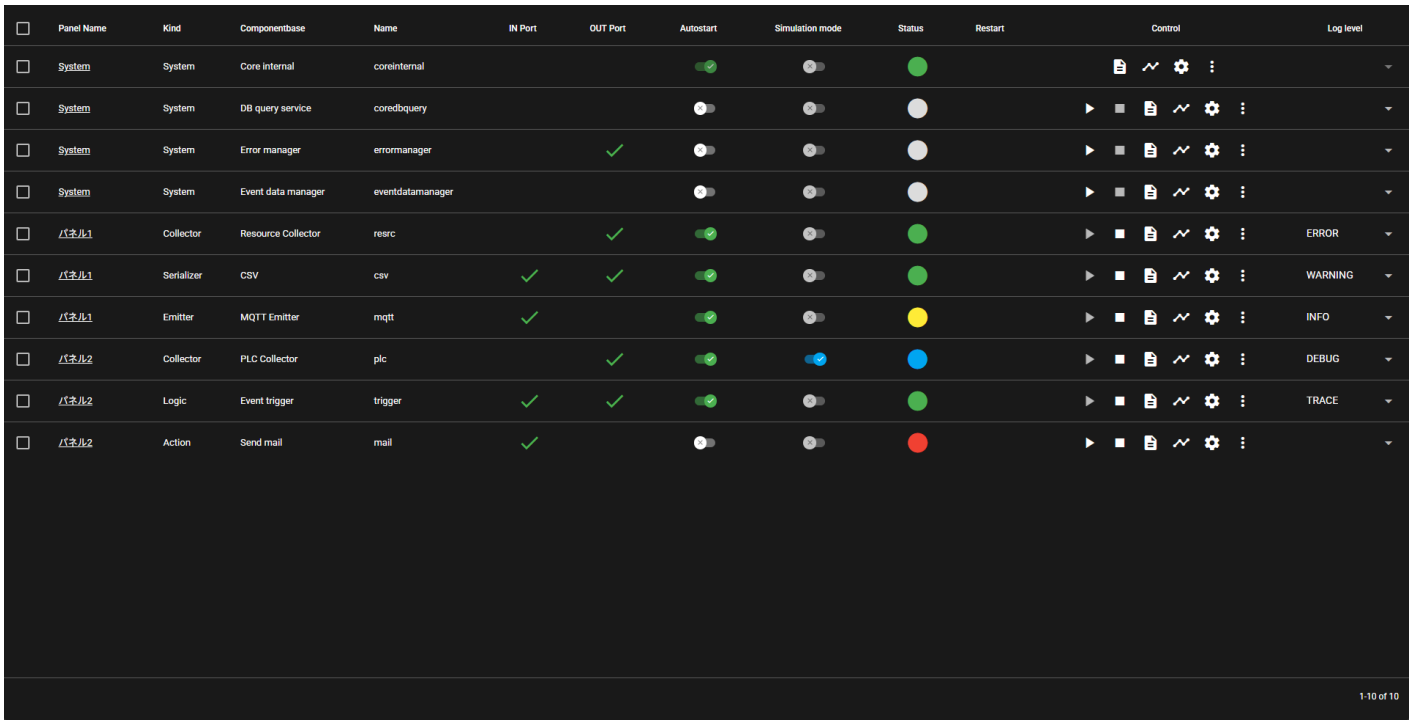
② List header



Number	Item	Description
①	Clear	Clicking this clears the filter conditions.
②	Panel name	Only components in the selected panel are displayed in the component list.
③	Type	Only components of the selected type are displayed in the component list.
④	Component base	Only components of the selected component base are displayed in the component list.

Number	Item	Description
⑤	Name	Only components whose names partially match the entered string are displayed in the component list. Note: Case is not distinguished.
⑥	Status	Only components with the selected status are displayed in the component list. Note: For details on status, see Status List.
⑦	Control the selected components	Clicking this opens a menu to control the components selected in the component list. Note: For detail, see Control the selected components.

③ Component list



Number	Item	Description
①	Selection	Select this to control the selected component.
②	Panel name	The name of the panel where the component is placed. Note: Clicking this displays the clicked panel in Edit Mode.
③	Kind	The kind of the component.

Number	Item	Description
④	Componentbase	The name of the componentbase.
⑤	Name	The name of the component.
⑥	IN port	Displayed if the component has an IN port.
⑦	OUT port	Displayed if the component has an OUT port.
⑧	Autostart	Displays whether autostart of the component is enabled or disabled. Click to toggle enabled/disabled.
⑨	Simulation mode	Displays whether simulation mode of the component is enabled or disabled. Click to toggle enabled/disabled.
⑩	Status	Displays the status of the component. Note: For details on status, see Status List.
⑪	Restart	Displays whether the component needs to be restarted.
⑫	Control	Performs various controls of the component. Note: For details, see Component list control.
⑬	Log level	Displays the individual log level of the component. Change the log level by selecting from the dropdown.

5.3.3 Screen Operation

This section describes how to operate the Flow Editor.

5.3.3.1 Setting menu

This section describes how to operate the Setting Menu.

> The Setting Menu can be operated only with Edit Mode.

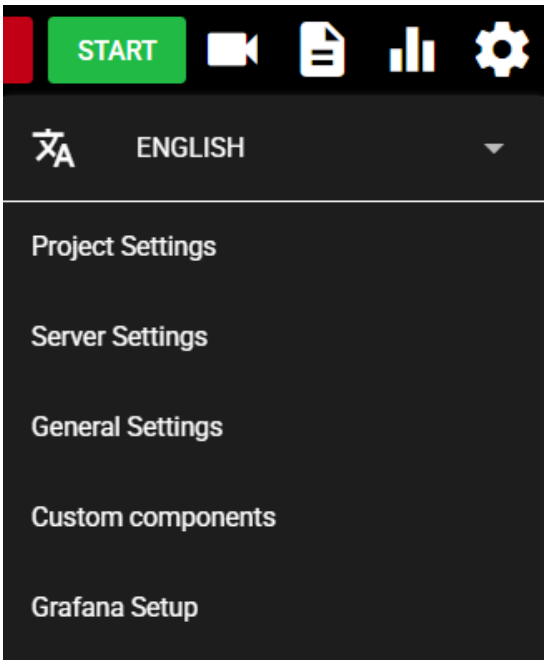
5.3.3.1.1 Language configuration

Languages can be switched by following the steps below.

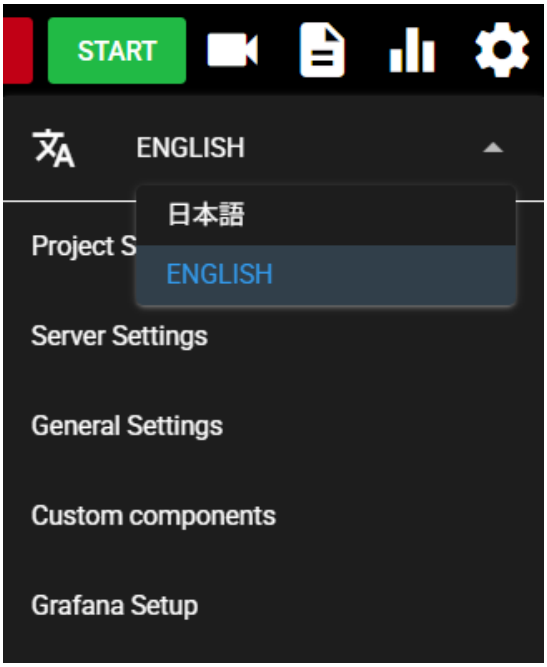
1. Press the Setting Menu icon.



2. Select the language selection pull-down menu.



3. Select a language.



4. A selected language is reflected in SpeeDBee Synapse.

The currently available languages are as follows:

Language	Code	Description
日本語	ja	Displays Synapse in Japanese.
ENGLISH	en	Displays Synapse in English.

5.3.3.1.2 Project Settings

This enables the configuration of a project. For detail, see Management of Configuration Information (Project).

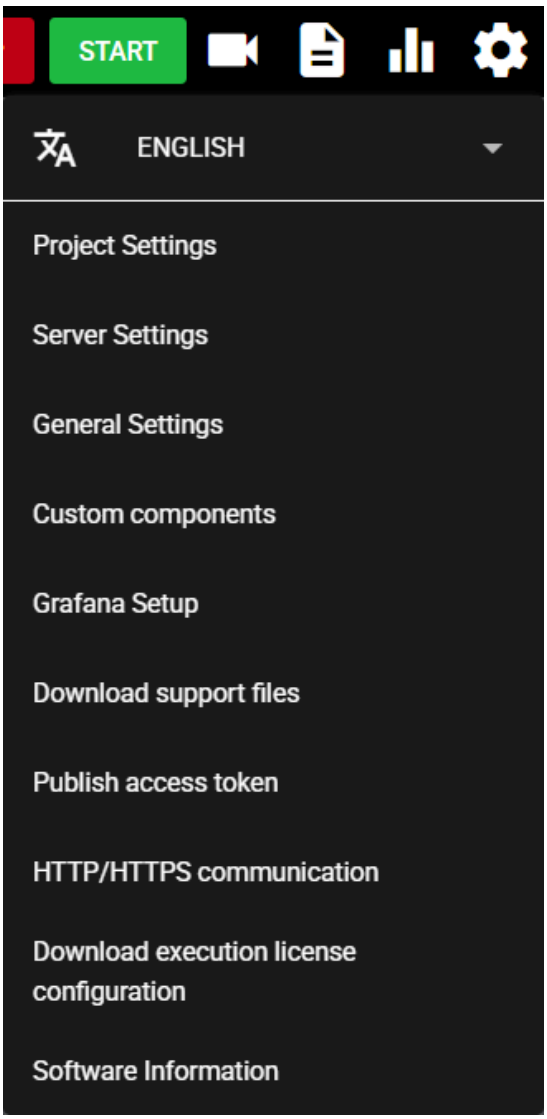
5.3.3.1.3 Server Settings

By following the steps below, server settings can be confirmed and updated, and the database can be reset.

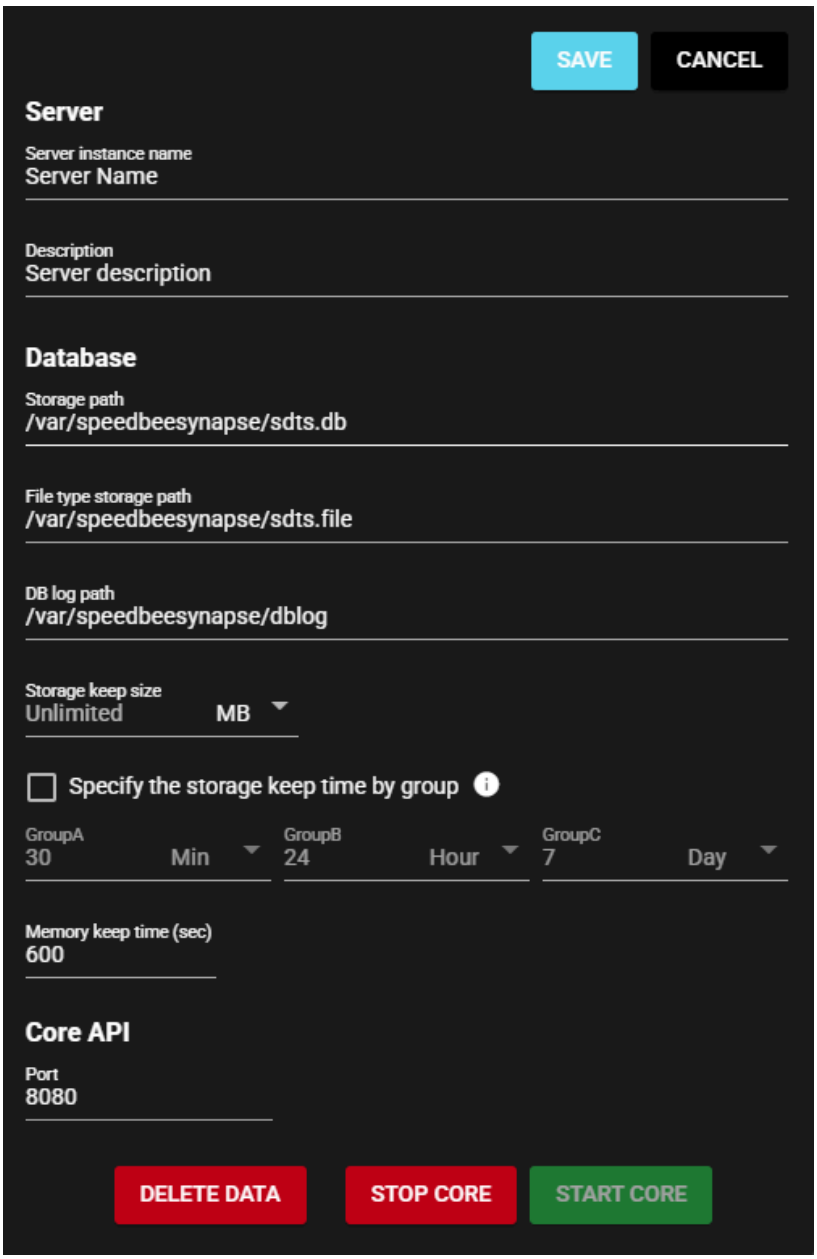
1. Press the Setting Menu icon.



2. Select "Server Settings".



3. From the following items, input any one and press "Save" to update the configuration.

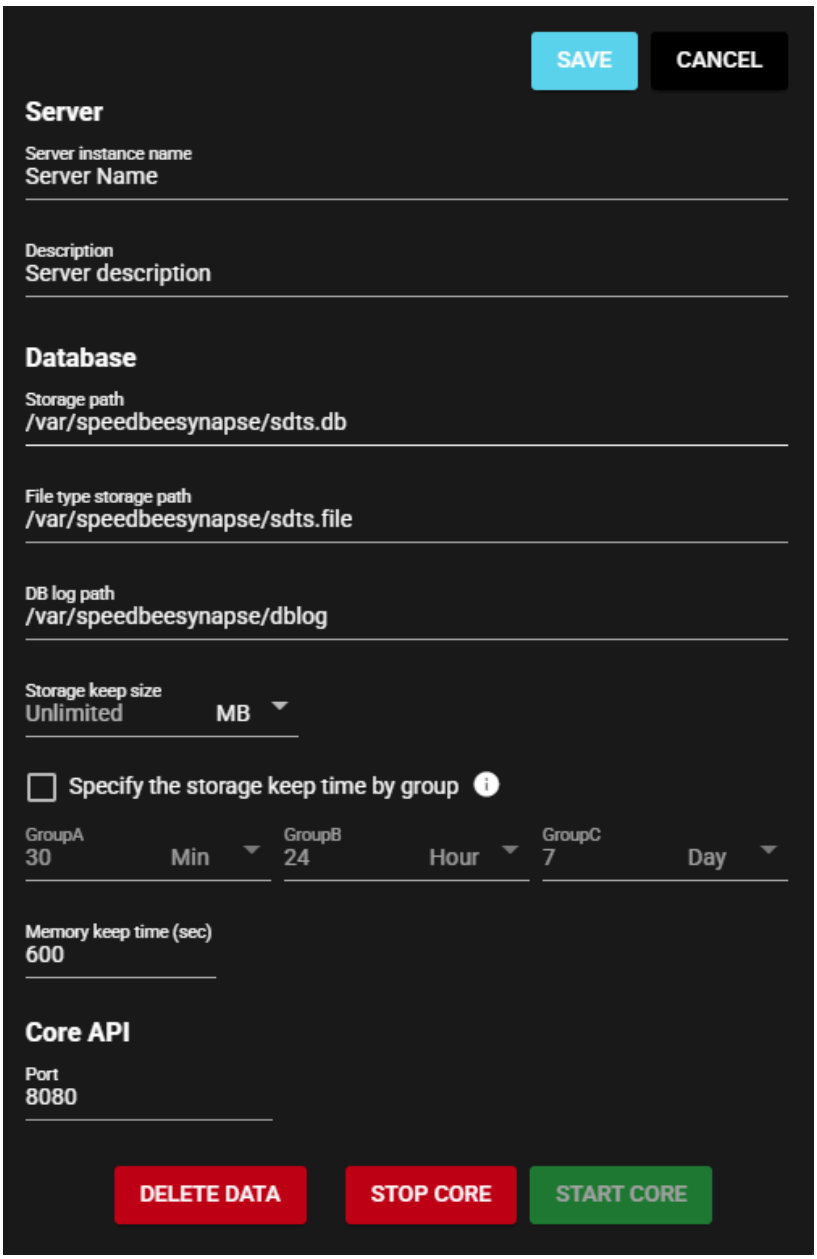


Item		Description
Server	Server instance name	This is a name assigned to SpeeDBee Synapse. The name is displayed in the general panel of Run Mode. (If multiple SpeeDBee Synapse are used, they can be distinguished by assigning different names individually.)
	Description	Descriptions including the purpose of SpeeDBee Synapse
Database	Storage path	The database save location directory Note: In some environments, this setting cannot be changed/edited.

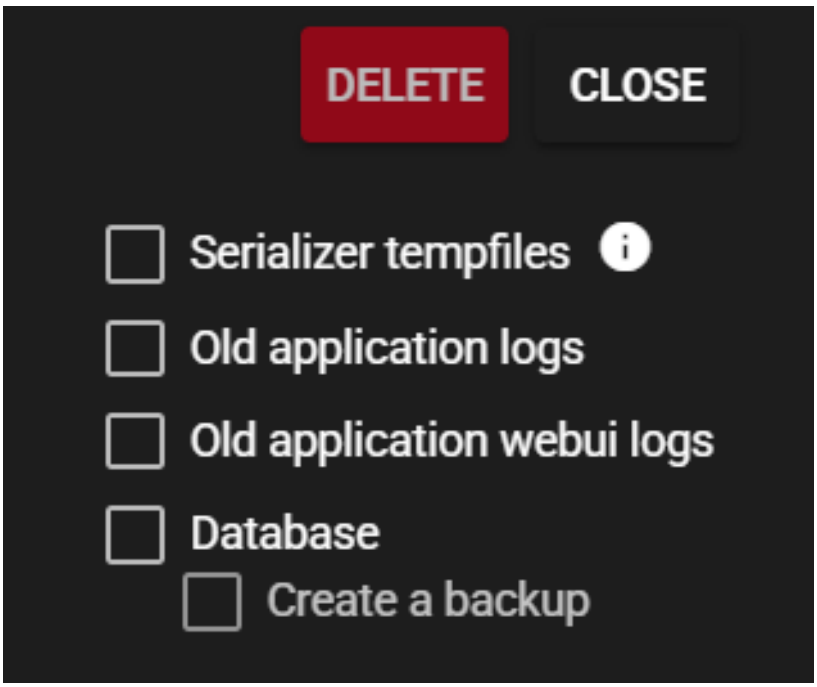
	File type storage path	File type database save location directory Note: In some environments, this setting cannot be changed/edited.
	DB log path	Directory for database log save location Note: In some environments, this setting cannot be changed/edited.
	Storage keep size	Maximum size to store data in storage If the data in the storage exceeds this keep size, the data is deleted starting from the oldest data. If the keep size is undefined, it is unlimited.
	Specify the storage keep time by group.	If ON is selected, the keep time—the period for retaining data in the storage—can be configured for each group. Data that exceeds the keep time is deleted. However, even before the keep time expires, if the storage keep size is exceeded, data is deleted starting from the oldest data.
	Memory keep time (sec)	This is the period during which data is stored in memory.
Core API	Port	This is the port number used by the SpeeDBee Synapse to communicate between the web interface and the internal server. Note: In some environments, this setting cannot be changed/edited.

Delete Data

1. Press "Delete data".

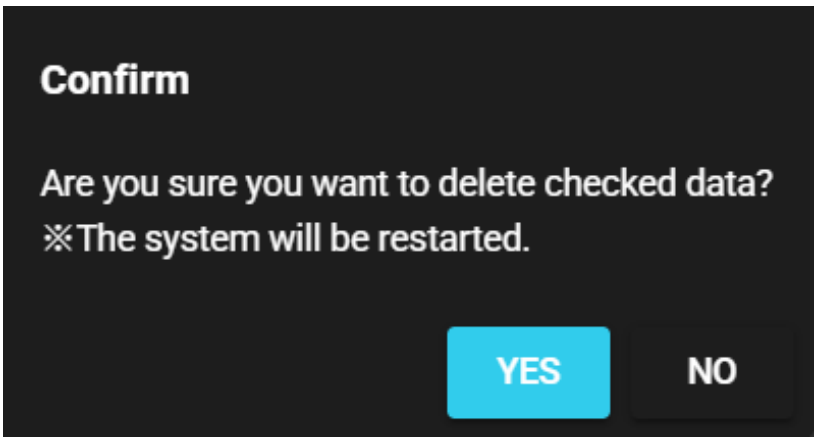


2. Select the data to delete and press "Delete".



Item	Description
Serializer tempfiles	These are temporary output files created by the serializer component. Note: If deleted, the system is restarted.
Application Log	System logs for the core, component, and database
Application log (UI)	WEBUI system log
Database	Time series database of SpeeDBee Synapse Note: If deleted, the system is restarted.
Create a backup	A backup is created before the database is deleted. Note: This can be selected only when the database is checked.

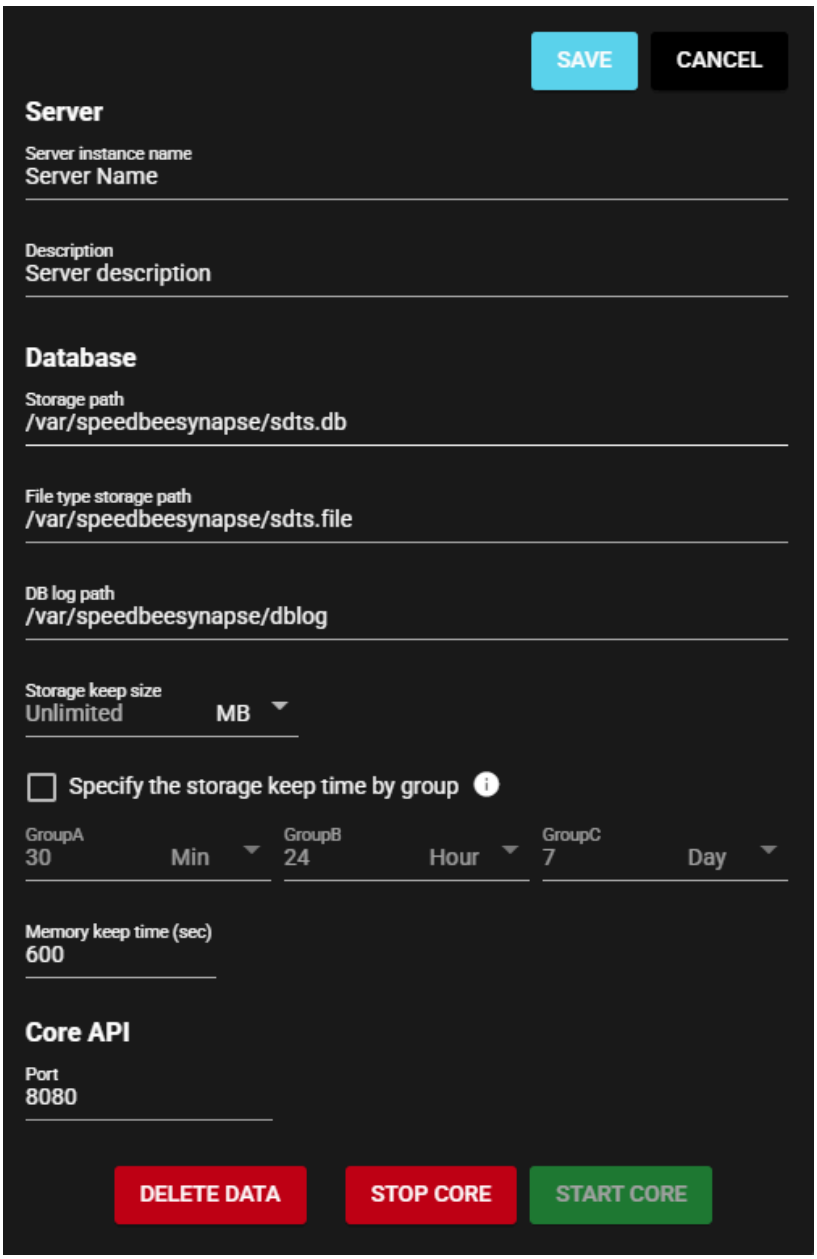
3. Press "Yes".



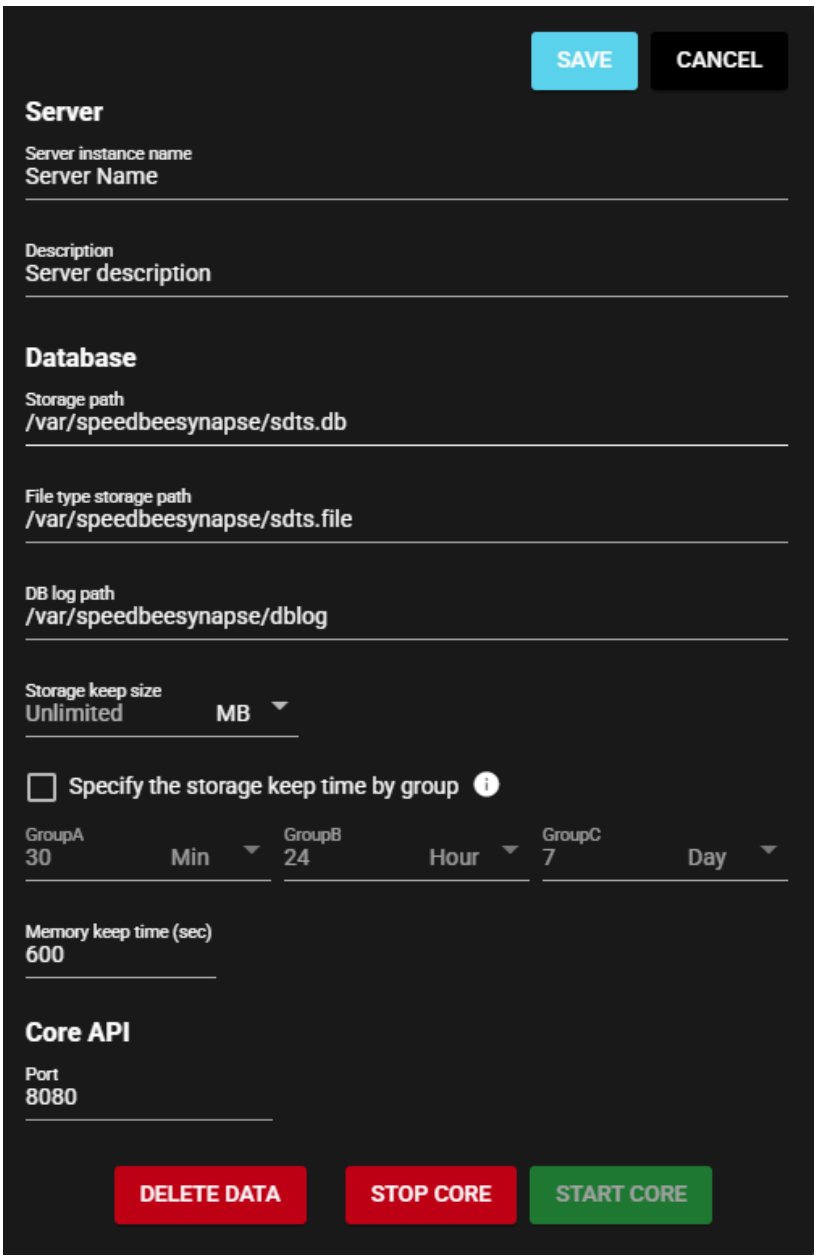
4. The selected data is deleted.

Start/Stop core

1. Press "Stop core" to stop the core function of SpeeDBee Synapse.



2. Press "Start core" to start the core function of SpeeDBee Synapse.



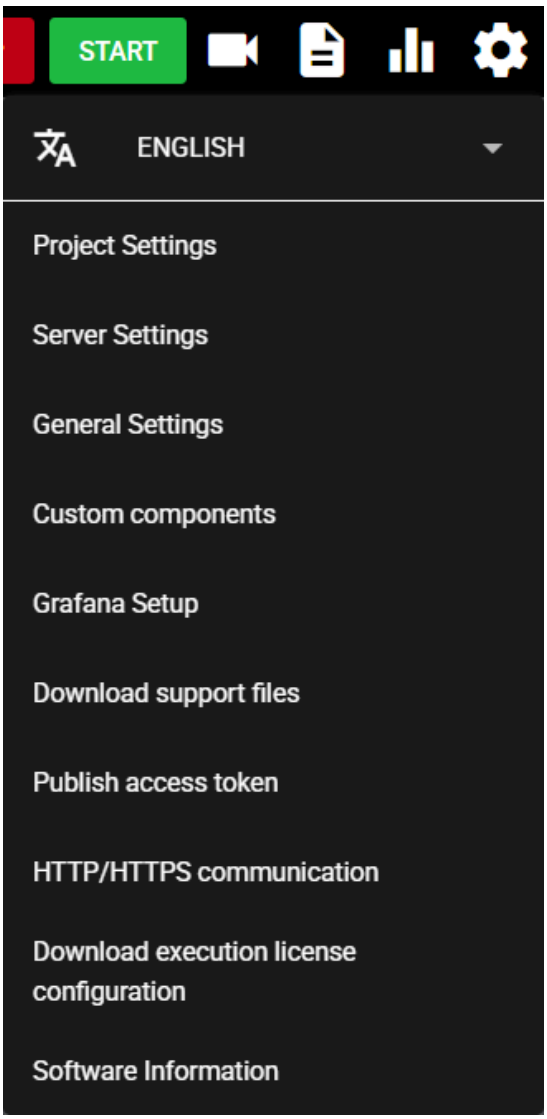
5.3.3.1.4 General Settings

The general settings can be confirmed and updated by following the steps below.

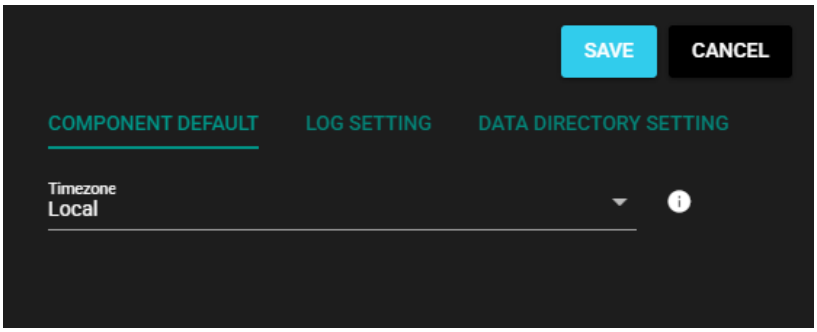
1. Press the Setting Menu icon.



2. Select "General Settings".



3. Switch the tab, edit the items of choice, and then press "Save" to update the configuration.



Item	Description
Component default	This sets the common initial default value for the component.
Log Setting	This configures the application log.
Data directory setting	This configures the data directory.

5.3.3.1.5 Component default

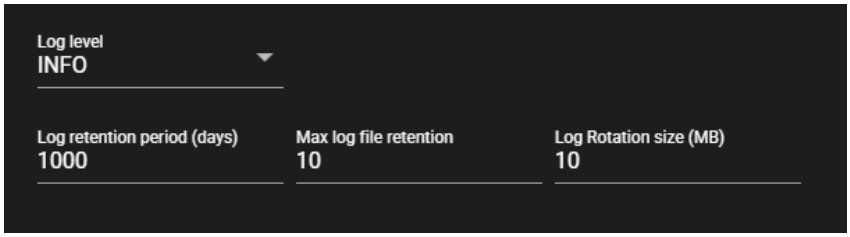
This confirms and edits initial values for common inputs of components.



Item	Description
Timezone	The default time zone that can be configured for CSV, JSON serializers, and File Emitters

5.3.3.1.6 Log configuration

This confirms and edits the application log configuration.



Item	Description
Log level	Log output level
Log retention period (days)	Log retention period in days
Max log file retention	Maximum number of log files to be saved
Log Rotation size (MB)	Size of log rotation (MB)

5.3.3.1.7 Data directory setting

This confirms and edits the data directory configuration.



Item	Description
Data directory	The base directory for save location and output location that can be configured with VIDEO Collector, File Emitter, and Event Recorder. Note: In some environments, this setting cannot be changed/edited.

Note: If you change the data directory, the changes will be reflected in the following components from the next startup, and the new directory will be used.

- VIDEO Collector
- File Emitter
- Event Recorder

5.3.3.1.8 Custom (Python)

The Python Custom Component can be configured. For detail, see Creation and Python Custom Component.

5.3.3.1.9 Custom (C)

C Custom Component can be configured. For detail, see Creation of C language Custom Component.

5.3.3.1.10 Grafana Setup

This enables the configuration of Grafana. For detail, see Integration with Grafana.

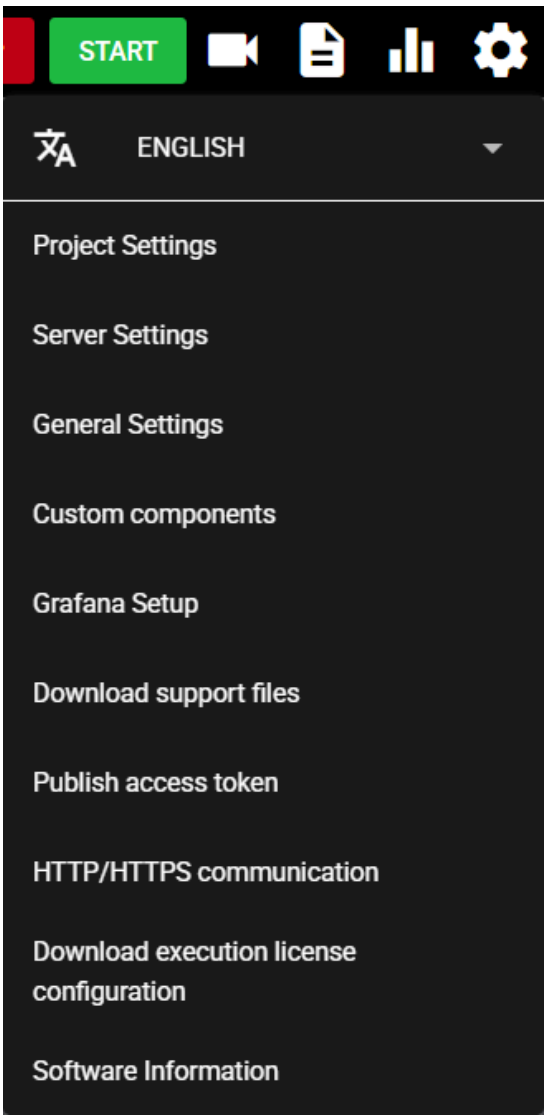
5.3.3.1.11 Download support files

Files to be given to the SpeeDBee Synapse support staff can be downloaded by following the steps below.

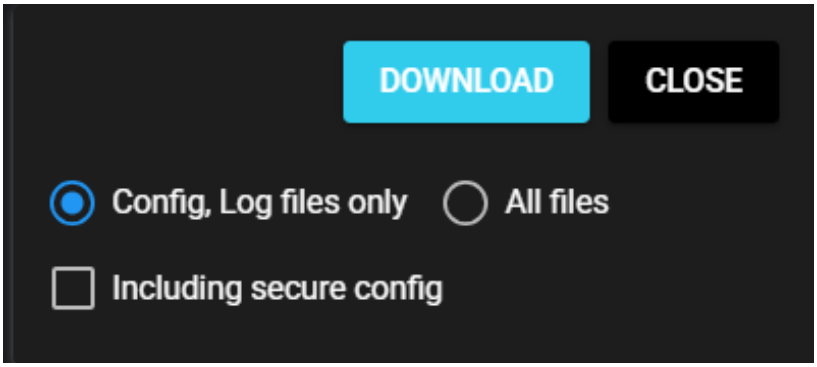
1. Press the Setting Menu icon.



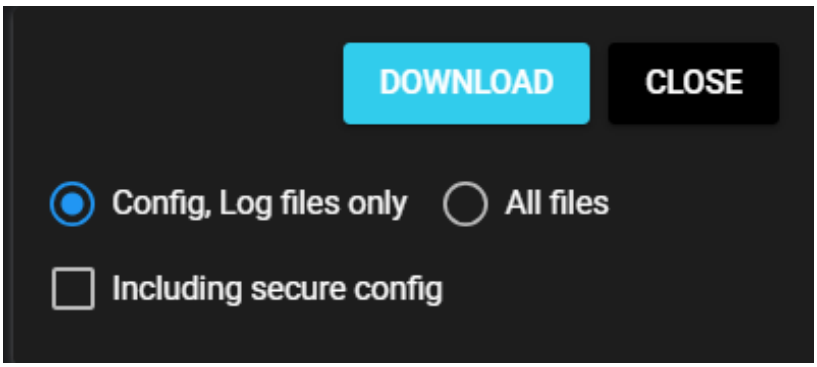
2. Select "Download support files".



3. Select the file to download.
   Note: Select "Config, Log files only" only as a standard procedure. Confidential information does not have to be included unless instructed by support staff to do so.



4. Press "Download".



5. Support files are downloaded.

Protected Custom Components

When you select "All files" for download, custom components are included.
However, protected custom components (Python, C, and SCC packages) are excluded from the support file download for security reasons.

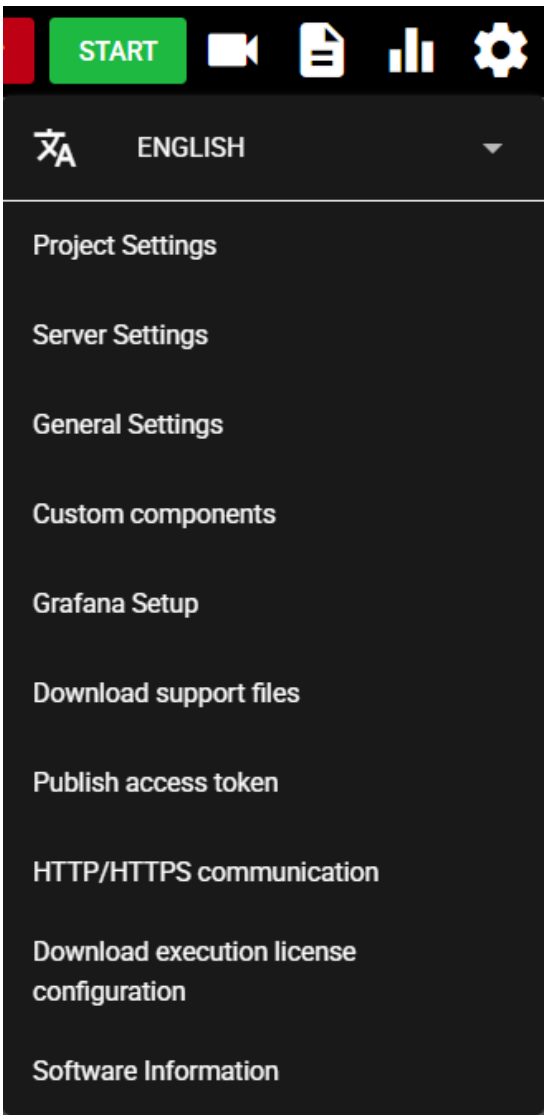
5.3.3.1.12 Publish access token

Access token can be published by following the steps below. The published access tokens can be used in "DB query service", "Integration with Grafana", and "Integration with control program".

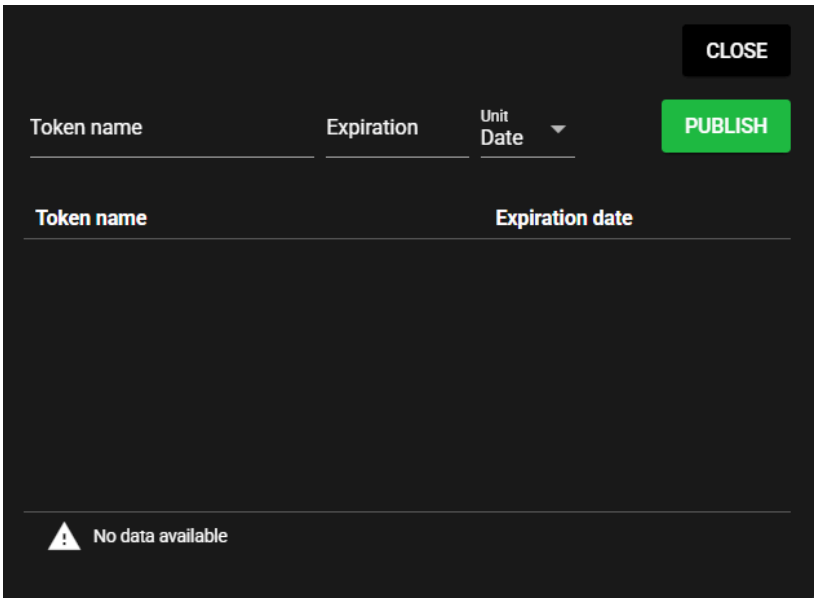
1. Press the Setting Menu icon.



2. Select "Publish access token".



3. Input the following items and press "Publish".



Item	Description
Token name	Name to identify a token
Expiration date	Expiration date for the token

About the expiration date of access tokens

If the expiration date for a token is set long, the security risk in the event of a token breach increases. Conversely, if the expiration date is set too short, security is enhanced, but the access token must be updated frequently. Depending on a predetermined operation policy, consider updating the access token regularly.

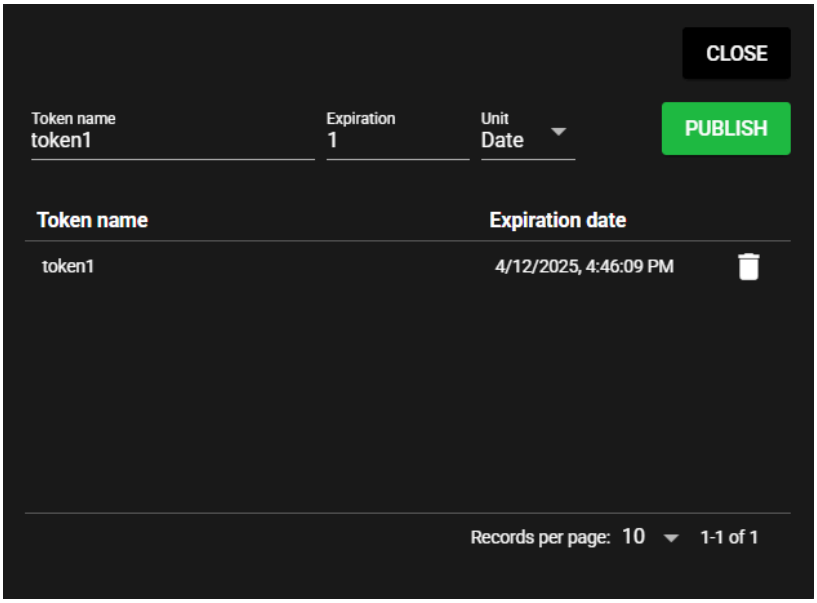
4. After copying a published access token, press "Close".

Storage of access token

The access token is displayed only once. The same token cannot be retrieved once this screen is closed, so be sure to make a copy.



5. Published access tokens are added to the list.



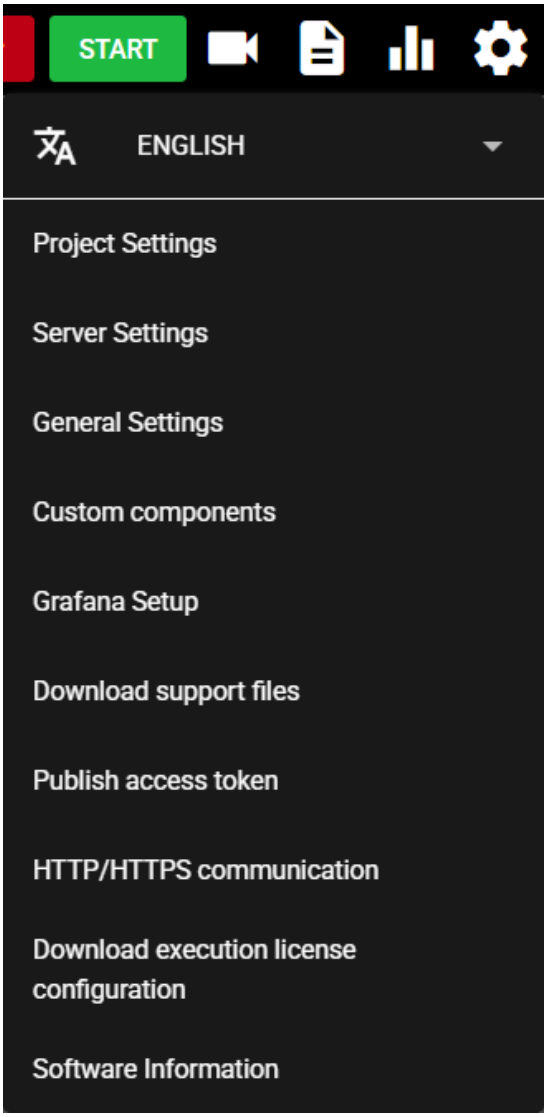
5.3.3.1.13 HTTP/HTTPS communication

The user device and SpeeDBee Synapse HTTP communication can be configured by following the steps below.

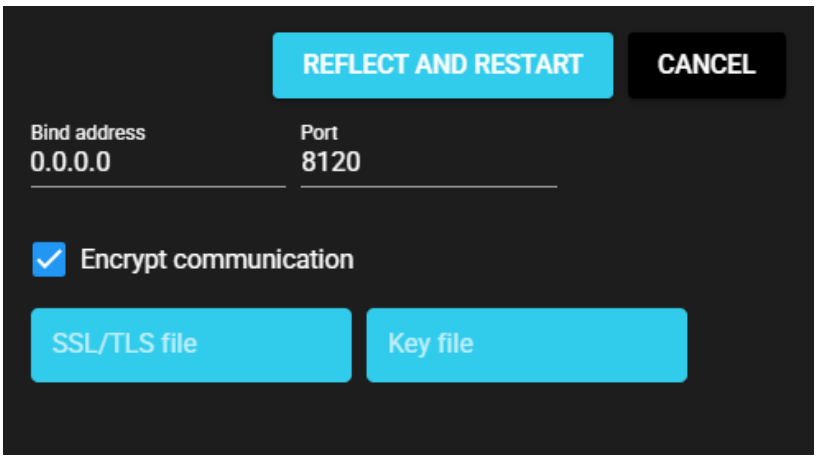
1. Press the Setting Menu icon.



2. Select "HTTP/HTTPS communication".



3. Input the following items and press "Reflect and restart".



Item	Description
Bind address	This is the address at which SpeeDBee Synapse accepts requests. Note: In some environments, this setting cannot be changed/edited.
Port	This is the port at which SpeeDBee Synapse accepts requests. Note: In some environments, this setting cannot be changed/edited.
Encrypt communication	Select ON to use encrypted communication (HTTPS). Note: In some environments, this setting cannot be changed/edited.
SSL/TLS certificates	This is a digital certificate used for encryption. Note: The certificate needs to be obtained from the certificate authority (CA).
Secret key	This is a secret key linked to the SSL/TLS certificate. Note: The secret key needs to be obtained from the certificate authority (CA).

4. The whole system is restarted to reflect configurations. After restarting, SpeeDBee Synapse can be accessed with the URL that matches the settings.

When using encrypt communication

```
https://< Host name or IP address of the device>:< This is the port at which SpeeDBee Synapse accepts requests.>/
```

When not using encrypt communication

```
http://< Host name or IP address of the device >:< This is the port at which SpeeDBee Synapse accepts requests.>/
```

5.3.3.1.14 Proxy Settings

This section describes the operation of Proxy Settings. The proxy information set here can be referenced from the settings of each component, and its communication can be route through a proxy.

Proxy-Compatible Components

- MQTT Collector
- MQTT Emitter
- Cloud Emitter for AWS
- Cloud Emitter for Azure
- MQTT Client

By setting a default proxy, communication for specific operations will be routed through the proxy.

Operations Affected

- Adding external libraries for Python custom components

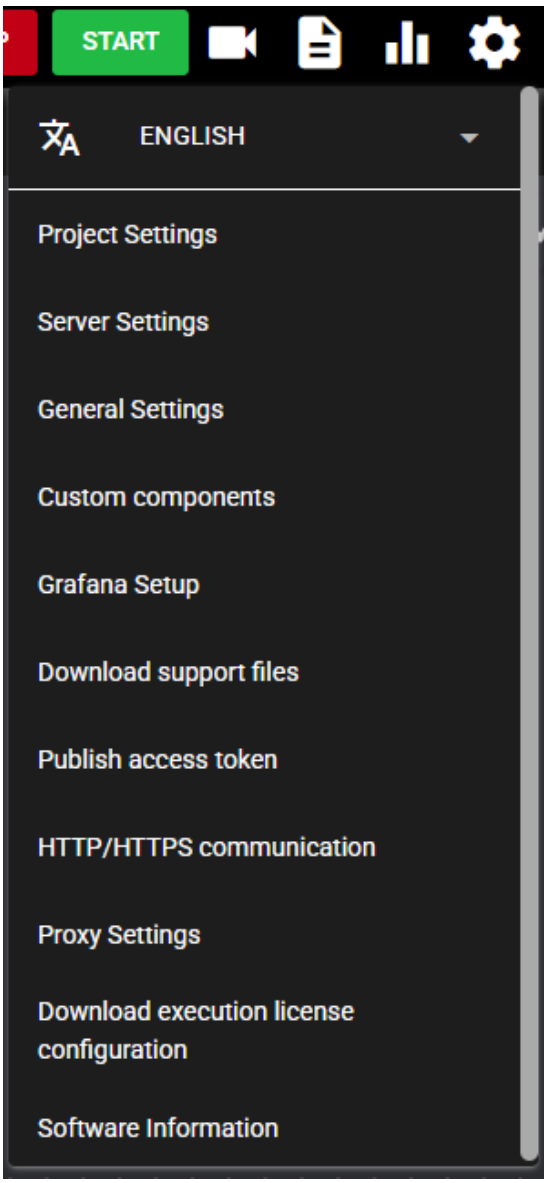
Add

Proxy settings can be added by following the steps below.

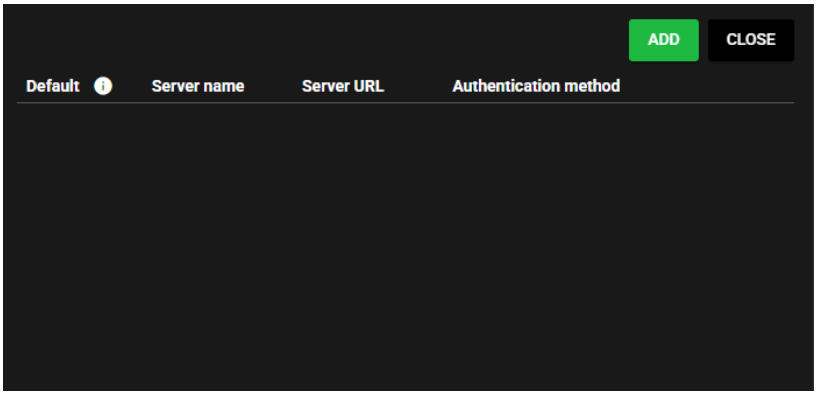
1. Press the Setting Menu icon.



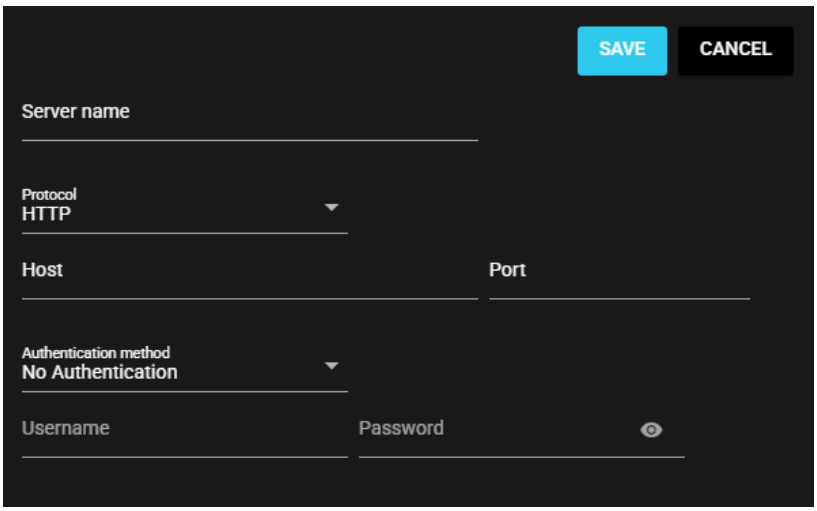
2. Select "Proxy Settings".



3. Select "ADD".

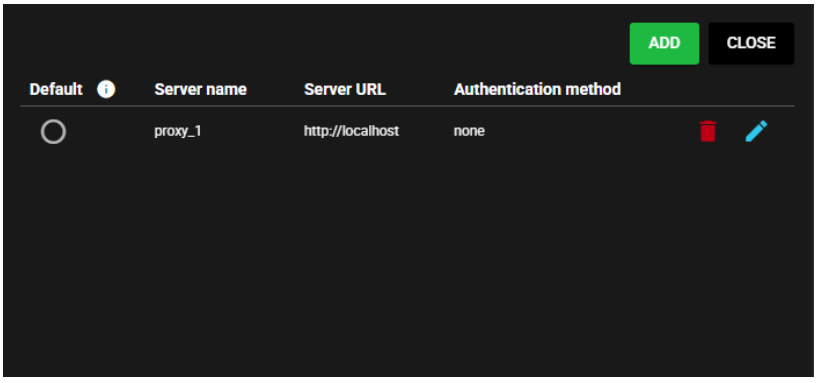


4. Input the following items and press "SAVE".



Item	Description
Server name	An optional name to identify the server. Note: Components configured to communicate via a proxy reference the server name to load their settings. Therefore, if the server name is changed, the components must be reconfigured.
Protocol	Type of proxy to use.(HTTP)
Host	Host name or IP address of the proxy server.
Port	Port number of the proxy server.
Authentication method	Proxy server authentication method. (No Authentication, Basic Authentication)
Username	Username for Basic Authentication.
Password	Password for Basic Authentication.

5. The saved setting will be added to the list.



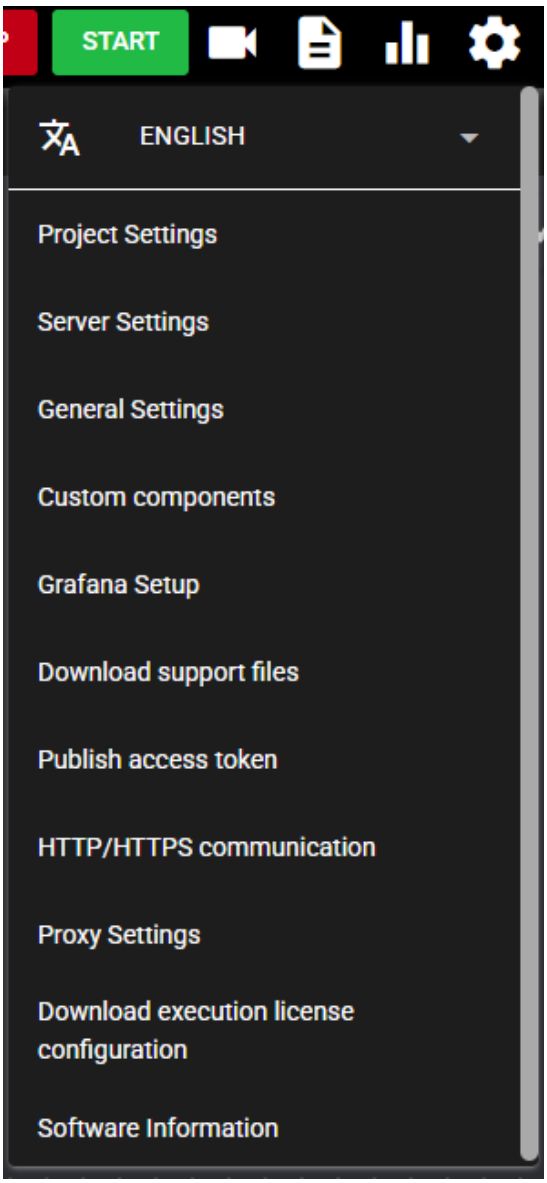
Edit

Proxy settings can be edited by following the steps below.

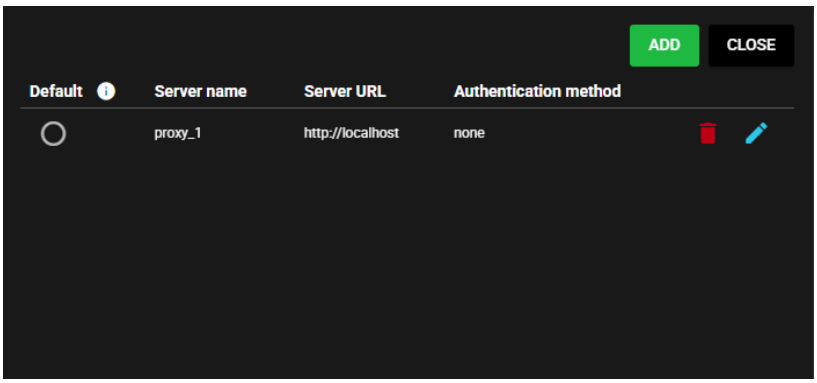
1. Press the Setting Menu icon.



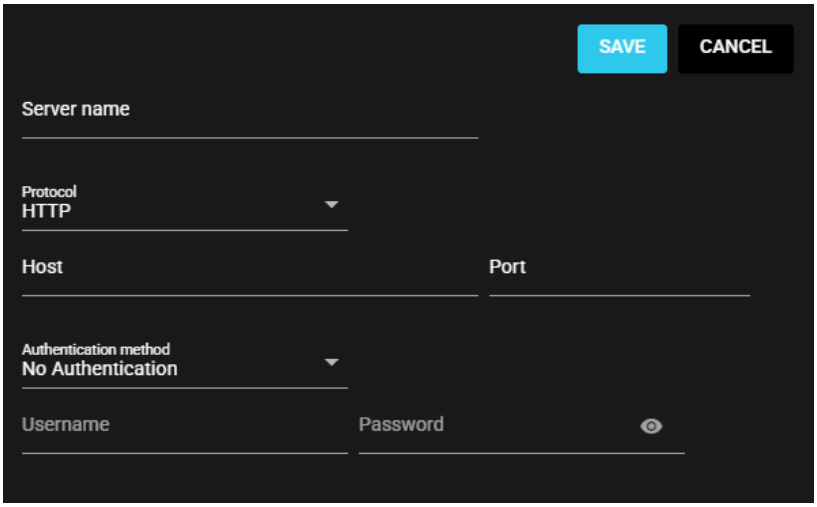
2. Select "Proxy Settings".



3. Select the edit icon.



4. Input the following items and press "SAVE".



Item	Description
Server name	An optional name to identify the server. Note: Components configured to communicate via a proxy reference the server name to load their settings. Therefore, if the server name is changed, the components must be reconfigured.
Protocol	Type of proxy to use.(HTTP)
Host	Host name or IP address of the proxy server.
Port	Port number of the proxy server.
Authentication method	Proxy server authentication method. (No Authentication, Basic Authentication)
Username	Username for Basic Authentication.
Password	Password for Basic Authentication.

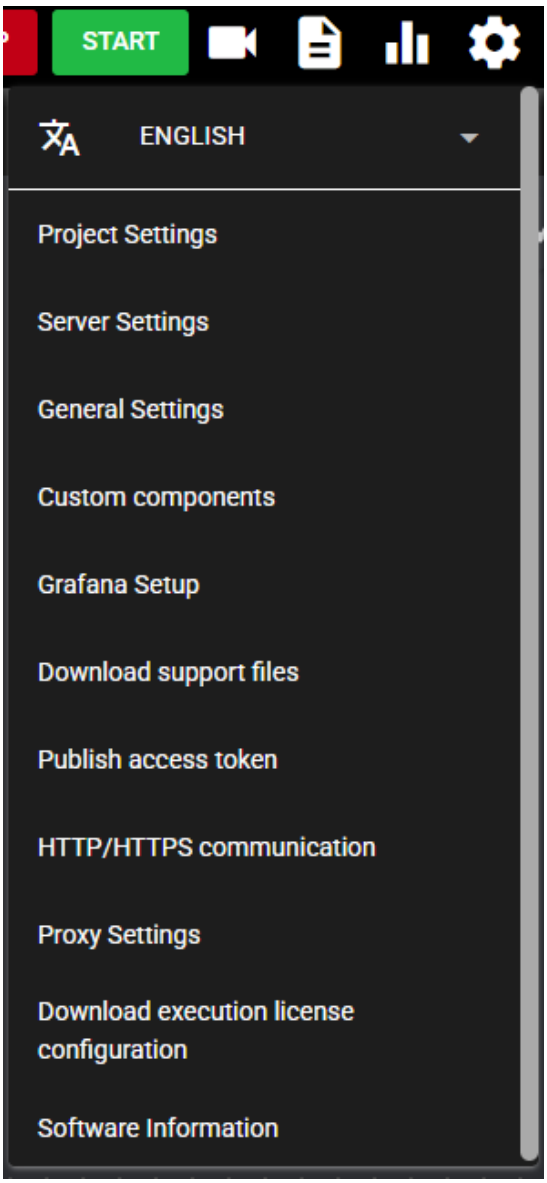
Default Proxy

Default proxy can be set by following the steps below.

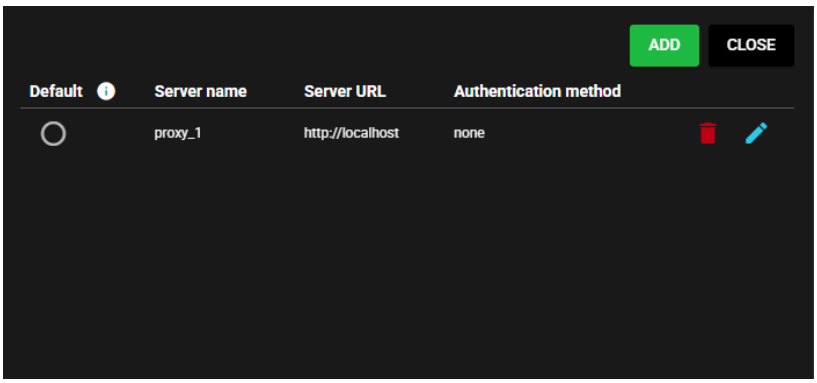
1. Press the Setting Menu icon.



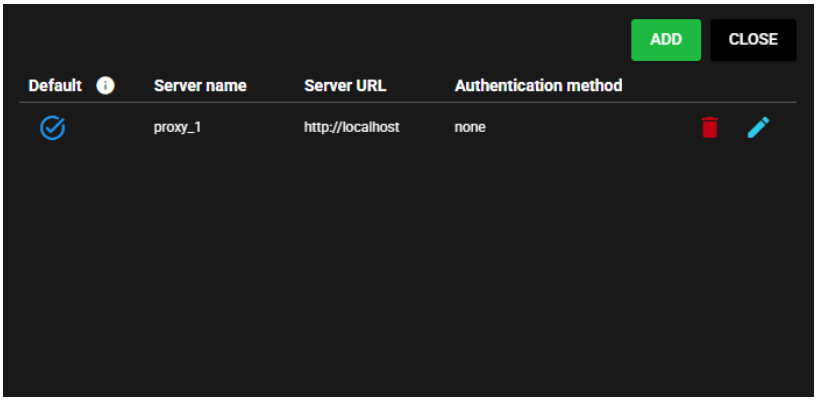
2. Select "Proxy Settings".



3. Select the proxy to set as the default.



Note: Pressing a proxy that is set as the default will unset it, leaving no default proxy configured.



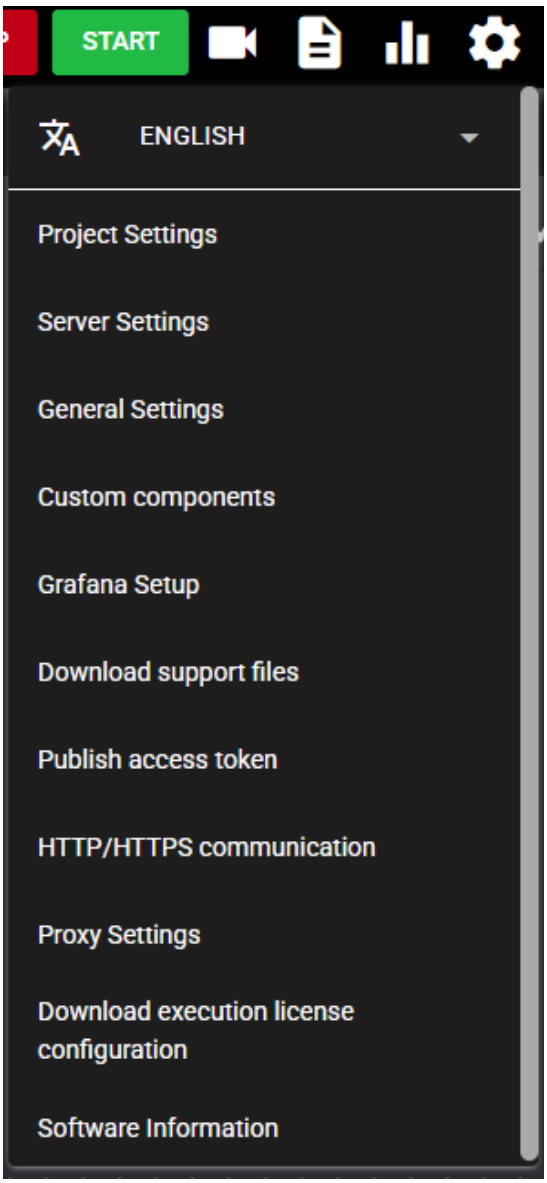
Delete

Proxy settings can be deleted by following the steps below.

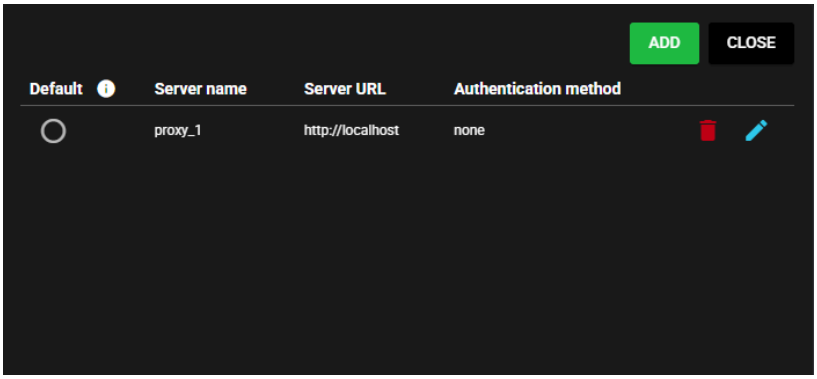
1. Press the Setting Menu icon.



2. Select "Proxy Settings".



3. Press the delete icon.



Note: If a deleted proxy setting was being referenced by a component, the component will need to be reconfigured.

5.3.3.1.15 Download execution license configuration

Configuration can be downloaded for the execution license environment. For details, see Execution license.

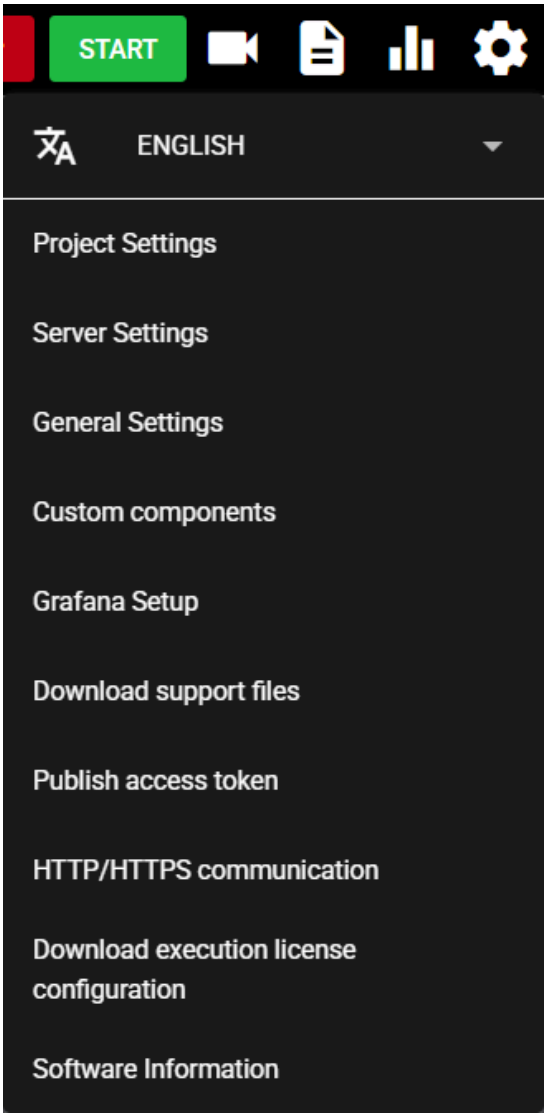
5.3.3.1.16 Software Information

Software information can be confirmed by following the steps below.

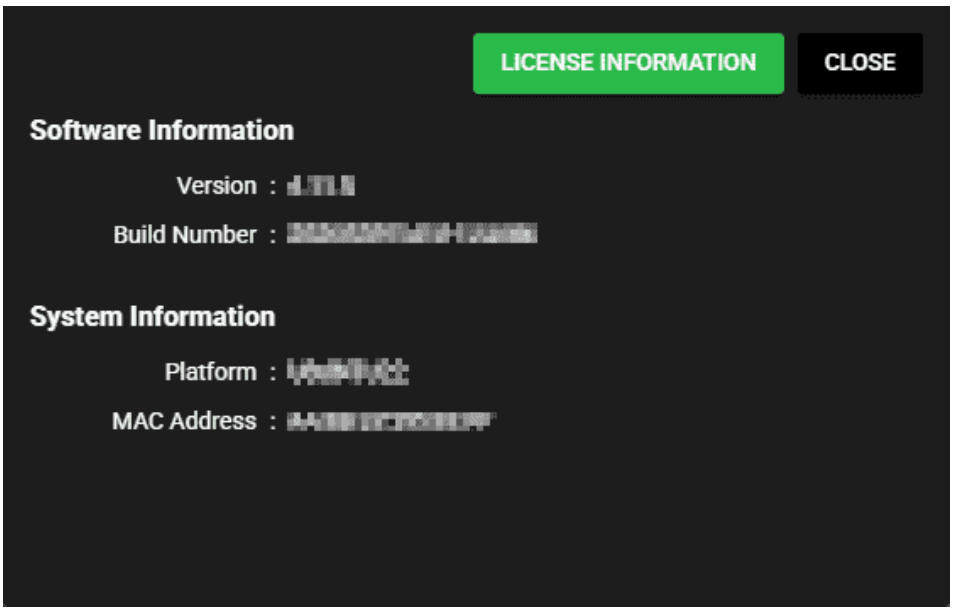
1. Press the Setting Menu icon.



2. Select "Software Information".



3. You can verify software and system information and register licenses.

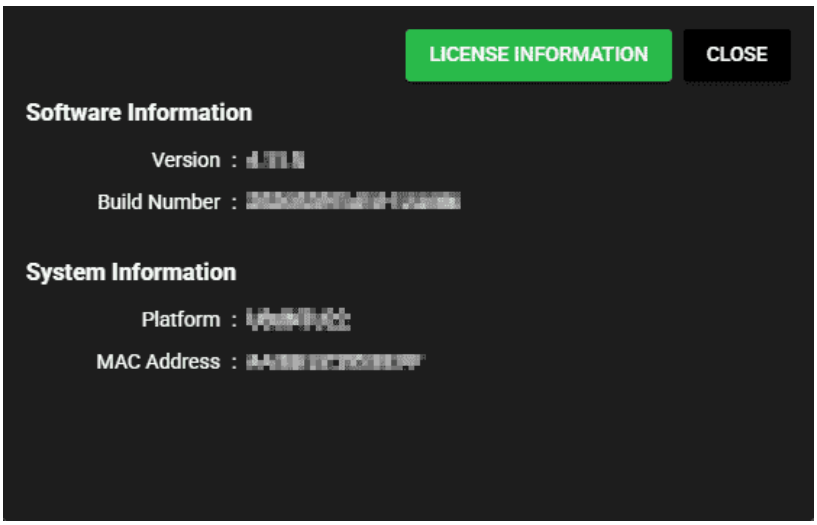


Item		Description
License Information		Verify and register license information
Software Information	Version	The version of SpeeDBee Synapse
	Build Number	The build number of SpeeDBee Synapse
System Information	Platform	Operating Equipment Platform
	MAC address	Operating Equipment MAC address

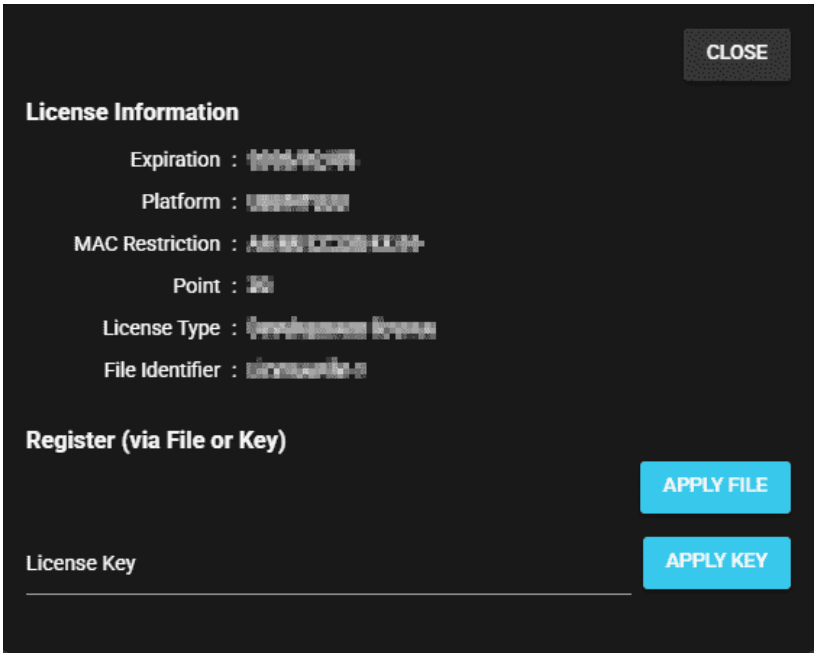
5.3.3.1.17 License Information

License information can be confirmed and registered by following the steps below.

1. Click “License Information” in Software Information.



2. The following items will be displayed.



Item		Description
License Information	Expiration	The expiration date of the registered license
	Platform	The platform that the registered license supports
	MAC address restriction	MAC addresses that the registered license supports
	Point	Points granted for registered licenses
	License type	The registered license type

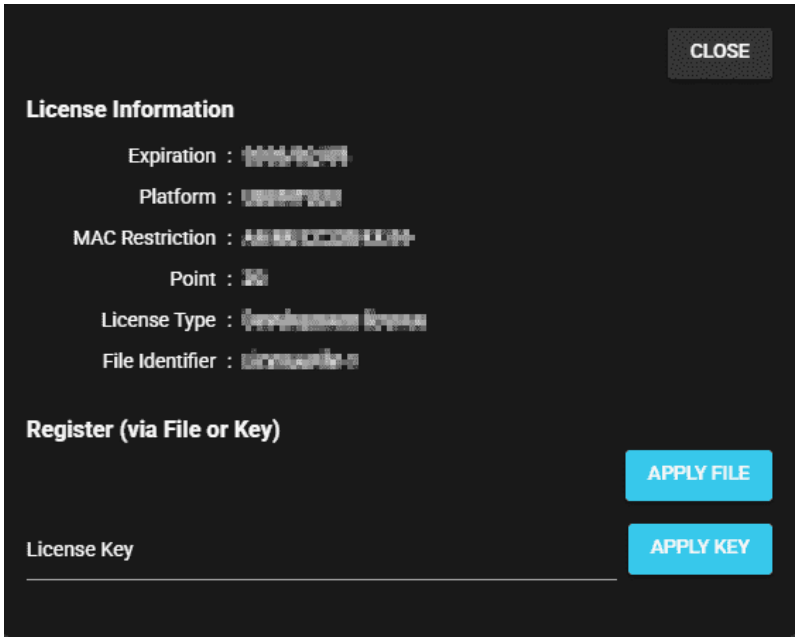
	File Identifier	Identifier of the registered license file ※Not displayed if not applied via file.

Registration

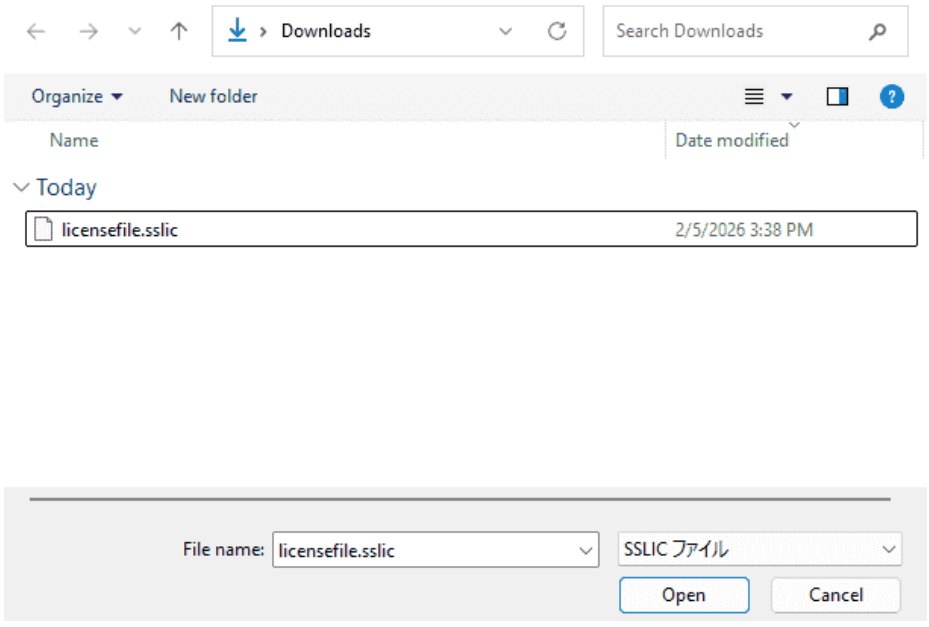
Licenses can be registered using two methods: file or key.

- Register by file

  a. Press the “Apply File” button.

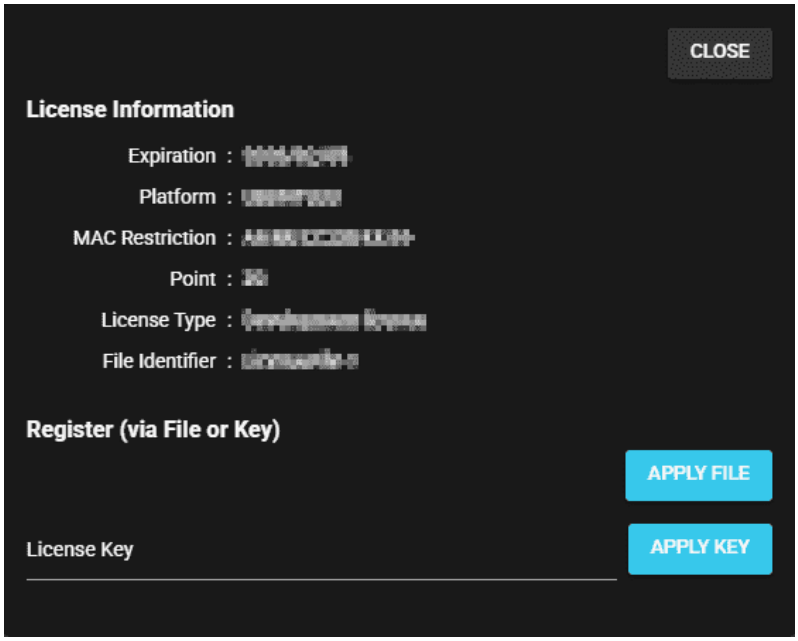


  b. Select the license file.

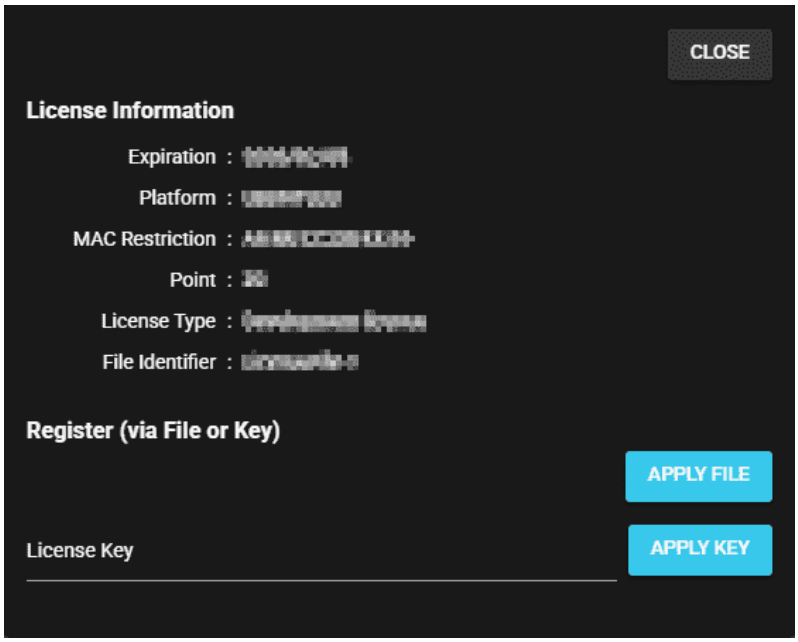


- Register with Key

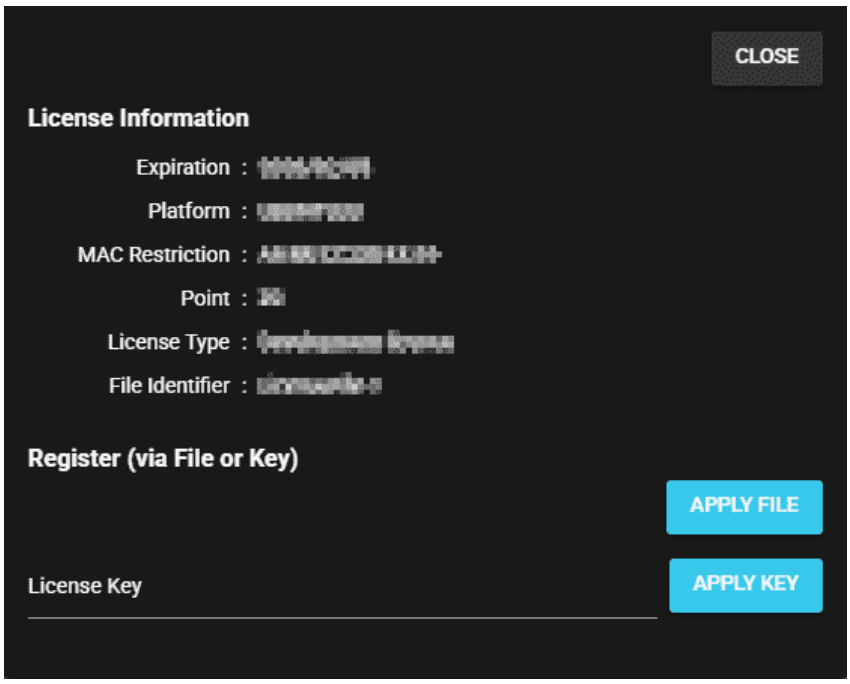
a. Enter the license key in the input field.



b. Click “Apply Key”.



Once the license is registered using either method, the license information will appear on the screen.

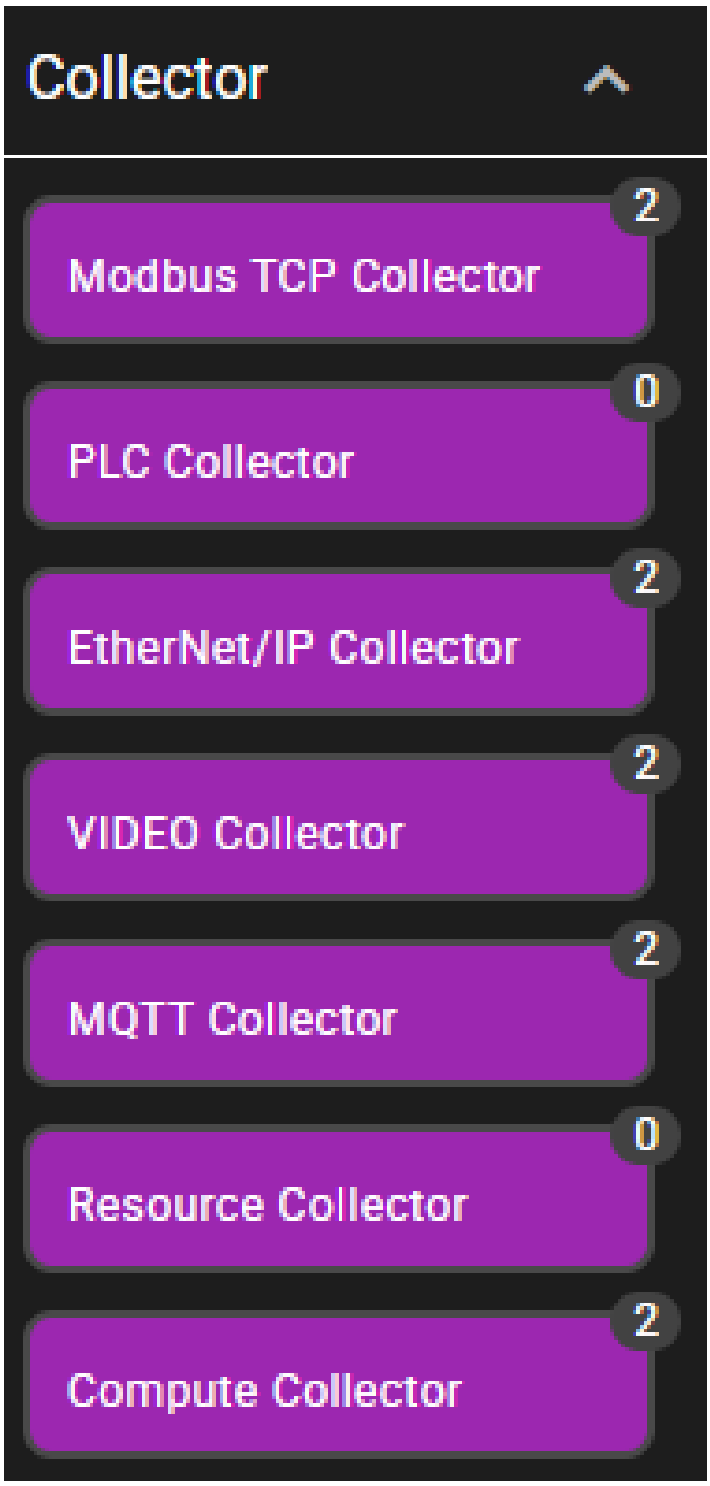


5.3.3.1.18 About Points

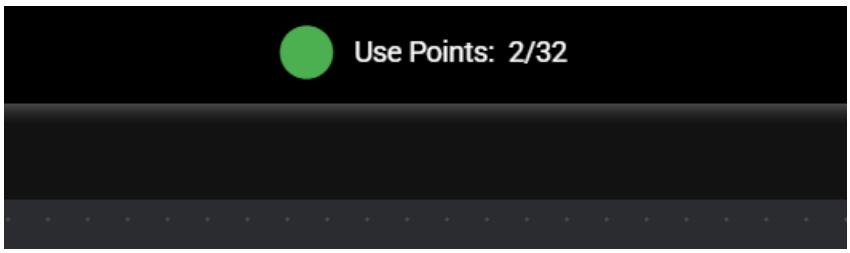
Points to use various functions are given to each SpeeDBee Synapse license. When SpeeDBee Synapse functions are used, the corresponding number of points is consumed.

Notes on Data Flow Controller and OMRON-SBS for Windows editions

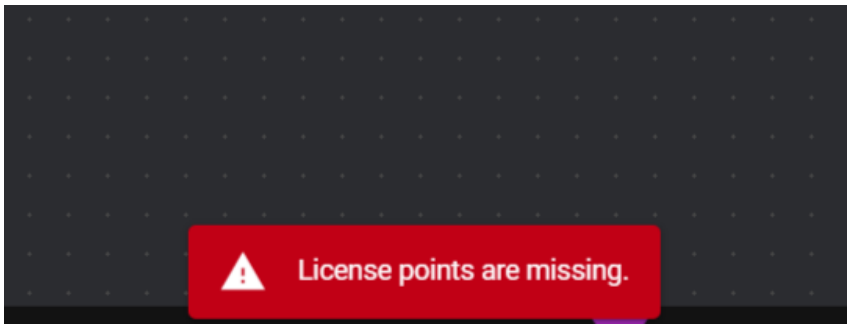
Data Flow Controller and OMRON-SBS for Windows editions can use points without limitation.
However, this does not guarantee unlimited operation. Performance may be affected by your hardware specifications, running applications, and related systems.



If the points are 0, no points are consumed.
If the point is 2, two points are consumed if the configuration is saved and operated.
Also, when the same function is used simultaneously, the number of points consumed remains two points.
For example, even when two MQTT Emitters are used, the number of points consumed remains two.



If there is a shortage of points for functions to run simultaneously, the component operation is disabled.



Cautions when using the same function at the same time

When the same function is used simultaneously, the number of points consumed remains the same. However, this does not guarantee operations without limit. This is affected by the specifications of the hardware in use, applications to use, and related systems.

5.3.3.2 User Management

This section describes user management operations.

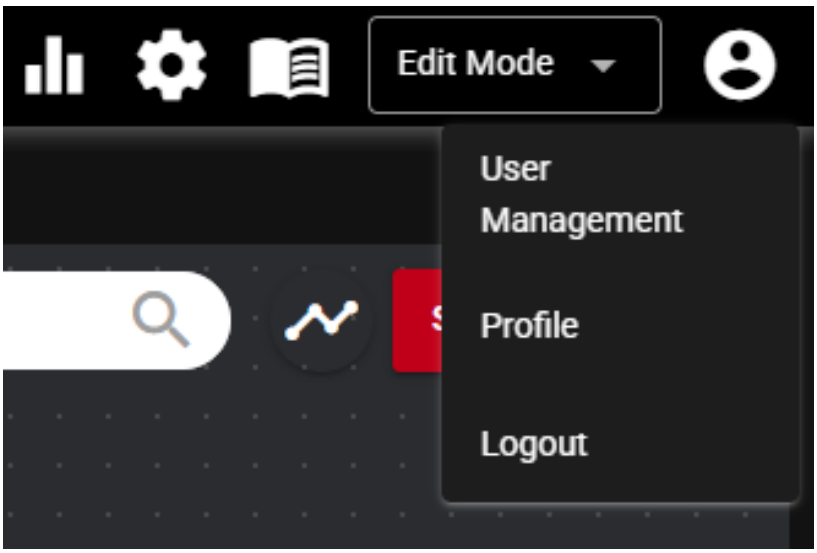
5.3.3.2.1 Add

Users can be added by following the steps below.

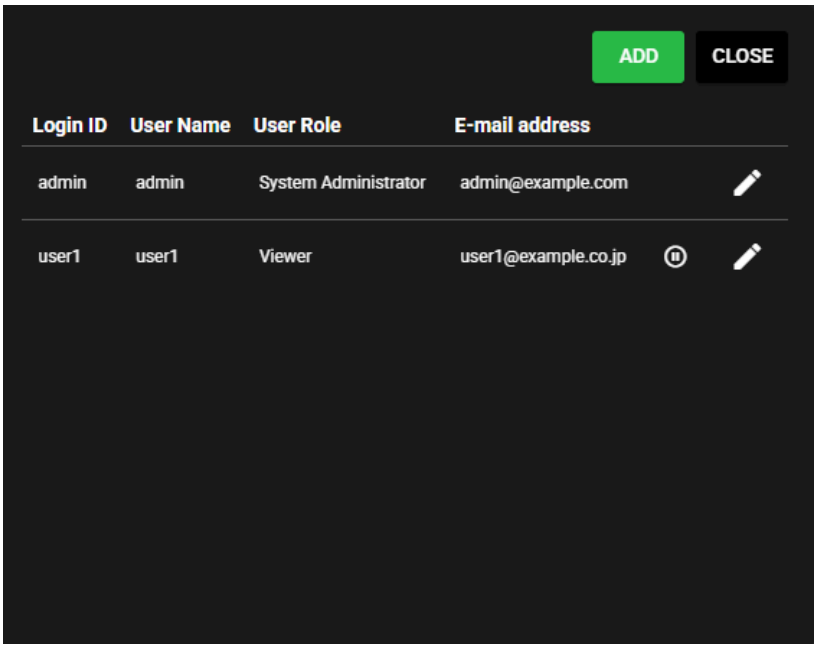
1. Press the User Management icon.



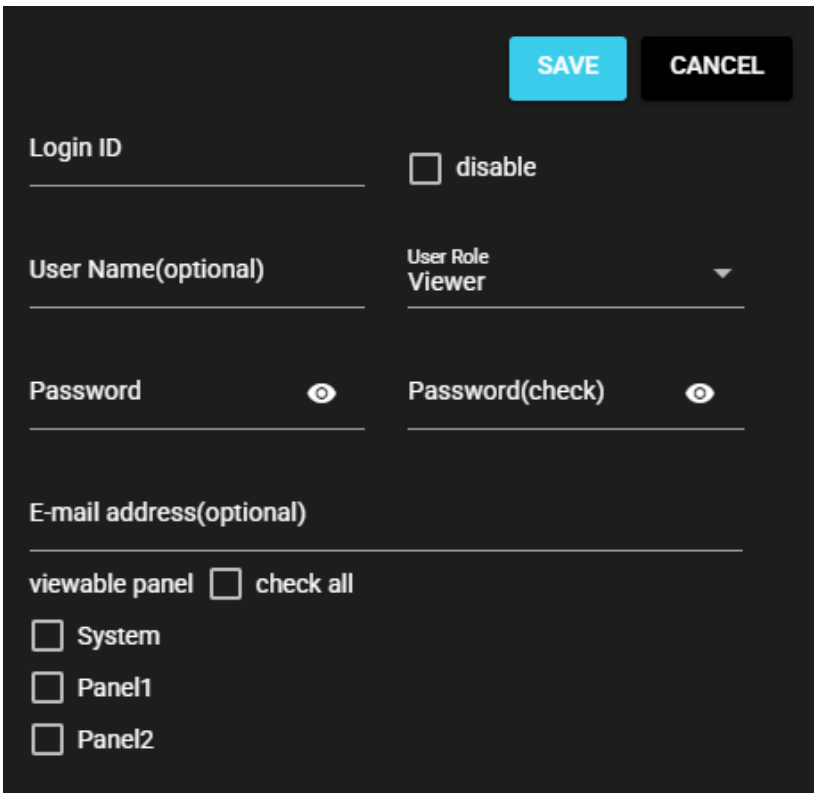
2. Select "User Management".



3. Press "Add".



4. Input the following items and press "Save".



Item	Description
Disable	Checking this disables a user, preventing them from logging in.
Login ID	ID to enter when logging in
Username	Name assigned to a user (optional)
User Role	System administrator: Can perform all operations. This role is granted to the initial user "admin", and at least one administrator is required. Manager: Can perform all operations. System administrators cannot be added, updated, or deleted. Operator: Can view Run Mode and start and stop components. Viewer: Can view Run Mode.
Password	Password to enter when logging in
E-mail address	E-mail address to be linked to the user (Optional)

Item	Description
Viewable panel	Panels that can be viewed by users (This is set only when the user role is Operator or viewer.)

5. The user is added.

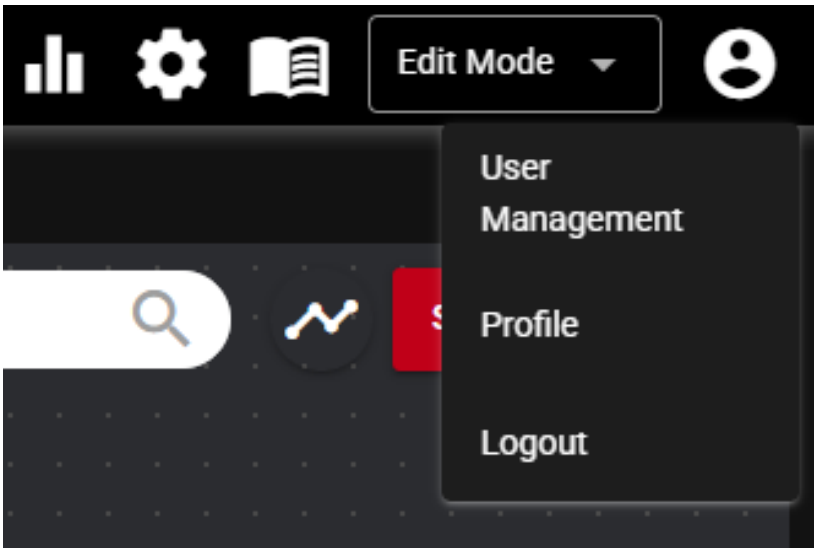
5.3.3.2.2 Edit

The user information can be edited by following the steps below.

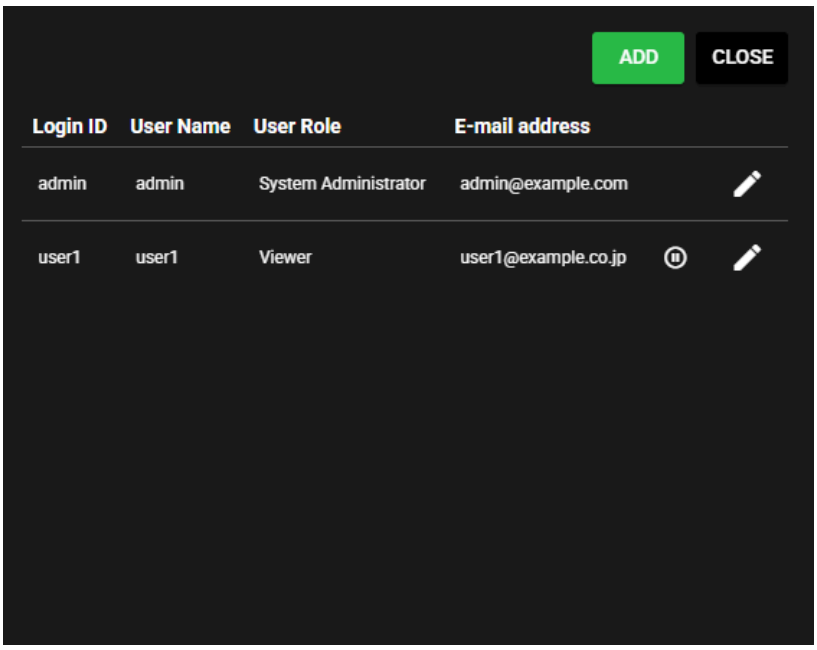
1. Press the User Management icon.



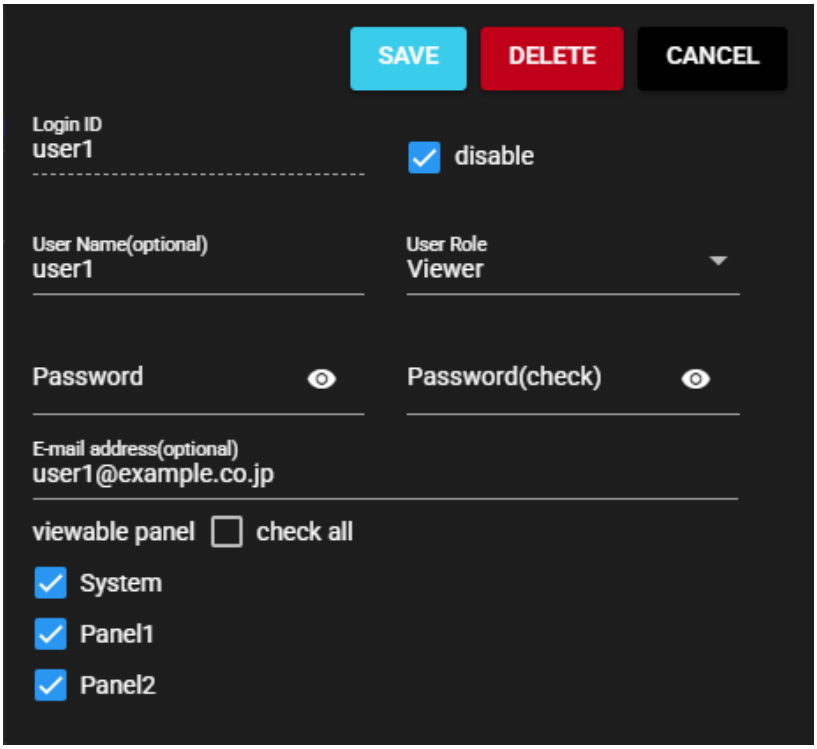
2. Select "User Management".



3. Press the edit icon of the user to edit.



4. Edit the following items and press "Save".



Item	Description
Disable	Checking this disables a user, preventing them from logging in.
Login ID	ID to enter when logging in
Username	Name assigned to a user (optional)

Item	Description
User Role	System administrator: Can perform all operations. This role is granted to the initial user "admin", and at least one administrator is required. Manager: Can perform all operations. System administrators cannot be added, updated, or deleted. Operator: Can view Run Mode and start and stop components. Viewer: Can view Run Mode.
Password	Password to enter when logging in
E-mail address	E-mail address to be linked to the user (Optional)
Viewable panel	Panels that can be viewed by users (This is set only when the user role is Operator or viewer.)

5. The edited content is saved.

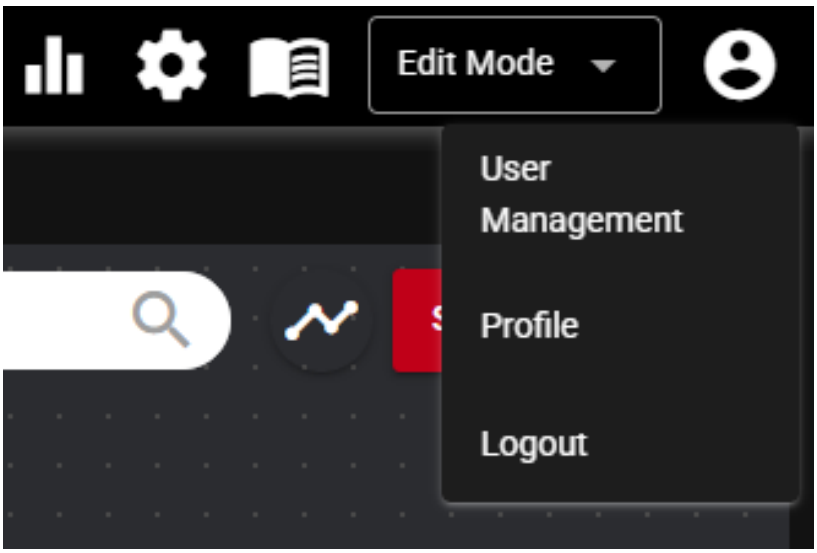
5.3.3.2.3 Delete

Users can be deleted by following the steps below.

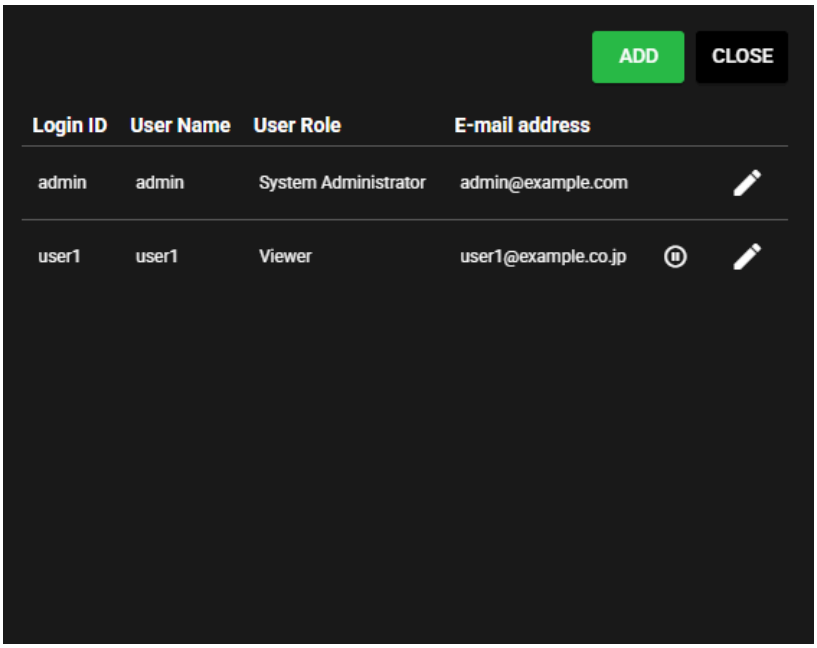
1. Press the User Management icon.



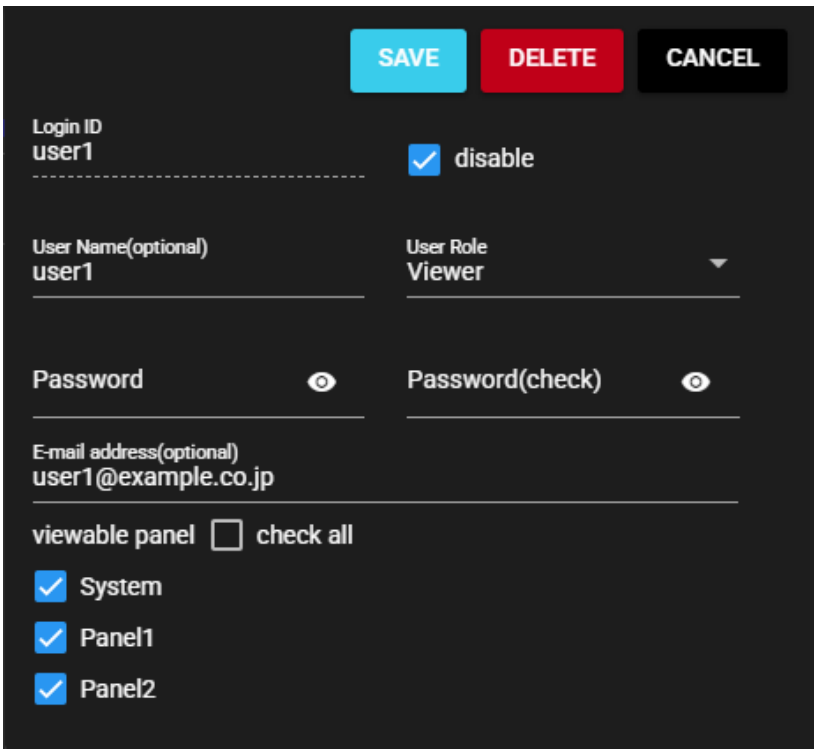
2. Select "User Management".



3. Press the edit icon of the user to delete.



4. Press "Delete".



5. The user is deleted.

> **About the System Administrator**
>
> At least one system administrator is required. If there is only one system administrator, it cannot be deleted.

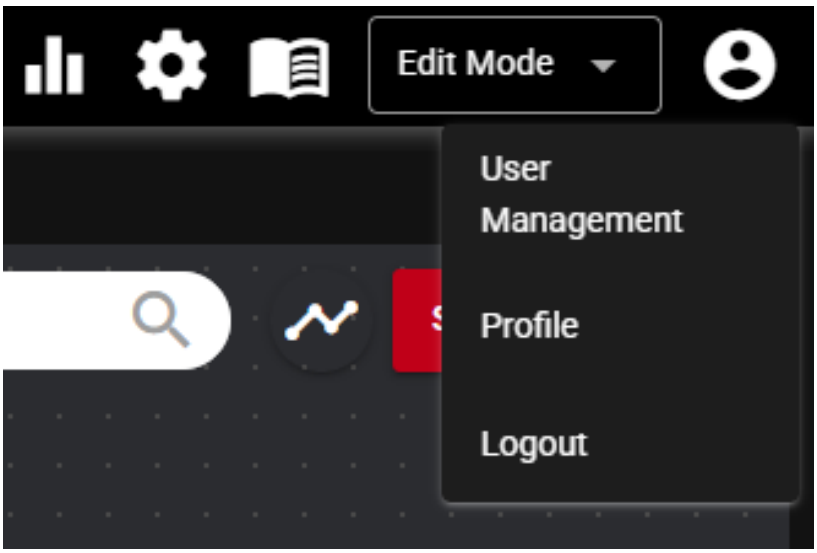
5.3.3.2.4 Profile

The information of a user that is currently logged in can be checked by following the step below.

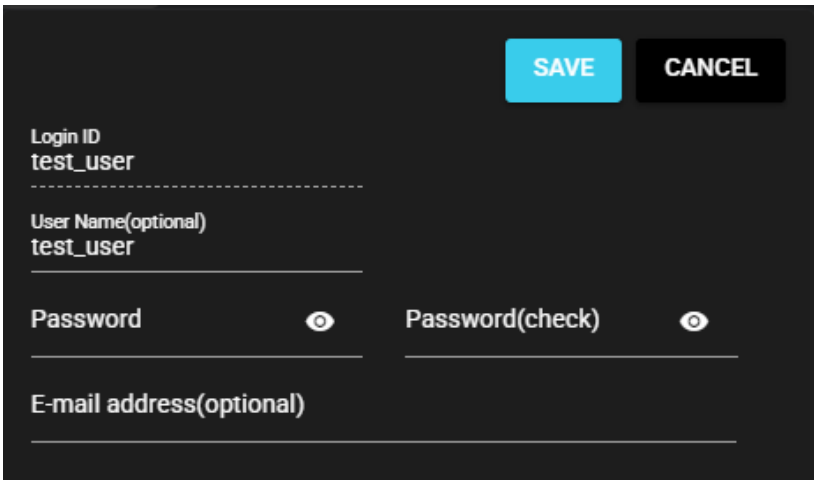
1. Press the User Management icon.



2. Select "Profile".

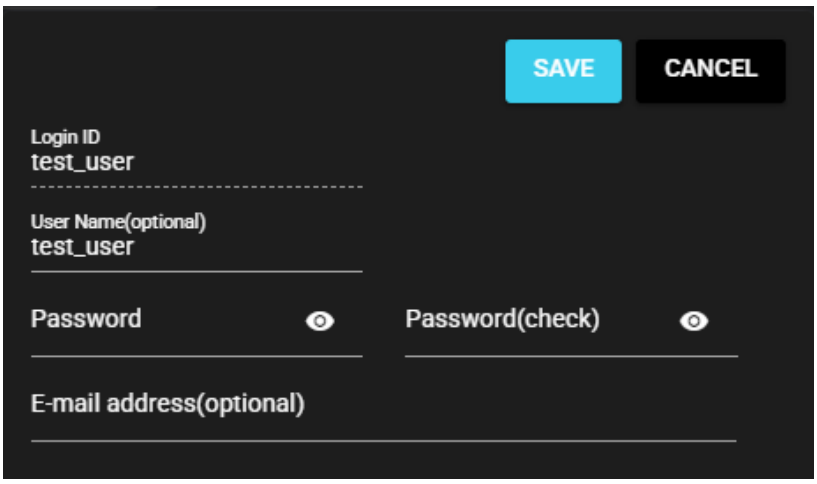


3. The following items are displayed for the user who is currently logged in.



Item	Description
Login ID	ID to enter when logging in
Username	Name assigned to a user (optional)
Password	Password to enter when logging in
E-mail address	E-mail address to be linked to the user (Optional)

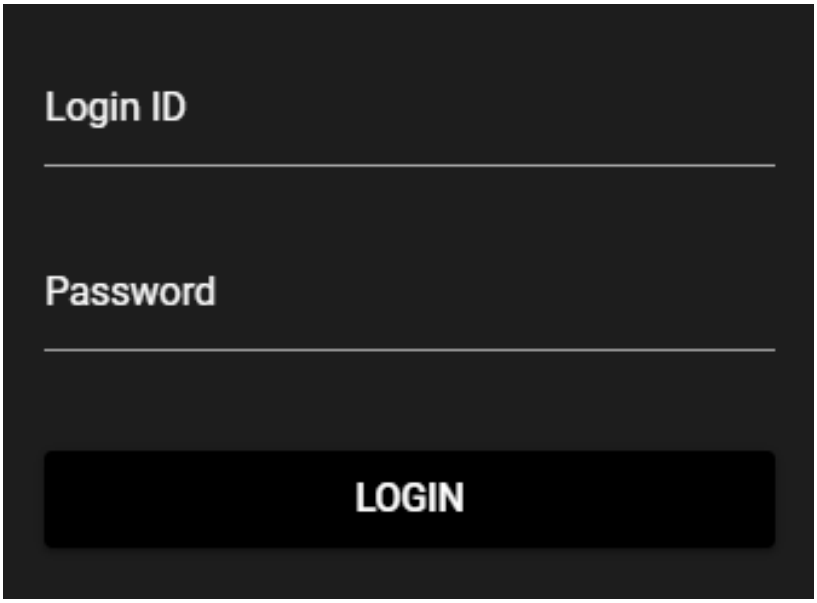
4. After editing the items and clicking "Save", the edited version is saved.



5.3.3.2.5 Login

Users can log in to SpeeDBee Synapse by following the steps below.

1. Go to "http://{ IP address of the environment where SpeeDBee Synapse is running }:8120".
2. Input the user's "login ID" and "password" and press "login".



3. Log the user in to SpeeDBee Synapse.

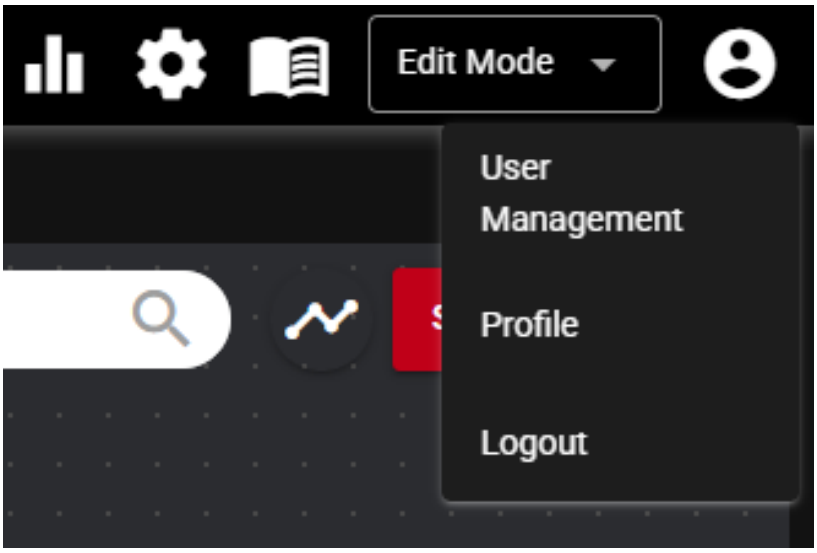
5.3.3.2.6 Logout

Users can log out of SpeeDBee Synapse by following the steps below.

1. Press the User Management icon.



2. Select "Logout".



3. This logs the user out of SpeeDBee Synapse.

5.3.3.3 Panel

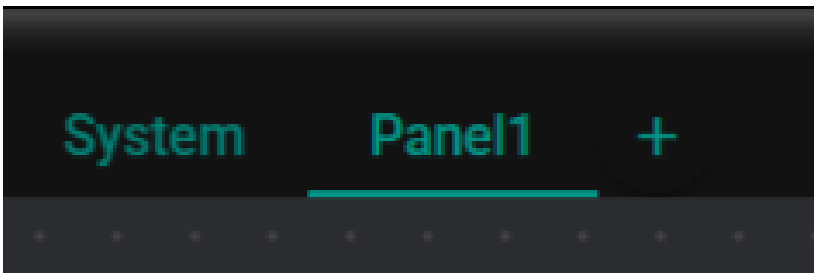
This section describes panel operations.

> **About panel browsing permissions**
>
> Users who are operators or viewers can set browsing user roles for individual panels through the User Management configurations. Using multiple panels, different flows from data collection to output and integration can be created, and only selected panels can be displayed to operators and viewers.

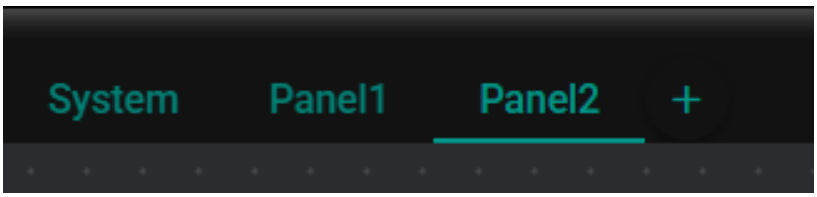
5.3.3.3.1 Add

Panels can be added by following the steps below.

1. Press the + icon on the tab menu of the panel.



2. The panel is added.

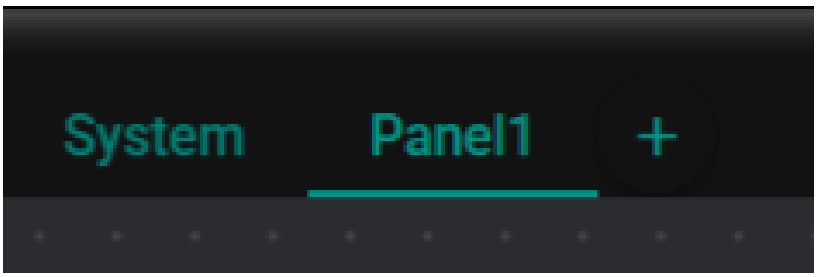


This operation can be performed only in Edit Mode.

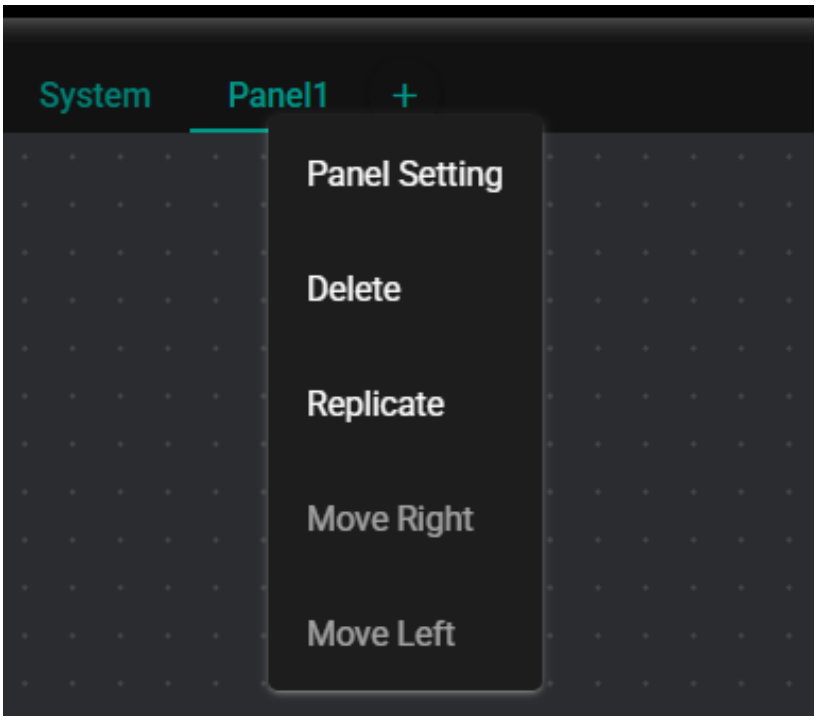
5.3.3.3.2 Changing a name

Panel names can be changed by following the steps below.

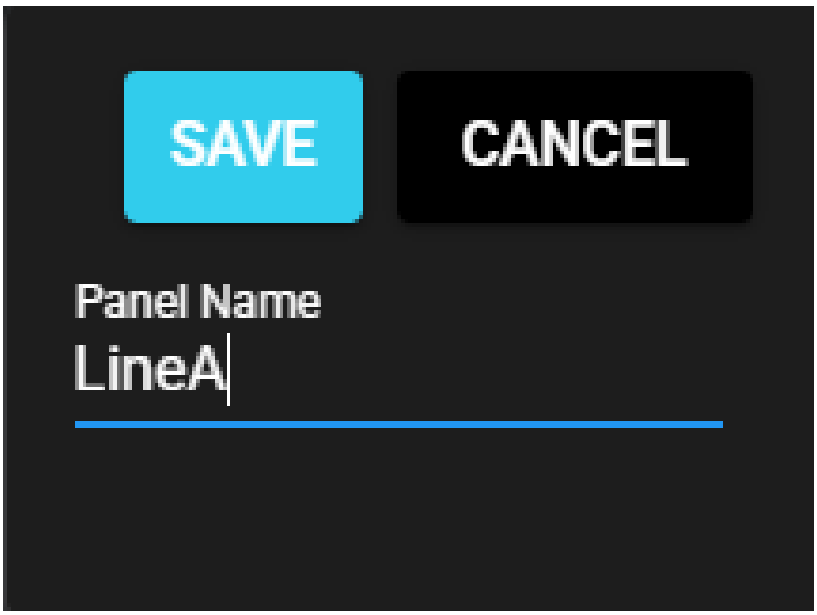
1. Right-click a panel tab.



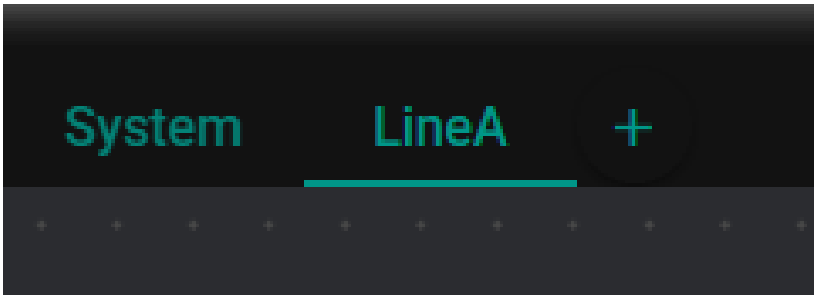
2. Select "Panel Settings".



3. Input the panel name and press "Save".



4. The panel name changes to the entered input.



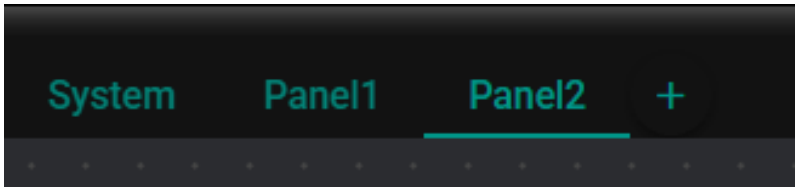
This operation can be performed only in Edit Mode.

5.3.3.3.3 Rearranging

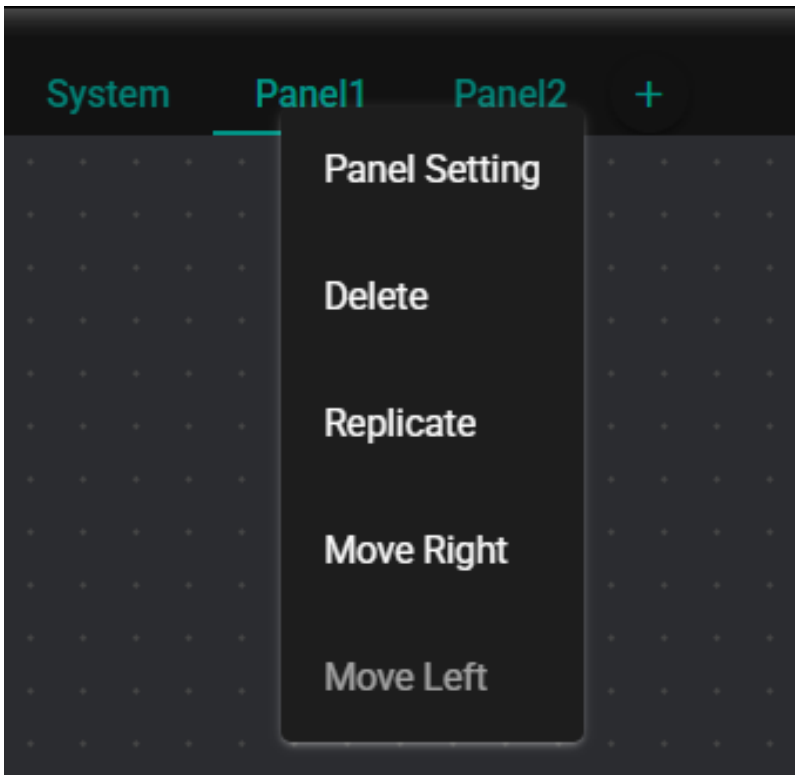
Panels can be rearranged by following either of the steps below.

- Rearranging from the panel tab context menu

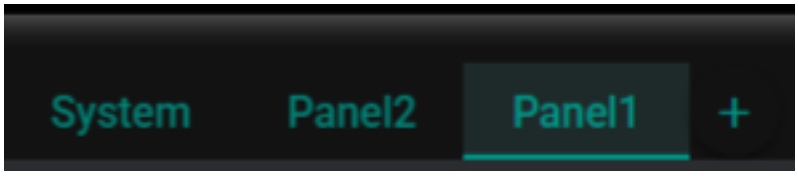
  a. Right-click a panel tab.



  b. Select "Move Right" or "Move Left".



c. The panels are rearranged.

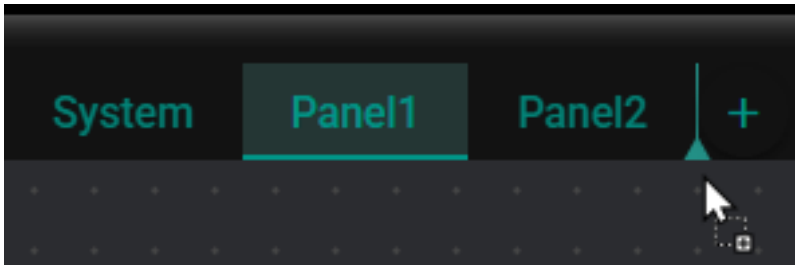


- Rearranging by dragging and dropping the panel tab

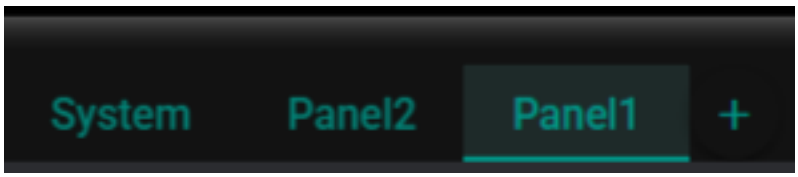
a. Drag a panel tab.



b. Drop it at the place where you want to move the panel.



c. The panels are rearranged.

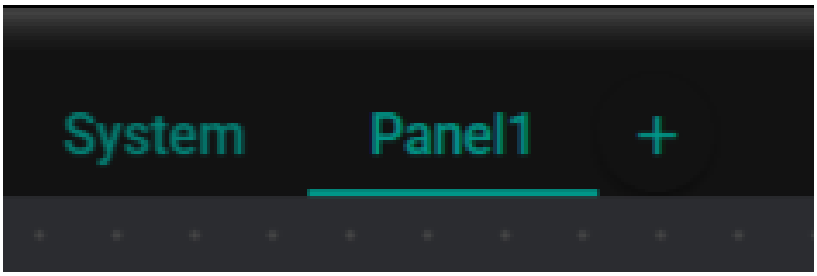


This operation can be performed only in Edit Mode.

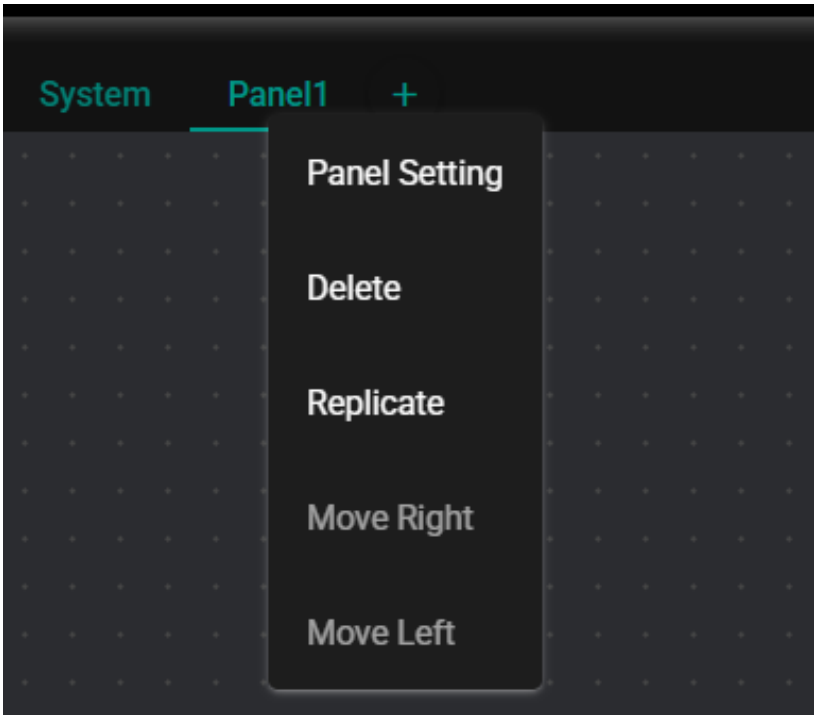
5.3.3.3.4 Delete

Panels can be deleted by following the steps below.

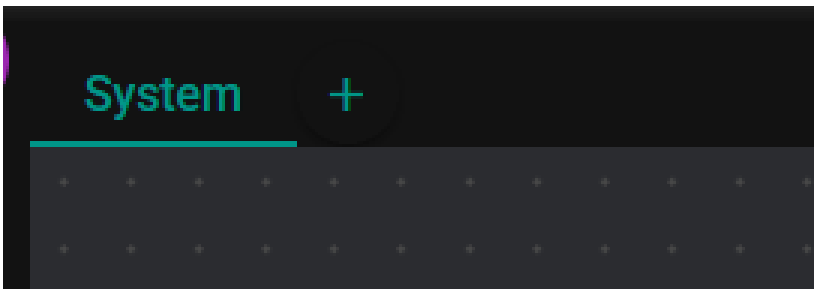
1. Right-click a panel tab.



2. Select "Delete".



3. The panel is deleted.

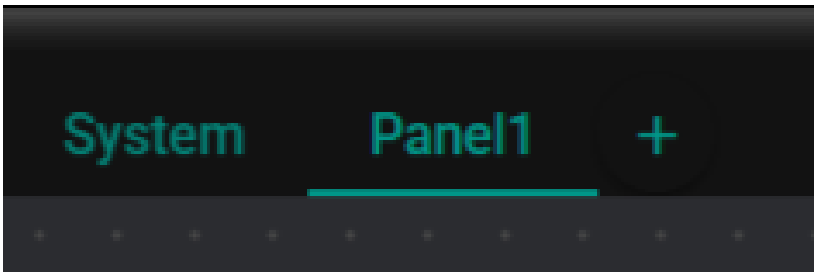


This operation can be performed only in Edit Mode.

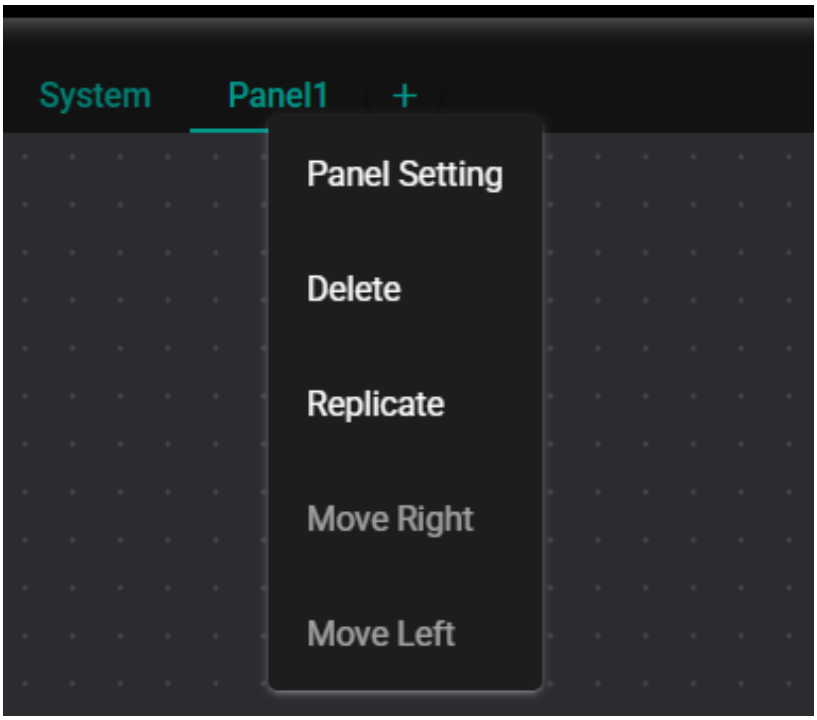
5.3.3.3.5 Replicate

Panels of the same configurations can be replicated by following the steps below.

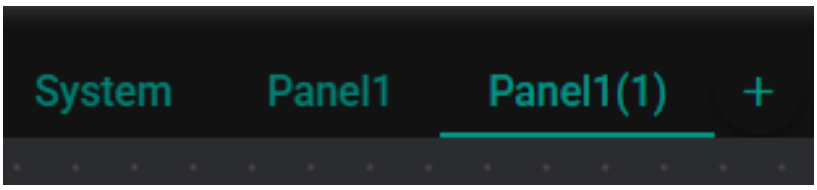
1. Right-click a panel tab.



2. Select "Replicate".



3. A panel with the same configurations of the original is replicated.

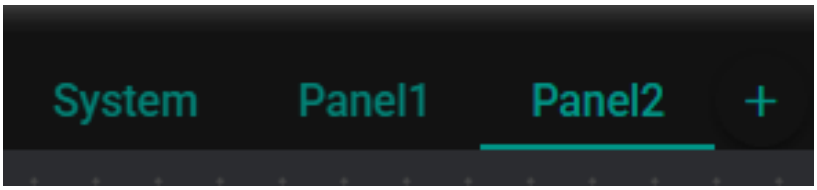


This operation can be performed only in Edit Mode.

5.3.3.3.6 Display

Panels can be displayed by following the steps below.

1. Click the tab of the panel to display.

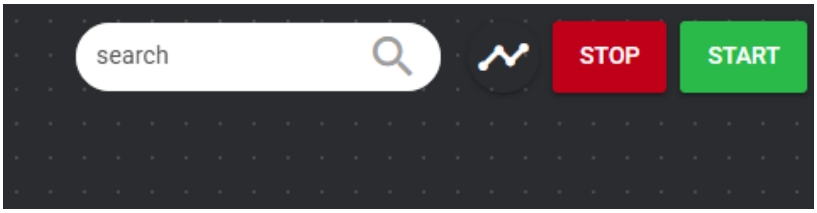


2. The panel whose tab is clicked is displayed.

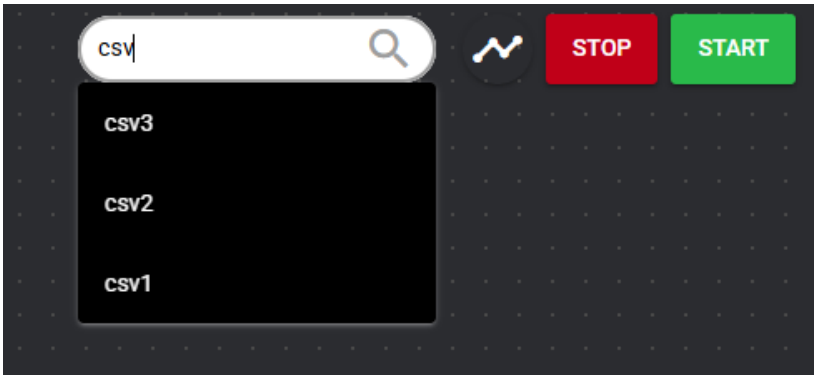
5.3.3.3.7 Component Search

The location of components registered in the panel can be searched by following the steps below.

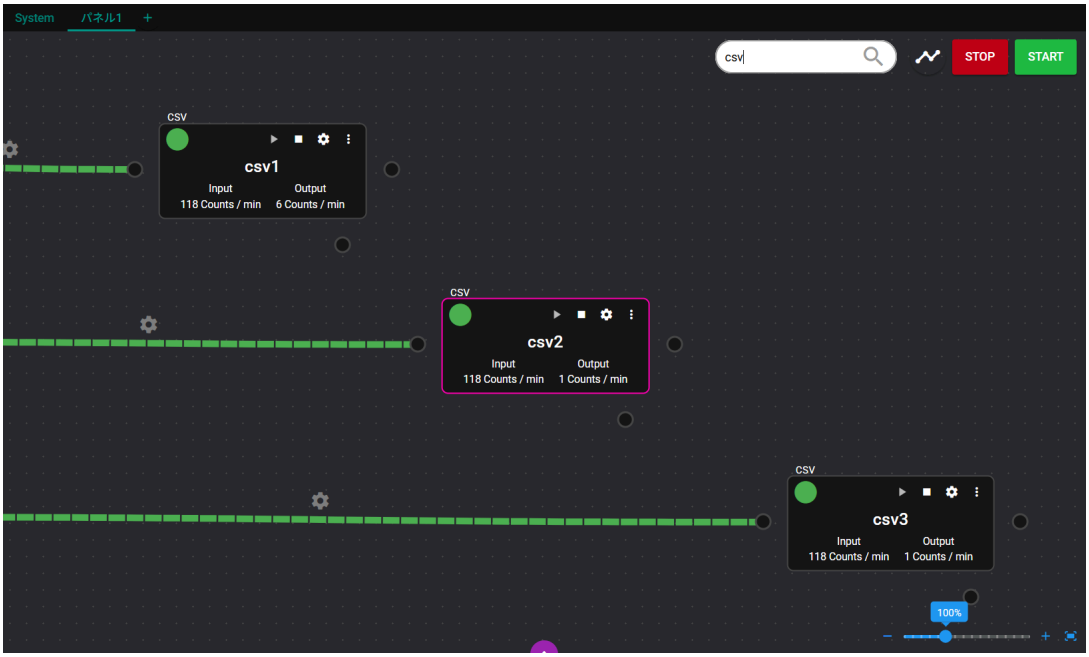
1. Input the component name in the search box at the top right of the panel.



2. Select from the suggestions.



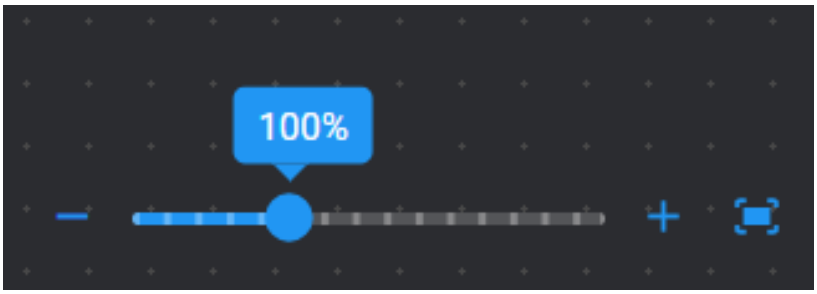
3. The selected component is displayed in the center of the panel.



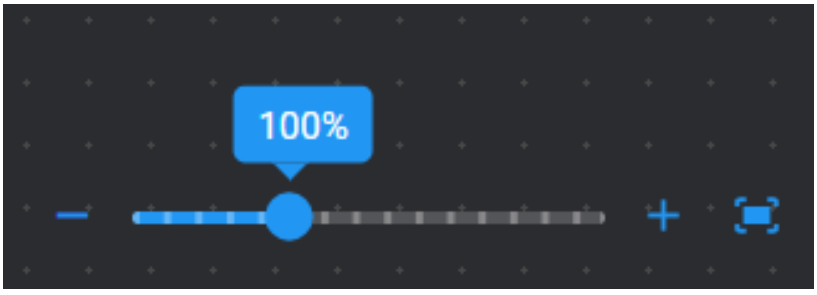
5.3.3.3.8 Adjusting the display magnification

The display magnification of the panel can be adjusted by following the steps below.

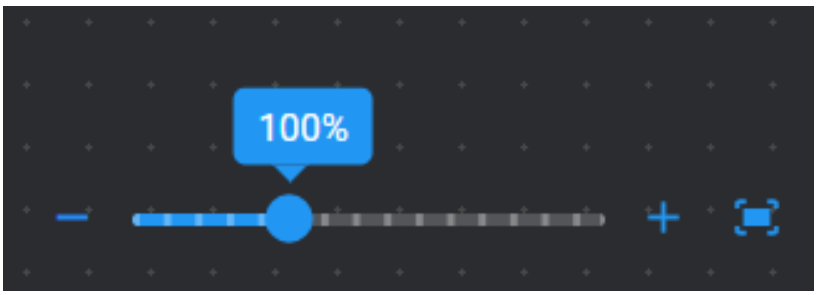
1. The current magnification of the panel is displayed at the bottom right of the panel.



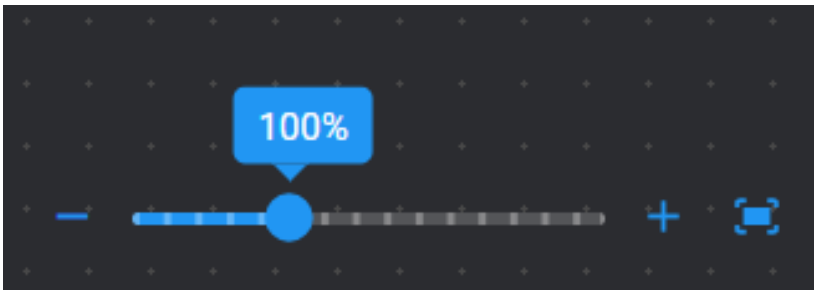
2. Move the slider in the + direction to enlarge the display to fullscreen, and move it in the – direction to reduce the display.



3. Pressing + enlarges the display magnification by 5%, and pressing - reduces the display magnification by 5%.



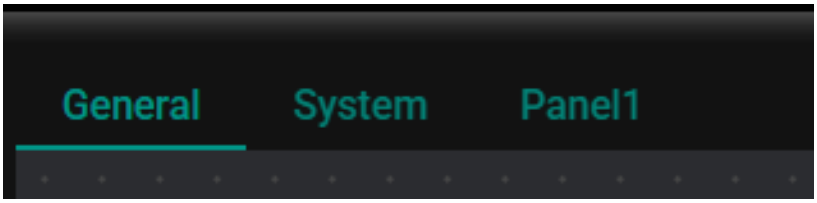
4. Pressing the Overall Display button shows the entire component arranged on the panel on the screen.



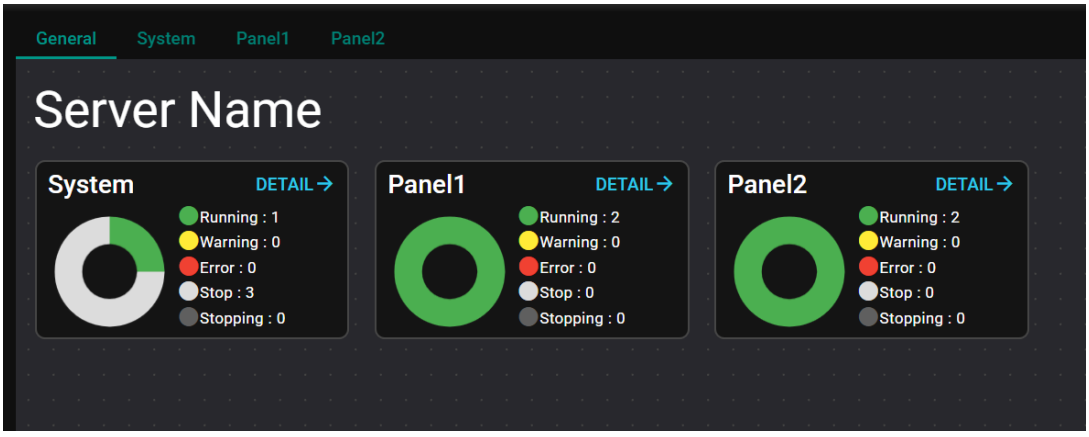
5.3.3.3.9 Overall Panel

The entire panel can be displayed by following the steps below. The entire panel shows the information of individual panels.

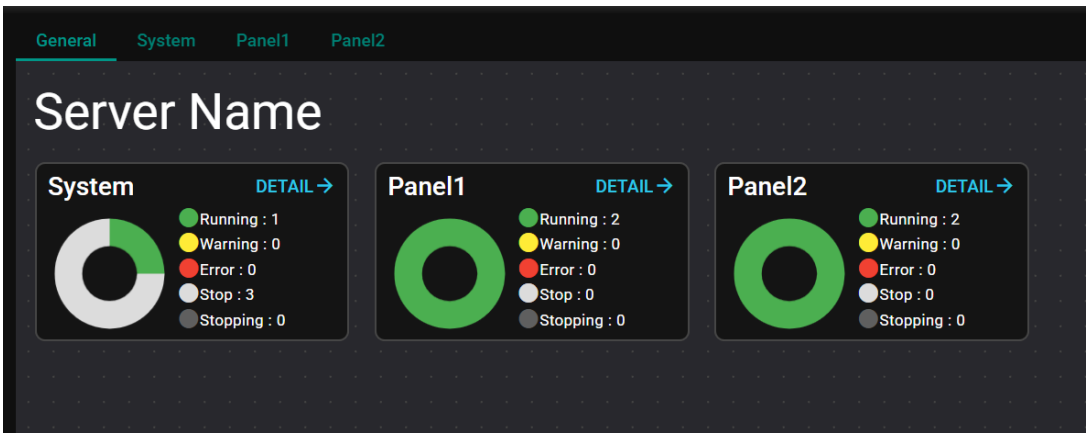
1. Select "General" in the panel tab.



2. This displays the server instance name, panel list, and operation status of individual panel component.



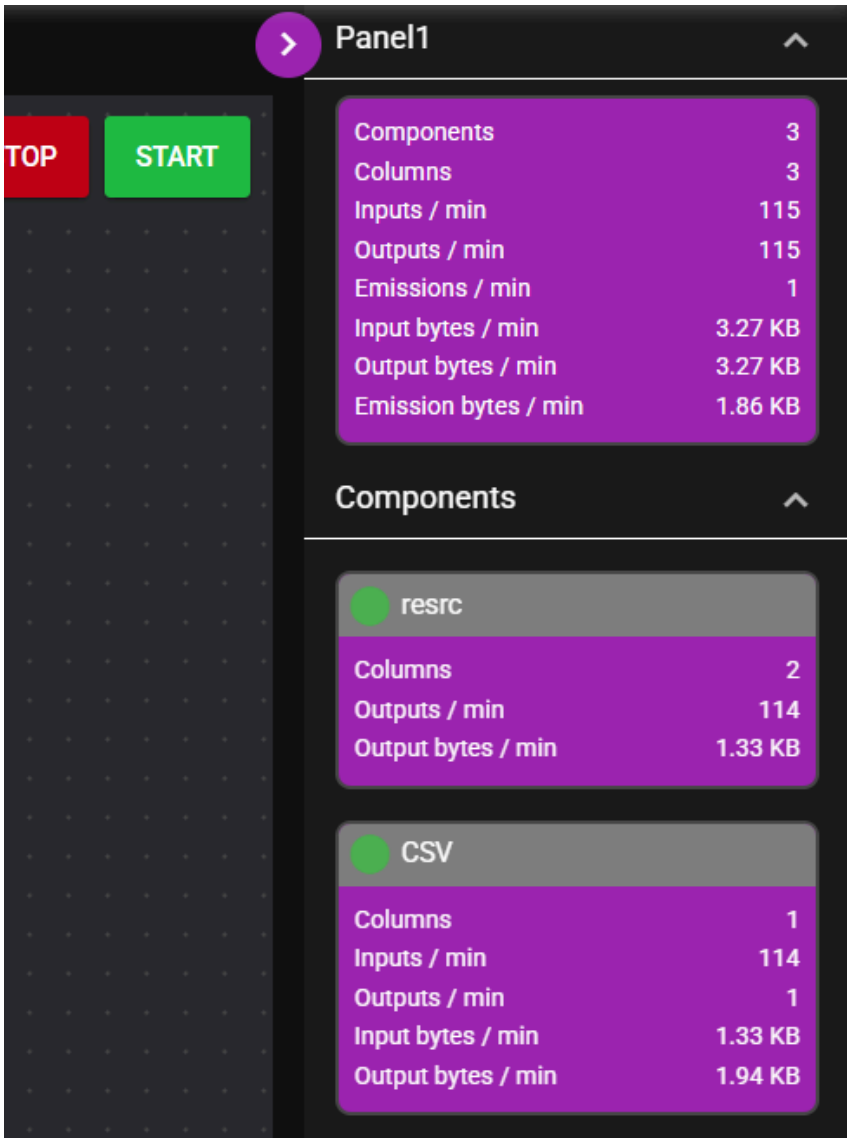
3. Press "DETAIL" to go to each panel.



This operation can be performed only in Run Mode.

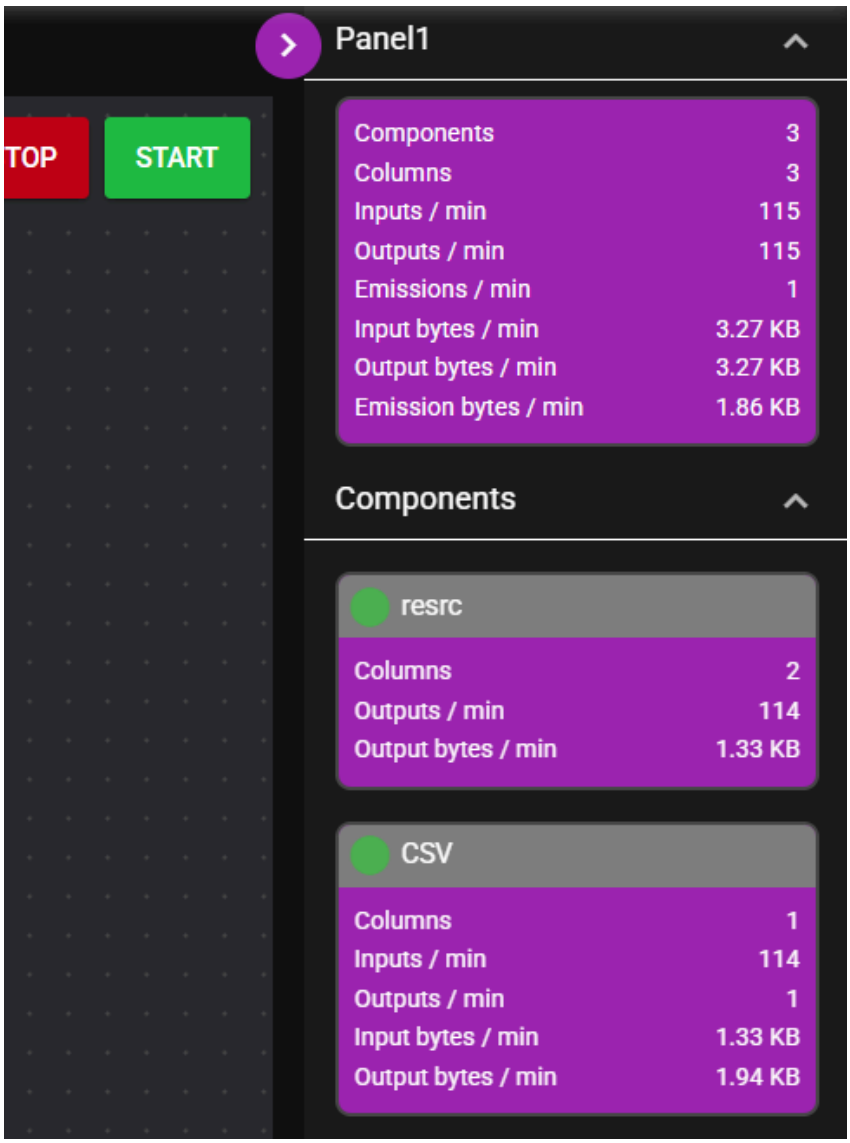
5.3.3.3.10 Panel Status

For each panel, the following information is displayed.



Item	Description
Number of components	Number of all components on the panel
Number of columns	Number of all component columns on the panel
Inputs / min	The total number of data inputs per minute for all components on the panel
Outputs / min	The total number of data outputs per minute for all components on the panel
Receives / min	The total number of data received per minute for all components on the panel
Emissions / min	The total number of data emitted per minute for all components on the panel
Input bytes / min	The total number of input bytes per minute for all components on the panel
Output bytes / min	The total number of output bytes per minute for all components on the panel
Receive bytes / min	The total number of received bytes per minute for all components on the panel
Emission bytes / min	The total number of emission bytes per minute for all components on the panel

The following information is displayed for operating components on the panel.



Item	Description
Number of columns	Number of data items that component outputs
Inputs / min	Number of data inputs per minute
Outputs / min	Number of data outputs per minute
Receives / min	Number of data received per minute
Emissions / min	Number of data emitted per minute
Input bytes / min	Number of data inputs bytes per minute
Output bytes / min	Number of data output bytes per minute

Item	Description
Receive bytes / min	Number of data reception bytes per minute
Emission bytes / min	Number of data emission bytes per minute

This operation can be performed only in Run Mode.

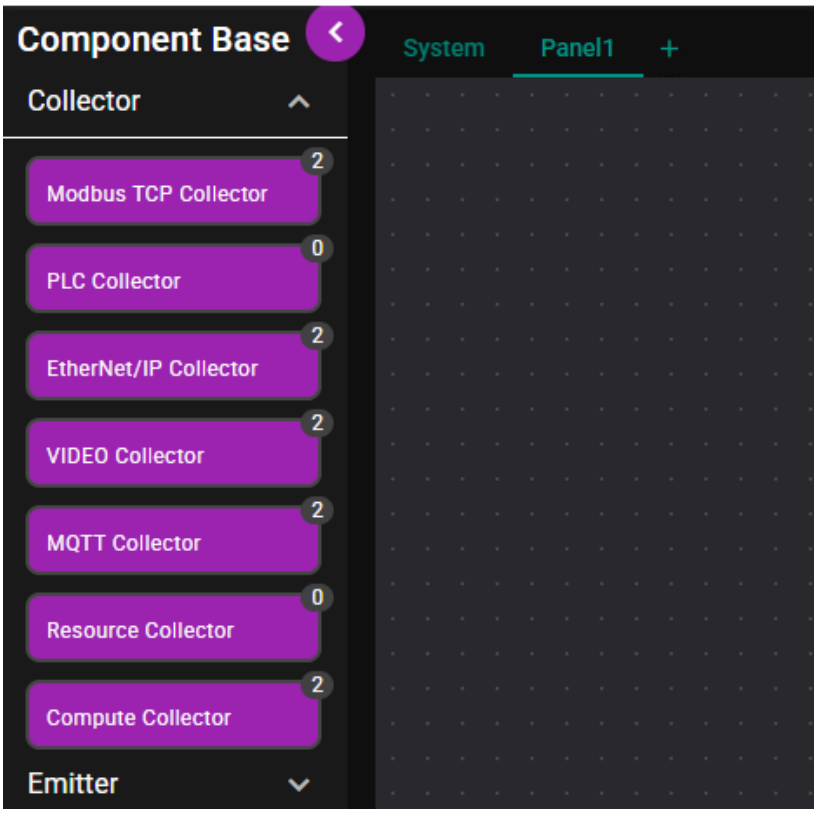
5.3.3.4 Component

This section describes component operations.

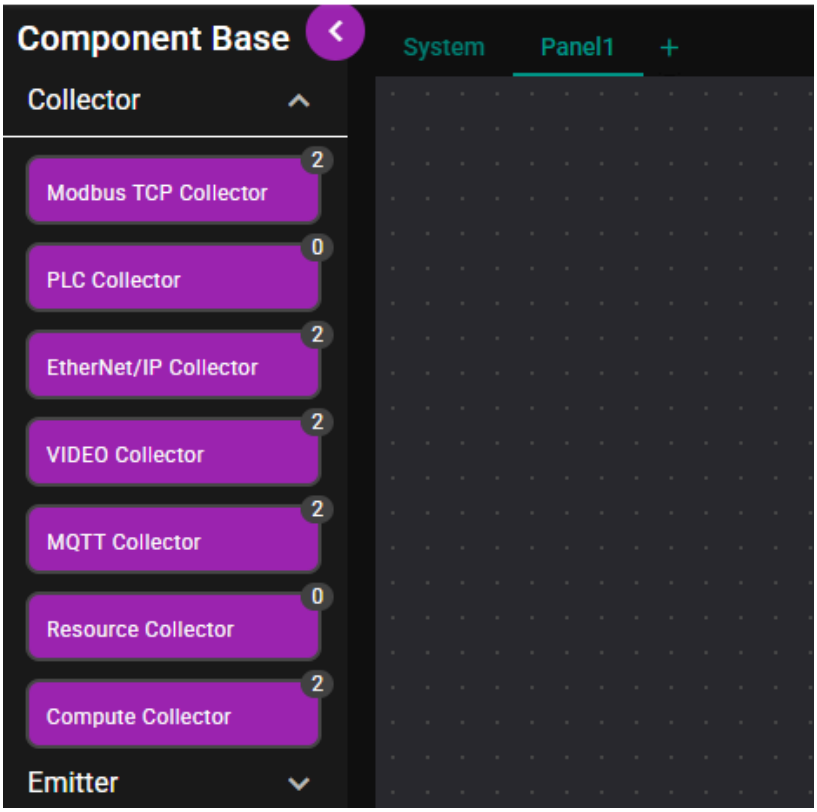
5.3.3.4.1 Add

Components can be added by following the steps below.

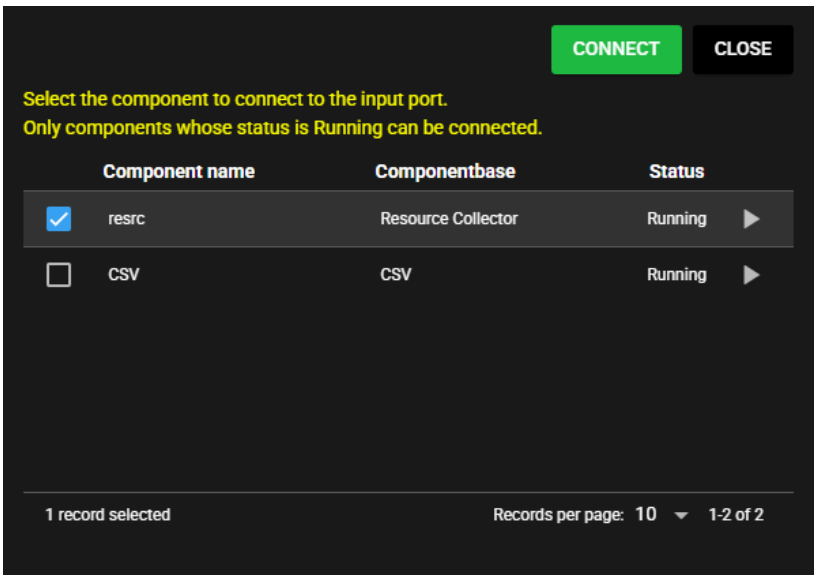
1. Select a component from the left menu and drag the icon.



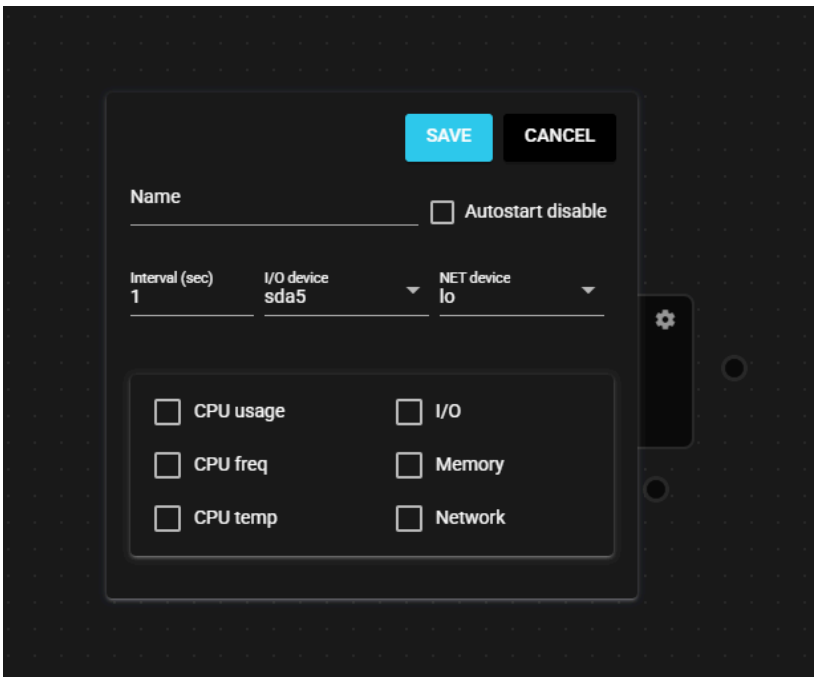
2. Drop it on top of the panel.



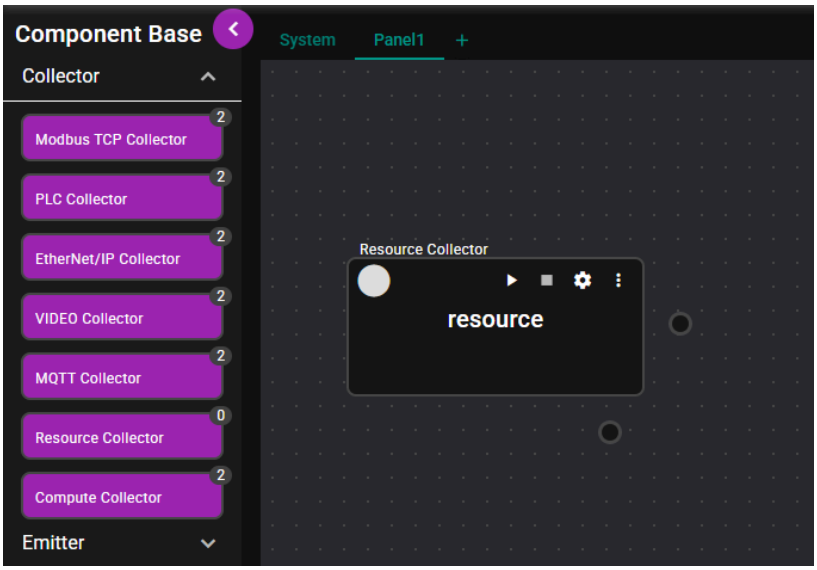
3. For some components, the dialog is displayed to select a component to connect to the IN Port. Select the component to connect and press "Select".



4. A configuration dialog opens. Input configuration items there and press "Save".



5. The component is added to the panel.

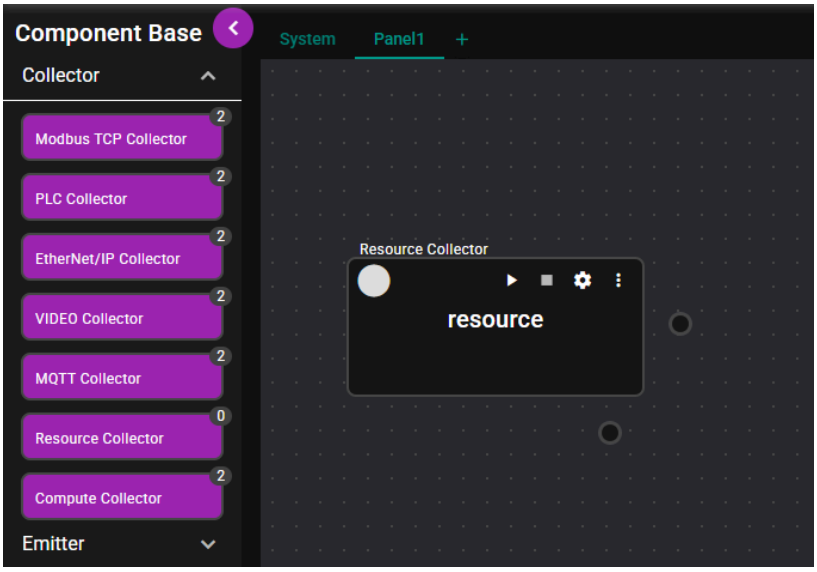


This operation can be performed only in Edit Mode.

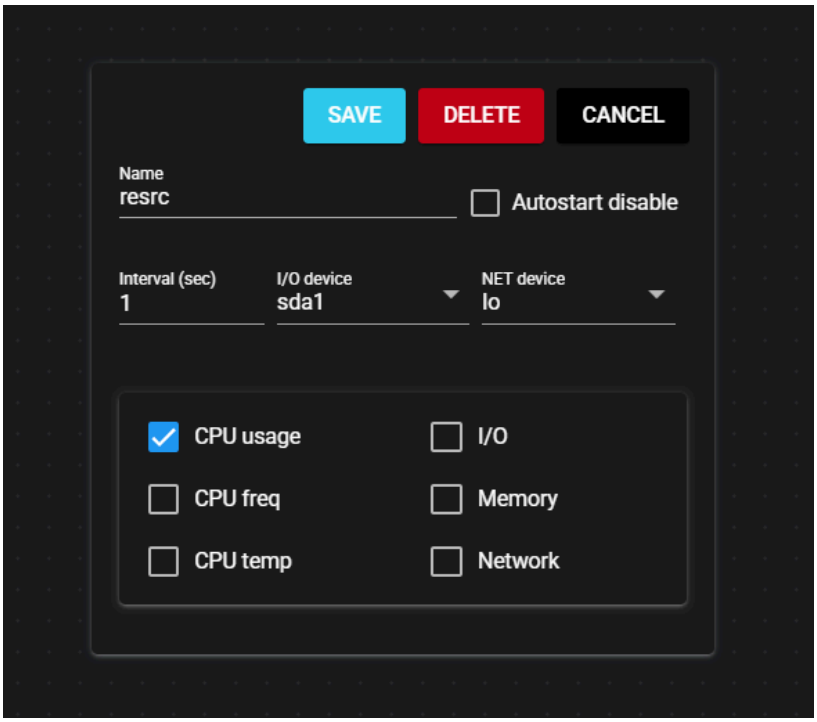
5.3.3.4.2 Edit

Component configurations can be edited by following the steps below.

1. Press the Configuration icon of the component.



2. A configuration dialog opens. Edit configuration items there and press "Save".



3. The edited version is saved.

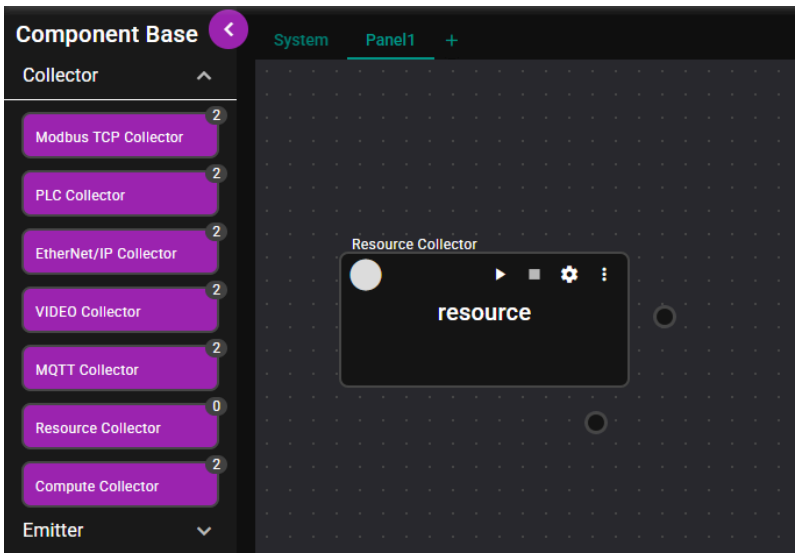
This operation can be performed only in Edit Mode.

5.3.3.4.3 Delete

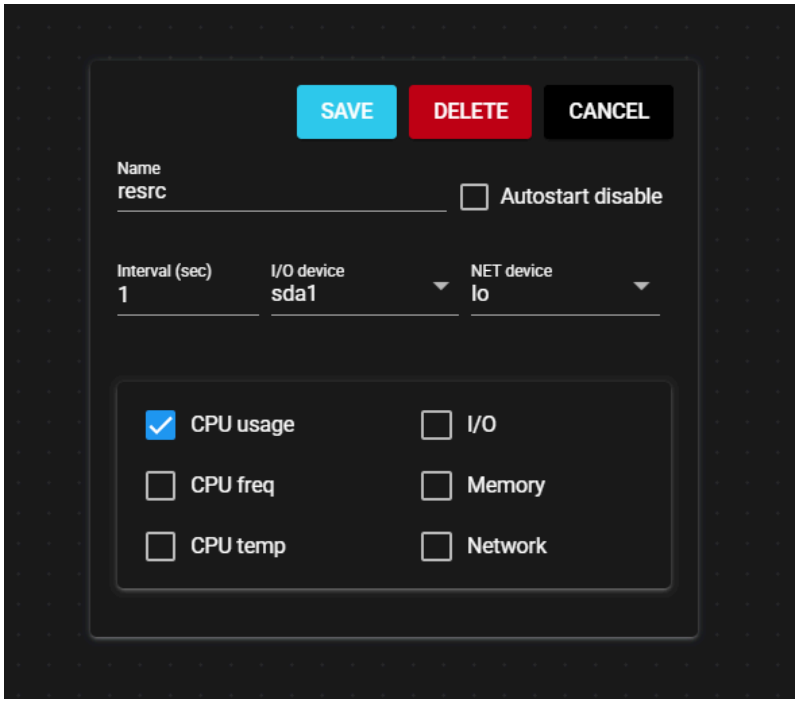
Components can be deleted by following the steps below.

- This is how to delete a component from the configuration icon of the component.

  a. Press the Configuration icon of the component.



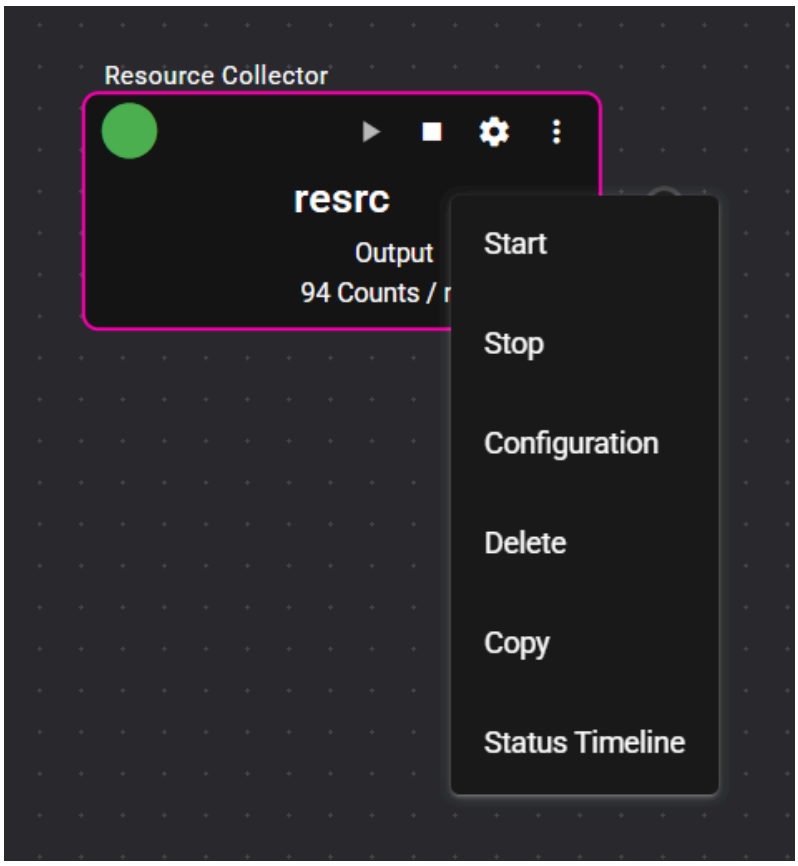
b. The configuration dialog opens. Press "Delete".



c. The component is deleted.

- This is how to delete a component using its context menu of the component.

  a. Right-click on the component and select "Delete".



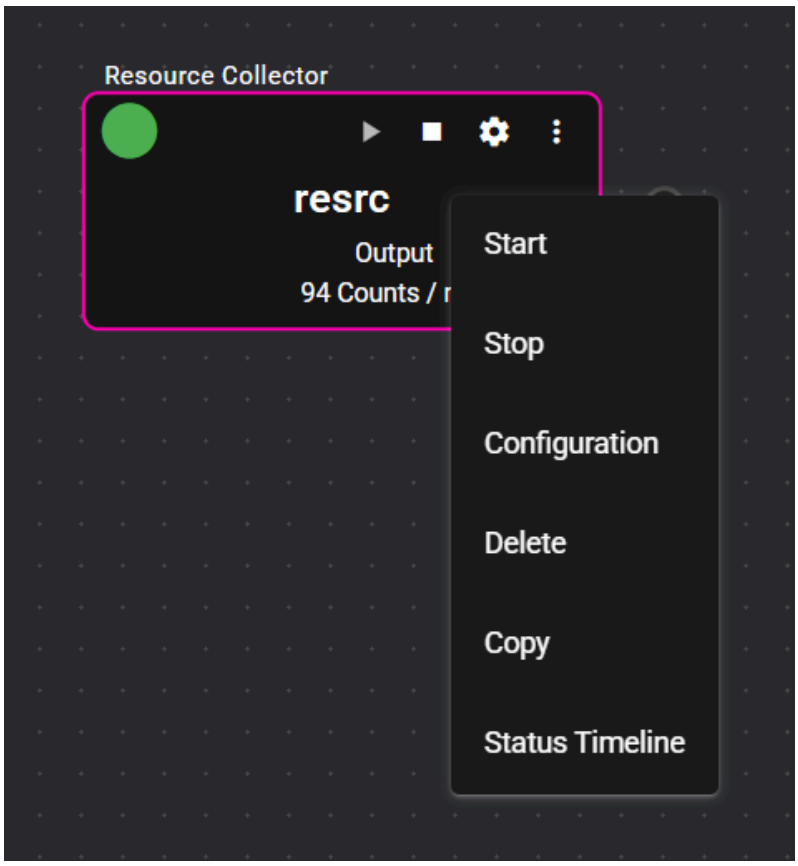
b. The component is deleted.

> This operation can be performed only in Edit Mode.

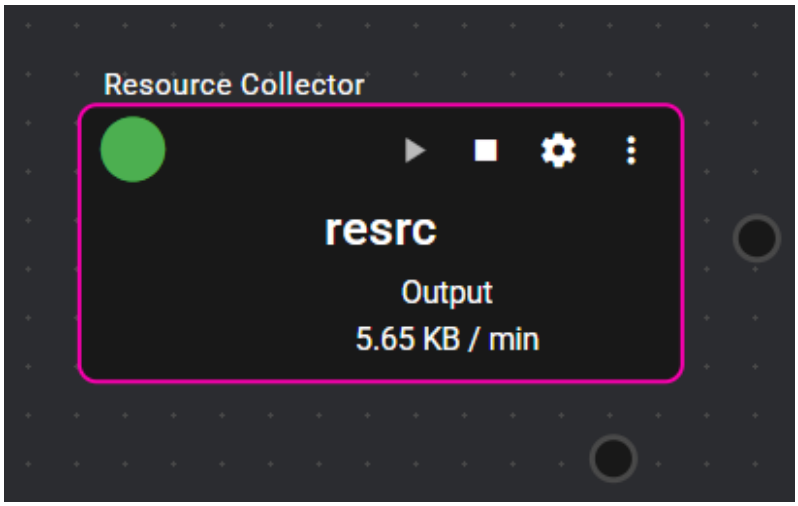
5.3.3.4.4 Copy & Paste

Components of the same configurations can be replicated by following the steps below.

1. Copy the component using one of the following ways.
    - Right-click on the component and select "Copy".

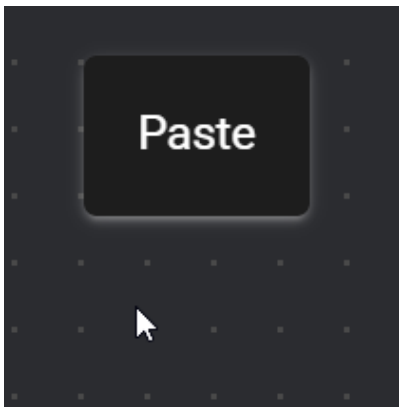


- While clicking to select the component, press the "C" key while holding down the "Ctrl" key on the keyboard.



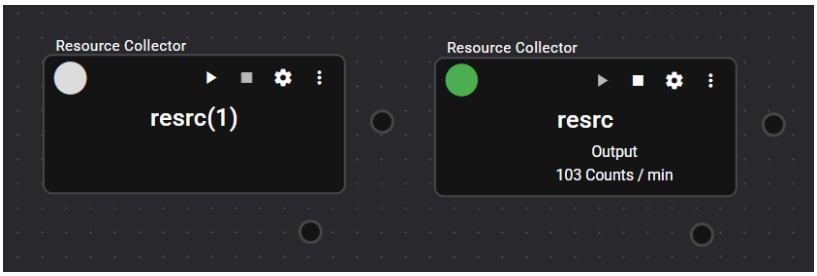
2. Paste the component in one of the following ways.

- Right-click the panel and select "Paste".



- Press the "V" key while holding down the "Ctrl" key on the keyboard.

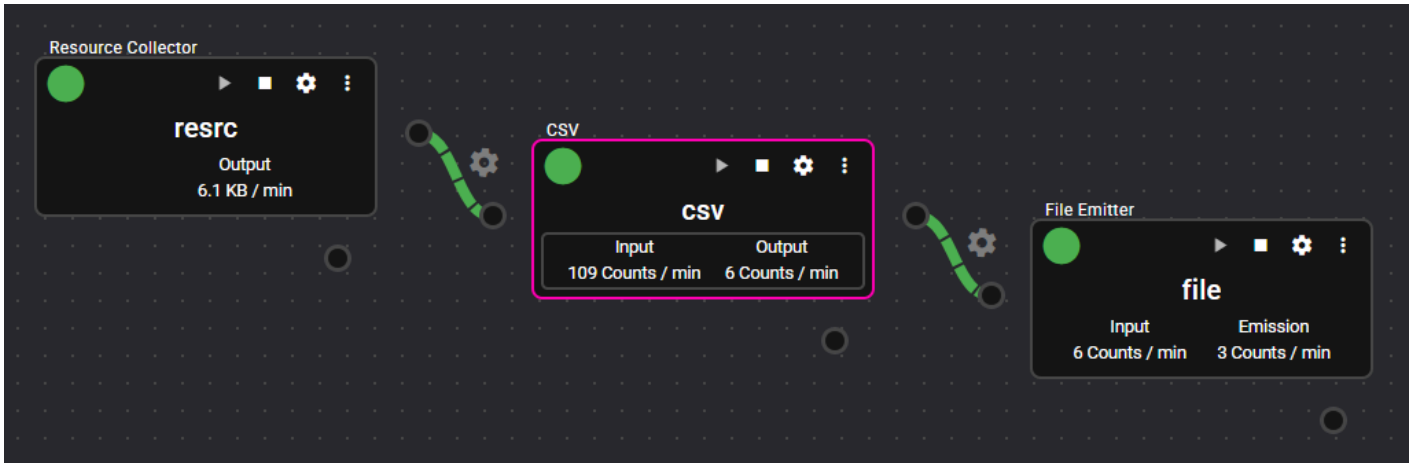
3. A component with the same configurations as the copied one is created.



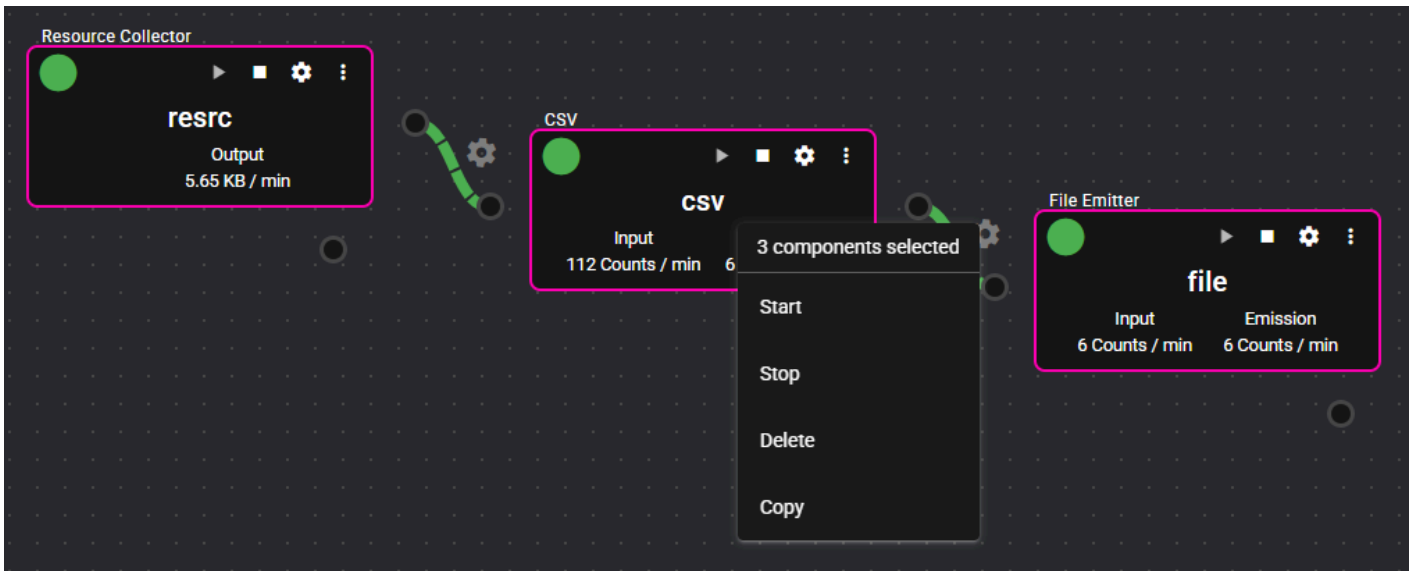
This operation can be performed only in Edit Mode.

5.3.3.4.5 Multiple Select

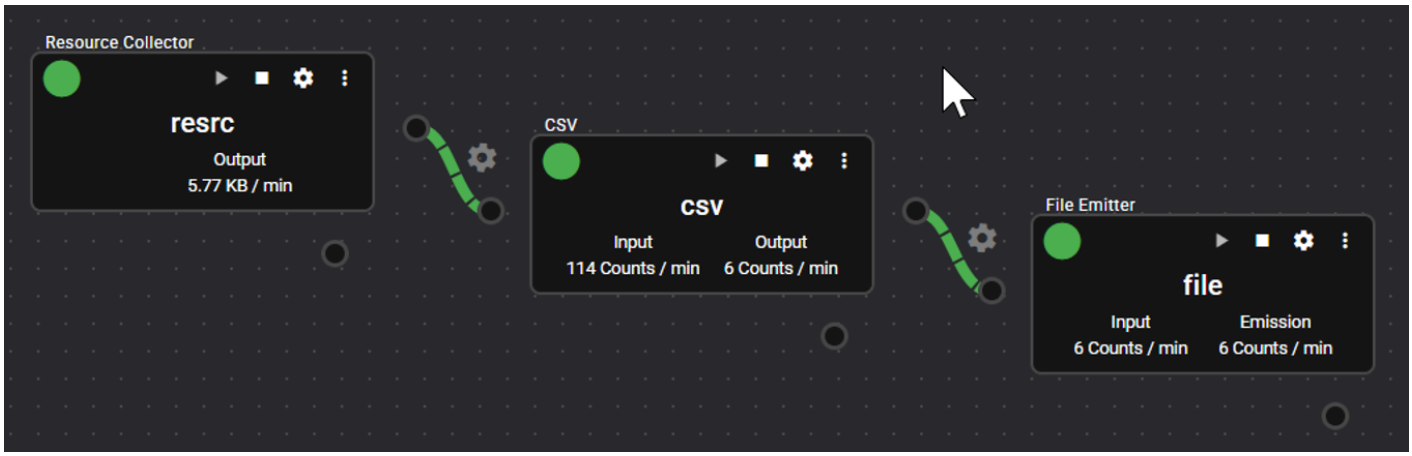
Multiple components can be selected by clicking each component while holding down the "Ctrl" key on the keyboard.



After selecting multiple components, components selected from the context menu or through keyboard operation can be Copied & Pasted or Deleted.



To deselect multiple components, click anywhere in the panel except the component.

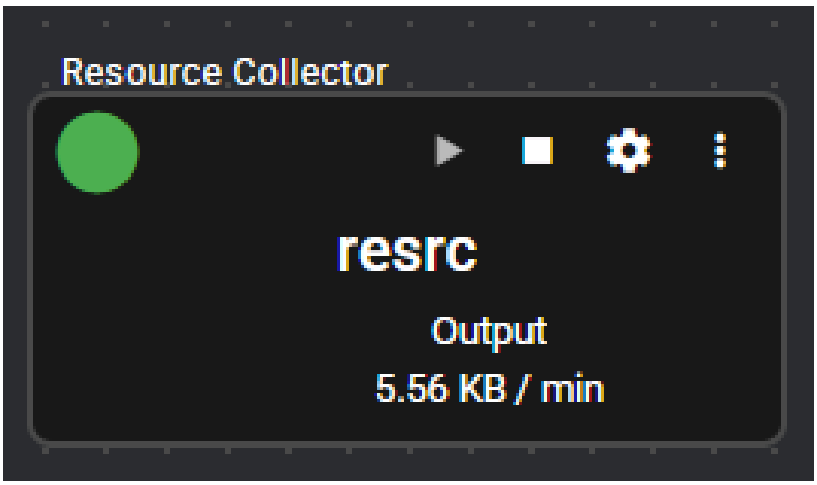


This operation can be performed only in Edit Mode.

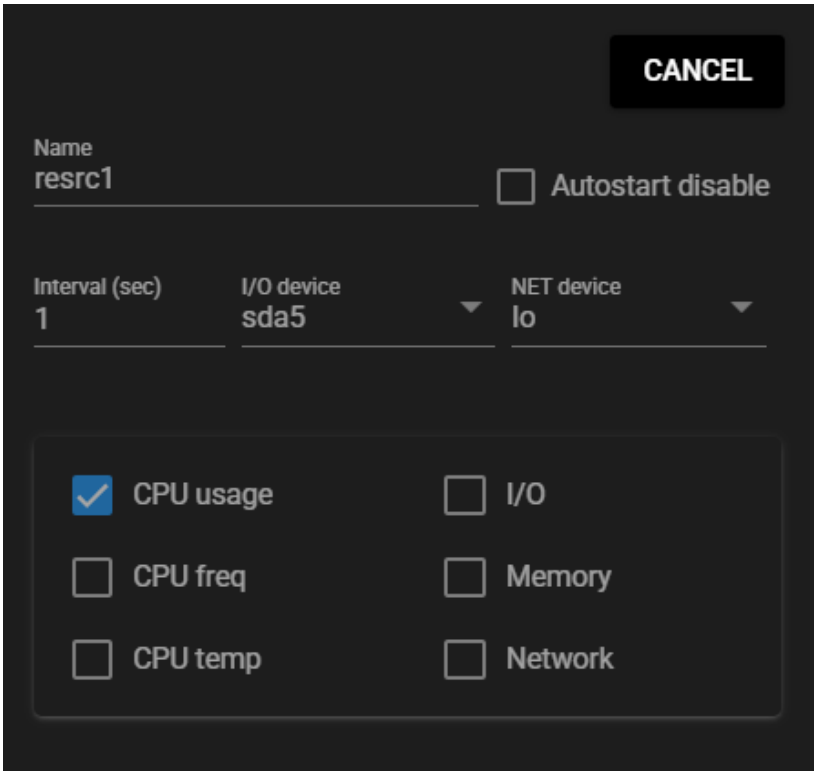
5.3.3.4.6 Confirming configurations

Component configurations can be confirmed by following the steps below.

1. Press the Configuration icon of the component.



2. A dialog opens and displays component configuration items and configuration values.



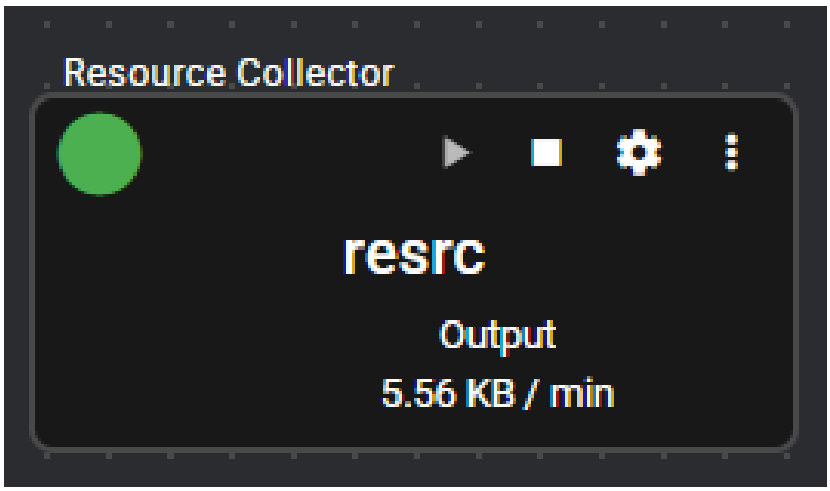
This operation can be performed only in Run Mode.

5.3.3.4.7 Component status

Component operation status can be checked by following the steps below.

- Status

  The status light displays the operational status of the component. For detail about the status, see Status List.

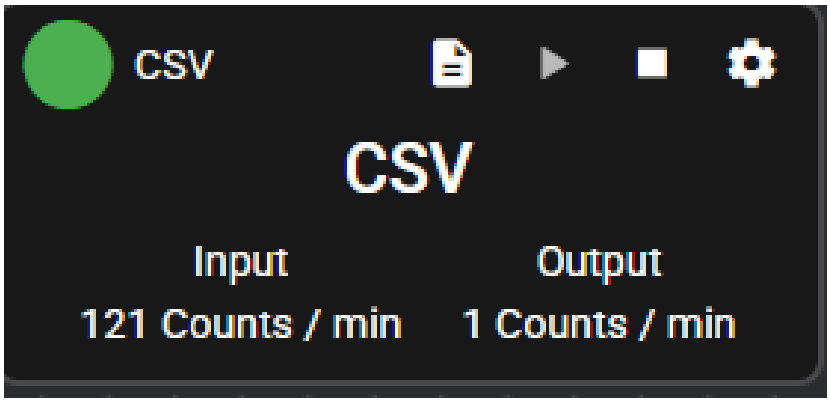


- Data volume

  This displays the volume of data for the following components.

  - Input and output data per minute
  - Emission and reception data per minute

  Click this to switch between displaying the number of data and the number of bytes.



5.3.3.4.8 Start/Stop

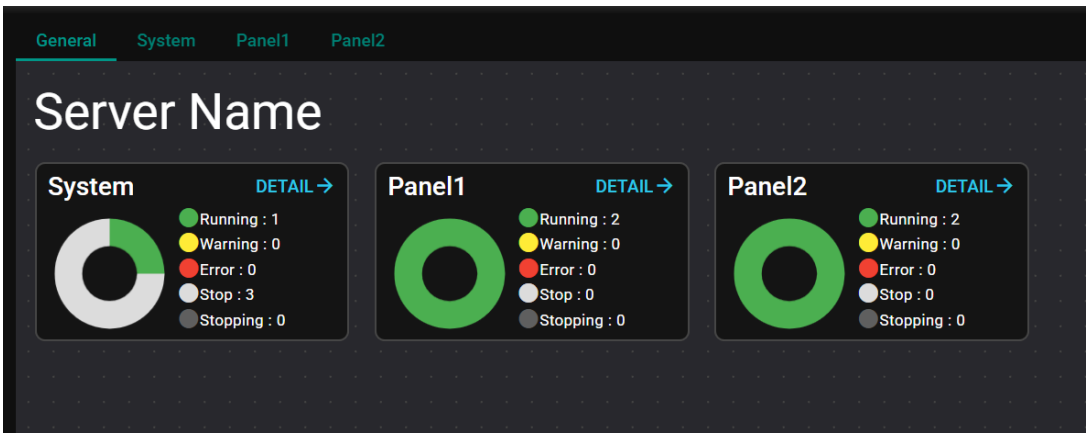
Components can be started/stopped by following the steps below.

(All components)

1. Press the "Start" button on the header.



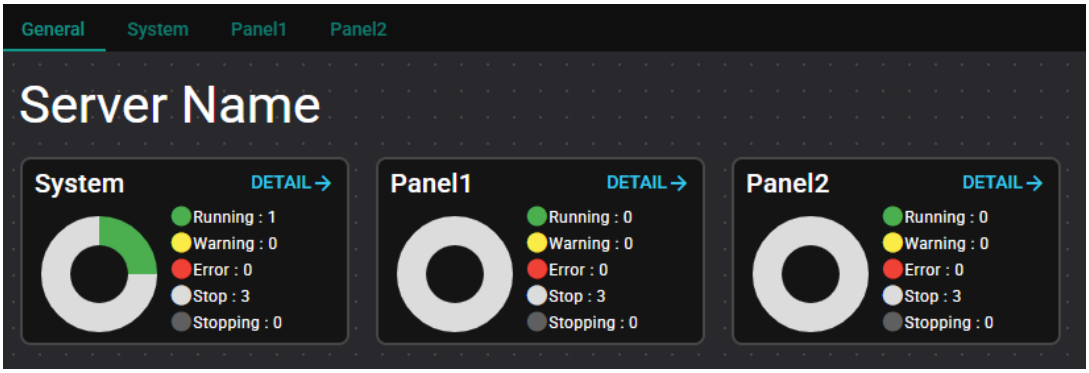
2. This starts all data collection flows registered in SpeeDBee Synapse.



3. Press the "Stop" button on the header.

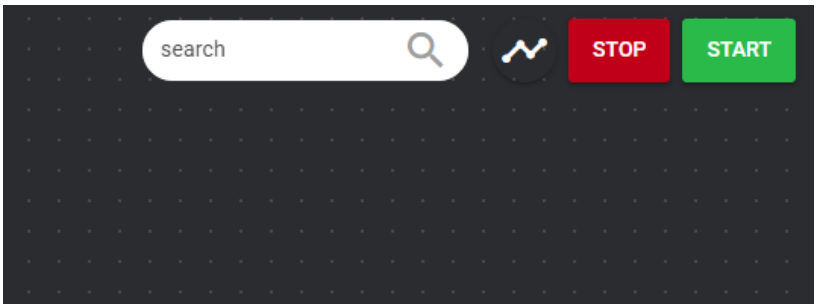


4. This stops all data collection flows registered in SpeeDBee Synapse.



(Panel Unit)

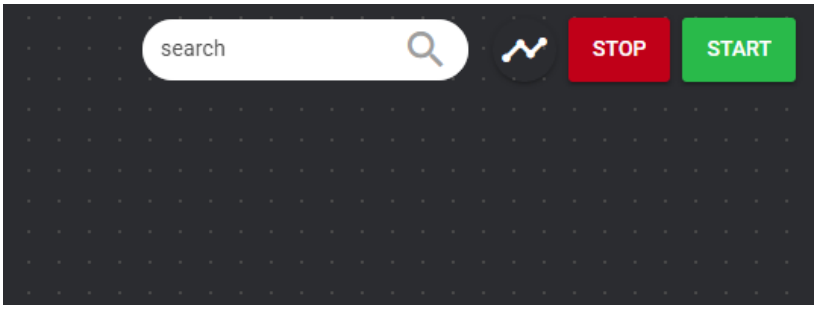
1. Press the "Start" button on the panel.



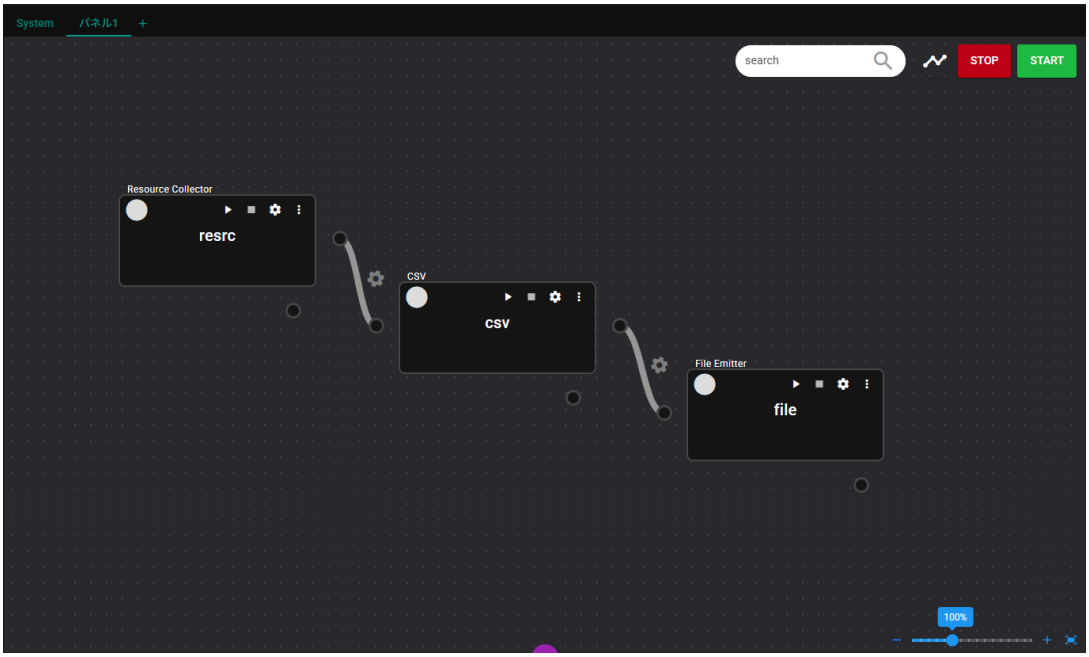
2. This starts the operation of the data collection flow registered in the panel.



3. Press the "Stop" button on the panel.



4. This stops the operation of data collection flow registered in the panel.

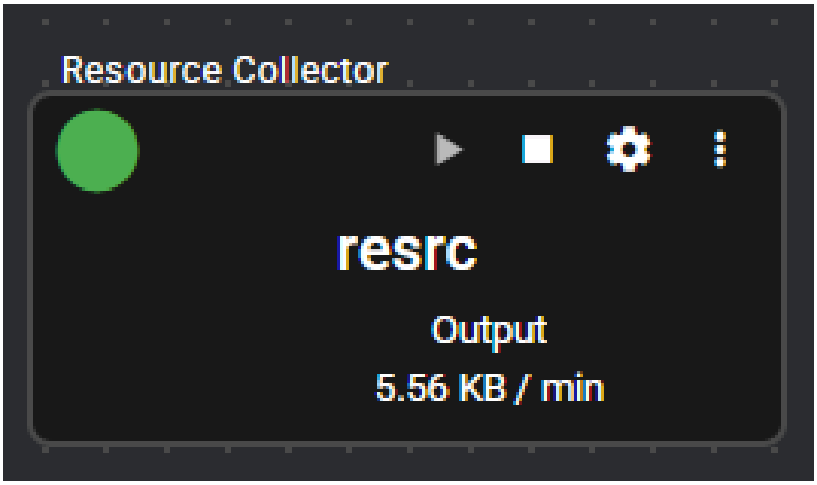


(Component Unit)

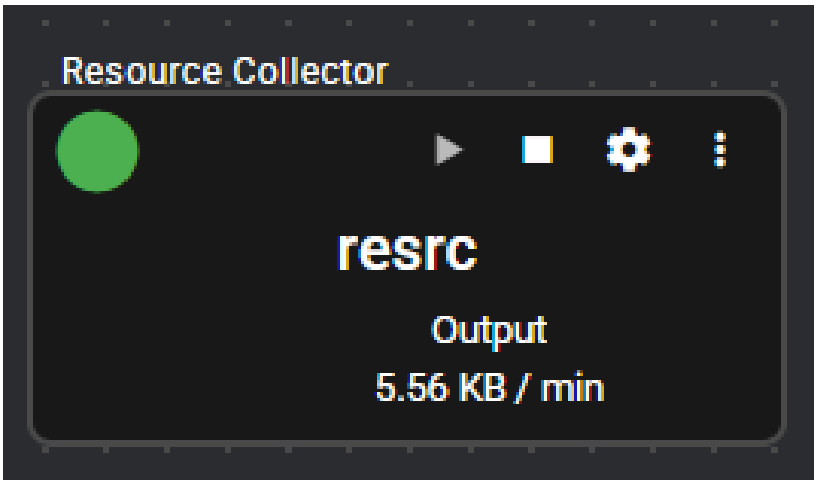
1. Press the Start icon of the component.



2. This starts the component and begins processing.



3. Press the Stop icon of the component.



4. This stops the component and ends processing.

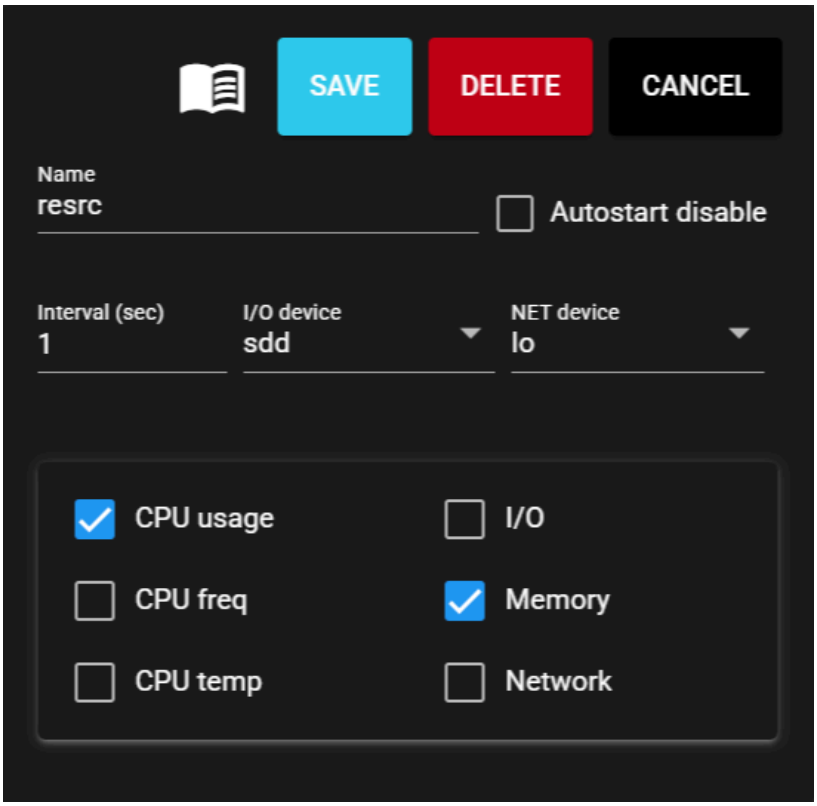


Users with the viewer role cannot start or stop components.

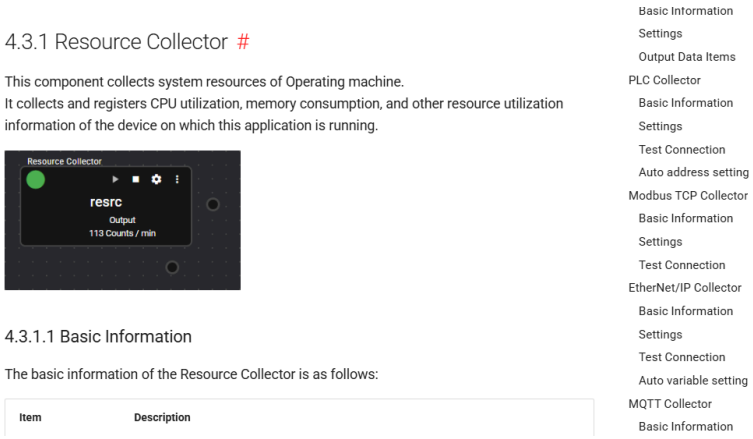
5.3.3.4.9 Display manual for component to operate

When the component to operate is on the display, the manual for the target screen can be shown.

1. Press the Manual icon when the component is displayed.



2. The manual for the component to operate opens in a separate window.



5.3.3.5 Flowlink

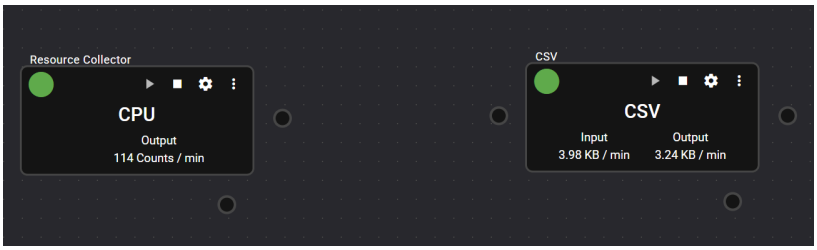
This section describes flowlink operations.

> This operation can be performed only in Edit Mode.

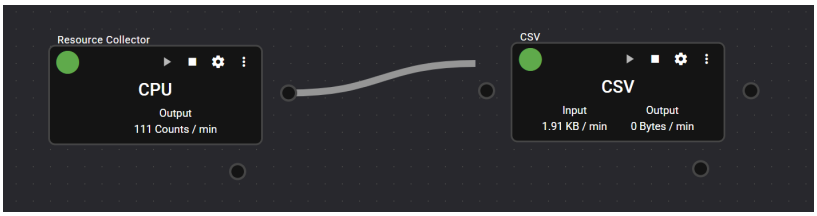
5.3.3.5.1 Add

Flowlink can be added by following the steps below.

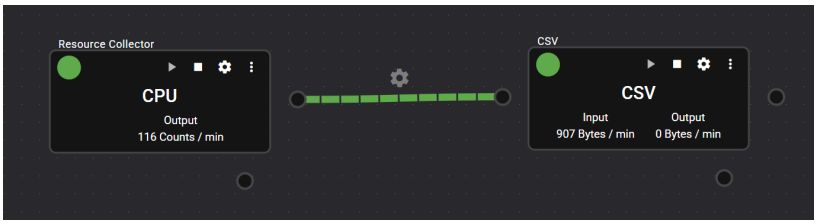
1. Drag the OUT Port or SYSTEM Port of the component.



2. Release the mouse button on the IN Port of another component.



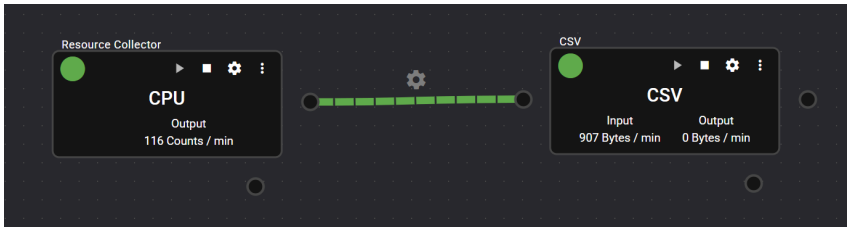
3. This adds a flowlink between these components.



5.3.3.5.2 Filter

Left-clicking the flow link icon or right-clicking the flow link allows filter configuration.

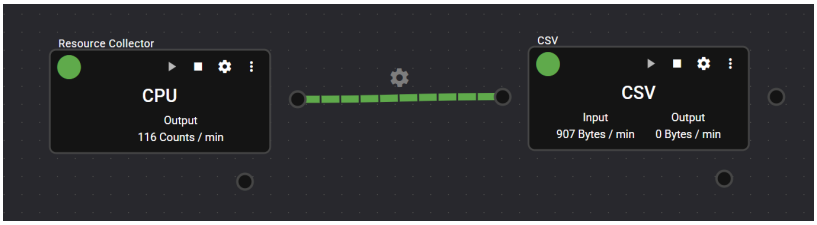
Configuration can restrict the data flowing through the flow link. For more details, see “Filter”.



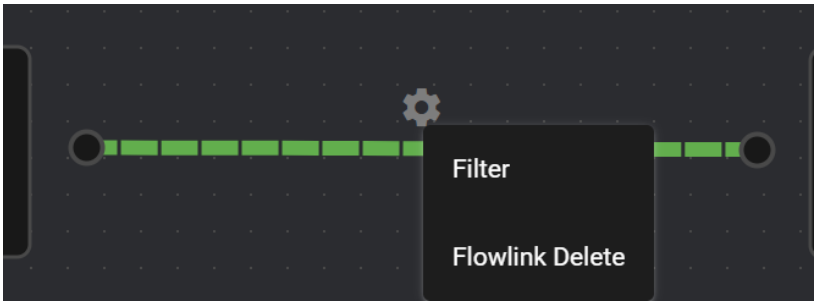
5.3.3.5.3 Delete

Flowlink can be deleted by following the steps below.

1. Right-click the flow link or left-click the icon.



2. Select "Flowlink Delete".



3. The flowlink is deleted.

5.3.3.6 Data monitor

By clicking the port of the component, the data that is flowing to the port can be checked on the data monitor.



5.3.3.6.1 Change configurations

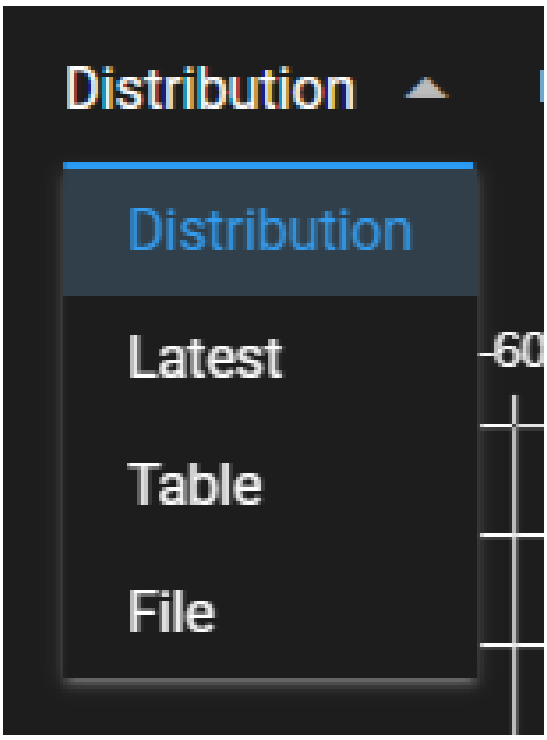
Change types

The data monitor types can be changed by following the steps below.

1. Click on the data monitor display type selection form.



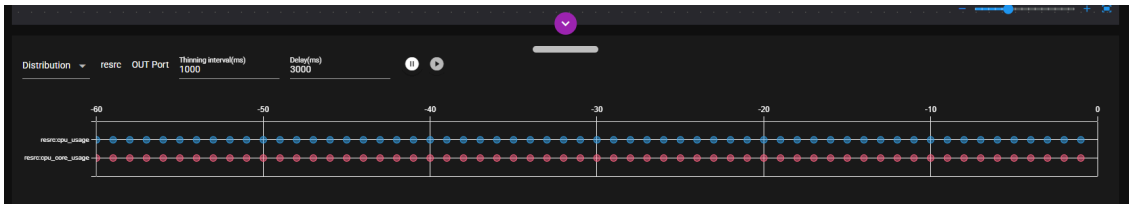
2. Select a type.



3. The data monitor of the selected type is displayed.

Configure thinning interval

The data monitor displays data by thinning them at specified intervals.
When the thinning interval is input in "Thinning interval(ms)", it is reflected in the data monitor.



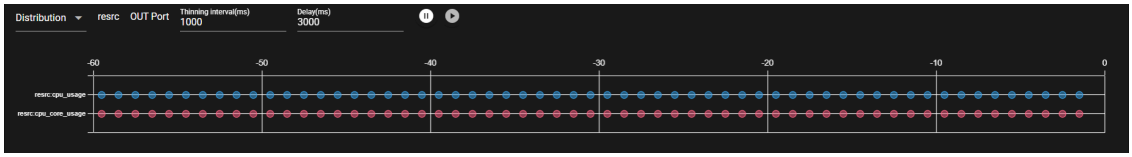
Configure delay time

The data monitor displays the latest value as the current time minus the "delay time(ms)".
When the value is input in "Delay time(ms)", it is reflected in the data monitor.



5.3.3.6.2 Distribution

The graph displays the data flowing through the port over a one-minute period. The graph displays the following information.



Item	Description
Horizontal axis	Time 0 is the current time minus the "Delay time(ms)", and -x is the time x seconds before that time.
Vertical axis	Data item name
Point	Data

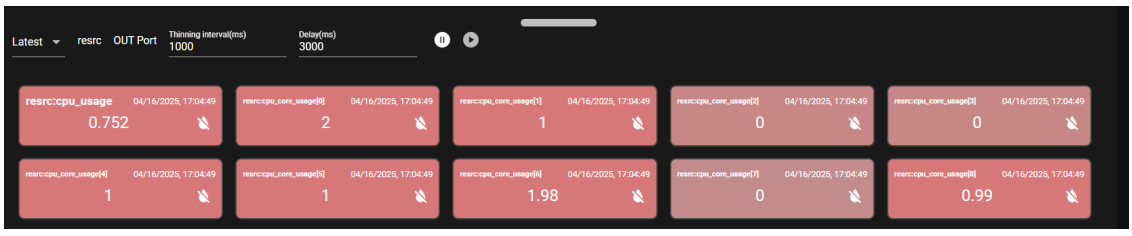
When the cursor is placed on the point, the data content and the time at which the data flowed are displayed.



5.3.3.6.3 Latest value

The following information is displayed for all data items on the port.

- Latest value
- Time at which the latest value flowed

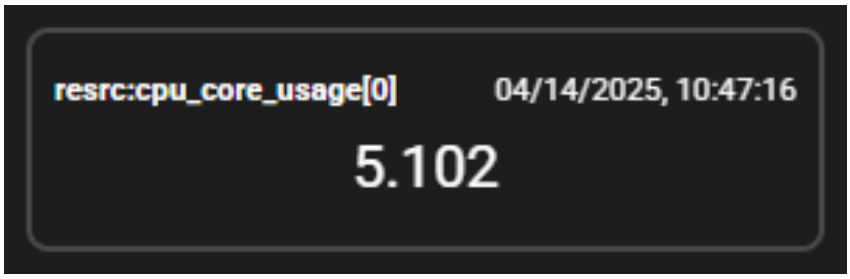


When data values are changed, they are displayed in red.



Pressing the Clear button restores the data items that have turned red to their original colors.

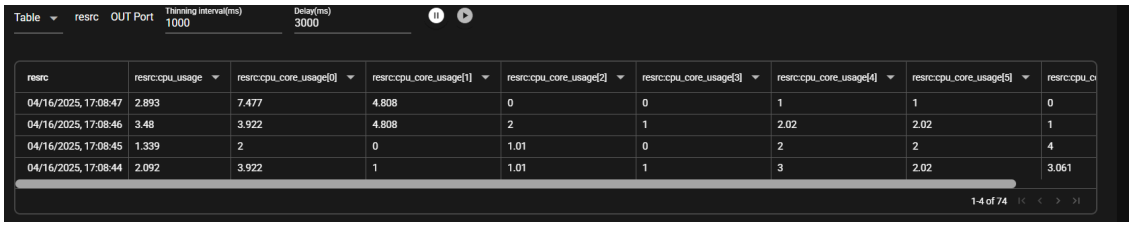




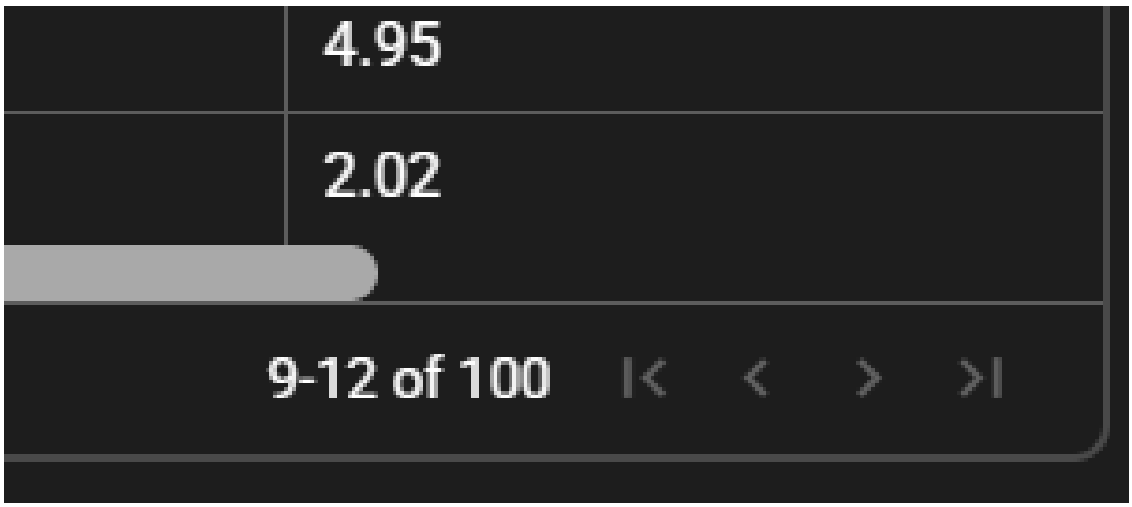
5.3.3.6.4 Table

The data that flowed into the port are displayed in table format. The table displays the following information. The number of table rows displayed per page is automatically adjusted according to the height of the data monitor.

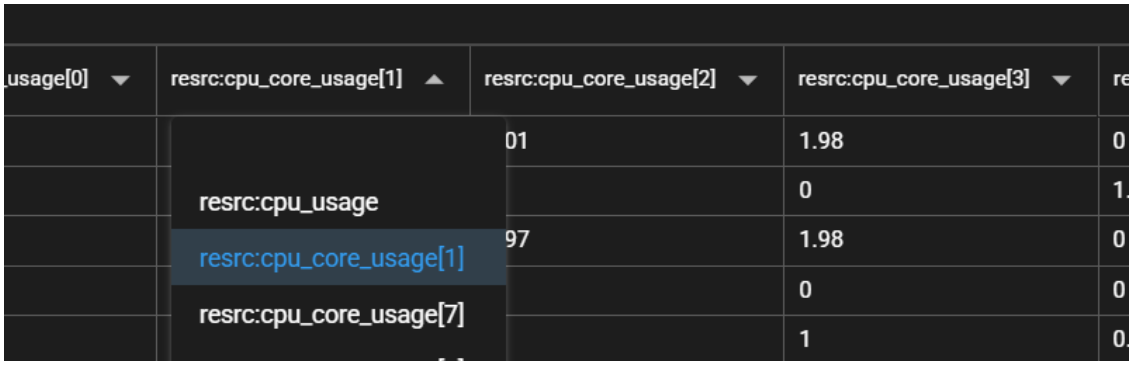
Item	Description
Columns	Data item name The maximum number of columns is 8.
Row	Time at which the data flowed The maximum number of rows is 100.
Content	Data



If the table contains more rows than can fit on one page, pages can be switched.



If there are more than eight data items on a port, selecting a column heading in the table changes the data items displayed in that column. When blank is selected, the corresponding section is not displayed.



5.3.3.6.5 File

If the data is in a file format, the following file information is displayed.

- File size ... Size of the file.
- Media type ... Media type of the file.
- Timestamp ... Time range of the data contained in the file
- File content

Item	Description
Text Content	Text is displayed in a monospaced font. If all data cannot be displayed within the view area, horizontal and vertical scroll bars are available for navigation.
Image Content	Images are automatically scaled up or down to fit within the display area while preserving their aspect ratio. Images smaller than the display area are shown at their original size. Note that supported image formats depend on the browser's rendering capabilities. Please check which image formats are supported by your browser.

Text Display Example

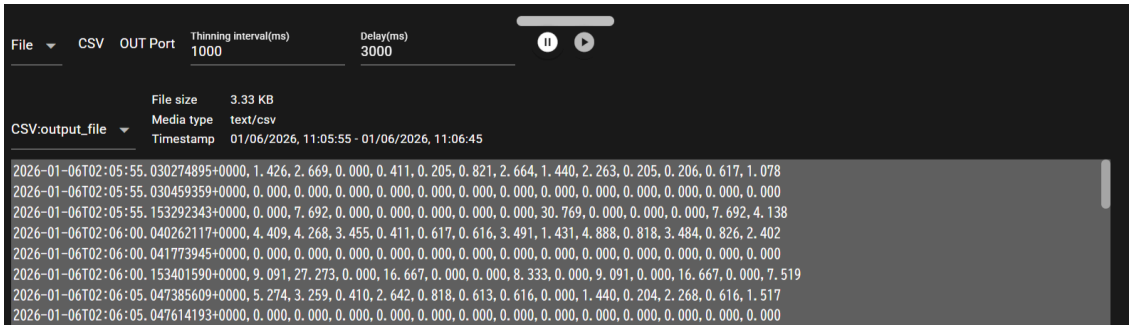
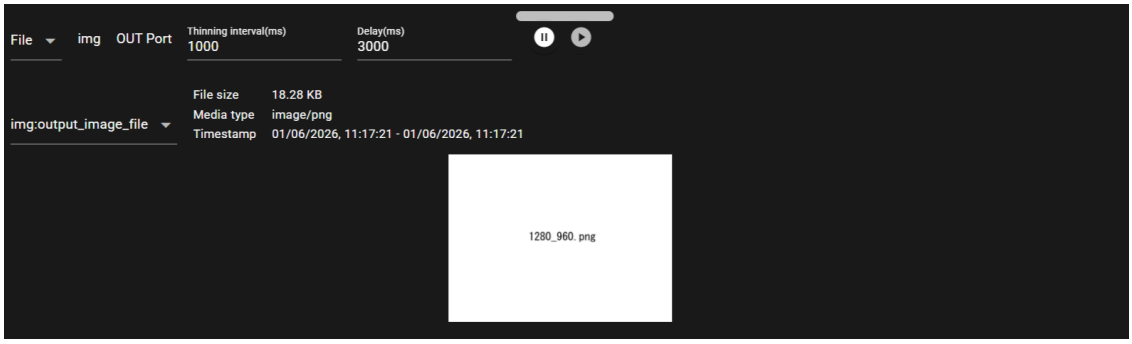


Image Display Example



Supported Image Formats in the Data Monitor

Supported image formats depend on the browser's rendering capabilities. As a result, images in formats other than common ones such as JPG, PNG, and GIF (for example, HEIF) may not be displayed in some browsers. Please verify which image formats are supported by your browser.

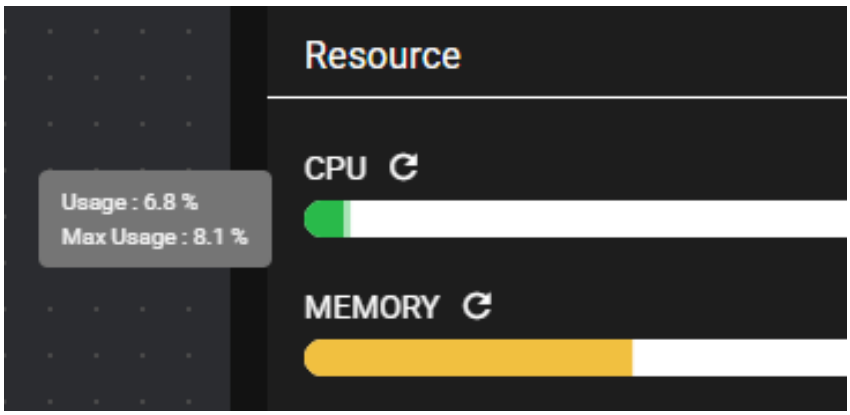
5.3.3.7 System resources

This section describes system resources.

This operation can be performed only in Run Mode.

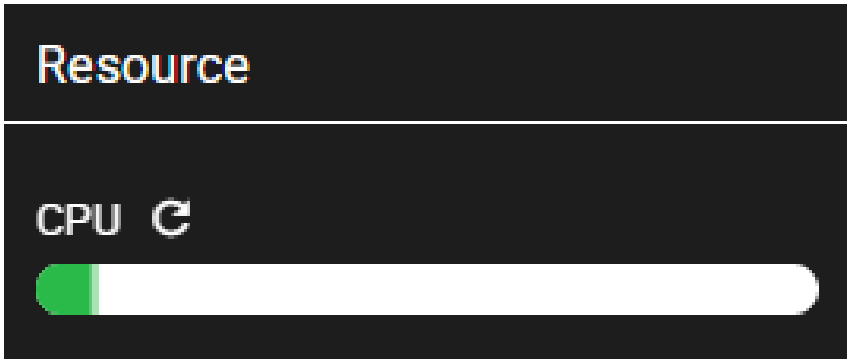
5.3.3.7.1 CPU

When the cursor is placed on the gauge, the following information is displayed.



Item	Description
Usage	The current CPU usage of SpeeDBee Synapse
Max Usage	The maximum CPU usage of SpeeDBee Synapse

The dark portion of the gauge represents the current CPU usage and the light portion represents the maximum CPU usage that has been reached during operation.



The gauge changes color according to the percentage of CPU usage, as shown in the table below.

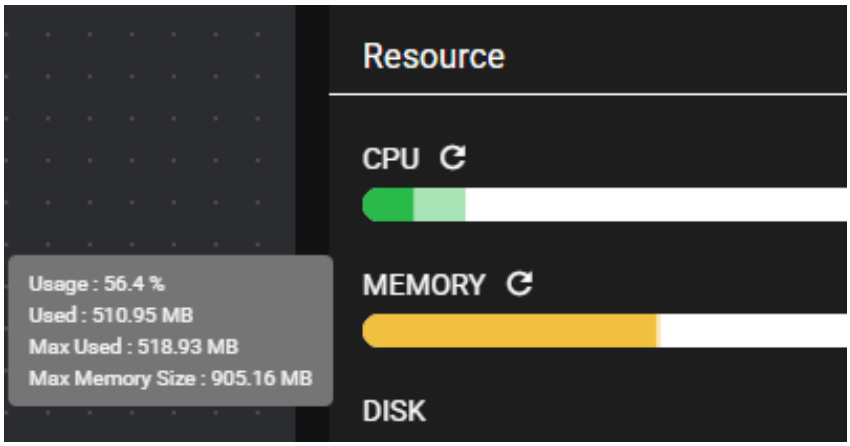
Percentage	Color
0~40	Green
40.1~70	Yellow
70.1~100	Red

Pressing the Refresh icon resets the Max Usage.



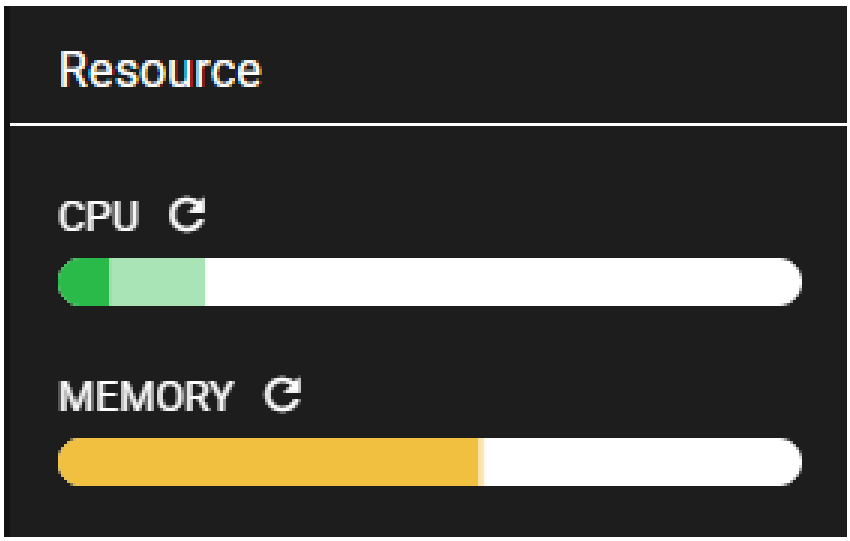
5.3.3.7.2 MEMORY

When the cursor is placed on the gauge, the following information is displayed.



Item	Description
Usage	The current memory usage of SpeeDBee Synapse
Used	The current memory being used in SpeeDBee Synapse
Max Used	The max value of the memory usage of SpeeDBee Synapse
Max Memory Size	Memory size of the entire system

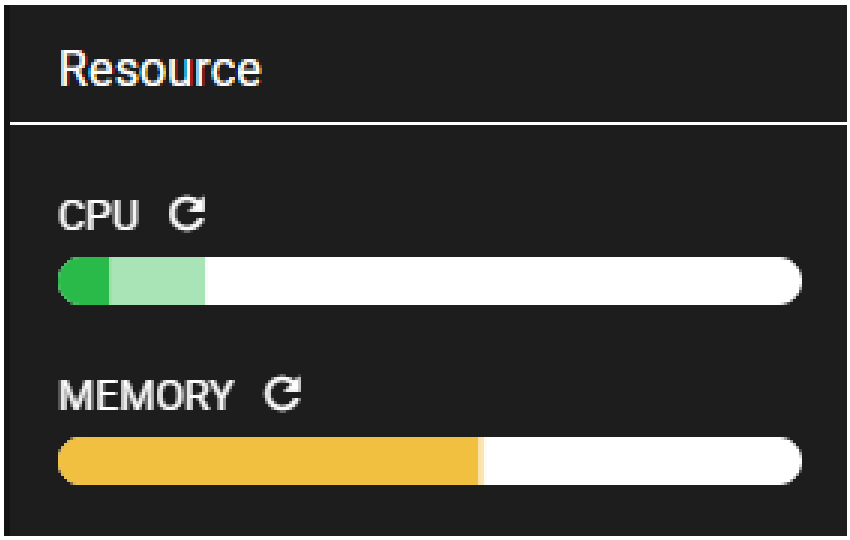
The colored portion of the gauge represents memory usage.



The gauge changes color according to the percentage of memory usage, as shown in the table below.

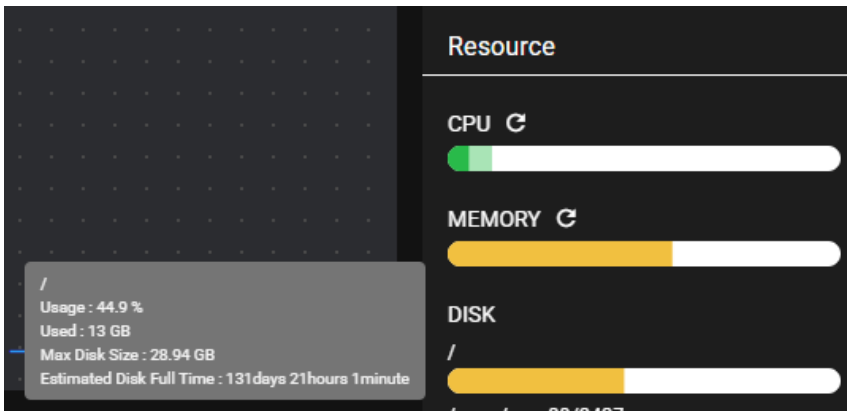
Percentage	Color
0~40	Green
40.1~70	Yellow
70.1~100	Red

Pressing the Refresh icon resets the Max Used.



5.3.3.7.3 DISK

When the cursor is placed on the gauge, the following information is displayed.



Item	Description
Usage	The current ratio of disk usage in SpeeDBee Synapse compared to the Max Disk Size
Used	The current disk being used in SpeeDBee Synapse
Max Disk Size	Maximum amount of disk space available for SpeeDBee Synapse
Estimated Disk Full Time	Estimated time for the disk used to reach maximum disk capacity, based on the 1-minute increase in disk used Note: This is not displayed if there is no increase in disk used

The colored portion of the gauge represents disk usage.

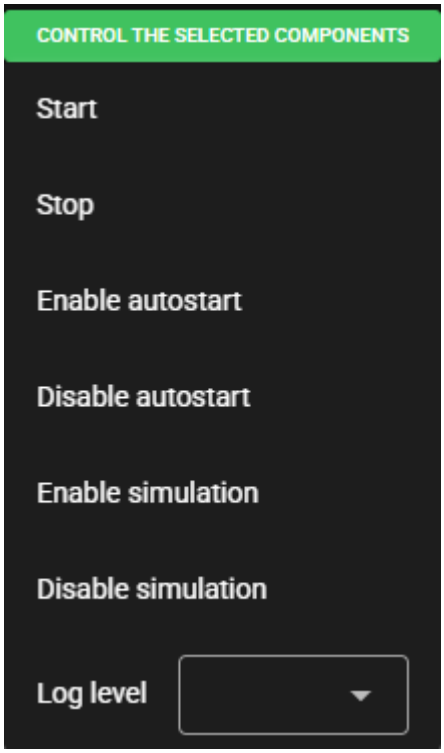


The gauge changes color according to the percentage of disk usage, as shown in the table below.

Percentage	Color
0~40	Green
40.1~70	Yellow
70.1~100	Red

5.3.3.8 Control the selected components

Control the Selected components is a function that controls the selected components in List Mode. Operations such as start, stop, toggle autostart, toggle simulation mode, and select log level can be performed collectively on the selected components.



- Start

  Starts the selected components collectively. Starts regardless of the autostart setting.
- Stop

  Stops the selected components collectively.
- Enable autostart

  Enables autostart for the selected components collectively.
- Disable autostart

  Disables autostart for the selected components collectively.
- Enable simulation

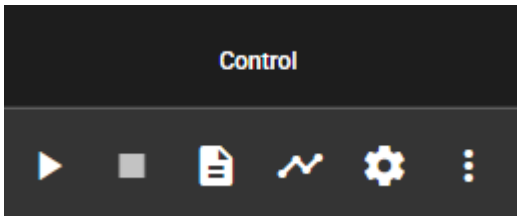
  Enables simulation for the selected components collectively.
- Disable simulation

  Disables simulation for the selected components collectively.
- Log level

  Changes the log level of the selected components collectively.

5.3.3.9 Component list control

Component list control is a function that controls components from the component list in List Mode. Operations such as start, stop, log viewing, and editing can be performed on the component list.



- Start

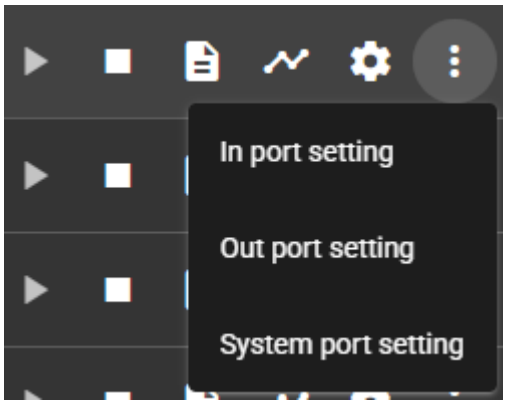
  Starts the component.
- Stop

  Stops the component.
- Log

  Opens the log dialog of the component.
- Status Timeline

  Opens the status timeline dialog of the component.
- Configuration

  Opens the configuration dialog of the component.
- Port settings

  - In port setting

    Opens the in port setting dialog of the component.
  - Out port setting

    Opens the out port setting dialog of the component.
  - System port setting

    Opens the system port setting dialog of the component.

5.3.3.10 Simulation mode

This section describes the simulation mode.

5.3.3.10.1 Outline of simulation mode

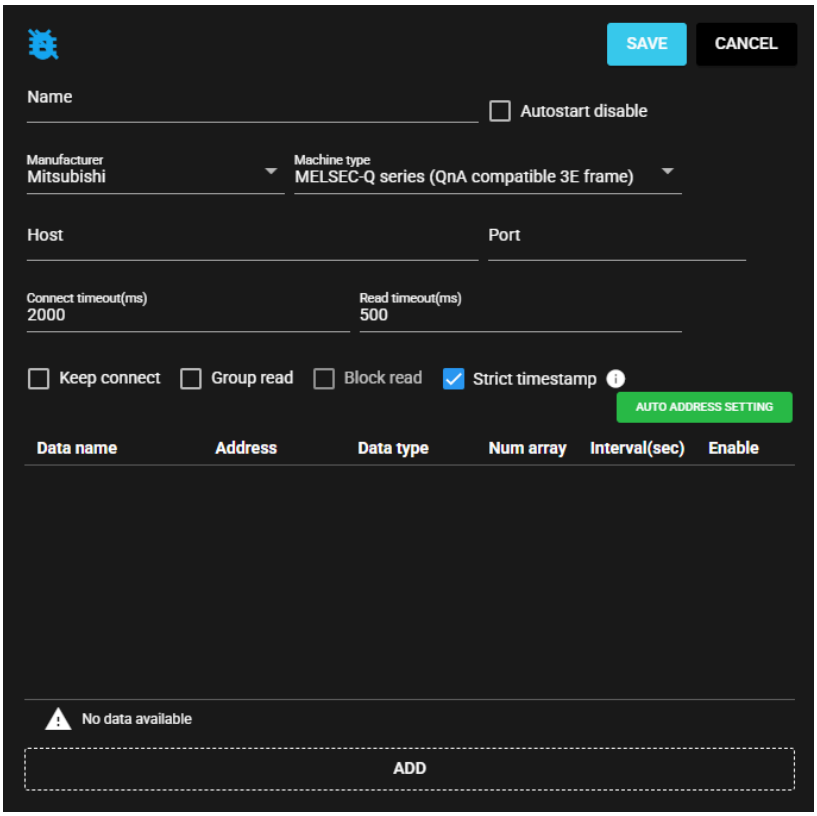
The simulation mode is a function that generates columns based on the contents of a CSV file and registers dummy data. This function enables configurations and operation verification even in an environment where there is no actual opposed equipment (such as PLC). Components that support simulation mode are as follows.

- Modbus Collector
- PLC Collector
- EtherNet/IP Collector
- MQTT Collector

5.3.3.10.2 Configuration and switching of simulation mode

This section describes using the PLC collector.

1. Click the Simulation settings button in the upper left of the component configuration window.



2. The simulation settings window opens. Input configuration items and click "Close".

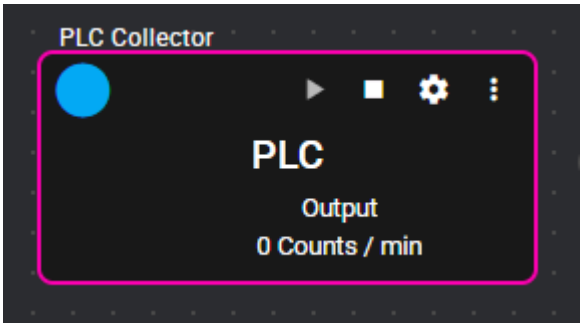


Item	Description
Simulation mode	Enable/disable simulation mode
Operation file (csv)	A CSV file that defines the columns and data used in simulation mode. For detail, see Formatting Driving Files.

Once a driving file is uploaded, its configurations are retained until deleted.



3. The window returns to the component configuration window. Input component configuration items and click "Save". Changes to simulation settings are not applied until "Save" is clicked on the component configuration screen.
4. Click the Start button to start. The component status light turns blue when the system is running while the simulation mode is ON.



5.3.3.10.3 Operation File Format

The operation file is in CSV format. This consists the header section (the first row) and the data section (the second row onward). The header section defines the columns, while the data section specifies the timestamps and data to register.

example.csv

```
timestamp_ms,data1:bool,data2:int32,data3:string,data4:binary
        1000,      true,          1,       test1,    YWl1ZW8=
        2000,     false,          2,       test2,    YWl1ZW8=
        3000,      true,          3,       test3,    YWl1ZW8=
```

Header section

```
timestamp_ms,data1:bool,data2:int32,data3:string,data4:binary
```

- The first row of the header section is fixed as timestamp_ms, and column definitions are input starting from the second row onward.
- Column definitions specify column names and data types separated by a `:`, such as `column name: data type.`
- For arrays, the number of arrays is enclosed in `[]` to specify, such as `column name: data type [Array Size].`

Data section

```
        1000,      true,          1,       test1,    YWl1ZW8=
        2000,     false,          2,       test2,    YWl1ZW8=
        3000,      true,          3,       test3,    YWl1ZW8=
```

- For the first row of the data section, input the value of timestamp_ms, and for the second row and onward, input data values to actually register.
- Data is registered based on the time elapsed from the start to the value of timestamp_ms. In the example of the above CSV, the data is registered at 1,000, 2,000, and 3,000 milliseconds rom the start.
- If timestamp_ms is less than the last value (back in time), no data is registered for that row.
- When the end of the file is reached, data returns to the beginning of the file and is registered from the beginning again.
- If the value of the data is blank, the data is not registered.
- Array data is specified by separating them with spaces.

Available data types for specification	**Remarks**
bool	Specify using true, false, TRUE, or FALSE.
int8	Specified in the range int8. Hex can be specified by adding 0x to the prefix.
int16	Specified in the range of int16. Hex can be specified by adding 0x to the prefix.
int32	Specified in the range int32. Hex can be specified by adding 0x to the prefix.
int64	Specified in the range int64. Hex can be specified by adding 0x to the prefix.
uint8	Specified in the range uint8. Hex can be specified by adding 0x to the prefix.
uint16	Specified in the range of uint16. Hex can be specified by adding 0x to the prefix.
uint32	Specified in the range uint32. Hex can be specified by adding 0x to the prefix.

Available data types for specification	Remarks
uint64	Specified in the range uint64. Hex can be specified by adding 0x to the prefix.
float	Specified in the float range. Hex can be specified by adding 0x to the prefix.
double	Specified in double range. Hex can be specified by adding 0x to the prefix.
string	When specifying special characters (such as `,` `\n`) in CSV, enclose them in `"`. The escape for `"` is `""`.
binary	Specified in Base64 format.

A sample driving file is shown below.

- All data type sample
- Array data sample

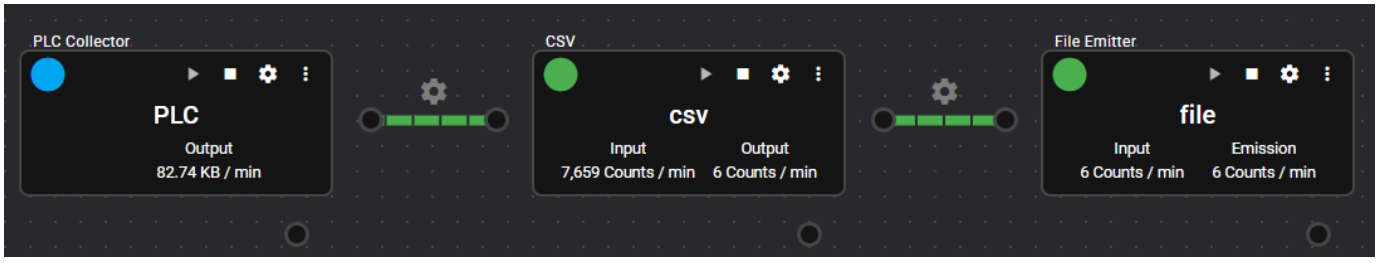
5.3.3.10.4 Operation of simulation mode

While the system is running on the simulation mode, only the dummy data is registered. The data of communication with actual devices is not collected. As for the component status while the simulation mode is running, the status lamp is shown with blue, but the status is the same as regular `Running`.

- The registered dummy data can be checked through data monitoring in the same way as normal components.



- Data can also be transferred to a different component by connecting flowlink.



Simulation mode poses no problem when validating data flows during development. However, be sure to turn it OFF when operating in an actual environment.

5.3.3.11 How to operate on a table device

SpeeDBee Synapse supports operation on a tablet device. The verified tablet devices are as follows.

Model name/model	OS version	Browser
iPad(A16)	iOS 18.5	Safari 605.1.15
HP Spectre x360 Convertible 14-ea0xxx	Windows 11 24H2	Google Chrome 136.0.7103.114

Some actions on a tablet device differ from operation on a PC using a mouse. The main differences are summarized below.

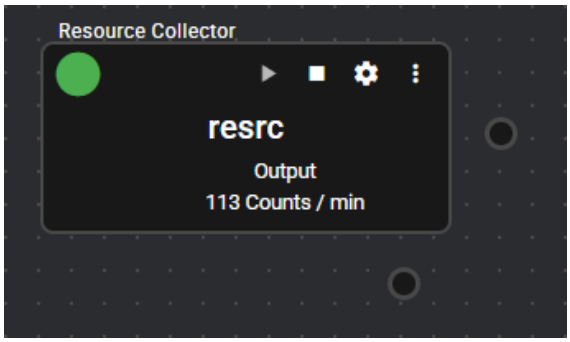
Function and operation	Operating on PC	Operation on a tablet
Context menu	Right-click	Long tap (hold) on a target element.
Drag & drop	Dragging with the mouse	Tap an object with a finger and drag it by moving your finger.
Hovering	Information is displayed by moving a mouse over it.	Hovering is not supported on devices with touch panels. However, some operations can be performed by tapping or long-tapping.

5.4 Collector

This page explains the collector components.

5.4.1 Resource Collector

This component collects system resources of Operating machine.
It collects and registers CPU utilization, memory consumption, and other resource utilization information of the device on which this application is running.



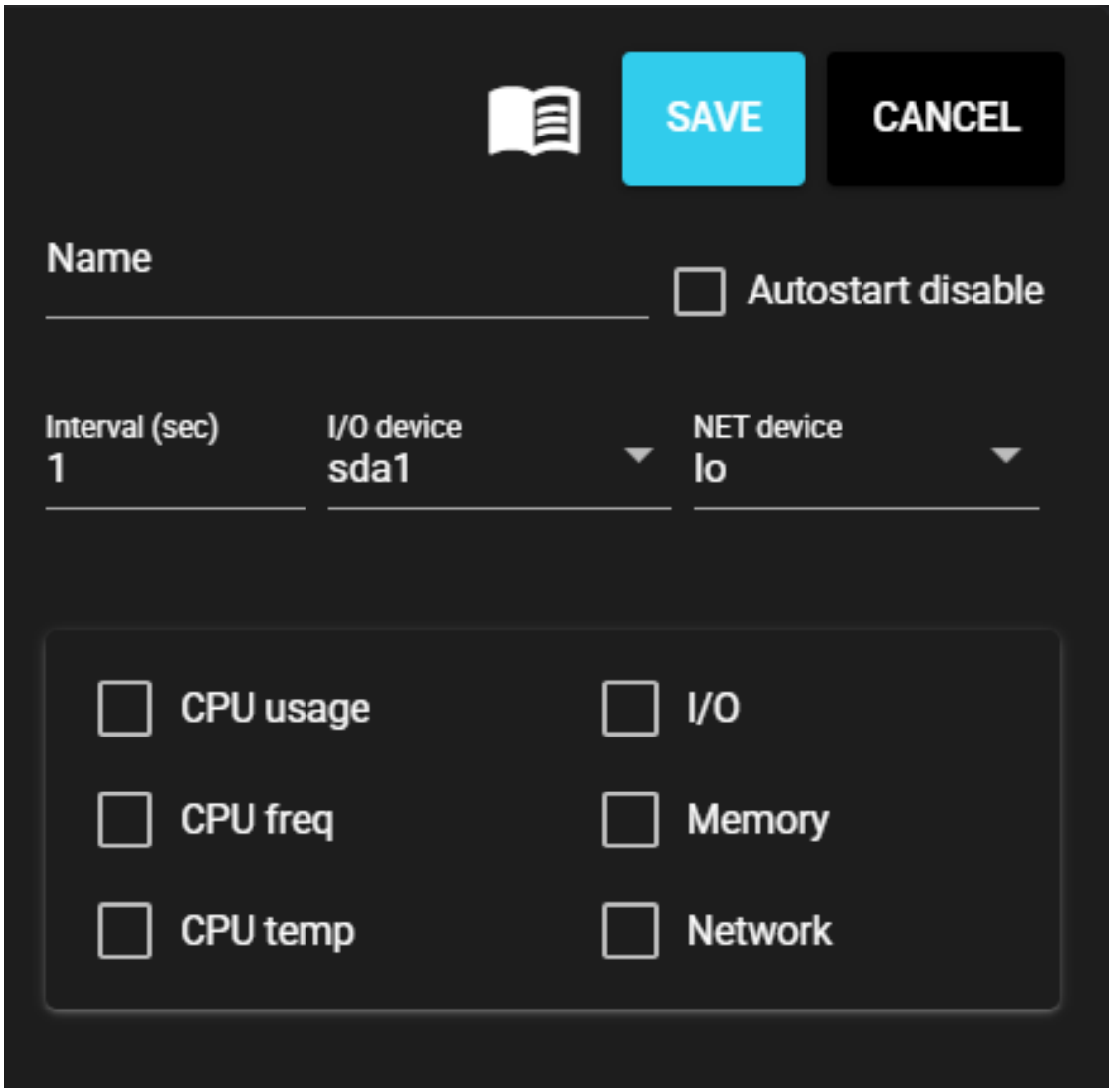
5.4.1.1 Basic Information

The basic information of the Resource Collector is as follows:

Item	Description
Base name	Resource Collector
Description	Collects system resources of Operating machine.
Consumption points	0
Port	Input: 0 Output: 1 Outputs the collected system resources of Operating machine. For details on output data, see Output Data Items. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.

5.4.1.2 Settings

The Resource Collector settings are as follows:



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Interval (sec)	Enter the interval at which the collector gets data.
I/O device	Select the I/O device from which data is collected.
NET device	Select the network device from which data is collected.

Item	Description
CPU usage	Set to ON to enable the collection of Operating machine's CPU utilization information.
CPU clock rate	Set to ON to enable the collection of Operating machine's CPU clock rate information. Note: Not available on Windows Note: Also may not be available on Linux
CPU temp	Set to ON to collect CPU temperature information. Note: Not available on Windows Note: Also may not be available on Linux
I/O	Set to ON to collect the I/O status information for the device selected in [I/O device]. Note: If I/O status collection is enabled in the Windows version, the processing may not be able to keep up with the acquisition interval of 1 second. (If delay occurs, make an adjustment by setting the interval to 5 seconds or more.)
Memory	Set to ON to enable the collection of Operating machine's used memory information.
Network	Set to ON to collect the network status information for the device selected in [NET device].

5.4.1.3 Output Data Items

The Resource Collector outputs the following data:

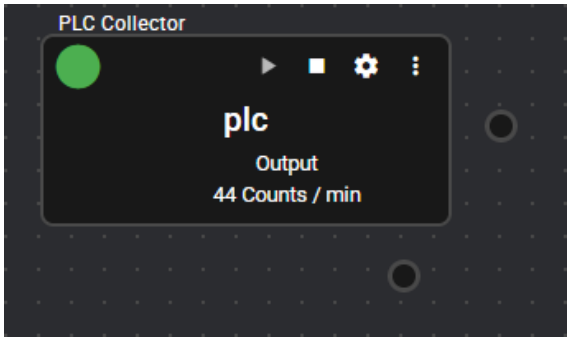
Item	Column name	Data type	Description
CPU usage	cpu_usage	double	Collects the CPU utilization information. Unit is % with a range from 0.000 to 100.000.
Utilization rate per CPU core	cpu_core_usage	double[n]	Collects the utilization information of each CPU core. Unit is % with a range from 0.000 to 100.000.
CPU execution frequency	cpu_core_freq	int32[n]	Collects the execution frequency information of each CPU core. Unit is kHz.
CPU temp	cpu_temp	double	Collects the CPU temperature information. Unit is °C.

Item	Column name	Data type	Description
I/O information collection device name	I_O	string	Name of device whose I/O information is being collected (physical only)
Average number of read requests processed	kb_r_s	double	Collects the average number of read requests processed. Unit is KB.
Average number of write requests processed	kb_w_s	double	Collects the average number of write requests processed. Unit is KB.
Average read/write count	IOPS	double	Collects the average read/write count. Unit is KB.
Free disk space (MB)	storage_free_space_mb	double	Collects information on free space on the specified disk. Unit is MB.
Disk usage (%)	storage_usage	double	Collects information on rate of usage of the specified disk. Unit is % with a range from 0.0 to 100.0.
Total memory	mem_total	int32	Collects total memory space information. Unit is KB.
Memory used	mem_used	int32	Collects information on memory space used. Unit is KB.
Memory usage rate	mem_usage	double	Collects information on rate of memory used. Unit is % with a range from 0.0 to 100.0.
Total SWAP space	swap_total	int32	Collects total SWAP space information. Unit is KB.
SWAP space used	swap_used	int32	Collects information on SWAP space used. Unit is KB.
SWAP usage rate	swap_usage	double	Collects information on rate of SWAP space used. Unit is % with a range from 0.0 to 100.0.

Item	Column name	Data type	Description
Total bytes received	rx_bytes1	uint64	Collects the number of bytes received. Unit is byte.
Total packets received	rx_pckt1	uint64	Collects the number of packets received.
Total bytes sent	tx_bytes1	uint64	Collects the number of bytes sent. Unit is byte.
Total packets sent	tx_pckt1	uint64	Collects the number of packets sent.

5.4.2 PLC Collector

This component collects data from PLC register/variable information.
It is connected to PLCs of various manufacturers such as Omron and Mitsubishi to collect and register data from register/variable information.
For some PLCs, you can set register information by importing the file output using the ladder editing tool.



5.4.2.1 Basic Information

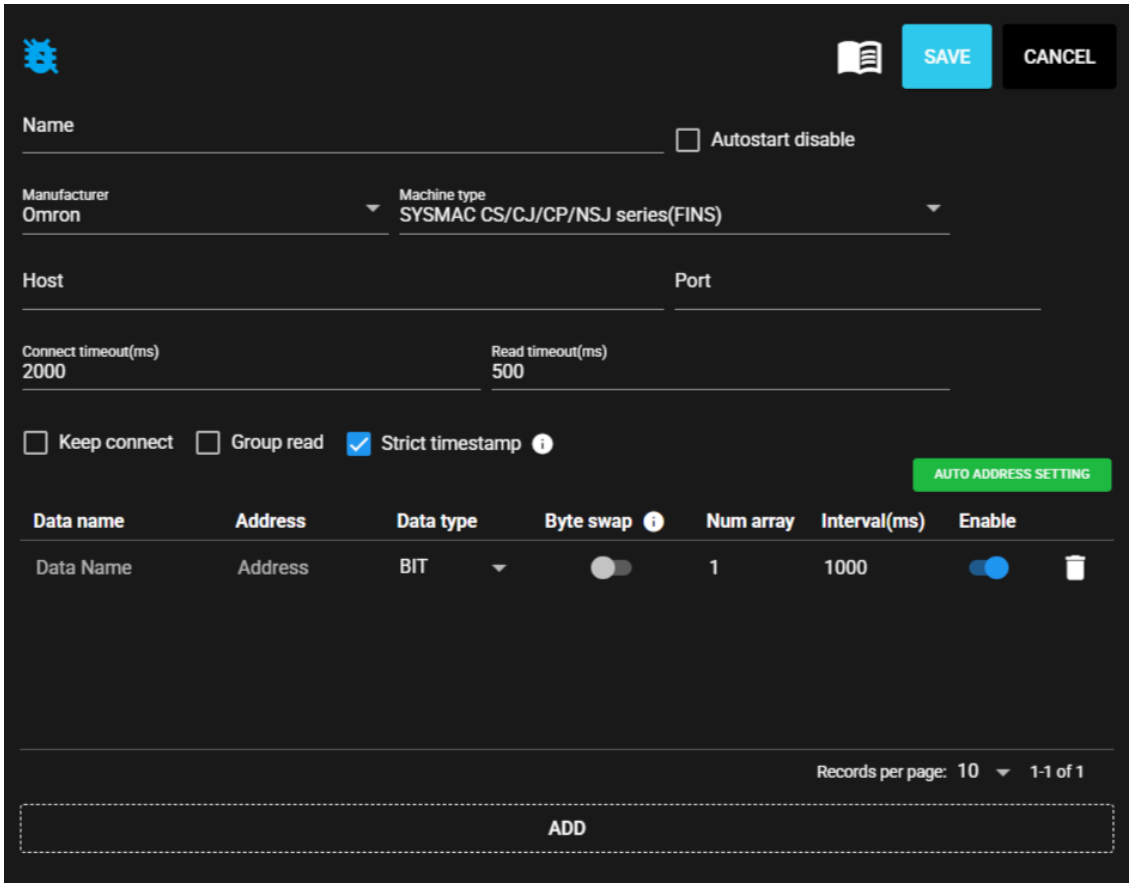
The basic information of the PLC Collector is as follows:

Item	Description
Base name	PLC Collector
Description	Collects data from PLC register/variable information.
Consumption points	0

Item	Description
Port	Input: 0 Output: 1 Outputs data collected from the PLC according to Settings. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	- You can test connection with a PLC. For more information, see Test Connection. - Addresses can be set automatically. For more information, see Auto address setting.

5.4.2.2 Settings

The PLC Collector settings are as follows:



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Manufacturer	Select the manufacturer of the connecting PLC. Note: You can select from the following manufacturers: - Mitsubishi - Jtekt - Omron - Keyence - Panasonic
Machine type	Select the machine type of the connecting PLC. Note: Machine types that can be selected depend on the manufacturer.
Host	Enter the host name or IP address of the connecting PLC.
Port	Enter the port for connecting PLC
Connect timeout(ms)	Enter the timeout duration for connecting.
Read timeout(ms)	Enter the timeout duration for reading.
Keep connect	Set to ON if PLC connection is not disconnected each time a query is made.
Group read	Set to ON to read multiple word registers at one time.
Block read	Set to ON to read multiple bit registers and word registers at one time.
Strict timestamp	Set to ON to set a strict timestamp based on collected data.

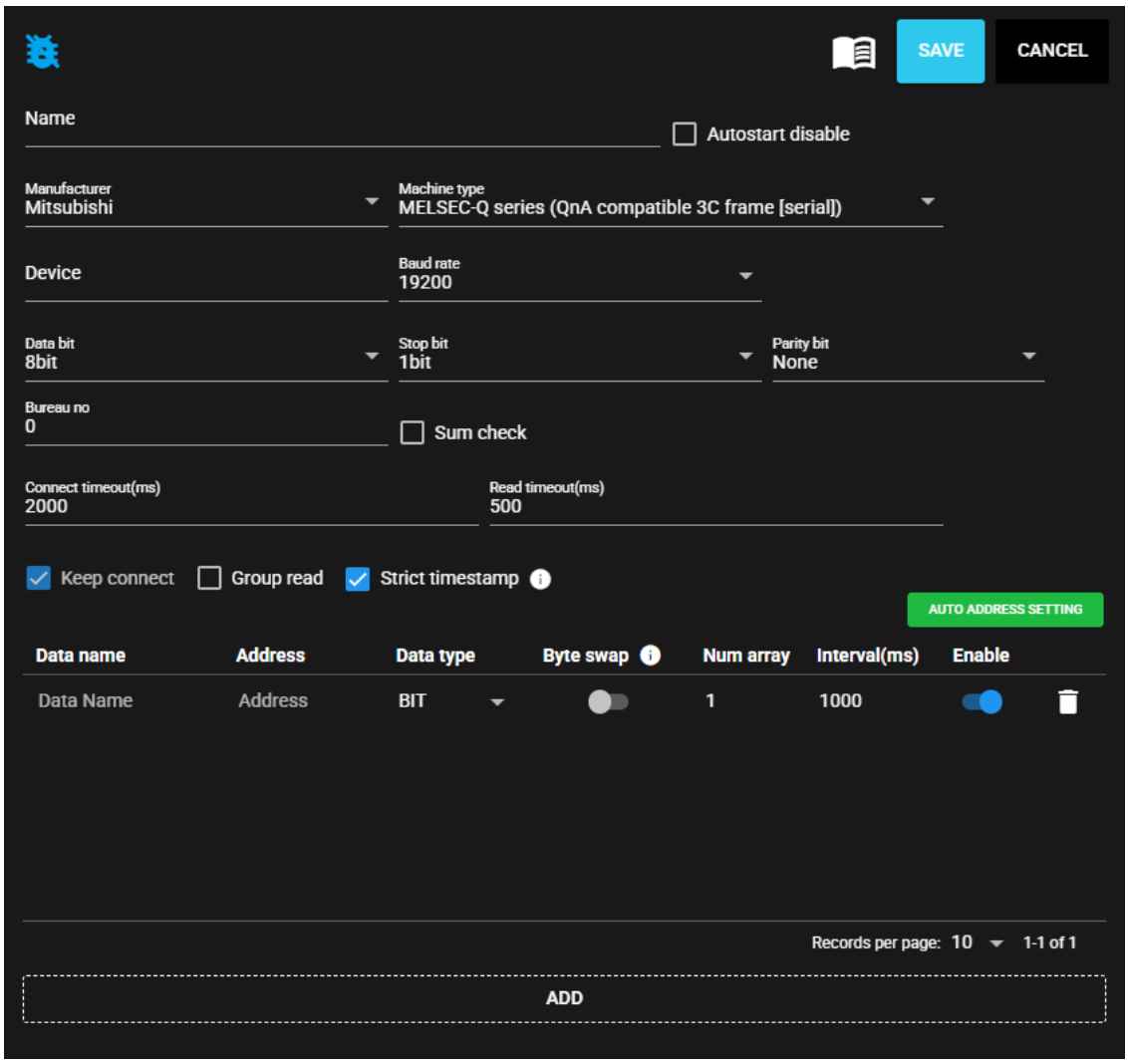
- The PLC Collector output data is defined by the following settings:

Item	Description
Data Name	Enter name of data.
Address	Enter the PLC register address. Note: When an address is entered, `(DEC)` / `(HEX)` / `(DDDX)` appears to the right of the address

Item	Description
	based on the protocol and device type. ・Decimal representation: `D00 (DEC)` ・Hexadecimal representation: `W00 (HEX)` ・Mixed representation (upper digits in decimal, least significant digit in hexadecimal): `X001F (DDDX)` (Example: In `X001F`, the `001` portion is interpreted as decimal, and the least significant digit `F` is interpreted as hexadecimal.) This notation indicates which numeral system is applied to the numeric portion of the entered address.
Program	Select the register program number. Note: Available only when the manufacturer is Jtekt.
Data type	Select the data type of the register value to be acquired.
Byte swap	Set to ON when swapping the order of two bytes within 1WORD value. Note: This operation is only available when the data type is set to `STRING`.
Num array	Enter the range from which value is retrieved from among the consecutive registers starting from the input address.
Interval(ms)	Enter the interval for retrieving register values.
Enable	Set to ON to retrieve register values.

5.4.2.3 Settings (Serial Connection)

The PLC Collector (Serial Connection) settings are as follows:



Serial Connection

For Omron products, SpeeDBee Synapse does not support serial connections. Therefore, the following models cannot be selected.

- MELSEC-Q series (QnA compatible 3C frame [serial])

Item	Description
Device	Enter the serial communication device to use.
Baud rate	Select the baud rate for serial communication.
Data bit	Select the data bit for serial communication.
Stop bit	Select the stop bit for serial communication.

Item	Description
Parity bit	Select the parity bit for serial communication.
Bureau no	Enter the bureau number for Mitsubishi 3C frame.
Sum check	Set to ON to perform checksum for Mitsubishi 3C frame.

- The PLC Collector output data is defined by the following settings:

Item	Description
Data Name	Enter name of data.
Address	Enter the PLC register address. Note: When an address is entered, `(DEC)` or `(HEX)` appears to the right of the address based on the protocol and device type. ・Decimal representation: `D00 (DEC)` ・Hexadecimal representation: `W00 (HEX)` This indicator shows whether the entered address is treated as decimal or hexadecimal.
Program	Select the register program number. Note: Available only when the manufacturer is Jtekt.
Data type	Select the data type of the register value to be acquired.
Byte swap	Set to ON when swapping the order of two bytes within 1WORD value. Note: This operation is only available when the data type is set to `STRING`.
Num array	Enter the range from which value is retrieved from among the consecutive registers starting from the input address.
Interval(ms)	Enter the interval for retrieving register values.
Enable	Set to ON to retrieve register values.

5.4.2.4 Test Connection

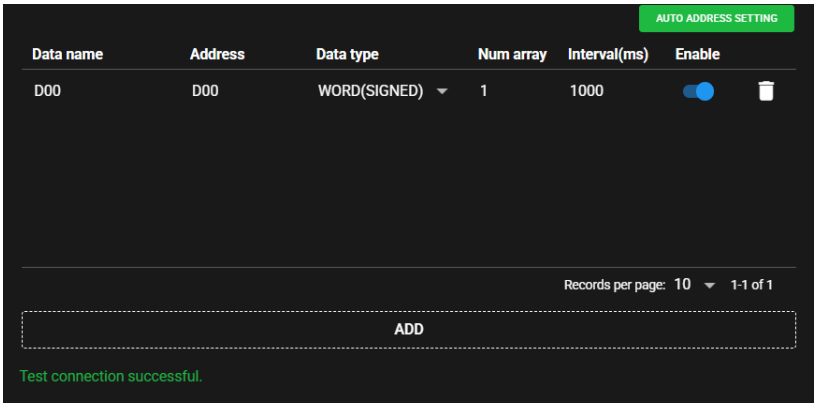
You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered PLC Collector.

2. Press [Test].



3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.4.2.5 Auto address setting

With some PLCs, you can set addresses automatically at once by reading the file output by the ladder editing tool.

1. In the ladder editing tool, output the created ladder as a file. The output file format differs depending on the PLC manufacturer and machine type. Sample files are provided below:

   - Mitsubishi
     - MELSEC-iQ-R series (QnA compatible 3E frame)
     - MELSEC-Q series (QnA compatible 3E frame)
     - MELSEC-F series (A compatible 1E frame)
     - MELSEC-Q series (QnA compatible 3C frame[serial])

     Download sample file
   - Omron
     - SYSMAC CS/CJ/CP/NSJ series (FINS)

     Download sample file
   - Keyence
     - KV-8000 series (QnA compatible 3E frame)

     Download sample file
     - KV-8000 series (Upper Link)

     Download sample file

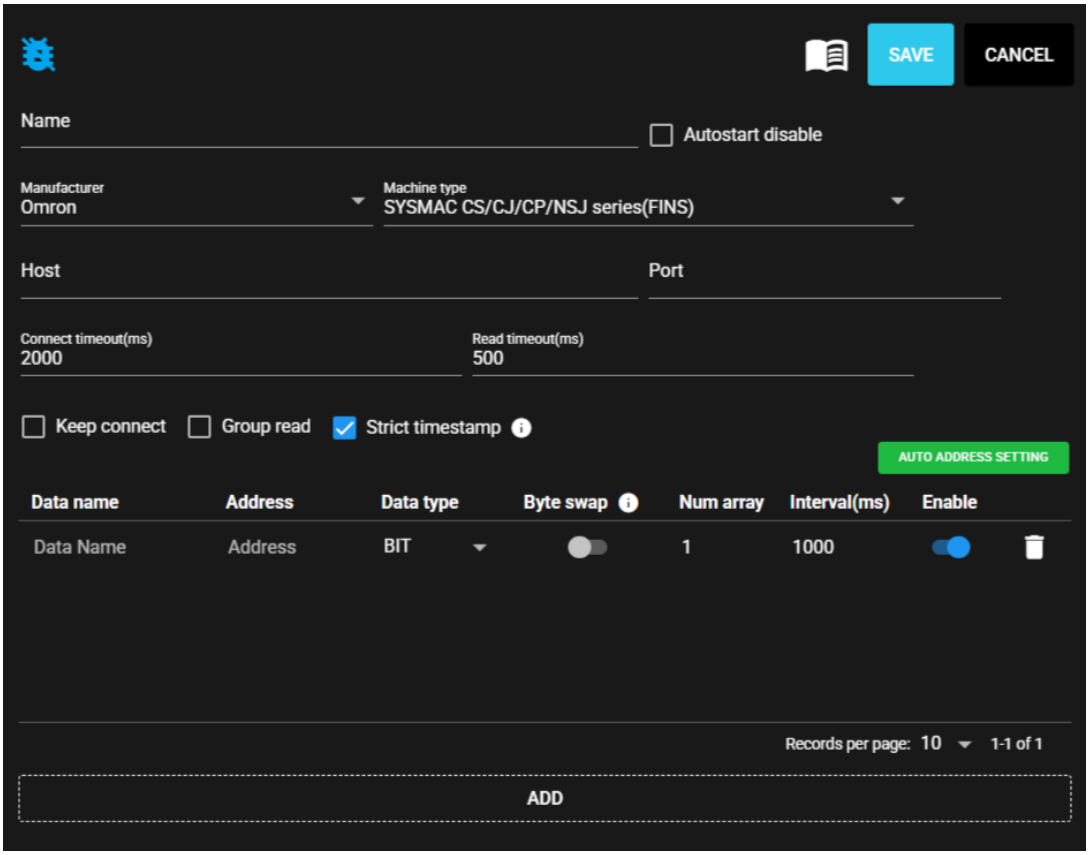
- Panasonic
  - FP7 Series (MEWTOCOL COM)
  - FP7 Series (MEWTOCOL COM FP2 Compatible)

  Download sample file
  - FP7 Series (QnA Compatible 3E Frame)

  Download sample file

> **About the Address Auto-Configuration File for Certain Machine type**
>
> The following models support the same format as the address auto-configuration file for Mitsubishi. However, this format cannot be directly output from each manufacturer's ladder editing software. Therefore, it is necessary to prepare a file in the same format separately to enable this functionality.
>
> - KV-8000 series (QnA compatible 3E frame)
> - FP7 Series (MEWTOCOL COM)
> - FP7 Series (MEWTOCOL COM FP2 Compatible)
> - FP7 Series (QnA Compatible 3E Frame)

2. Press [Auto address setting].

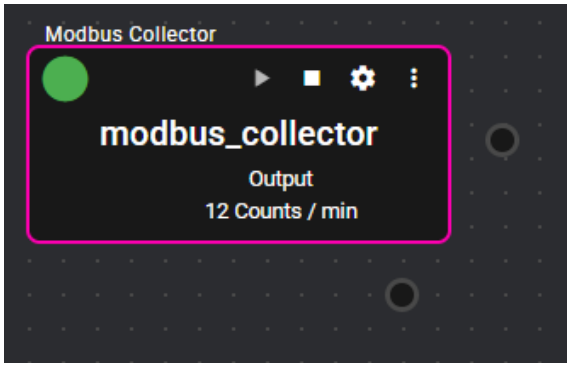


3. Select the file that you output with the ladder editing tool.

4. The address is automatically set based on the file.

5.4.3 Modbus Collector

This component collects data from Modbus compatible devices.
This component is the Modbus master and collects and registers data from Modbus slave devices.



This component supports three communication methods: TCP, RTU, and RTU over TCP.

Communication Method	Description
TCP	Ethernet communication based on Modbus TCP standard
RTU	Modbus RTU via serial communication such as RS-485
RTU over TCP	Modbus RTU via pass-through device (transferred over TCP) ※The performance and settings of the pass-through device significantly affect communication quality

5.4.3.1 Basic Information

The basic information of the Modbus Collector is as follows:

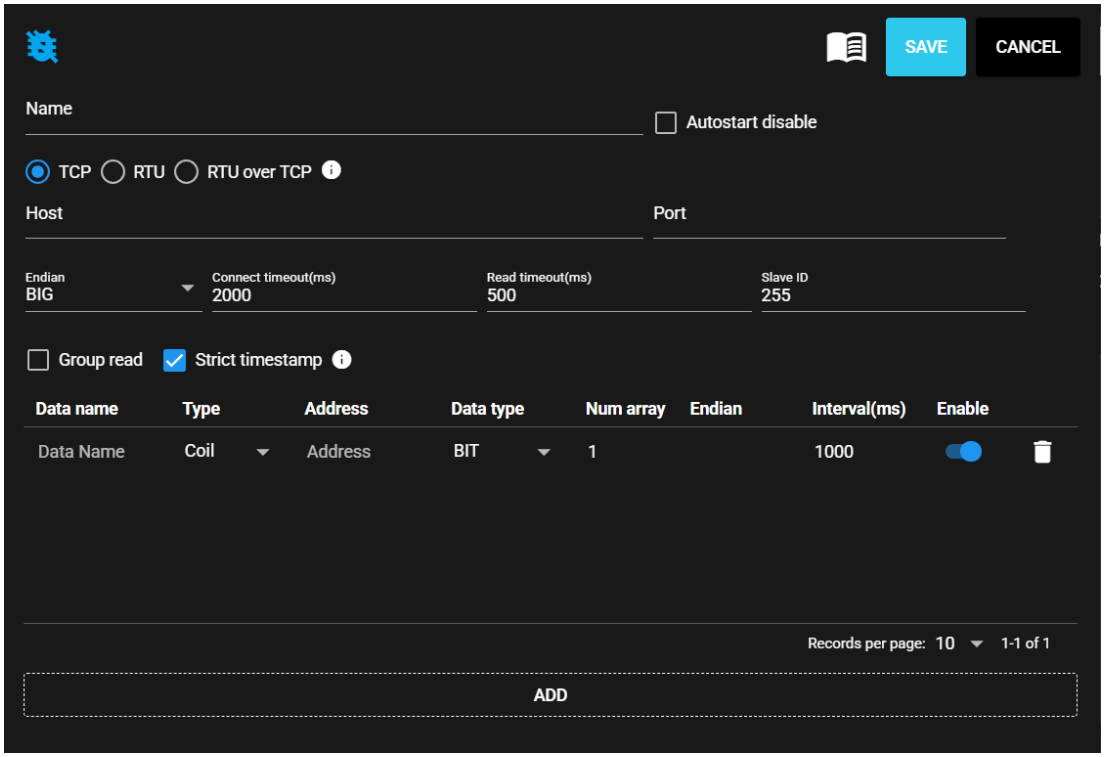
Item	Description
Base name	Modbus Collector
Description	Collects data from Modbus compatible devices.
Consumption points	2
Port	Input: 0

Item	Description
	Output: 1 Outputs data collected from Modbus compatible devices according to Settings. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	You can test connection with Modbus compatible devices. For more information, see Test Connection.

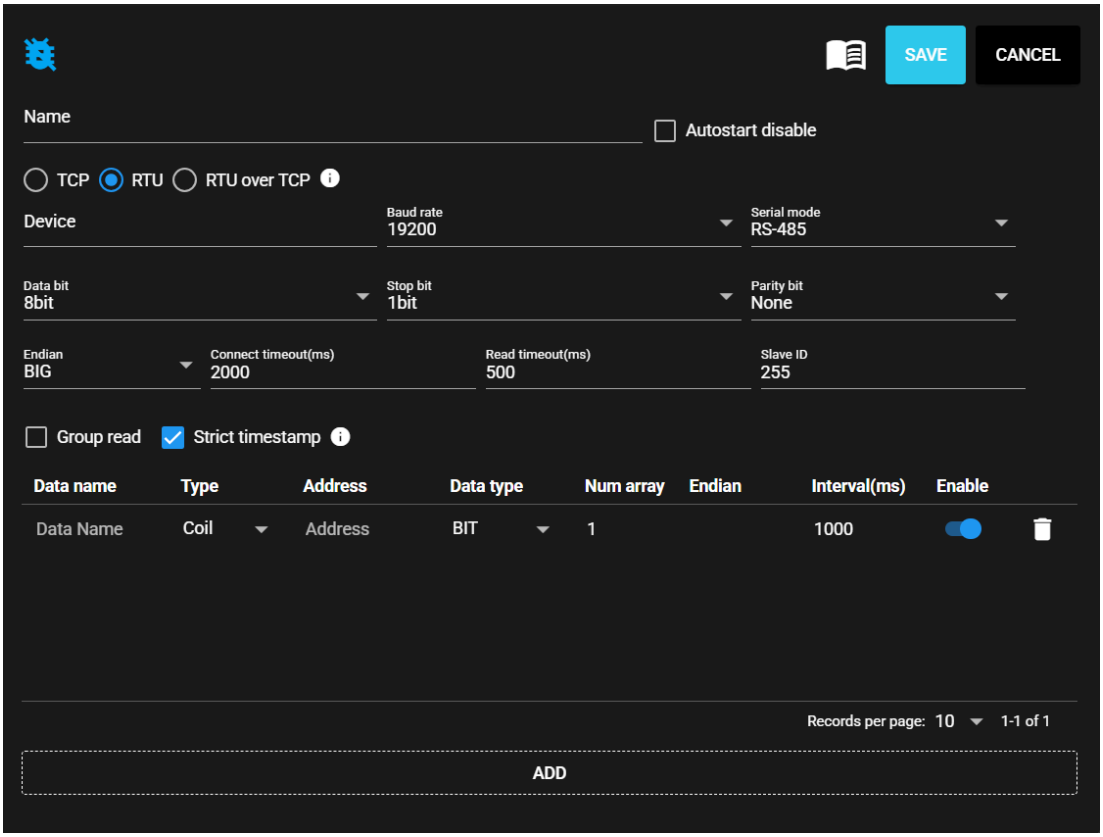
5.4.3.2 Settings

The Modbus Collector settings are as follows:

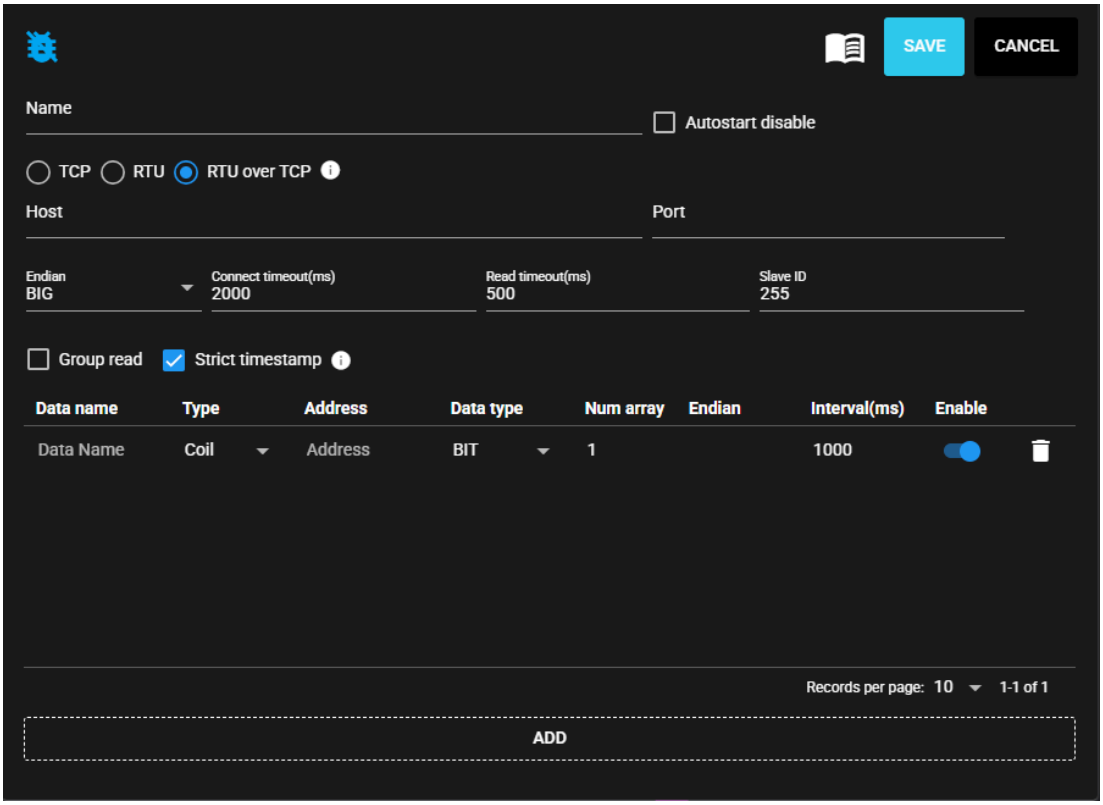
- When TCP is selected as the communication method



- When RTU is selected as the communication method



- When RTU over TCP is selected as the communication method



Communication Method

For Omron products, SpeeDBee Synapse does not support serial connections. Therefore, the following communication method cannot be selected.

- RTU

RTU over TCP Settings

With RTU over TCP, enter the pass-through unit connection information (host, port), and configure serial communication parameters such as communication speed and data bits on the pass-through unit side. Since the configuration screen and setting items differ depending on the pass-through unit product, please refer to the manual for each device.

※Verified pass-through units:

- ADAM-4571
- GW DEVICE SERVER 1E/1DB9

Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Communication method	Select TCP, RTU, or RTU over TCP
Host	Enter the host name or IP address of the connecting Modbus device. Note: Only available when the communication method is TCP or RTU over TCP.
Port	Enter the port for connecting to the Modbus device. Note: Only available when the communication method is TCP or RTU over TCP.
Device	Enter the serial communication device to use. Note: Only available when the communication method is RTU.
Baud rate	Select the baud rate for serial communication. Note: Only available when the communication method is RTU.
Serial mode	Select the serial communication mode. Note: Only available when the communication method is RTU.

Item	Description
Data bit	Select the data bit for serial communication. Note: Only available when the communication method is RTU.
Stop bit	Select the stop bit for serial communication. Note: Only available when the communication method is RTU.
Parity bit	Select the parity bit for serial communication. Note: Only available when the communication method is RTU.
Endian	Select a byte order.
Connect timeout(ms)	Enter the timeout duration for connecting.
Read timeout(ms)	Enter the timeout duration for reading.
Slave ID	Enter the Slave ID of the connecting device.
Group read	Set to ON to read multiple word registers at one time.
Strict timestamp	Set to ON to set a strict timestamp based on collected data.

- The Modbus Collector output data is defined by the following settings:

Item	Description
Data Name	Enter name of data.
Type	Select the type of register value.
Address	Enter the register address. Note: According to Modbus specifications, the address range is usually set to "1 to 65536," but when setting in SpeeDBee Synapse, it is necessary to set an address range shorter by 1, specifically, as "0 to 65535."
Data type	Select the data type of the register value to be acquired.
Num array	Enter the range from which value is retrieved from among the consecutive registers starting from the input address.

Item	Description
Endian	Select the byte order in register units.
Interval(ms)	Enter the interval for retrieving register values.
Enable	Set to ON to retrieve register values.

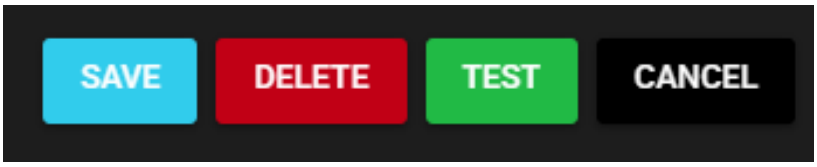
Hexadecimal Address Notation

Address can be specified in hexadecimal format by prefixing with `0x`.
Example: 0x64 (decimal 100), 0xffff (decimal 65535)

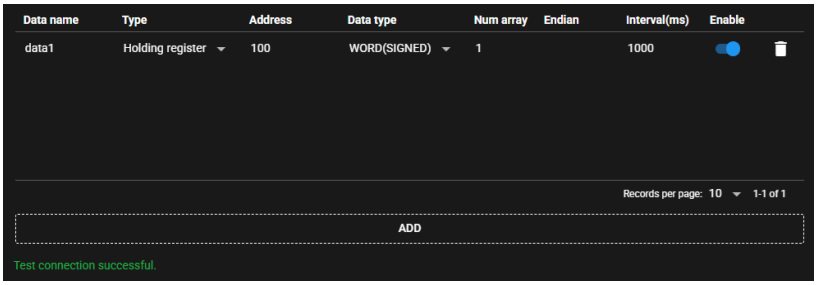
5.4.3.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered Modbus Collector.
2. Press [Test].

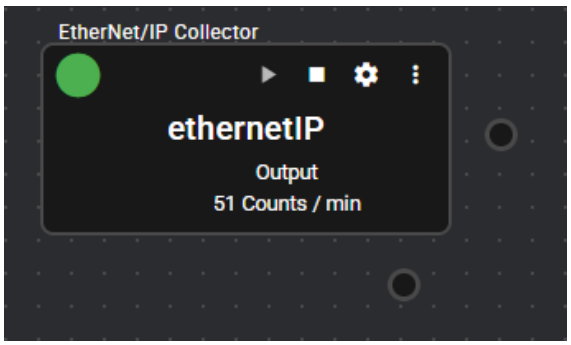


3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.4.4 EtherNet/IP Collector

This component collects data from EtherNet/IP-enabled devices.
This component acts as an originator and collects and registers data from Ethernet/IP-enabled devices.



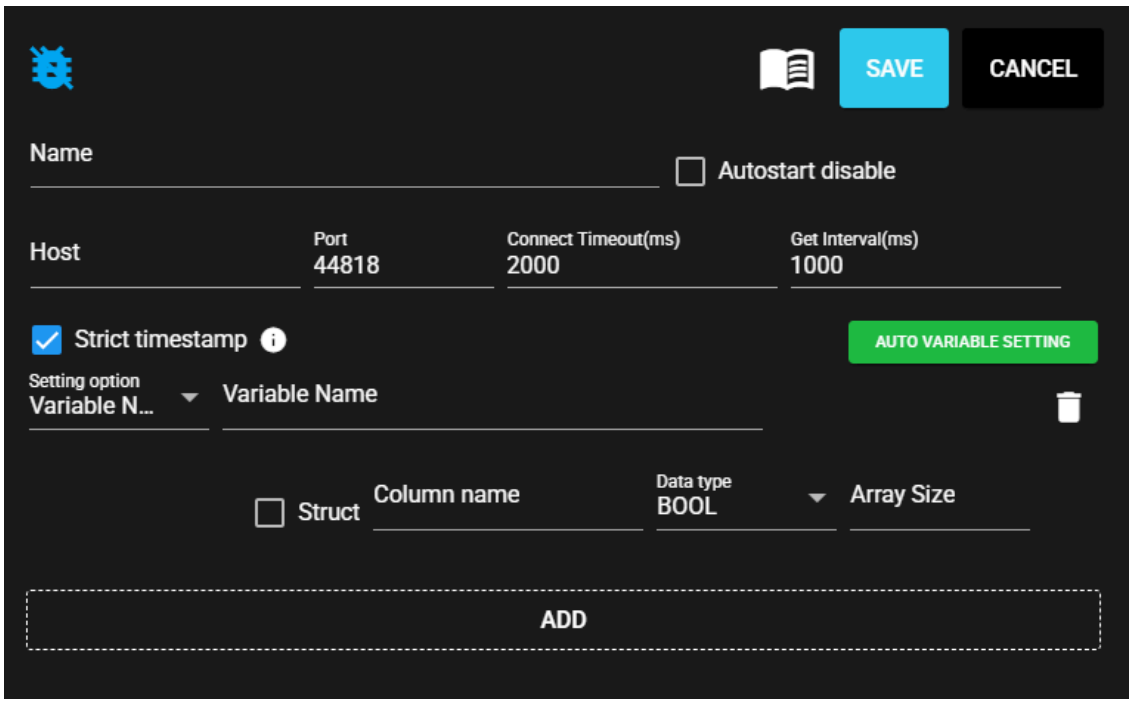
5.4.4.1 Basic Information

The basic information of the EtherNet/IP Collector is as follows:

Item	Description
Base name	EtherNet/IP Collector
Description	Collects data from EtherNet/IP-enabled devices.
Consumption points	2
Port	Input: 0 Output: 1 Outputs data collected from EtherNet/IP-enabled devices according to Settings. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	Variables can be set automatically. For more information, see Auto variable setting.

5.4.4.2 Settings

The EtherNet/IP Collector settings are as follows:



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Host	Enter the host name or IP address of the EtherNet/IP-enabled device.
Port	Enter the port for the EtherNet/IP-enabled device.
Connect timeout(ms)	Enter the timeout duration for connecting.
Get Interval(ms)	Enter the interval for retrieving data.
Strict timestamp	Set to ON to set a strict timestamp based on collected data.

- The EtherNet/IP Collector output data is defined by the following settings:

Item	Description
Setting Option	Select the setting option for the data to be acquired.

Item	Description
Variable Name	Enter a tag name. Note: This item can be entered only when the Setting Option is Variable Name.
Struct	Set to ON to register multiple members. Note: This item can be set only when the Setting Option is Variable Name.
Class ID	Enter the Class ID of CIP object. Note: This can be entered only when the Setting Option is CIPObject.
Instance ID	Enter the Instance ID of CIP object. Note: This can be entered only when the Setting Option is CIPObject.
Attribute ID	Enter the attribute ID of CIP object. Note: This can be entered only when the Setting Option is CIPObject.
Column name	Name of member
Data type	Data type of member
Array size	Array size of member

Hexadecimal ID Notation

Class ID, Instance ID, and Attribute ID can be specified in hexadecimal format by prefixing with `0x`.
Example: 0x64 (decimal 100), 0xffff (decimal 65535)

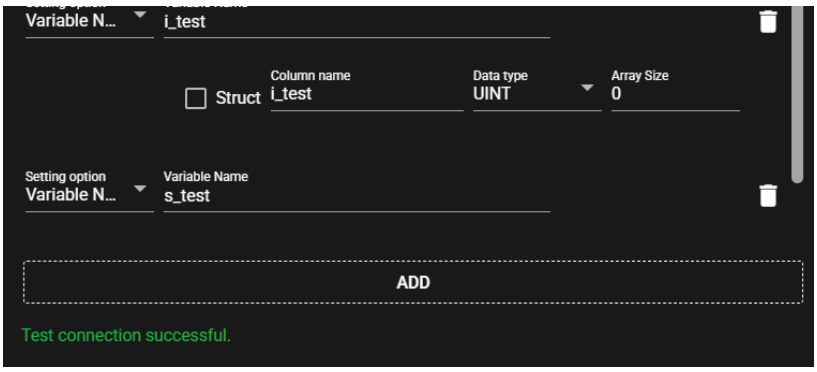
5.4.4.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered EtherNet/IP Collector.
2. Press [Test].



3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.4.4.4 Auto variable setting

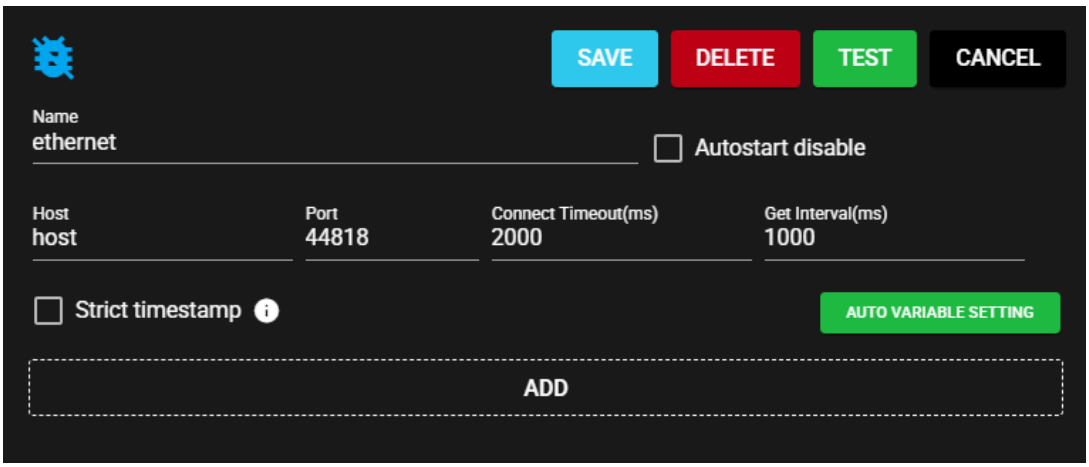
Variables can be automatically set by converting the global variable information exported by Sysmac Studio to a text file into a specified format and reading it.

1. Convert the global variable information exported by Sysmac Studio to a text file with the following column information:
   - HOST
   - NAME
   - DATATYPE
   - ADDRESS
   - COMMENT
   - TAGLINK
   - RW
   - POU

   Example

```
HOST    NAME    DATATYPE    ADDRESS COMMENT TAGLINK RW  POU
        b_test  BOOL        test1           TRUE    RW
        i_test  UINT        test2           TRUE    RW
        s_test  STRING(256) test3           TRUE    RW
```

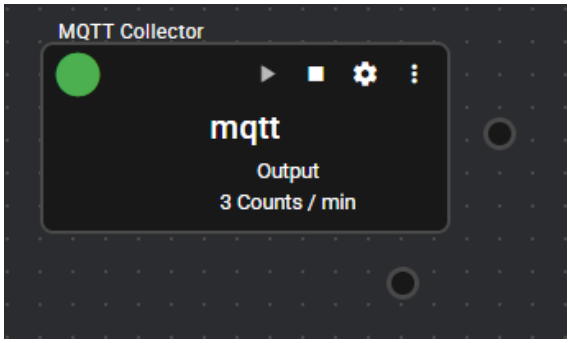
2. Press [Auto variable setting].



3. Select the prepared text file.
4. The variable is automatically set based on the file.

5.4.5 MQTT Collector

This component collects JSON data using the MQTT protocol.
This component subscribes to a specific topic name in MQTT. It parses the received data as JSON data and registers the content as collected data.
The format of JSON data that can be received is the same as that of JSON serializer output.
In addition to connecting with the regular MQTT broker, it can also connect with AWS IoT Core.



5.4.5.1 Basic Information

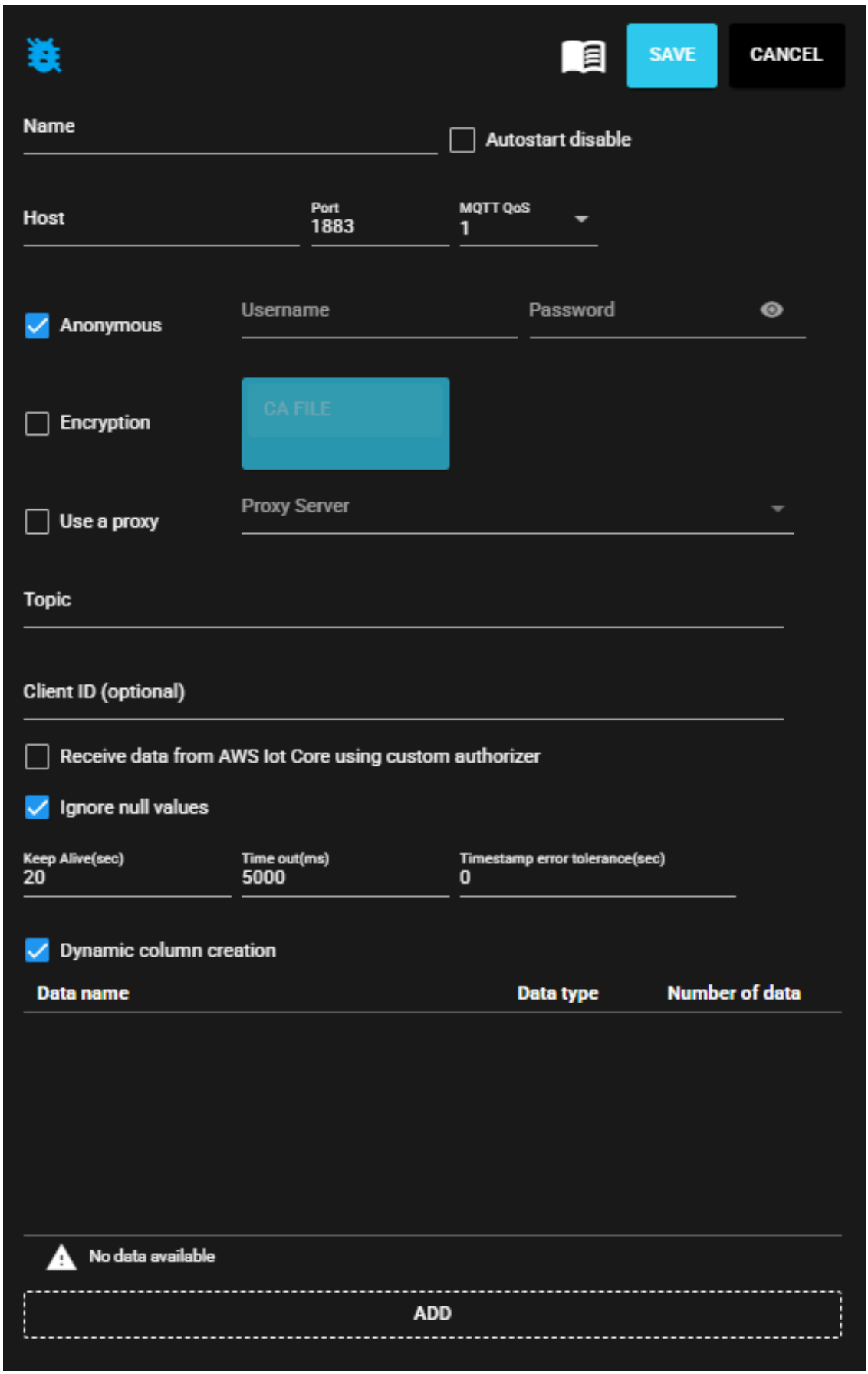
The basic information of the MQTT Collector is as follows:

Item	Description
Base name	MQTT Collector
Description	Data collection is performed using the MQTT protocol (Subscribe).

Item	Description
Consumption points	2
Port	Input: 0 Output: 1 Outputs the data received from the MQTT broker specified in Settings. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	You can test connection with the MQTT broker. For more information, see Test Connection.

5.4.5.2 Settings

The MQTT Collector settings are as follows:


SAVE
CANCEL
Name
Autostart disable
Host
Port
1883
MQTT QoS
1
Anonymous
Username
Password
Encryption
CA FILE
Use a proxy
Proxy Server
Topic
Client ID (optional)
Receive data from AWS Iot Core using custom authorizer
Ignore null values
Keep Alive(sec)
20
Time out(ms)
5000
Timestamp error tolerance(sec)
0
Dynamic column creation
Data name
Data type
Number of data
No data available
ADD

Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Host	Enter the host name or IP address of the MQTT broker.
Port	Enter the port for the MQTT broker.
MQTT Qos	Select the Quality of Service for MQTT.
Anonymous	Set to ON to disable user authentication.
User name	Enter the authenticated user name. Note: Can be entered only when Anonymous is disabled.
Password	Enter the authenticated password. Note: Can be entered only when Anonymous is disabled.
Encryption	Set to ON to enable encryption.
CA Certificate	Upload the CA Certificate to be used for encrypt communication. Note: You can only upload it when encryption is enabled.
Use a proxy	Set to ON to communicate with the MQTT broker via a proxy server.
Proxy server	Select a proxy server. Note: You must configure the server to be used in advance on the Proxy Settings screen.
Topic	Enter the topic for collection.
Client ID	MQTT Client ID Note: You may need to specify this item depending on the specifications of the connecting device. If omitted, a random string will be used. Note: If multiple components are connecting to the same MQTT broker, make sure that there is no duplication in the client IDs. Otherwise, communication will not be performed correctly.
Receive data from AWS Iot Core using custom	Set to ON if you want to use the AWS IoT Core custom authentication feature to receive data from AWS IoT Core.

Item	Description
authorizer	Note: You must have enabled custom authentication settings in AWS Iot Core. For more information, see here.
Ignore null values	This affects the behavior when the value in the received data is null. When this option is selected, NULL is ignored and no error occurs. When this option is not selected, error occurs when a NULL value is received.
Keep Alive(sec)	Enter MQTT keepalive interval.
Time out(ms)	Enter timeout seconds for MQTT broker connection.
Timestamp error tolerance(sec)	Enter timestamp error tolerance.
Dynamic column creation	Set to ON to dynamically create columns from data received from the MQTT broker. Note: 1 Columns are generated only after data is received from the MQTT broker. Note: 2 Dynamically generated columns disappear when the component stops. If you want to use the data, register it as a fixed column.
Data Name	Enter the name of fixed column.
Data type	Select the data type of fixed column.
Number of data	Enter the array size of the fixed column.

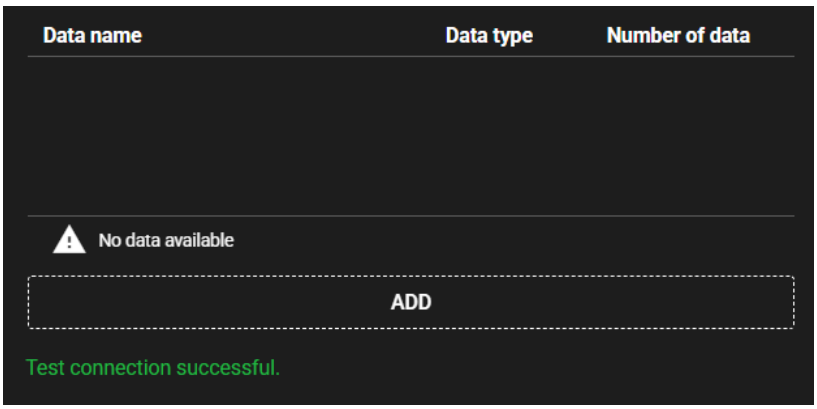
5.4.5.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered MQTT Collector.
2. Press [Test].



3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



Precautions for MQTT Collector Test Connection

Test connection is only available in the following cases:

- Component has stopped.
- Component status is Error.

Note: For more information on status, see Status List.

5.4.5.4 Format of received data

The format of the data received by MQTT Collector is compatible with the JSON serializer. Check the output data items of the JSON serializer.

Example of Data Transmission Using mosquitto

To receive data with the MQTT Collector, the key must be specified as "records" because the JSON "records" contains the data.
For example, to publish data to a topic using mosquitto_pub, the following command is necessary:

```
    $ mosquitto_pub -d -h 192.168.1.10 -p 1883 -t "topic" -m '{"records": [{"data1":1,
"data2":"Hello,World"}, {"data3":[111,222,333], "data4":3.14}]}'
```

Data cannot be received if you don't have "records" in the first level of json. Anything other than "records" is ignored.

5.4.5.5 Receiving Data from Cloud Service

The MQTT Collector can receive data from the following cloud services. See the following notes for each:

AWS IoT Core

In order for the MQTT Collector to receive data sent to AWS IoT Core,
you must use the “Receive data from AWS IoT Core using custom authorizer” option.
You also need to set it up on the cloud side.
For more information, see Custom Authentication and Authorization on the official AWS page.
This information is current as of June 2024.

Azure IoT Hub

To receive data from Azure IoT Hub, you need a setup to meet the IoT Hub specifications. The MQTT Collector can receive data sent as “cloud-to-device.”
The MQTT Collector cannot receive data sent to IoT Hub from other devices.
Data cannot be broadcast to multiple devices in a cloud-to-device setup.
This information is current as of June 2024. For more information, please check the official pages below:

- Communicating with IoT Hub using MQTT protocol
- Sending cloud-to-device messages from IoT Hub
- Sending messages to devices from the cloud using IoT Hub (Python)
- Controlling access to IoT Hub using Shared Access Signature (SAS token)

A summary of the settings is provided below. Be sure to check the above official pages for a complete setup.

- Your IoT Hub must be Standard tier. The IoT Hub cannot be used with Basic or Free tier.
- Host: {IoT Hub name}.azure-devices.net
- Port: 8883
- User name: {IoT Hub name}.azure-devices.net/{Device Id}/?api-version=2021-04-12
- Password: SAS token
- Topic: devices/{Device Id}/messages/events/"
- Client ID: {Device Id}
- Encryption: Enabled
- CA Certificate: DigiCert root certificate (DigiCert Global Root G2)

The MQTT broker in Azure Event Grid is not currently supported.

5.4.5.6 Receiving Data from Various Emitters

Data generated by a JSON serializer can be sent to various emitters and received by other MQTT Collectors.

Sender	Receiver	Yes/No
MQTT Emitter	MQTT Collector	○ Note: Connection with the same MQTT broker must be set up.
Cloud Emitter for AWS	MQTT Collector	○ Note: The MQTT Collector must have a custom authentication setup.
Cloud Emitter for Azure	MQTT Collector	× Note: Data for devices cannot be received by the MQTT Collector.

Precautions for Receiving Data from Other Emitters

When receiving data from various emitters, note the following:

1. When sending data by the emitter one set at a time:

Example:
Switch the data of the sending JSON serializer at one set.
Set the receiving MQTT Collector error tolerance (in seconds) to 10.

In this case, if the data can be received within 10 seconds from data registration, the data is registered with the original timestamp. However, keep in mind that data is sent to the cloud frequently.

2. When sending aggregated values by the emitter:

Example:
Switch the data of the sending JSON serializer at one set.
The sending JSON serializer takes the average of data aggregated at the input port, with a window width of 60 seconds.
Set the receiving MQTT Collector error tolerance (in seconds) to 10.

In this case, the average of the values is registered and sent at the emitter, with the specified width. The Collector receives this and registers the data with the original timestamp. The load on the network is small because not all data is sent.

3. When the sender stores data for a specified time and sends it by the emitter:

Example:
Switch the data of the sending JSON serializer in 60 seconds.
Set the receiving MQTT Collector error tolerance (in seconds) to 0.

In this case, the data is registered with the current time at which the entire data was received. Note that data is not registered with the original timestamp.

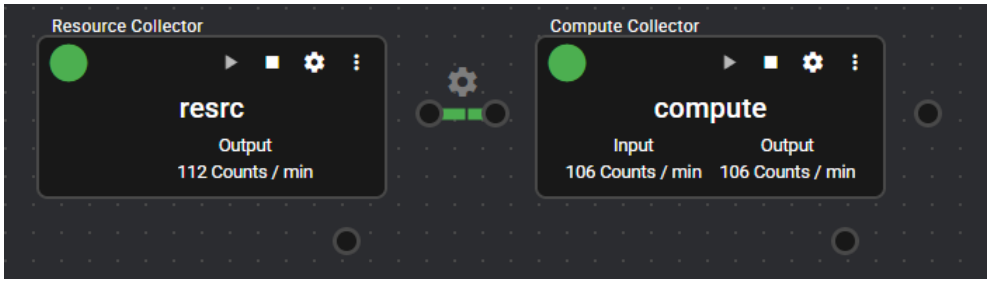
4. When the sender collects data for a specified time and sends it by the emitter (with the timestamp registered without change):

To achieve this, you need at least the following setup:
Example:
Switch the data of the sending JSON serializer in 10 to 30 seconds.
Set the receiving MQTT Collector error tolerance (in seconds) to 60.
Set the delay time to 61000 (milliseconds) on the input port of the component connected to the MQTT Collector.

This setting allows data to be registered if the data timestamp sent falls within 60 seconds. However, check the transmission and reception and make sure that there is no problem with the performance, sufficiently verifying that the received data is being processed as expected.

5.4.6 Compute Collector

This component performs calculations with any expression.
It receives data output by another component from the input port, and outputs as new data the result of calculating the data by applying any calculation formula.
You can use arithmetic and logical operations, as well as a variety of other functions, in the calculation formula.



5.4.6.1 Basic Information

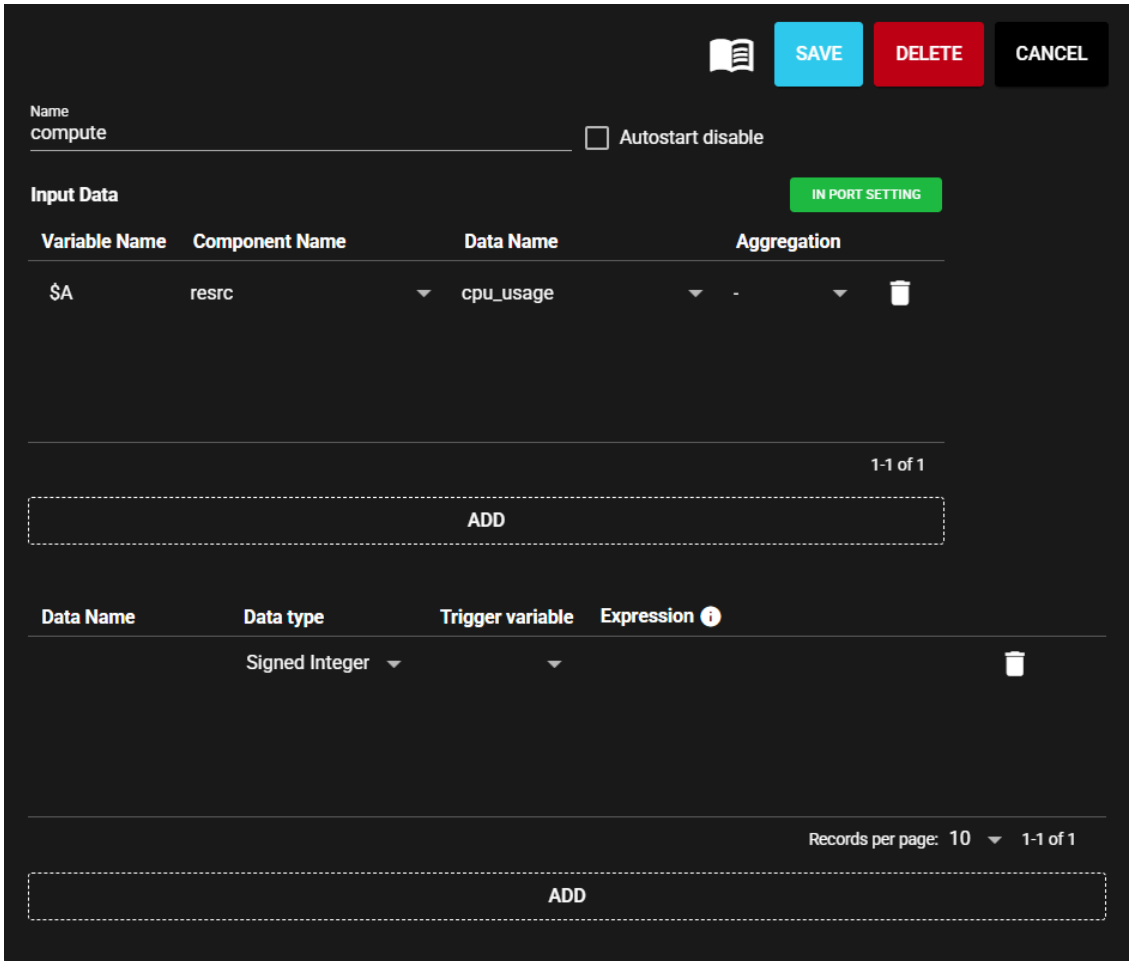
The basic information of the Compute Collector is as follows:

Item	Description
Base name	Compute Collector
Description	Receives data from other collectors, performs calculations, and treats it as new data.
Consumption points	2
Port	Input: 1 Receives data from other components.

Item	Description
	Output: 1 Outputs the calcuation result according to the Settings. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	This component can open the input port settings window and set up aggregation of data received from other components. For more information, see Import setting.

5.4.6.2 Settings

The Compute Collector settings are as follows:



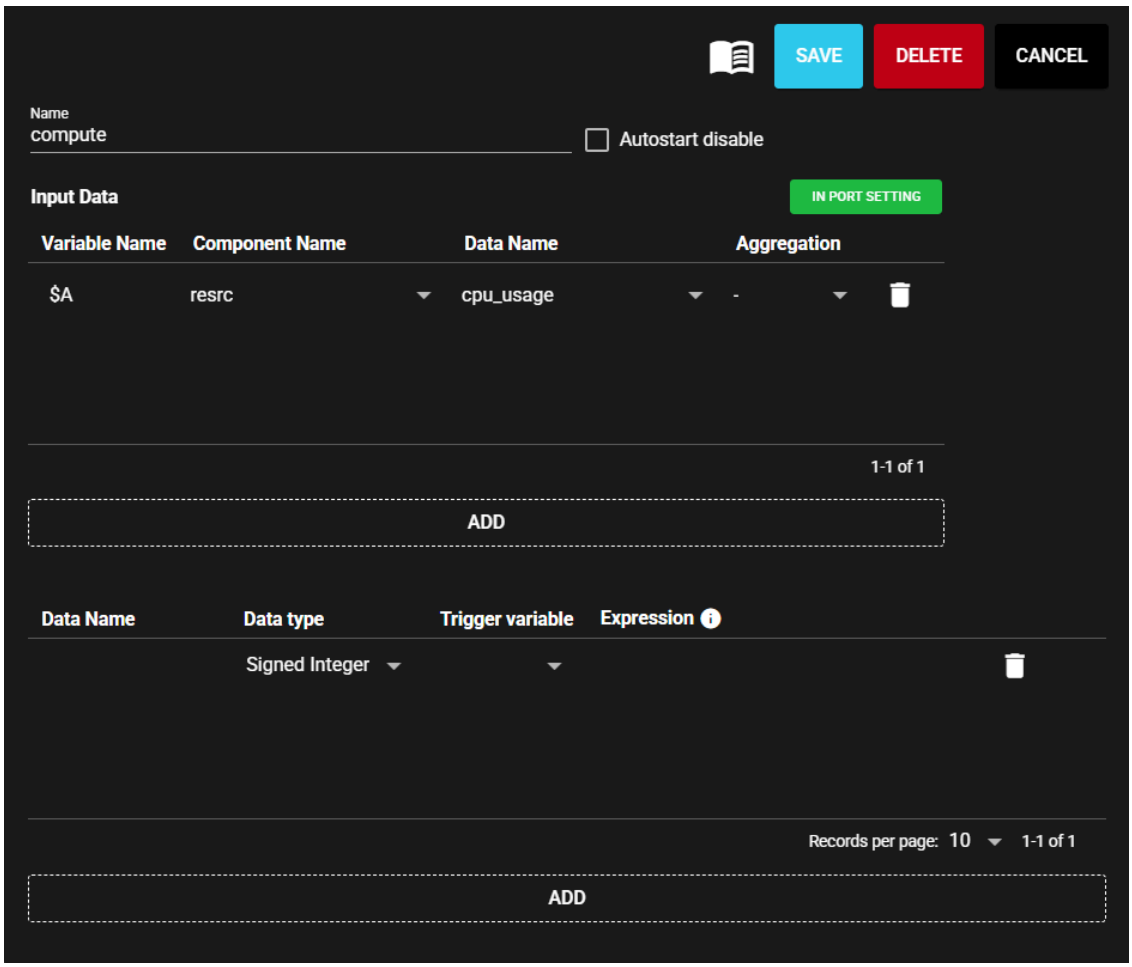
Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Variable Name	Name automatically assigned to the input data
Component Name	Select a component from the input port data list.
Data Name	Select data from the data list for the component selected above.
Aggregation	Select aggregate value. Note: Available only if you have applied the aggregation option to the component's data. For more information on aggregation, see IN Port.

- The Compute Collector output data is defined by the following settings:

Item	Description
Data Name	Enter name of data.
Data type	Select a data type.
Trigger variable	Variable that triggers the execution of calculation The variable set in the input data can be selected. Calculation is executed once the selected variable is updated.
Expression	Enter the expression to be executed by the Collector. Note: Click the icon to the right of "Expression" in the title to display an example and a list of elements that can be used in the expression.

5.4.6.3 Import setting

For added Compute Collectors, you can open the Compute Collector input port settings window by pressing "Import setting" on the setting screen. In the input port settings window, you can set up aggregation of data received from other components. For more information, see IN Port.

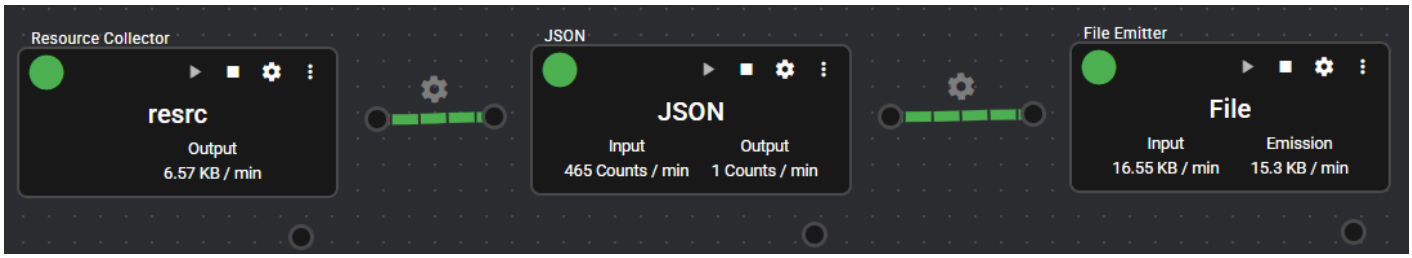

SAVE
DELETE
CANCEL
Name
compute
Autostart disable
Input Data
IN PORT SETTING
Variable Name
Component Name
Data Name
Aggregation
$A
resrc
cpu_usage
-
1-1 of 1
ADD
Data Name
Data type
Trigger variable
Expression
Signed Integer
Records per page: 10
1-1 of 1
ADD

5.5 Serializer

This section explains each serializer component.

5.5.1 JSON Serializer

This component processes specified data into JSON format.
It receives data from other components connected to the input port, processes it into JSON format with "component name:data name" as the key at a certain time/number of times, and outputs it.



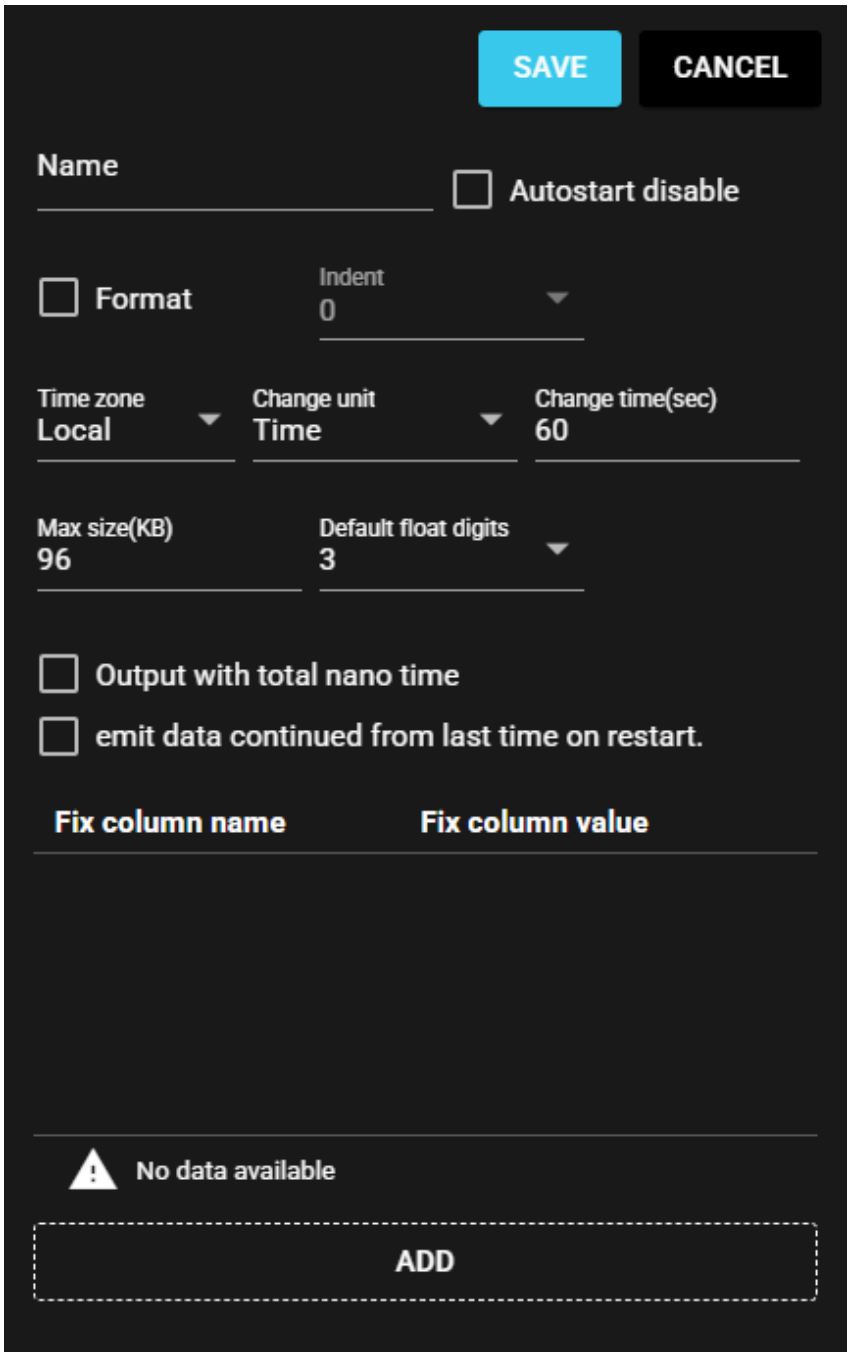
5.5.1.1 Basic Information

The basic information on the JSON serializer is as follows:

Item	Description
Base name	JSON
Description	Receives data from a component, processes it into a fixed JSON format, and passes it to another component.
Consumption point	0
Port	Input: 1 Receives data from other components. Output: 1 Outputs a JSON file created by processing input data. For more information on output data, see Output Data Items. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.5.1.2 Configuration Items

The JSON serializer configuration items are as follows:



Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

Item	Description
Format	Select ON to format JSON.
Indent	You can specify the indentation to be given to JSON. Note: You can input when you select Format ON.
Time zone	You can select the time zone for the timestamps given to the data in JSON.
Change unit	You can select the criteria for switching JSON files.
Change time(sec)	You can input how many seconds of data to write to a single JSON file. Note: You can input when the change unit is time.
Change count	You can input how many counts of data to write in a single JSON file. Note: You can input when the change unit is a count.
Change window	You can input how many window width units of data to write in a single JSON file. Note: You can input this only when the change unit is a window. Note: A window is a unit in which input data is grouped over a fixed period of time. The window width can be changed by enabling aggregation on the input port.
Max size(KB)	You can input the maximum size for a single JSON file.
Default float digits	You can set the number of digits to be emitted when the data is a floating decimal number. Used as the default value when the number of decimal digits is not specified for each data.
Output with total nano time	You can add total nanoseconds to the data.
After the start, data is output continuing from where the previous emission left off.	When the component is stopped and restarted, data is output continuing from where the previous emission left off.

You can optionally output fix values to a JSON file.

Item	Description
Fix column name	The column name of the fix value
Value	The value to be output as a fix value

5.5.1.3 Output Data Items

The JSON serializer outputs the following data:

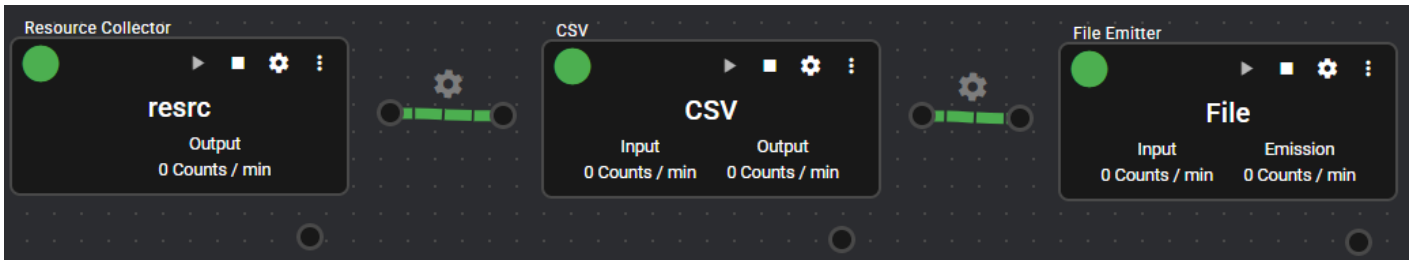
Item	Column Name	Data Type	Description
JSON Data Output	output file	FILE	This column outputs the created JSON file. The media type is fixed to "application/json".

The following is an example of an output file generated after receiving I/O data from the Resource Collector. "records" will contain the data. "timestamp" is the date and time the data was generated. "resrc: \~~" is the I/O data received from the Resource Collector named "resrc".

```
{
  "records": [
    {
      "timestamp": "2023-10-02T05:54:54.837742200Z",
      "resrc:IOPS": 0,
      "resrc:I_O": "sdb",
      "resrc:kb_r_s": 0,
      "resrc:kb_w_s": 0
    },
    {
      "timestamp": "2023-10-02T05:54:55.837652700Z",
      "resrc:IOPS": 0,
      "resrc:I_O": "sdb",
      "resrc:kb_r_s": 0,
      "resrc:kb_w_s": 0
    },
    ~~~
  ]
}
```

5.5.2 CSV Serializer

This component processes specified data into CSV format.
It receives data from other components connected to the input port, processes it into CSV format at a certain time/number of times, and outputs it.
You can change the sort order of the CSV files in the input port configuration.



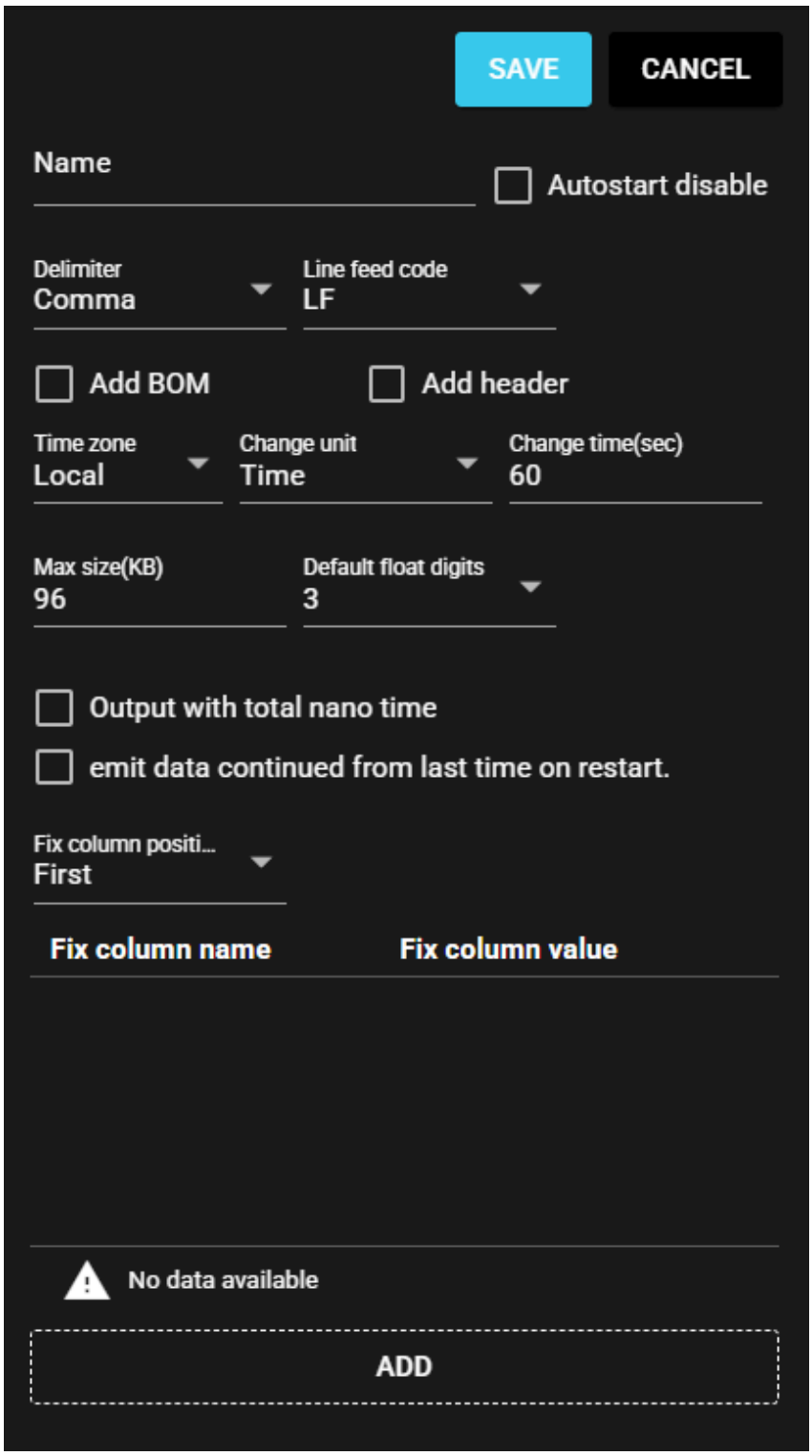
5.5.2.1 Basic Information

The basic information on the CSV serializer is as follows:

Item	Description
Base name	CSV
Description	Receives data from a component, processes it into CSV format, and passes it to another component.
Consumption point	0
Port	Input: 1 Receives data from other components. Output: 1 Outputs a CSV file created by processing input data. For more information on output data, see Output Data Items. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.5.2.2 Configuration Items

The CSV serializer configuration items are as follows:



Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

Item	Description
Delimiter	You can select the delimiter for the CSV file.
Line feed code	You can select the line feed code for the CSV file.
Add BOM	Select ON to add a file BOM to the CSV file.
Add header	Select ON to add a header to the CSV file.
Time zone	You can select the time zone for the timestamps given to the CSV data.
Change unit	You can select the criteria for switching CSV files.
Change time(sec)	You can input how many seconds of data to write to a single CSV file. Note: You can input when the change unit is time.
Change count	You can input how many counts of data to write in a single CSV file. Note: You can input when the change unit is a count.
Change window	You can input how many window width units of data to write in a single CSV file. Note: You can input this only when the change unit is a window. Note: A window is a unit in which input data is grouped over a fixed period of time. The window width can be changed by enabling aggregation on the input port.
Max size(KB)	You can input the maximum size for a single JSON file.
Default float digits	You can set the number of digits to be emitted when the data is a floating decimal number. Used as the default value when the number of decimal digits is not specified for each data.
Output with total nano time	You can add total nanoseconds to the data.
After the start, data is output continuing from where the previous emission left off.	When the component is stopped and restarted, data is output continuing from where the previous emission left off.
Fix column position	You can select the position of the fix value column in the CSV file.

You can optionally output fix values to a CSV file.

Item	Description
Fix column name	The column name of the fix value
Value	The value to be output as a fix value

5.5.2.3 Output Data Items

The CSV serializer outputs the following data:

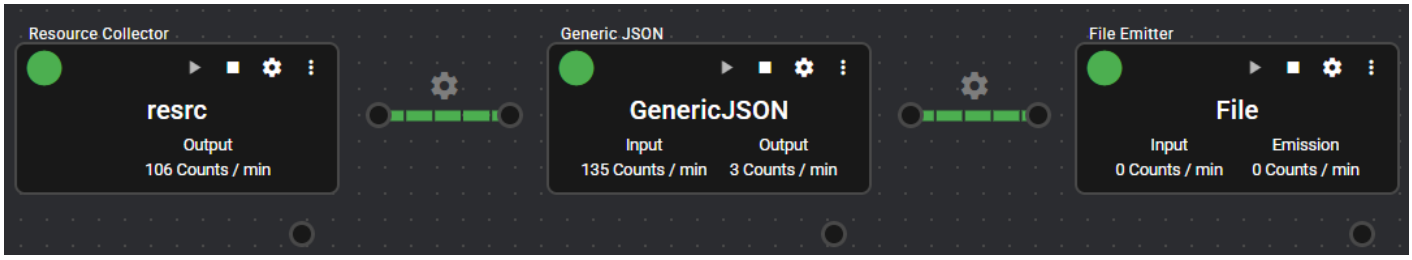
Item	Column Name	Data Type	Description
CSV Data Output	output file	FILE	This column outputs the created CSV file. The media type is fixed to "text/csv".

The following is an example of an output file generated after receiving I/O data from the Resource Collector. The leftmost column is the data timestamp. The remaining columns are the I/O data received from the Resource Collector.

```
2023-10-02T06:19:52.837670200Z,0.000,"sdb",0.000,0.000
2023-10-02T06:19:53.837656800Z,0.000,"sdb",0.000,0.000
2023-10-02T06:19:54.837805700Z,0.000,"sdb",0.000,0.000
2023-10-02T06:19:55.837734700Z,0.000,"sdb",0.000,0.000
~~~
```

5.5.3 Generic JSON Serializer

This component processes specified data into JSON format.
It receives data from other components connected to the input port, processes it into JSON format at a certain time/number of times, and outputs it. You can select a JSON format from presets or specify any format.



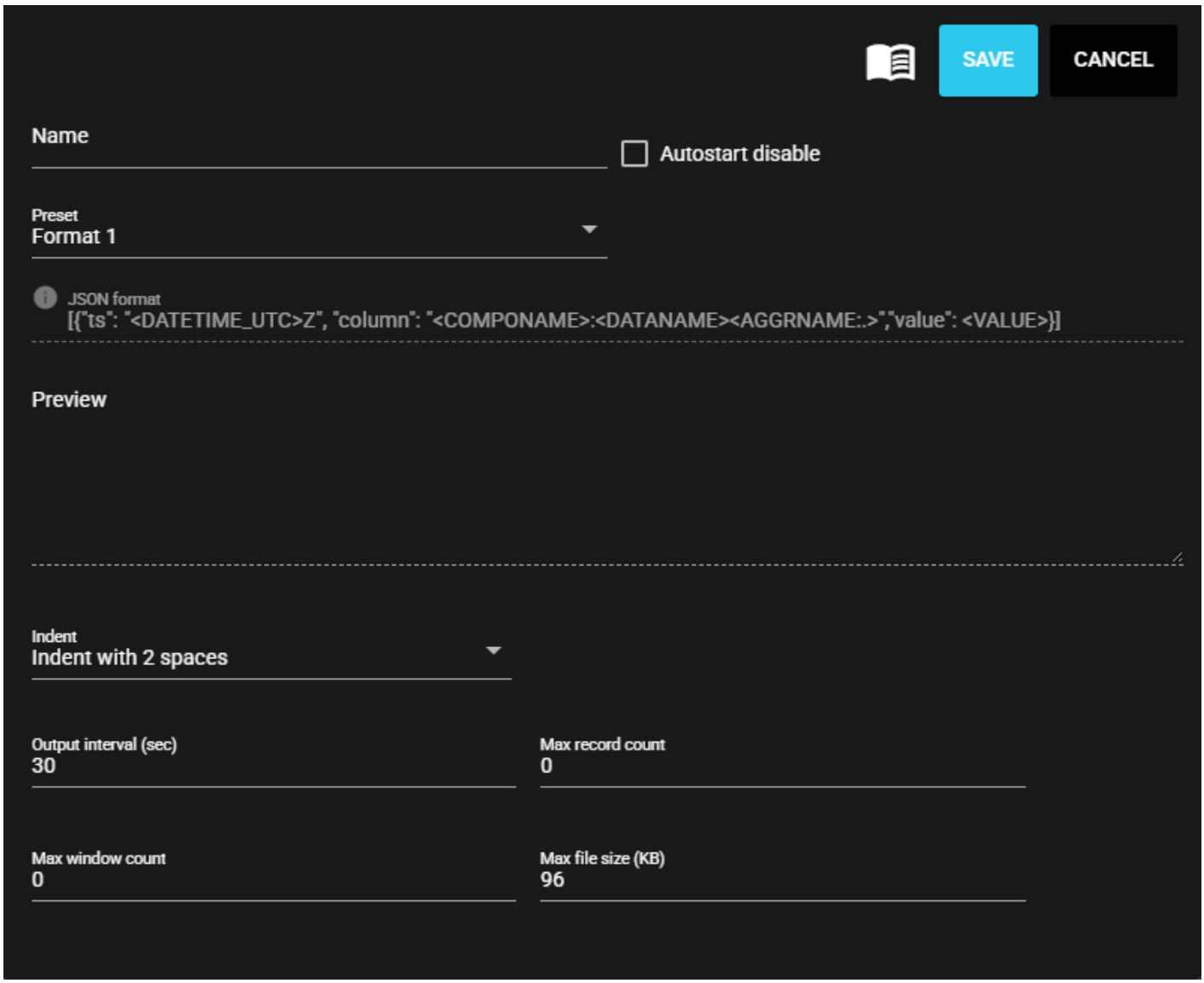
5.5.3.1 Basic Information

The basic information on the Generic JSON serializer is as follows:

Item	Description
Base name	Generic JSON
Description	Receives data from a component, processes it into any JSON format, and passes it to another component.
Consumption point	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs a JSON file created by processing input data. For more information on output data, see Output Data Items. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.5.3.2 Configuration Items

The Generic JSON serializer configuration items are as follows:



Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Preset	You can select JSON format.
JSON format	Format selected in the preset. Note: You can edit when you select Any format. Note: Use Replacement string to apply the corresponding information. For more information, see Available Replacement string for Generic JSON.
Preview	Display JSON output preview.

Item	Description
Indent	You can specify the indentation to be given to JSON.
Output interval (sec)	You can input how many seconds of data to write to a single JSON file.
Max record count	You can input how many records of data to write in a single JSON file.
Max window count	You can input how many windows of data to write in a single JSON file.
Max file size (KB)	You can input the maximum size for a single JSON file.

5.5.3.2.1 Available Replacement string for Generic JSON

Replacement string	Quotation marks	Description	Example of converted string
<COMPONAME>	Required	Source component name of data	plc-collector
<DATANAME>	Required	Data name	D110
<DATATYPE>	Required	Data type	double
<AGGRNAME>	Required	Aggregate method name (empty if not an aggregate)	count
<AGGRNAME:.>	Required	Prefix + Aggregate method name (empty if not an aggregate)	.count
<VALUE>	Unavailable	Value	123.45
<DATETIME_UTC>	Required	Date of data timestamp（UTC）	2024-11-06T00:00:00.000000
<DATETIME_UTC:format>	Required	Date of data timestamp（UTC） with format	2024/11/06 00:00:00

Replacement string	Quotation marks	Description	Example of converted string
<DATETIME_LOCAL>	Required	Date of data timestamp（Local Time Zone）	2024-11-06T00:00:00.000000
<DATETIME_LOCAL:format>	Required	Date of data timestamp（Local Time Zone） with format	2024/11/06 00:00:00
<TS>	Available	Data timestamp（in seconds）	1733356200
<TS:n>	Available	Data timestamp（in seconds） display up to n decimal places (max n: 9)	1733356200.123
<TS_MS>	Available	Data timestamp（in milliseconds）	1733356200123
<TS_MS:n>	Available	Data timestamp（in milliseconds） display up to n decimal places (max n: 9)	1733356200123.456
<TS_US>	Available	Data timestamp（in microseconds）	1733356200123456
<TS_US:n>	Available	Data timestamp（in microseconds） display up to n decimal places (max n: 9)	1733356200123456.789
<TS_NS>	Available	Data timestamp（in nanoseconds）	1733356200123456789

Description of Replacement string

- If quotation marks are permitted, please enclose the replacement string within quotation marks. Example "<COMPONAME>"
- The format can use strftime-compatible specifiers. Example "<DATETIME_UTC:%Y/%m/%d %H:%M:%S.%f>" → "2024/11/06 00:00:00.000000"

5.5.3.3 Output Data Items

The Generic JSON serializer outputs the following data:

Item	Column Name	Data Type	Description
JSON Data Output	output file	FILE	This column outputs the created JSON file. The media type is fixed to "application/json".

The following is an example of an output file generated after receiving I/O data from the Resource Collector. The selected preset is “Format 1”, and the selected indentation is “Indent with 2 spaces”. "<DATETIME_UTC>" is replaced with the data timestamp. "<COMPONAME><DATANAME><AGGRNAME:.>" are replaced with the Resource collector name, data name, and aggregate method name, respectively. "<VALUE>" is replaced with the I/O data received from the Resource collector.

Format 1

```
[{"ts": "<DATETIME_UTC>Z", "column": "<COMPONAME>:<DATANAME><AGGRNAME:.>","value": <VALUE>}]
```

output file

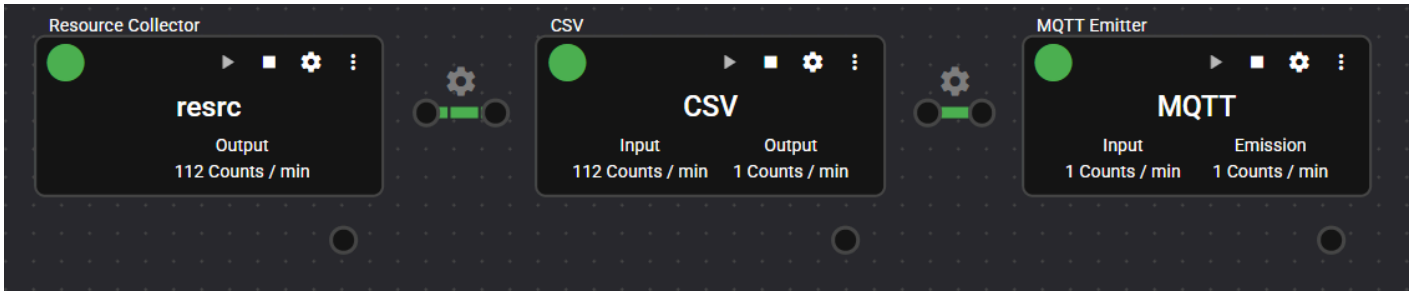
```
[
  {
    "ts": "2023-10-02T05:54:54.837742Z",
    "column": "resrc:IOPS",
    "value": 0
  },
  {
    "ts": "2023-10-02T05:54:54.837742Z",
    "column": "resrc:I_O",
    "value": "sdb"
  },
  {
    "ts": "2023-10-02T05:54:54.837742Z",
    "column": "resrc:kb_r_s",
    "value": 0
  },
  {
    "ts": "2023-10-02T05:54:54.837742Z",
    "column": "resrc:kb_w_s",
    "value": 0
  },
]
```

5.6 Emitter

This section describes emitter components.

5.6.1 MQTT Emitter

This component emits any given data files (publish) using the MQTT protocol.
It receives data from various serializers connected to the IN port and emits it.



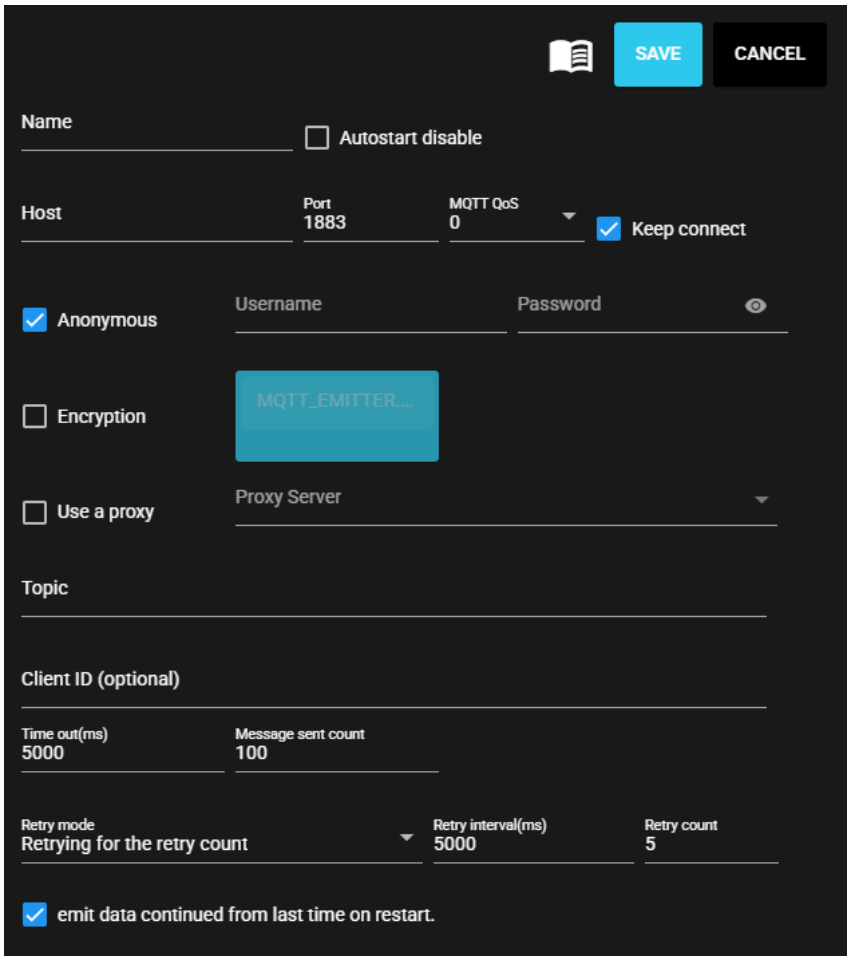
5.6.1.1 Basic Information

The basic information of the MQTT Emitter is as follows.

Item	Description
Base name	MQTT Emitter
Description	This emits data using the MQTT protocol (publish).
Consumption point	2
Port	Input: 1 This receives data in file format from the serializer component. The received data is emitted using the MQTT protocol (publish). Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	Test connection to the MQTT broker can be performed. For detail, see Test Connection.

5.6.1.2 Configuration item

The configuration items for the MQTT Emitter is as follows.



Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Host	Specify the host name or IP address of the MQTT broker.
Port	Specify MQTT broker port number.
MQTT Qos	Select the MQTT QoS level to use (0: No reception confirmation, 1: Reception confirmed, 2: Reception confirmed with duplicate suppression)
Keep connect	Do not disconnect server connection after each emission.

Item	Description
Anonymous	Select ON to access the broker anonymously.
Username	Input a username if you are not accessing the broker anonymously.
Password	Input a password if you are not accessing the broker anonymously.
Encryption	Select ON to perform encryption.
CA certificate	Upload a CA certificate. Note: Uploadable only when encryption is ON.
Use a proxy	Set to ON to communicate with the MQTT broker via a proxy server.
Proxy server	Select a proxy server. Note: You must configure the server to be used in advance on the Proxy Settings screen.
Topic	Topics to be assigned when publishing
Client ID	MQTT Client ID Note: This may need to be specified depending on the connection destination's specifications. If omitted, a random string will be used. Note: When connecting multiple components to the same MQTT broker, ensure each client ID is unique and not duplicated. Otherwise, communication will not function correctly.
Time out (ms)	Input the timeout duration for the emission process in milliseconds.
Message sent count	Maximum number of emissions in a single event
Retry mode	Select the behavior when retrying after an emission failure.
Retry interval (ms)	Input the emission retry interval.
Retry count	Input the emission retry count. Note: This can be input only when the retry mode is "Retrying for the retry count".

Item	Description
After the start, data is output continuing from where the previous emission left off.	When the component is stopped and restarted, data is output continuing from where the previous emission left off.

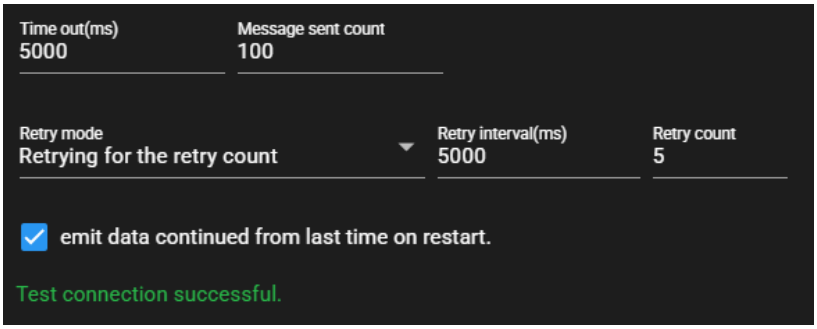
5.6.1.3 Test connection

A test connection to the configured destination can be performed by following the steps below.

1. Open the configuration screen for the registered MQTT Emitter.
2. Press Test.



3. A test connection is performed to the configured destination, and the result is displayed at the bottom of the configuration screen.



Cautions for MQTT Emitter Test Connection

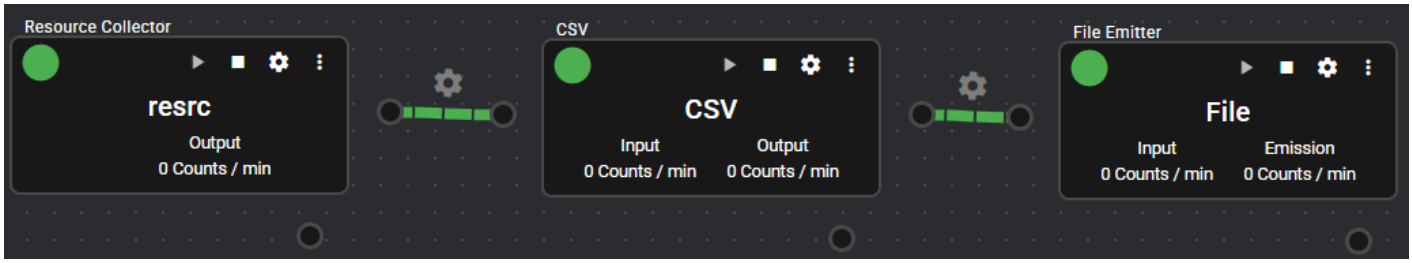
The test connection can only be performed under the following conditions.

- When the component is stopped
- When the component status is in error

Note: For details about the status, refer to Status List.

5.6.2 File Emitter

This component saves the collected data as a file in storage.
Data received from the serializers connected to the IN port is saved to a file path specified by the parameter.
The saved files can be downloaded by specifying a time range in the configuration dialog.



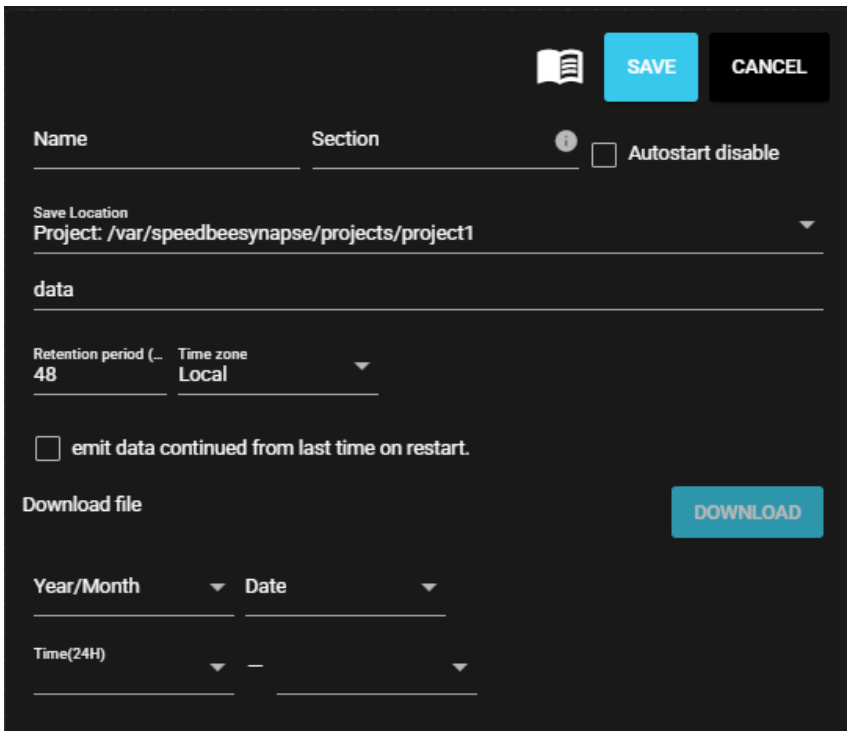
5.6.2.1 Basic Information

The basic information of the File Emitter is as follows.

Item	Description
Base name	File Emitter
Description	This saves the collected data in the storage.
Consumption point	0
Port	Input: 1 This receives data in file format from the serializer component. The received data is saved in the storage. Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	The output file can be downloaded. For details, see Download file.

5.6.2.2 Configuration item

The configuration items for the File Emitter is as follows.



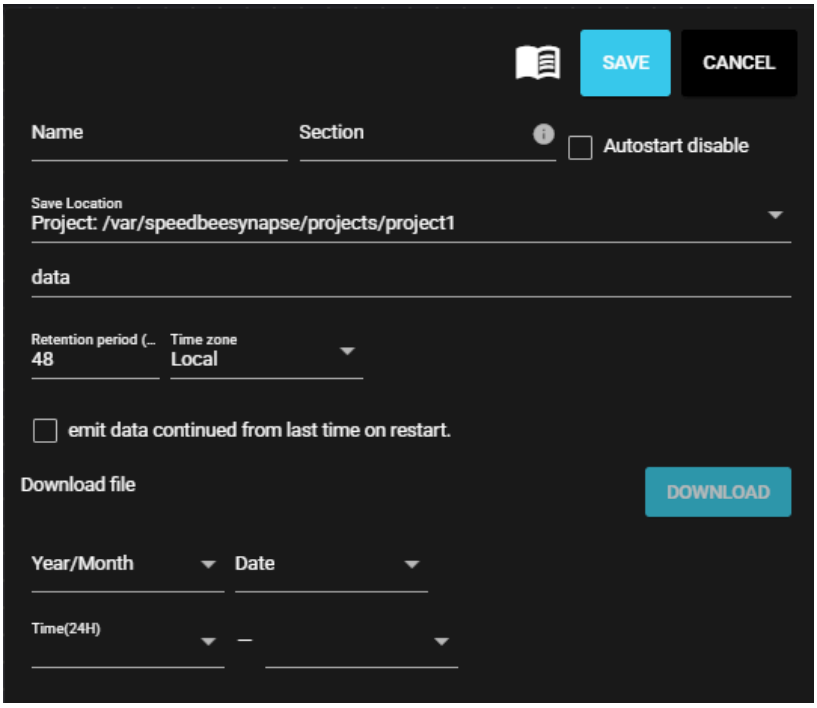
Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Section	Input any value that is uniquely assigned to each File Emitter. Note: The section is used as the subdirectory name of the output directory. Only one-byte characters can be entered.
Autostart disable	Select ON to disable the Autostart of the component.
Output directory	This is the directory where files are output. Specify a base directory and its relative path. The base directory can be selected from the options below. • Project path (This is saved under the project. If a project is replicated, the save folder will be under the new project.) Note: In some environments, this setting cannot be changed/edited. • Any given path (Usually a relative path to the base directory is specified, but for this, directly specify any given absolute path. Even when a project is replicated, the specified path becomes a save folder.)
Retention period (hour)	Input the duration for which the file will be kept in storage.
Time zone	Select a time zone for a file.

Item	Description
After the start, data is output continuing from where the previous emission left off.	emit data continued from last time on restart.

5.6.2.3 Download File

The file output by the File Emitter can be downloaded by performing the following steps.

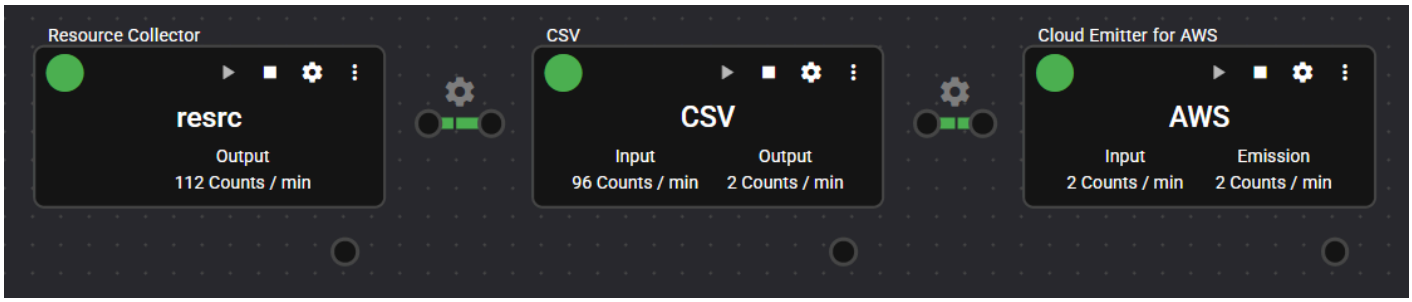
1. Input the time range in the "Download file" section.



2. Press Download to download the files output within the time range input in the step above, in ZIP format.

5.6.3 Cloud Emitter for AWS

This is a component that emits data to AWS IoT Core.
It receives data from various serializers at the IN Port and emits the data to AWS IoT Core.



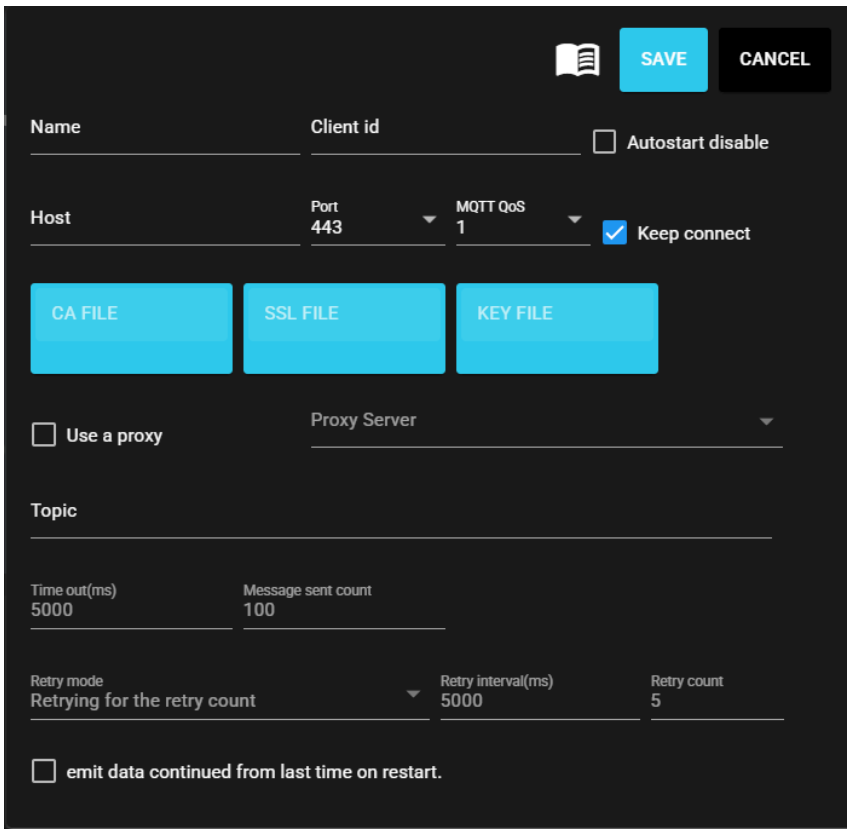
5.6.3.1 Basic Information

The basic information for the Cloud Emitter for AWS is as follows.

Item	Description
Base name	Cloud Emitter for AWS
Description	This emits data to AWS IoT Core.
Consumption point	2
Port	Input: 1 This receives data in file format from the serializer component. The received data is emitted to AWS IoT Core. Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	Test connection to AWS IoT Core can be performed. For detail, see Test Connection.

5.6.3.2 Configuration item

Configuration items for Cloud Emitter for AWS are as follows.



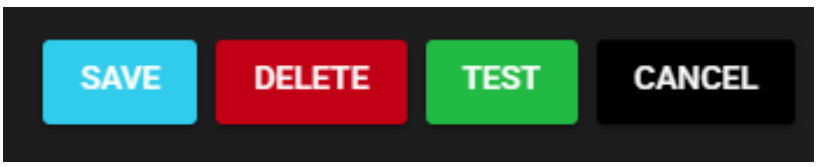
Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Client ID	AWS IoT Core Client ID
Autostart disable	Select ON to disable the Autostart of the component.
Host	Input the host name for AWS IoT Core.
Port	Select the port number for AWS IoT Core (445 or 8883).
MQTT Qos	Select the QoS of MQTT to use (0: No reception confirmation, 1: Reception confirmed).
Keep connect	Do not disconnect server connection after each emission.
CA certificate	Select Amazon Root CA1.
SSL certificate	Select device certificate.

Item	Description
Secret key	Select private key file.
Use a proxy	Set to ON to communicate with the AWS IoT Core via a proxy server.
Proxy server	Select a proxy server. Note: You must configure the server to be used in advance on the Proxy Settings screen.
Topic	Topics to be assigned when publishing
Time out (ms)	Timeout duration for the emission process in milliseconds. Note: This item cannot be edited.
Message sent count	Maximum number of emissions in a single event. Note: This item cannot be edited.
Retry mode	The behavior when retrying after an emission failure.
Retry interval (ms)	The interval between emission retries.
Retry count	The number of emission retries.
After the start, data is output continuing from where the previous emission left off.	emit data continued from last time on restart.

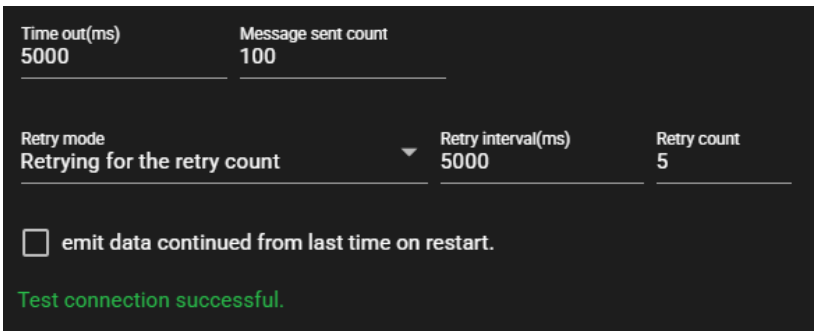
5.6.3.3 Test connection

A test connection to the configured destination can be performed by following the steps below.

1. Opens the configuration window for the registered Cloud Emitter for AWS.
2. Press Test.

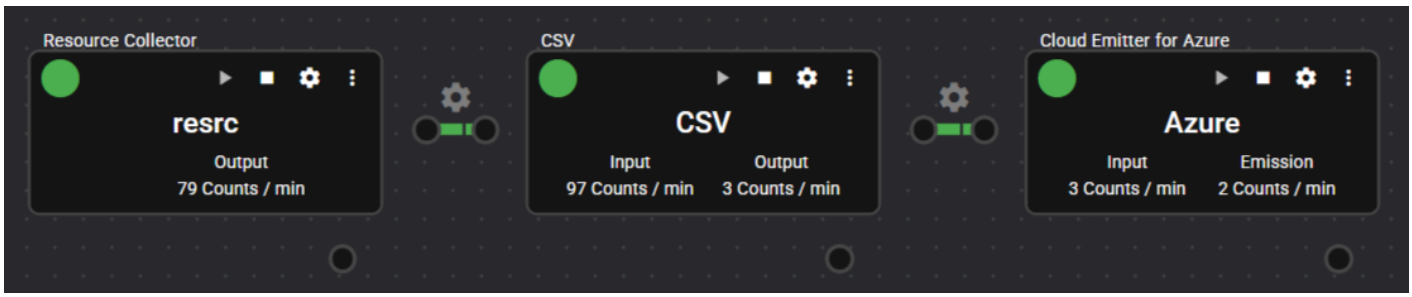


3. A test connection is performed to the configured destination, and the result is displayed at the bottom of the configuration screen.



5.6.4 Cloud Emitter for Azure

This is a component that emits data to Azure IoT Hub.
It receives data from various serializers at the IN Port and emits the data to Azure IoT Hub.



5.6.4.1 Basic Information

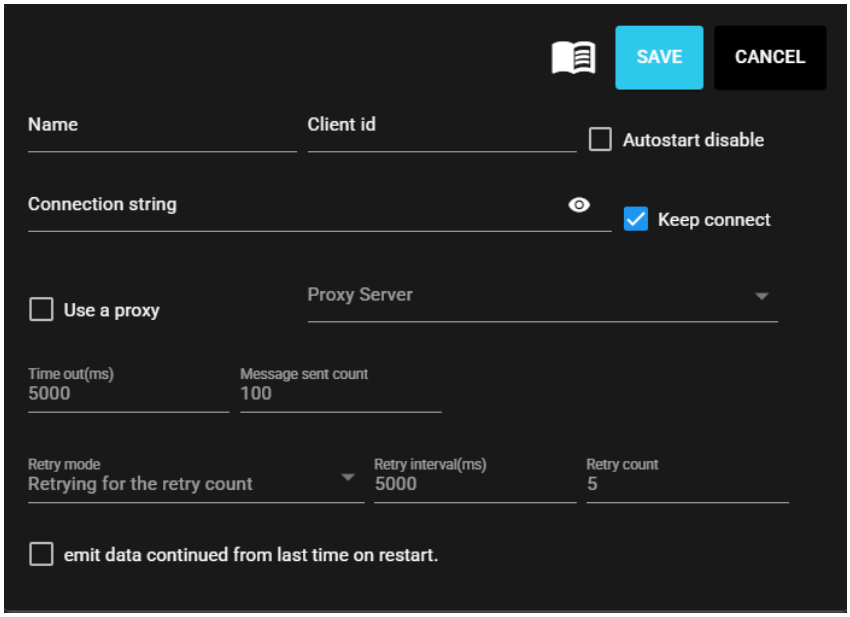
The basic information for the Cloud Emitter for Azure is as follows.

Item	Description
Base name	Cloud Emitter for Azure
Description	This emits data to Azure IoT Hub.
Consumption point	2

Item	Description
Port	Input: 1 This receives data in file format from the serializer component. The received data is emitted to Azure IoT Hub. Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	Test connection to Azure IoT Hub can be performed. For detail, see Test Connection.

5.6.4.2 Configuration item

The configuration information for the Cloud Emitter for Azure is as follows.



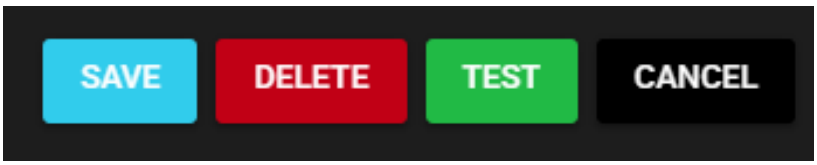
Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
Client ID	Azure Client ID (Client ID used at MQTT)

Item	Description
Autostart disable	Select ON to disable the Autostart of the component.
Connection string	Input Azure IoT Hub connection string.
Keep connect	Do not disconnect server connection after each emission.
Use a proxy	Set to ON to communicate with the Azure IoT Hub via a proxy server.
Proxy server	Select a proxy server. Note: You must configure the server to be used in advance on the Proxy Settings screen.
Time out (ms)	Timeout duration for the emission process in milliseconds. Note: This item cannot be edited.
Message sent count	Maximum number of emissions in a single event. Note: This item cannot be edited.
Retry mode	The behavior when retrying after an emission failure. Note: This item cannot be edited.
Retry interval (ms)	The interval between emission retries. Note: This item cannot be edited.
Retry count	The number of emission retries. Note: This item cannot be edited.
After the start, data is output continuing from where the previous emission left off.	emit data continued from last time on restart.

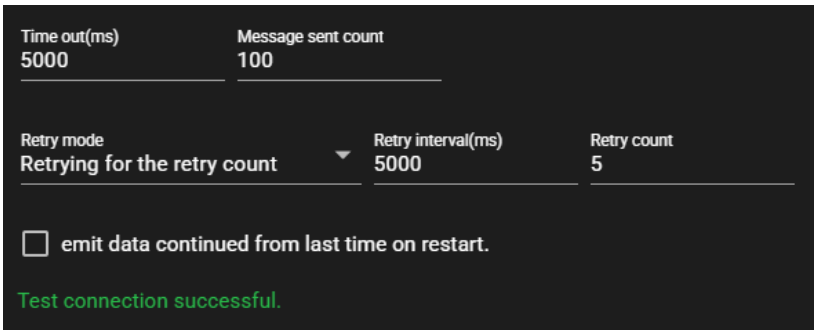
5.6.4.3 Test connection

A test connection to the configured destination can be performed by following the steps below.

1. Open the configuration window for the registered Cloud Emitter for Azure.
2. Press Test.

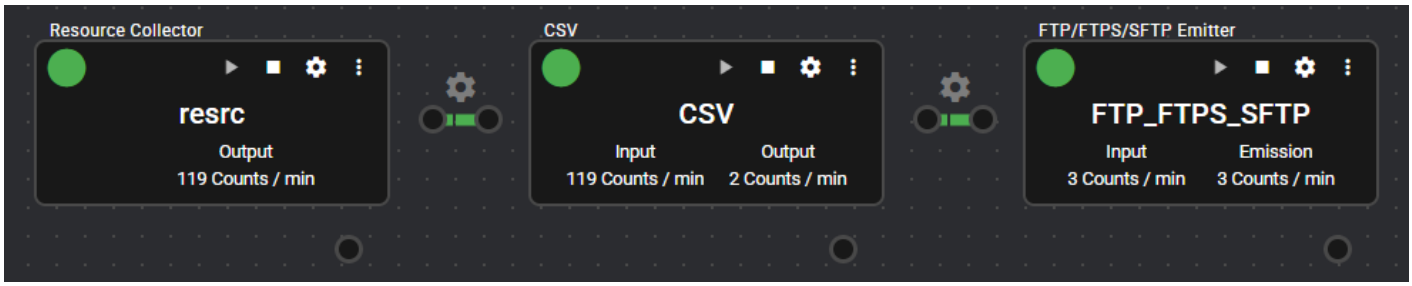


3. A test connection is performed to the configured destination, and the result is displayed at the bottom of the configuration screen.



5.6.5 FTP/FTPS/SFTP Emitter

This is a component that transfers data using FTP, FTPS, or SFTP.
At the IN Port, data is received from serializers, and the data is transferred using FTP, FTPS, or SFTP as a file in the format of the serializer that provided the data.
This component is not compatible with communication via a Proxy Server.



5.6.5.1 Basic Information

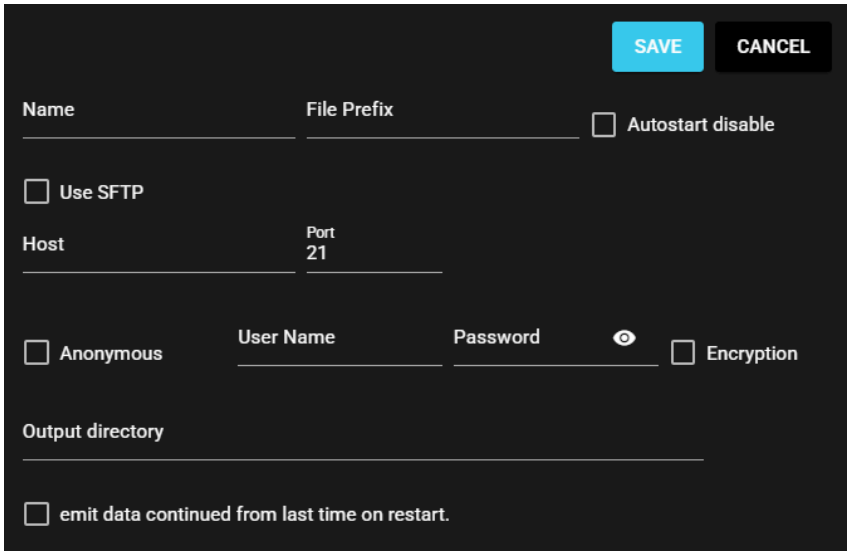
The basic information about FTP/FTPS/SFTP Emitters is as follows.

Item	Description
Base name	FTP/FTPS/SFTP Emitter
Description	This transfers data via FTP or SFTP.
Consumption point	2

Item	Description
Port	Input: 1 This receives data in file format from the serializer component. The received data is transferred by FTP or SFTP. Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	Test connection to FTP or SFTP server can be performed. For detail, see Test Connection.

5.6.5.2 Configuration item

The configuration information about FTP/FTPS/SFTP Emitters is as follows.



Item	Description
Name	Name to be assigned to the component Note: The same name as another component name cannot be used.
File Prefix	Prefix assigned to the file name
Autostart disable	Select ON to disable the Autostart of the component.

Item	Description
Use SFTP.	Select ON to use SFTP.
Host	Specify the host name or IP address of the destination server.
Port	Specify the port number of the destination server.
Anonymous	Select ON when user authentication is not performed. Note: This option is available only when Use SFTP is set to OFF.
Username	Input the authentication user name. Note: This can be input in the following cases. • Use SFTP is set to OFF, and anonymous access is set to OFF. • Use SFTP is set to ON.
Password	Input the authentication password. Note: This can be input in the following cases. • Use SFTP is set to OFF, and anonymous access is set to OFF. • Use SFTP is set to ON, and public key authentication is set to OFF.
Passphrase	Input passphrase. Note: This can be input only when Use SFTP is set to ON and public key authentication set to is ON.
Public key	Select ON to perform public key authentication. Note: This option is available only when Use SFTP is set to ON.
Secret key	Upload secret key. Note: This can be uploaded only when Use SFTP is set to ON and public key authentication set to is ON.
Encryption	Select ON when encrypting emitting/receiving data. Note: This option is available only when Use SFTP is set to OFF.
Output directory	Specify the output directory in the destination server.

Item	Description
After the start, data is output continuing from where the previous emission left off.	emit data continued from last time on restart.

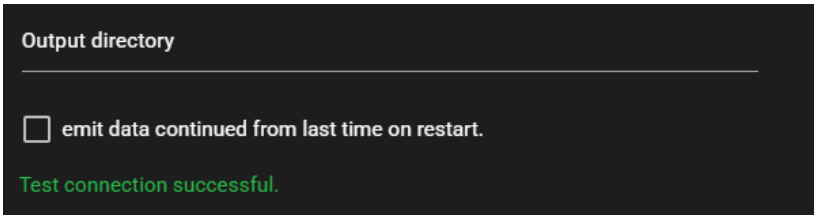
5.6.5.3 Test connection

A test connection to the configured destination can be performed by following the steps below.

1. This Open the configuration window for the registered FTP/FTPS/SFTP Emitter.
2. Press Test.



3. A test connection is performed to the configured destination, and the result is displayed at the bottom of the configuration screen.

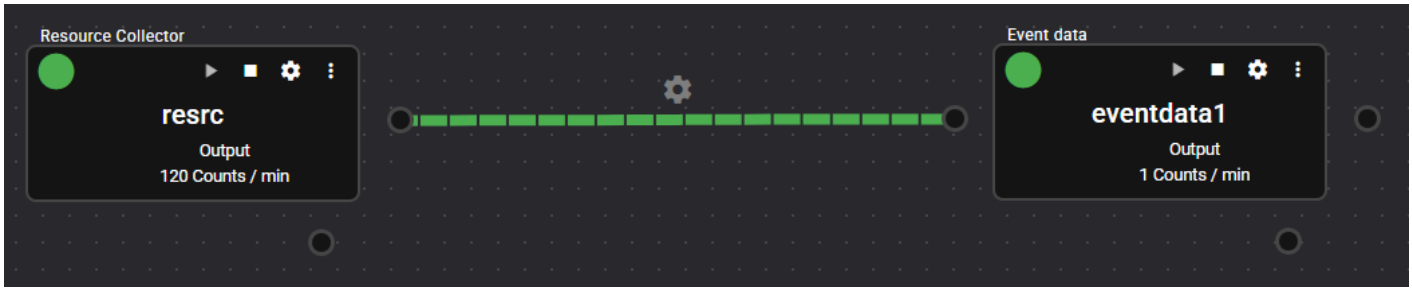


5.7 Logic (Event)

This section explains the logic components associated with events.

5.7.1 Event Data

This component determines the condition of the collected data and registers it as new separate data only when the condition is met.
It receives data from the input port, performs conditional judgment on the data each time, and outputs the data as new separate data if the conditions are met.



5.7.1.1 Basic Information

The basic information of the event data is as follows.

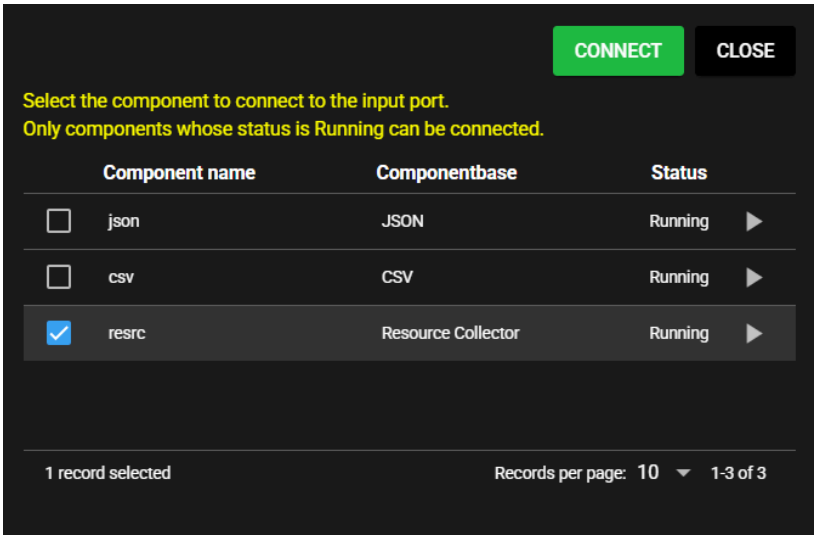
Item	Description
Base name	Event data
Description	Outputs only the collected data that matches the condition as new separate data. Note: The memory usage, etc. increases because the data is registered as different data. This is not suitable when you want to handle high sampling rate data as is.
Consumption point	2

Item	Description
Port	Input: 1 Receives data from other components. Output: 1 Outputs received data that matches the condition. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	For a setting example using event data, see Setting Example.

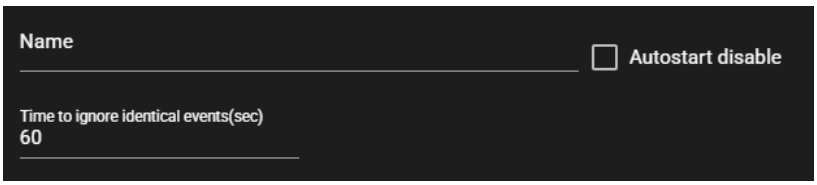
5.7.1.2 Setting Method

You can configure event data by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add event data. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

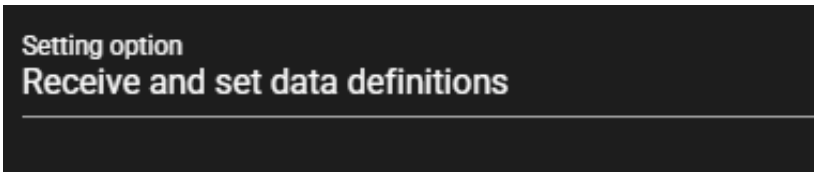


5. Enter the following component information.



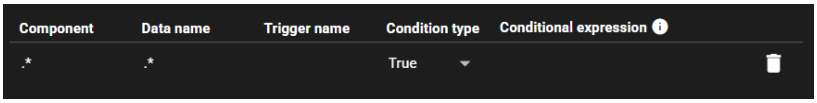
Item	Description
Name	Enter a component name. Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Time to ignore identical events(sec)	Enter time to ignore the identical events

6. Select a Setting option from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

7. Enter the following items to set the event condition.



Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression

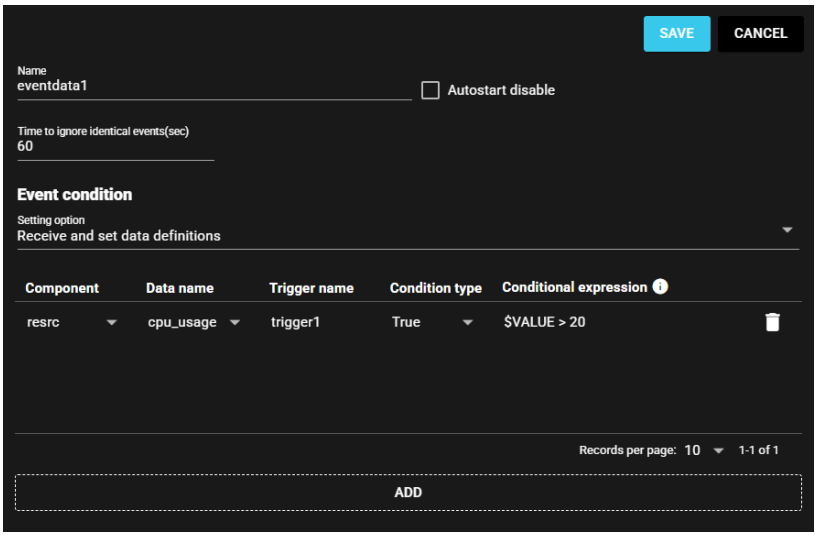
Item	Description
Data name	Regular expression to indicate the data name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression
Trigger name	Enter a name for the trigger
Condition type	True: A trigger occurs if the conditional expression is true False: A trigger occurs if the conditional expression is false Fall: A trigger occurs when the conditional expression changes from true to false Raise: A trigger occurs when the conditional expression changes from false to true Anytime: Occurs when a value is registered
Conditional expression	Enter the conditional expression for trigger Note: Cannot be entered when the condition type is Anytime

Regular expression input examples

The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

8. Press Save.

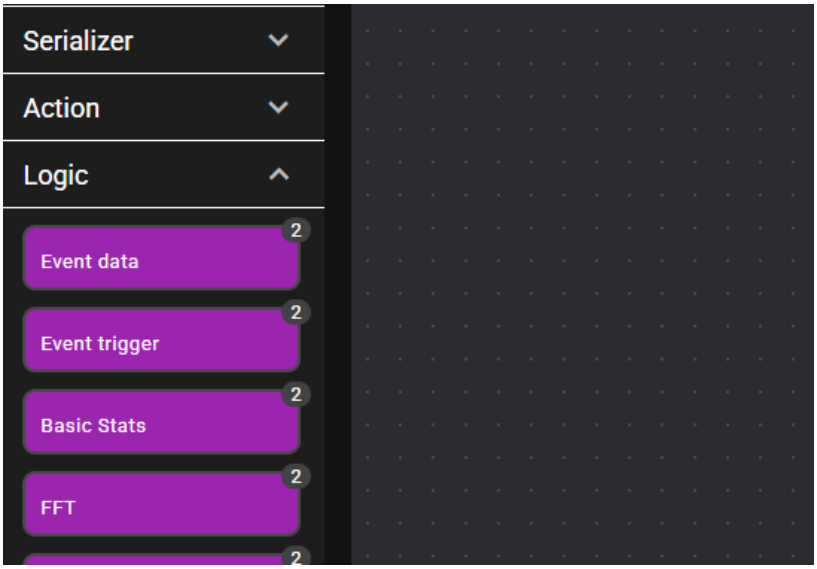


9. Start the component. For more information on how to start components, see Start/Stop Components.

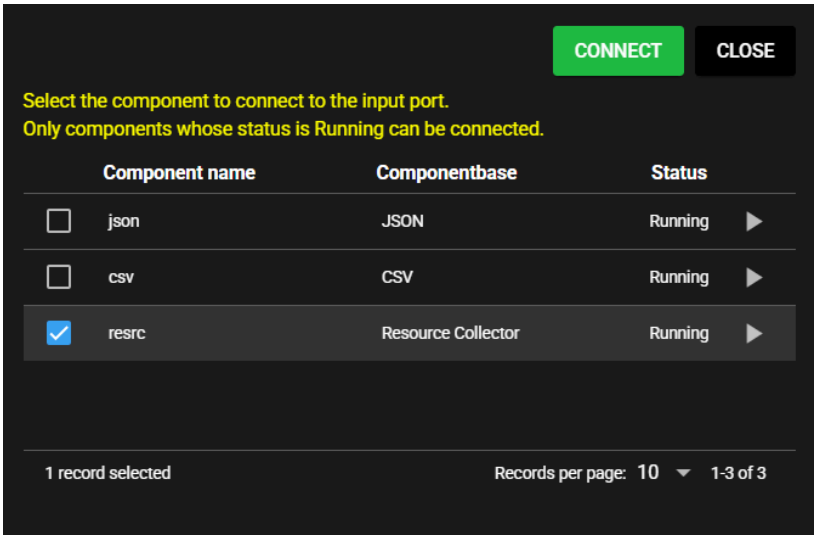
5.7.1.3 Setting Example

An example of setting event data is as follows. Here, the first step is to collect CPU usage using a resource collector. Then, use event data so that the CPU usage is registered as new separate data only when it exceeds 20%.

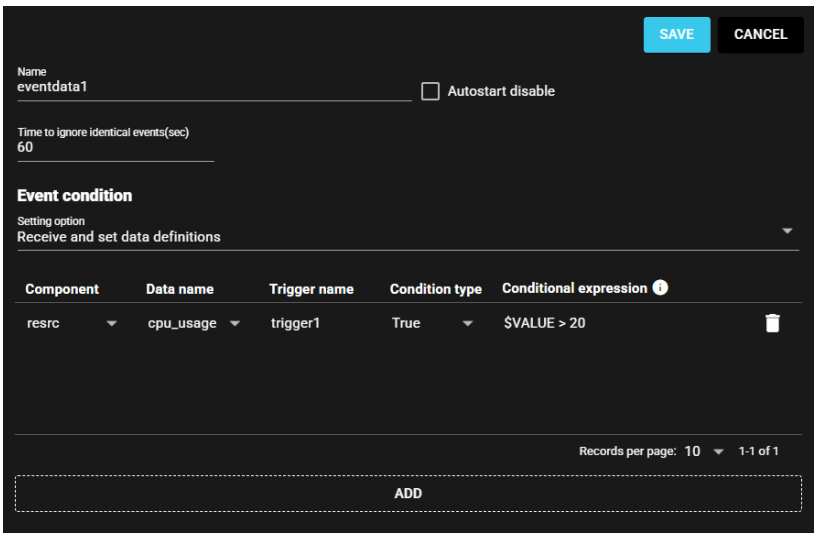
1. Add a collector beforehand. Here we create a resource collector named "resrc" and configure it to output CPU usage.
2. Drag and drop the event data icon onto the panel.



3. Select the component to connect to the input port. In this case, select the resource collector named "resrc".



4. Enter the following items and press Save.

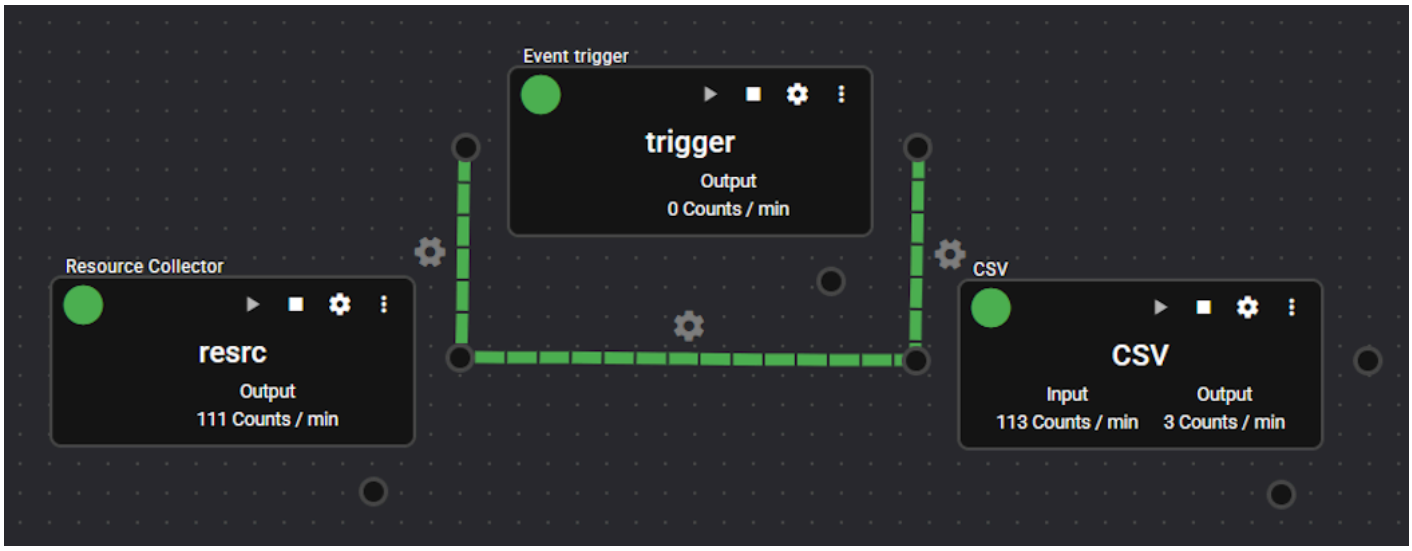


Item	Description
Name	Event Data 1
Autostart disable	OFF
Time to ignore identical events(sec)	60
Setting option	Receive and set data definitions
Component	resource
Data name	cpu_usage
Trigger name	trigger1
Condition type	True
Conditional expression	$VALUE > 20

5. CPU usage is received from the resource collector and registered as new separate data when it exceeds 20%. The registered data is output from the output port.

5.7.2 Event Trigger

This component determines the condition of the collected data and outputs the information of the intervals that match the condition.
This component is used in combination with the "Use Event" component of the input port.
It receives data from the input port, performs conditional judgment on the data each time, and outputs the information of the interval that matches the condition as data.



5.7.2.1 Basic Information

The basic information of the event trigger is as follows.

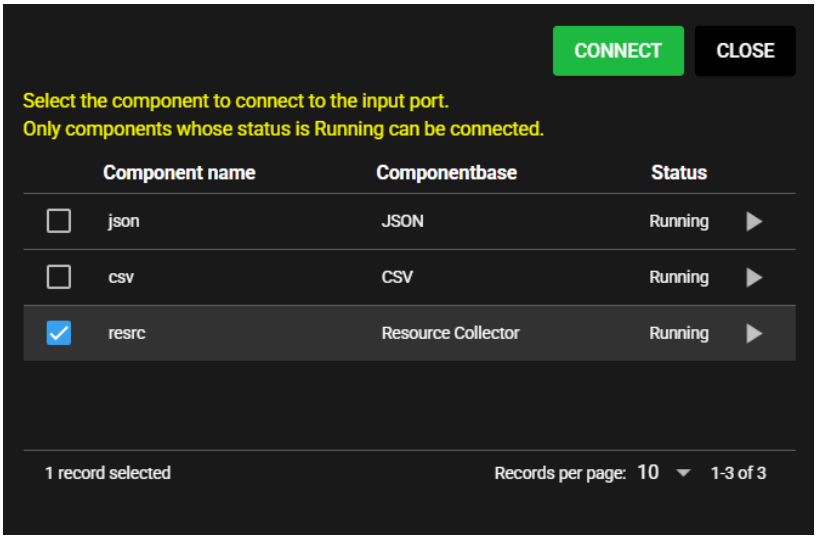
Item	Description
Base name	Event trigger
Description	Determines the condition of the collected data and outputs as data the information of the intervals that match the condition. This component is used in combination with the "Use Event" component of the input port. To transmit the data in the event to the flowlink destination, a separate flowlink must be connected. Unlike the event data, which registers the original data separately, this component registers the information of the event intervals, so memory usage can be saved.
Consumption point	2
Port	Input: 1 Receives data from other components. Output: 1 Determines the condition of the received data and outputs whether the condition is met as a true/false value.

Item	Description
	System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.
Other	For a setting example using an event trigger, see Setting Example.

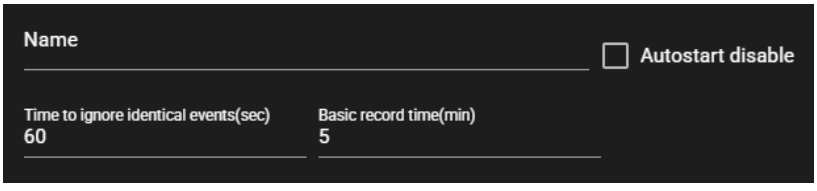
5.7.2.2 Setting Method

You can configure an event trigger by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add an event trigger. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

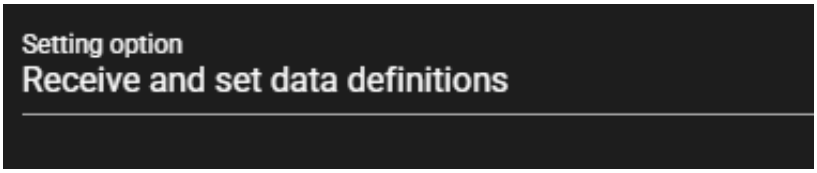


5. Enter the following component information.



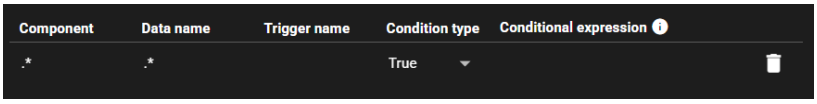
Item	Description
Name	Enter a component name. Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Time to ignore identical events(sec)	Enter time to ignore the identical events
Basic record time(min)	Enter the basic record time in minutes

6. Select a Setting option for the start event from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

7. Enter the following items to set the start event.



Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression
Data name	Regular expression to indicate the data name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression
Trigger name	Enter a name for the trigger

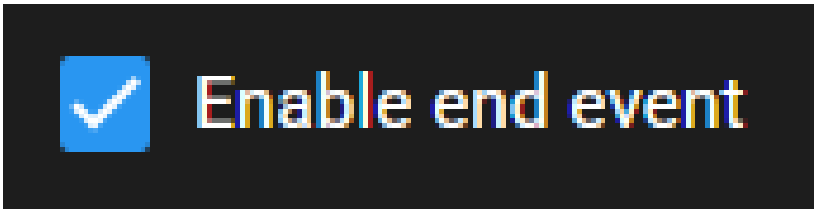
Item	Description
Condition type	True: A trigger occurs if the conditional expression is true False: A trigger occurs if the conditional expression is false Fall: A trigger occurs when the conditional expression changes from true to false Raise: A trigger occurs when the conditional expression changes from false to true Anytime: Occurs when a value is registered
Conditional expression	Enter the conditional expression for trigger Note: Cannot be entered when the condition type is Anytime

Regular expression input examples

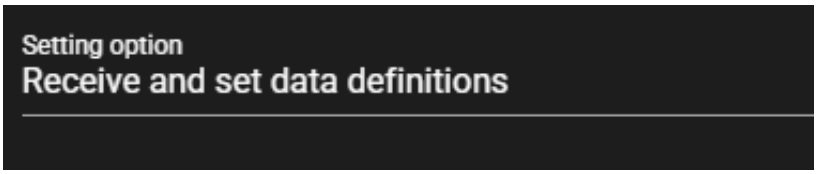
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

8. If you want to set an end event, set End event ON.

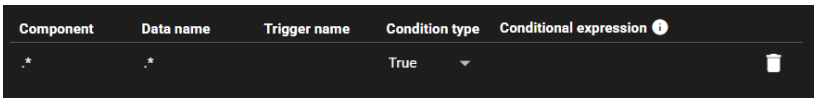


9. To set an end event, select a Setting option for the end event from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

10. To set an end event, enter the following items to set the end event.



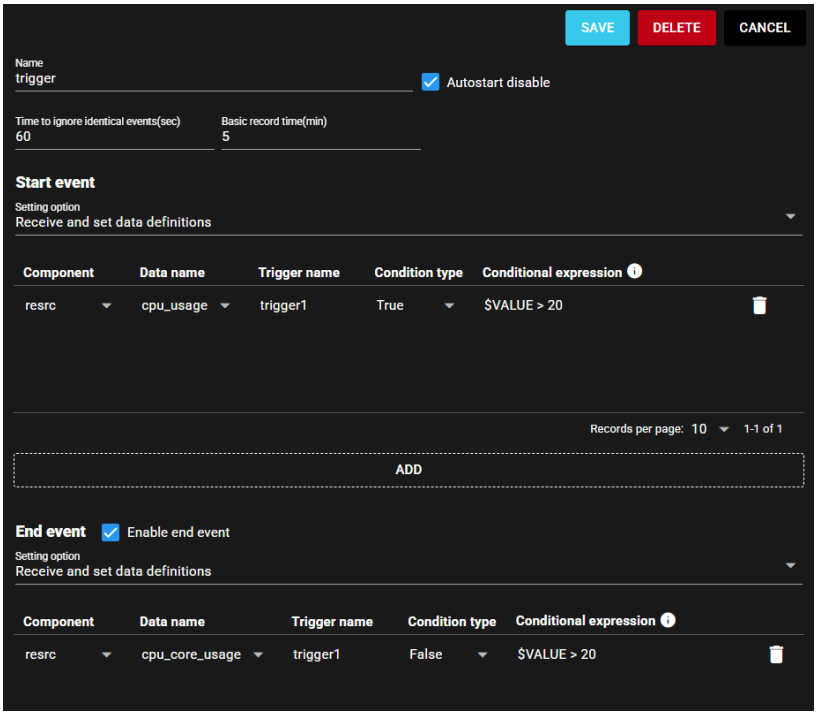
Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression
Data name	Regular expression to indicate the data name of the data received at the input port that is to be set Note: Can be entered only when the setting option is Set by regular expression
Trigger name	Enter a name for the trigger
Condition type	True: A trigger occurs if the conditional expression is true False: A trigger occurs if the conditional expression is false Fall: A trigger occurs when the conditional expression changes from true to false Raise: A trigger occurs when the conditional expression changes from false to true Anytime: Occurs when a value is registered
Conditional expression	Enter the conditional expression for trigger Note: Cannot be entered when the condition type is Anytime

Regular expression input examples

The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

11. Press Save to save the settings.



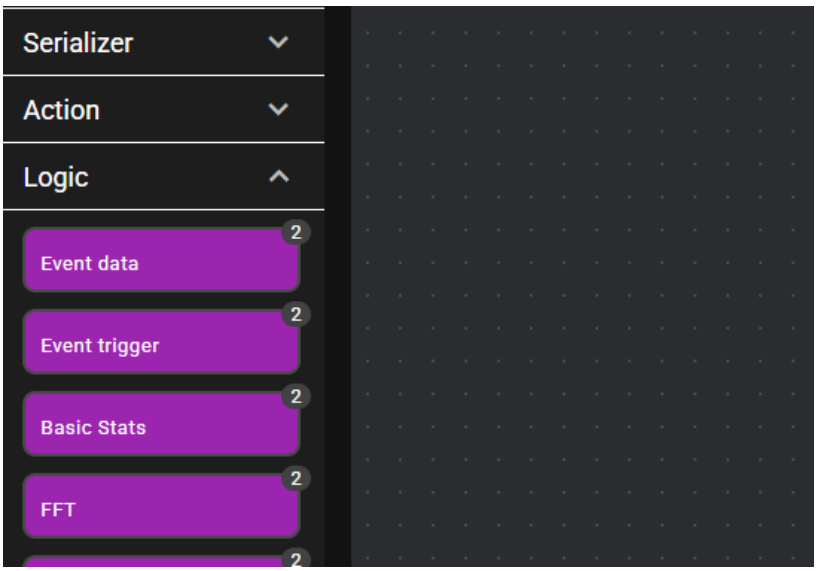
12. Start the component. For more information on how to start components, see Start/Stop Components.
13. Events set in an event trigger can be used to determine the condition when another component receives data. For more information, see Event Settings in Input Port.

> **Components for which the event intervals of event triggers are not available**
>
> The following components can be connected (set) to an event trigger but cannot use the event interval.
> - Logic components (event data, basic stats, FFT, moving average)

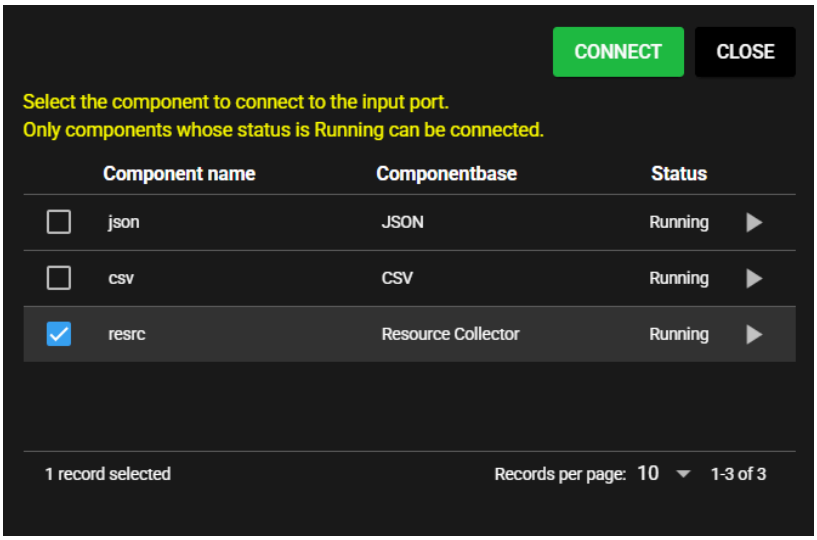
5.7.2.3 Setting Example

An example of setting an event trigger is as follows. Here, the first step is to make settings to allow the CPU usage collected by a resource collector to be processed by a CSV serializer. Then, use a event trigger so that processing occurs only when the CPU usage exceeds 20%.

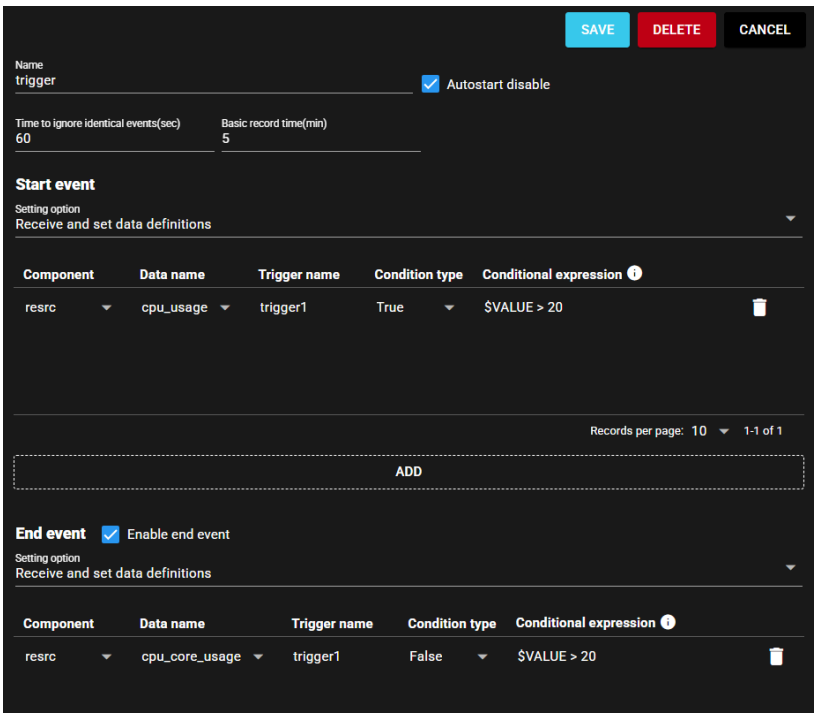
1. Add a collector beforehand. Here we create a resource collector named "resrc" and configure it to output CPU usage.
2. Drag the event trigger icon from the left menu and drop it on the panel.



3. Select the component to connect to the input port. In this case, select the resource collector named "resrc".



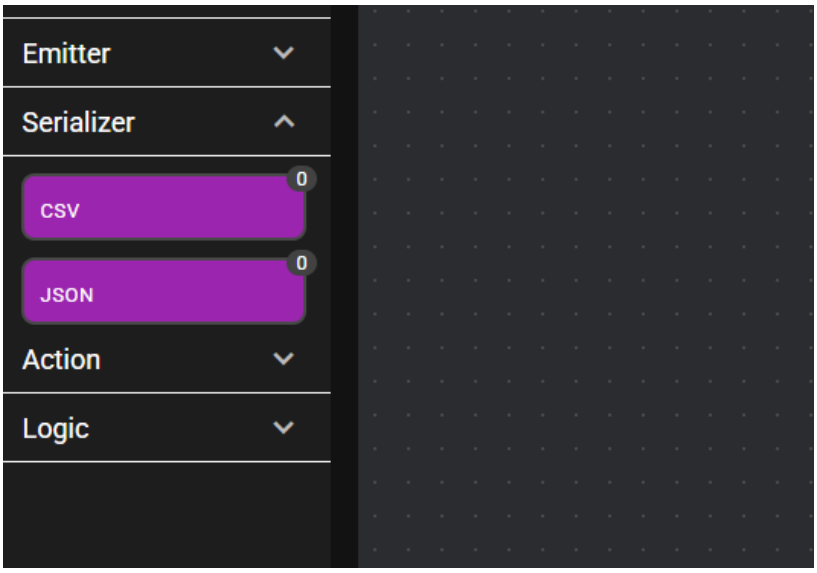
4. Enter the following items and press Save.



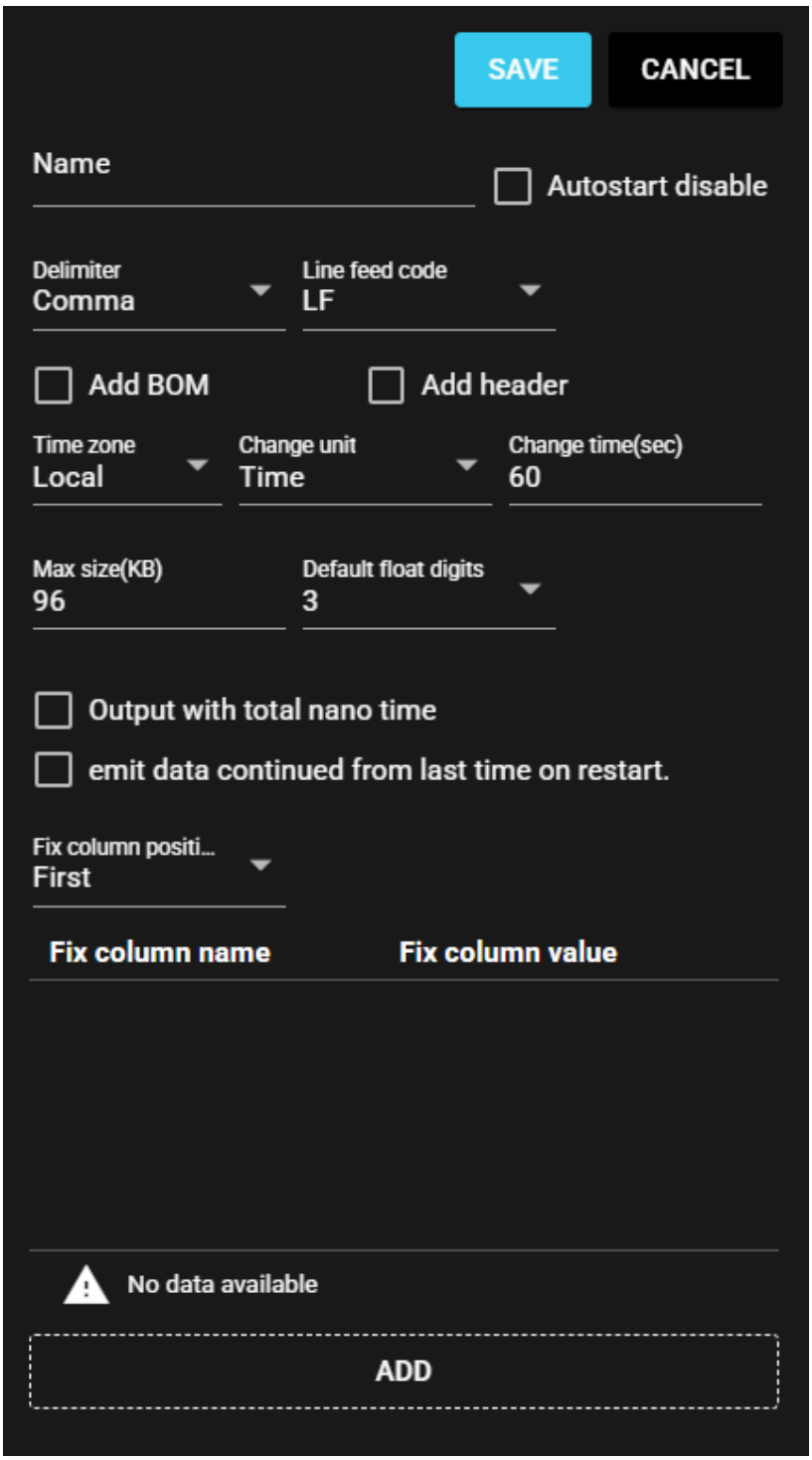
Item	Description
Name	Event Trigger 1
Autostart disable	OFF
Time to ignore identical events(sec)	60
Basic record time(min)	5
Setting option	Receive and set data definitions
Component	resource
Data name	cpu usage
Trigger name	trigger1
Condition type	True
Conditional expression	$VALUE > 20
Enable end event	ON

Item	Description
Setting option	Receive and set data definitions
Component	resource
Data name	cpu_usage
Trigger name	trigger1
Condition type	False
Conditional expression	$VALUE > 20

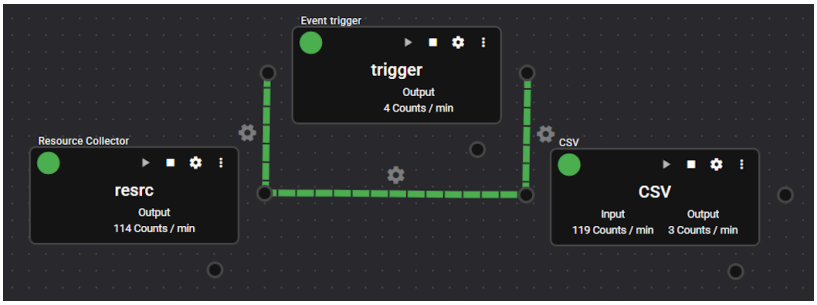
5. Drag the CSV serializer icon from the left menu and drop it on the panel.



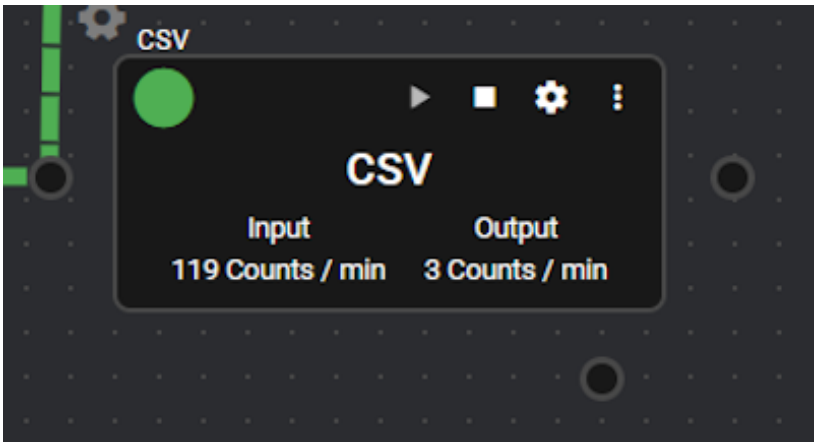
6. Enter the setting items as needed and press Save.



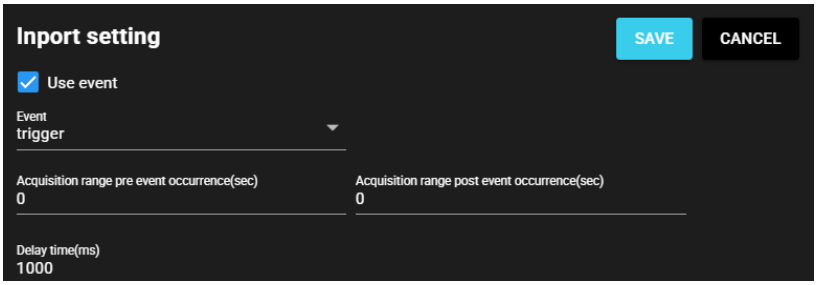
7. Connect the output port of the resource collector/event trigger and the input port of the CSV serializer with a flowlink.



8. Right-click the input port of the CSV serializer.



9. Check Use event, select "Event Trigger 1" in Event, and press Save.



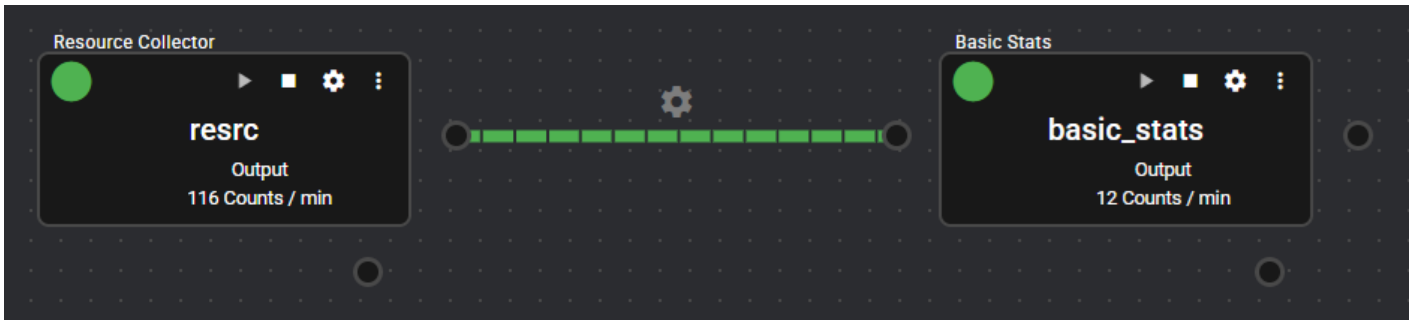
10. With the component running in this state, the CPU usage collected by the resource collector is processed into a CSV file by the CSV serializer only when it exceeds 20%.

5.8 Logic (Real Time Analysis and Data Manipulation)

This section explains the logic components associated with real time analysis.

5.8.1 Basic Stats

This component computes basic statistics.
It receives data from the input port and outputs the results of basic statistics of the data within a certain range (count/seconds) as new data.



5.8.1.1 Basic Information

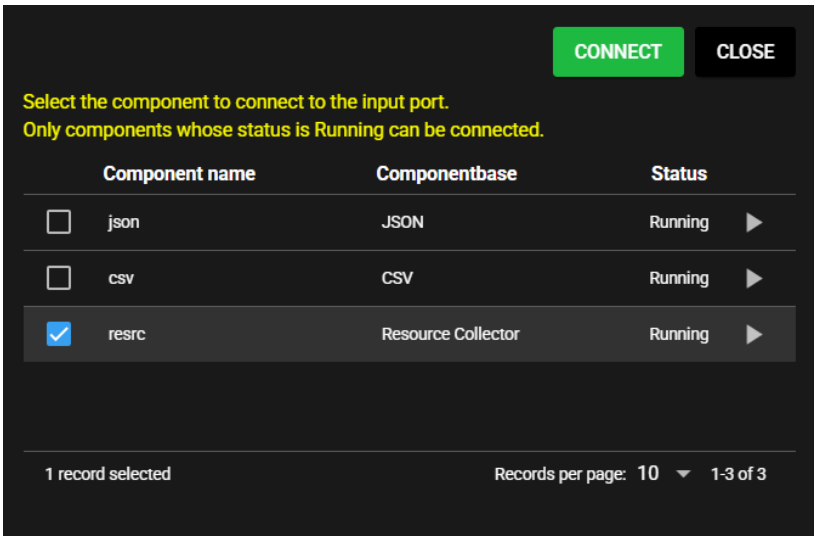
The basic information of the basic stats is as follows.

Item	Description
Base name	Basic stats
Description	Outputs as data the results of basic statistics of the collected data.
Consumption point	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs the results of computing basic statistics of the input data based on the settings specified by the Setting Method. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.8.1.2 Setting Method

You can configure basic stats by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add basic stats. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

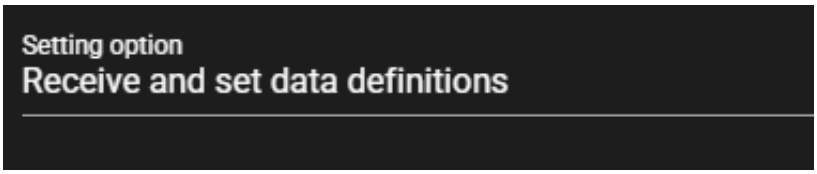


5. Enter the following component information.



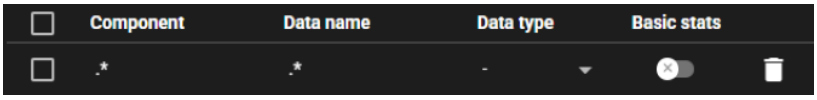
Item	Description
Name	Enter a name for the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

6. Select a Setting option from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

7. Enter the following items for the data to be set only when the Setting option is Set by regular expression.



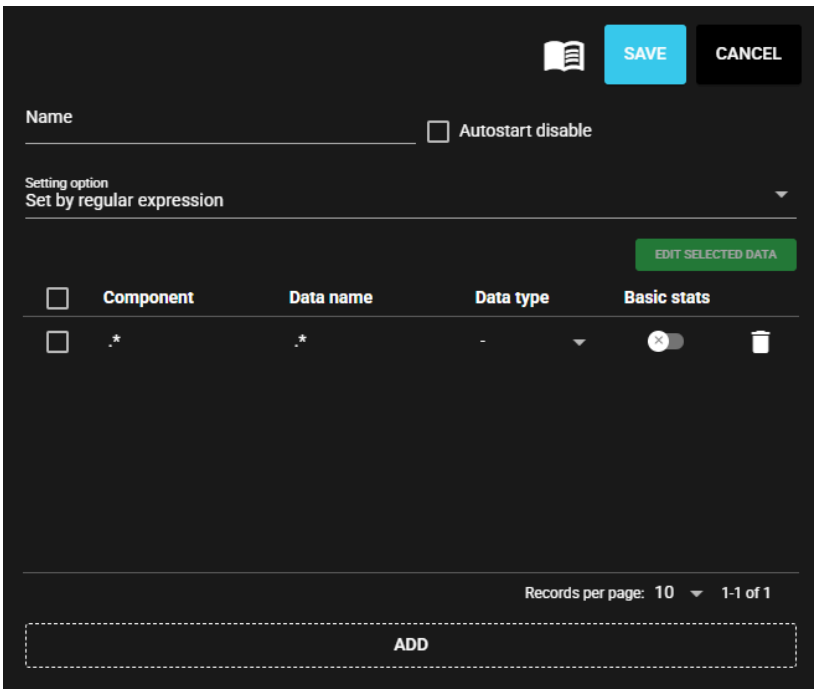
Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set
Data Name	Regular expression to indicate the data name of the data received at the input port that is to be set
Data Type	Data type of the data received at the input port that is to be set

Regular expression input examples

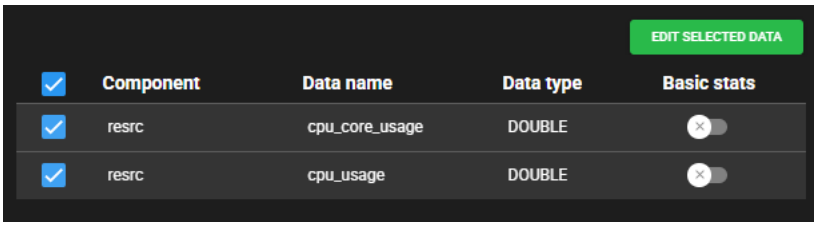
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

8. Do one of the following to open the basic stats setting dialog.
   - Press the Basic Stats icon for the data for which you want to set basic stats.



- Check the data for which you want to set basic stats and press Edit selected data. The same settings will be applied to the checked data.



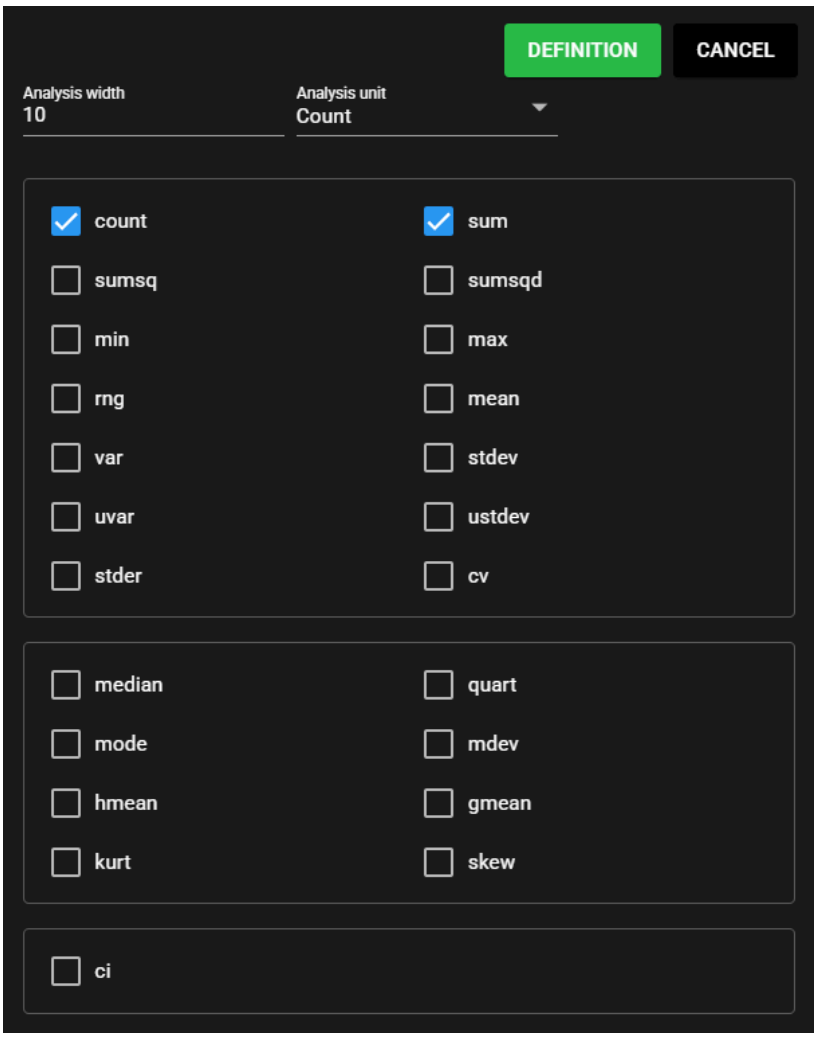
9. Enter the following items to set basic stats.



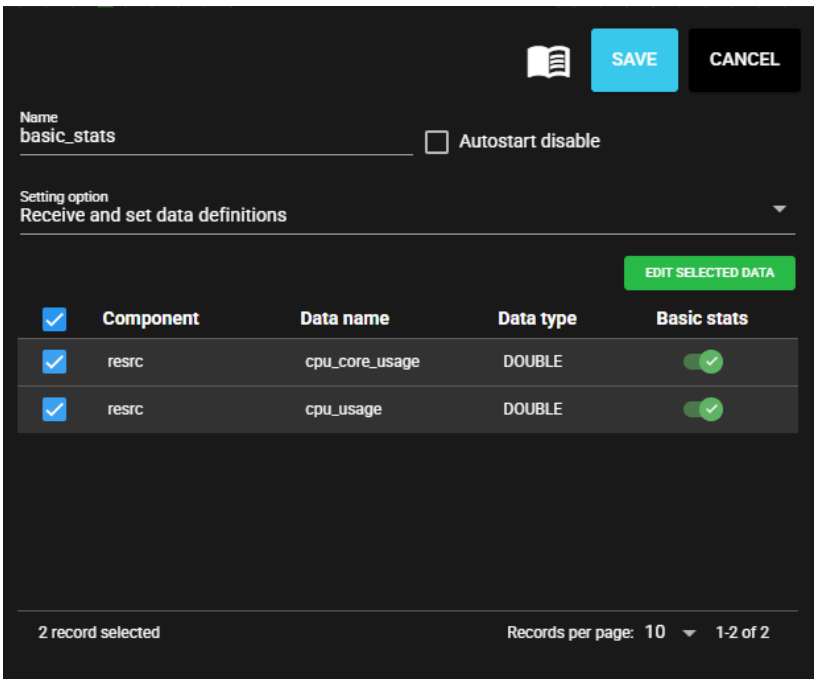
Item	Description
Analysis width	Set the width of analysis
Analysis unit	Set the unit for analysis width (Count or Seconds)
count	Number of data values in the data being analyzed
sum	Total value of the data being analyzed
sumsq	Sum of the squares of each value in the data being analyzed
sumsqd	Sum of the squared deviations of each value in the data being analyzed from the mean
min	Smallest value in the data being analyzed
max	Largest value in the data being analyzed

Item	Description
rng	Difference between the maximum and minimum values of the data being analyzed
mean	Average value of the data being analyzed
var	Square of the average of how far each value in the data being analyzed is from the mean
stdev	Square roots of variance
uvar	Variance divided by count -1
ustdev	Standard deviation of the sample
stder	Standard deviation of the sample mean
cv	Standard deviation divided by the mean
median	Value in the middle of the data being analyzed, ordered from smallest to largest
quart	Values that divide the data being analyzed, ordered from smallest to largest, into four equal parts
mode	Value that appears most frequently in the data being analyzed
mdev	Average of how far each value in the data being analyzed is from the mean
hmean	Reciprocal of the average of the reciprocals of each value in the data being analyzed
gmean	Nth root of the product of each value in the data being analyzed Note: N is the count
kurt	Sharpness of the distribution
skew	Distortion of the distribution
ci	Range of values that can be trusted to contain the mean of the data being analyzed

10. Press Definition.



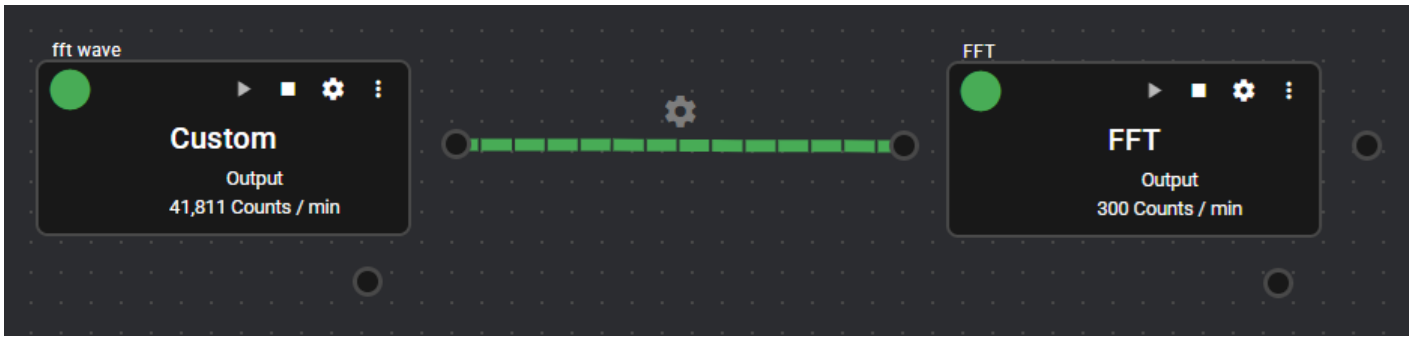
11. Press Save.



12. Start the component. For more information on how to start components, see Start/Stop Components.

5.8.2 FFT

This component registers the FFT results of collected data as data.
It receives data from the input port and outputs the results of FFT analysis on the data as separate data.
This component is available only for column type middle data for which the sampling rate is defined.



About column type middle data

There are no standard-feature components that use column type middle with a defined sampling rate. Develop and use custom components that handle column type middle (regular sampling rate) data.

5.8.2.1 Basic Information

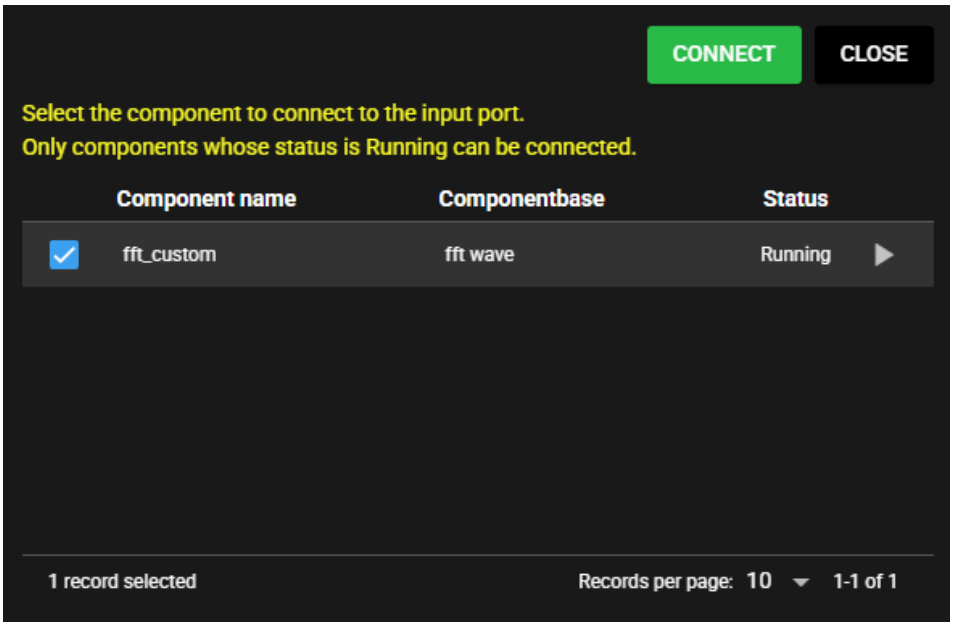
The basic information of the FFT is as follows.

Item	Description
Base name	FFT
Description	Outputs as data the results of FFT analysis on the collected data.
Consumption point	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs the results of FFT analysis on the input data based on the settings specified by the Setting Method. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.8.2.2 Setting Method

You can configure FFT by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add FFT. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

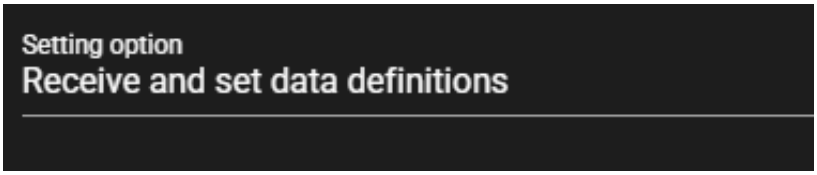


5. Enter the following component information.



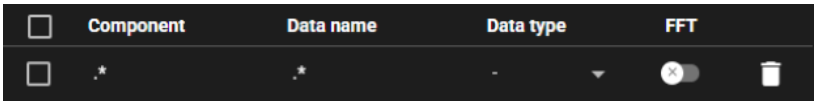
Item	Description
Name	Enter a name for the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

6. Select a Setting option from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

7. Enter the following items for the data to be set only when the Setting option is Set by regular expression.



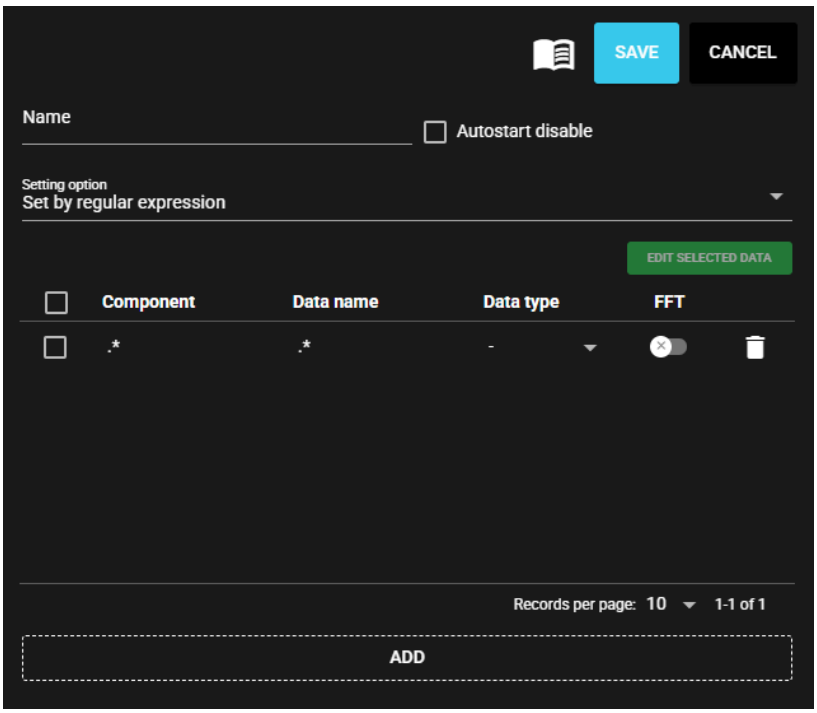
Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set
Data Name	Regular expression to indicate the data name of the data received at the input port that is to be set
Data Type	Data type of the data received at the input port that is to be set

Regular expression input examples

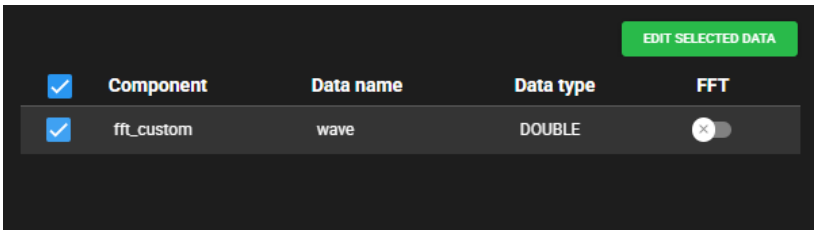
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

8. Do one of the following to open the FFT setting dialog.
   - Press the FFT icon for the data for which you want to set FFT.



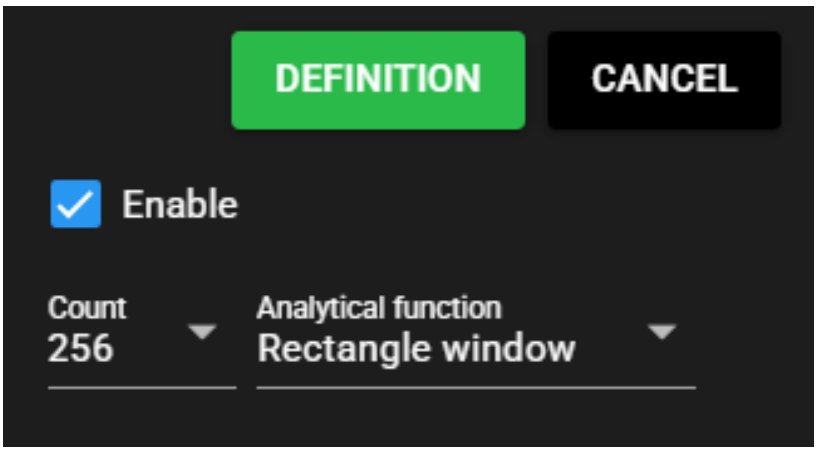
- Check the data for which you want to set FFT and press Edit selected data. The same settings will be applied to the checked data.



9. Set Enable ON.

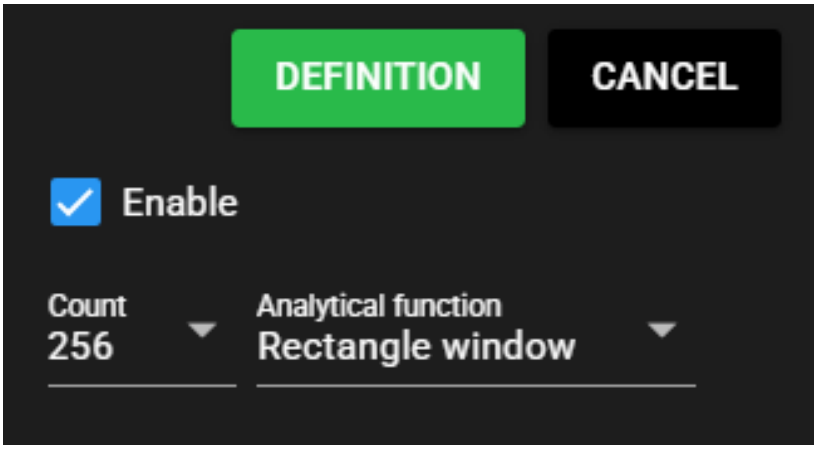


10. Enter the following items to set FFT.

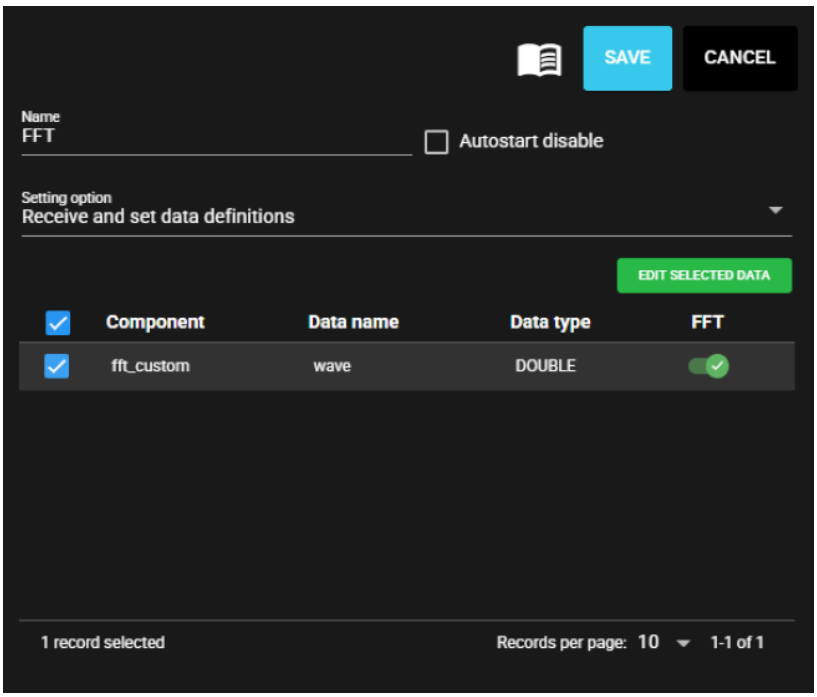


Item	Description
Count	Number of data values for which FFT is executed
Analytical function	Function applied to data before executing FFT

11. Press Definition.



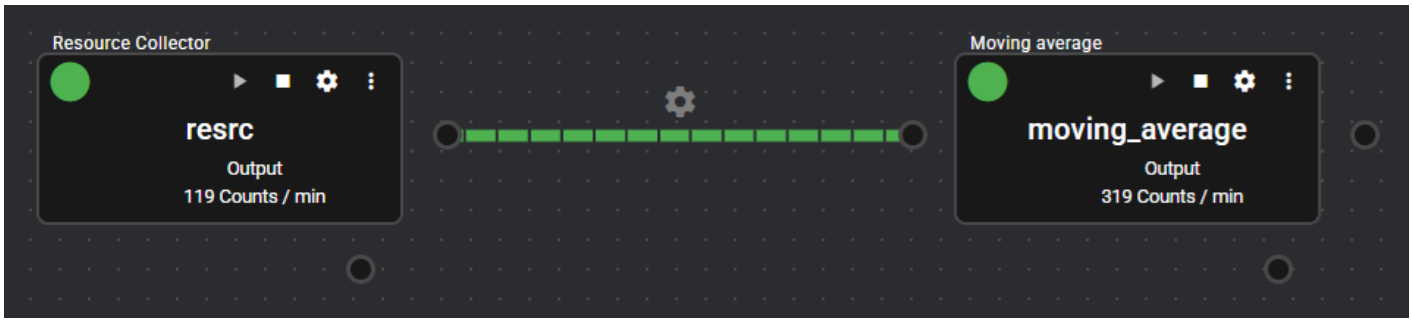
12. Press Save.



13. Start the component. For more information on how to start components, see Start/Stop Components.

5.8.3 Moving Average

This component computes moving average values.
It receives data from the input port, computes moving averages within a period of time from that data, and outputs them as new data.



5.8.3.1 Basic Information

The basic information of the moving average is as follows.

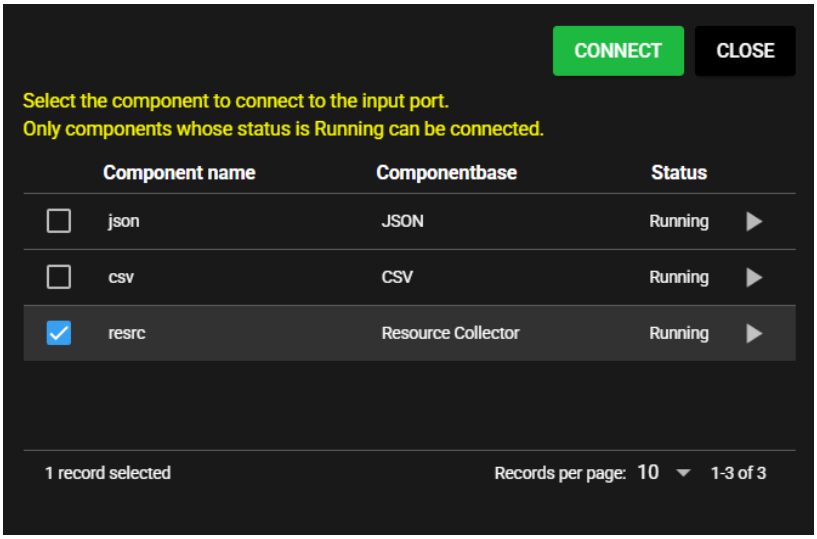
Item	Description
Base name	Moving average
Description	Outputs as data the moving average values of the collected data.

Item	Description
Consumption point	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs the moving average values of the input data based on the settings specified by the Setting Method. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.8.3.2 Setting Method

You can configure moving average by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add a moving average. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

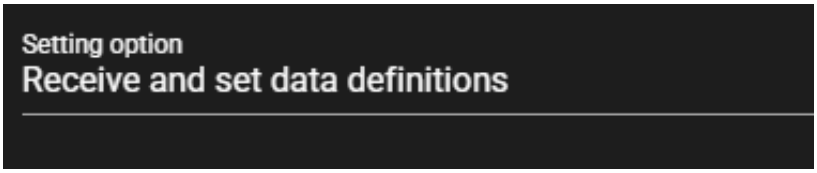


5. Enter the following component information.



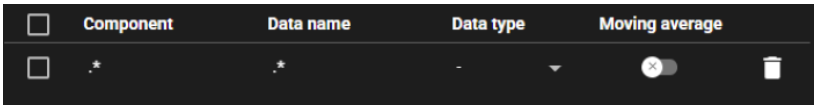
Item	Description
Name	Enter a name for the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

6. Select a Setting option from the following.



Item	Description
Receive and set data definitions	Displays a list of data received at the input port for individual settings
Set by regular expression	Makes settings for the data received at the input port that matches a regular expression entered

7. Enter the following items for the data to be set only when the Setting option is Set by regular expression.



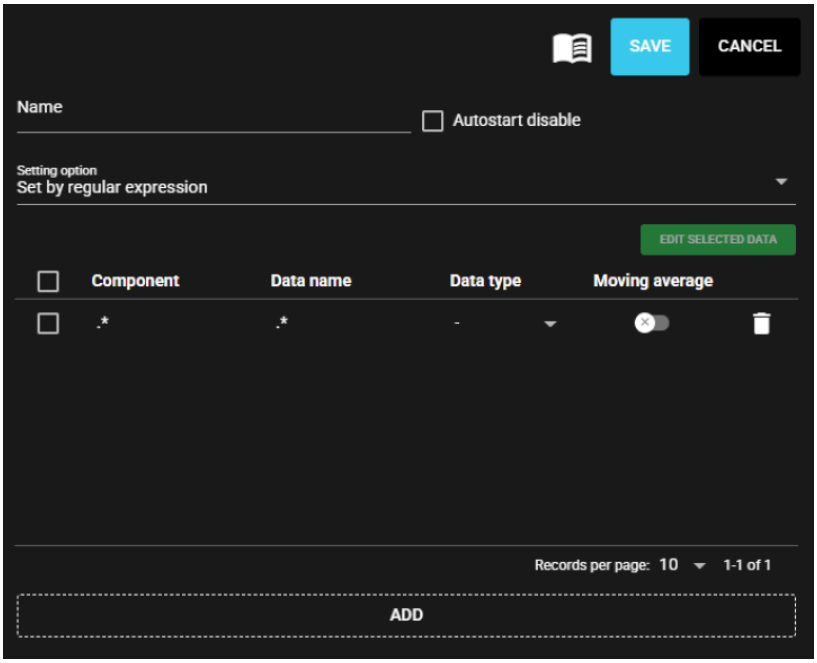
Item	Description
Component	Regular expression to indicate the component name of the data received at the input port that is to be set
Data Name	Regular expression to indicate the data name of the data received at the input port that is to be set
Data Type	Data type of the data received at the input port that is to be set

Regular expression input examples

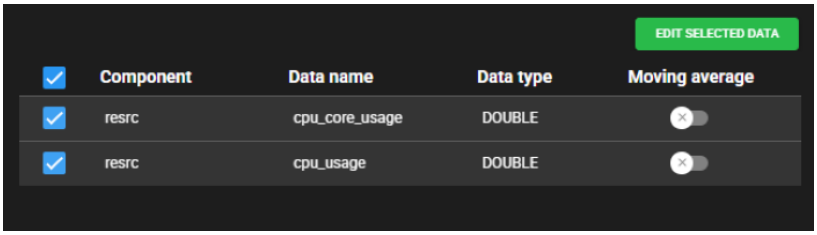
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

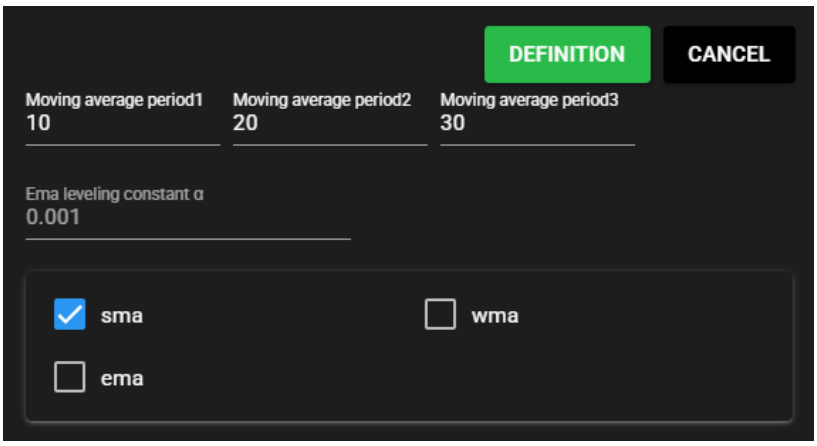
8. Do one of the following to open the moving average setting dialog.
   - Press the Moving average icon for the data for which you want to set moving average.



   - Check the data for which you want to set moving average and press Edit selected data. The same settings will be applied to the checked data.

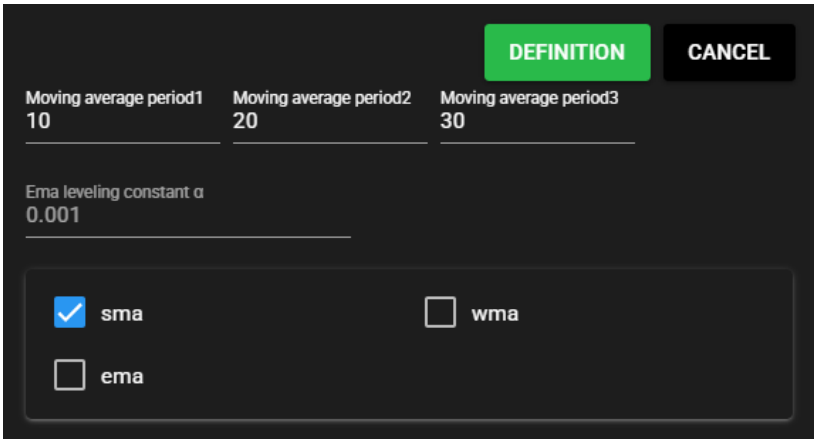


9. Enter the following items to set moving average.

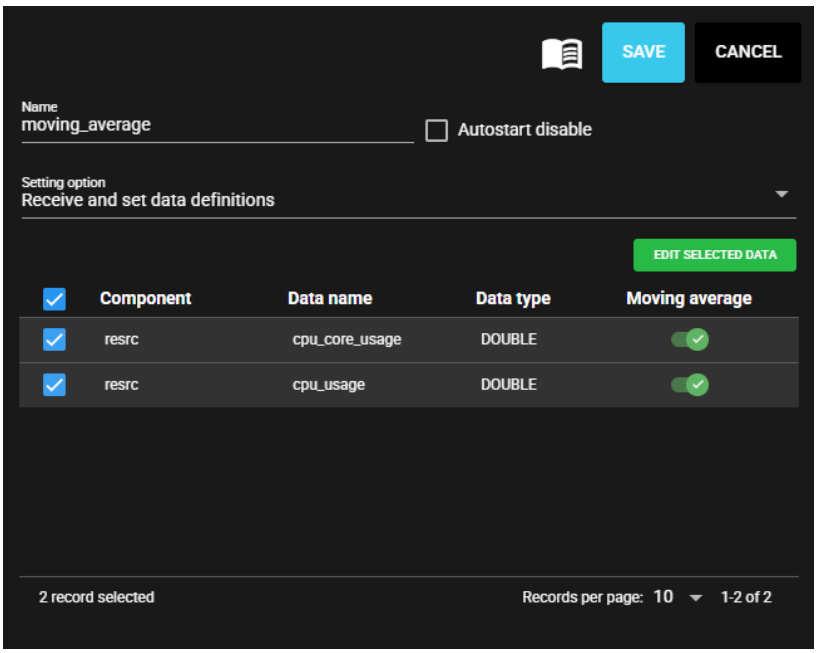


Item	Description
Moving average period 1 to 3	Period covered for computation
Ema leveling constant α	Coefficient for balancing exponentially weighted moving averages (ema) Note: Can be entered only when ema is ON
sma	Simple average of the data in the period
wma	Average computed by assigning linear weight to each data in the period
ema	Average computed by assigning exponential weight to the data for all periods

10. Press Definition.



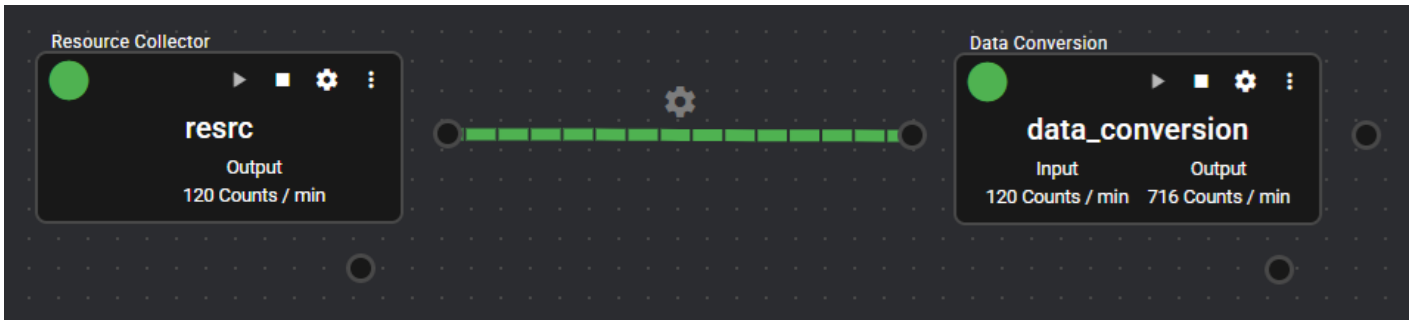
11. Press Save.



12. Start the component. For more information on how to start components, see Start/Stop Components.

5.8.4 Data Conversion

This component retrieves various types of data from the input port column, and outputs it to the output port column after various transformations.



The data conversion functions currently provided are as follows.

Conversion function	Description
Array split	Decomposes the data in an array column into individual elements and registers them in individual columns. The name of each column after decomposition is automatically set by default. [Target column] INT8/INT16/INT32/INT64 array UINT8/UINT16/UINT32/UINT64 array BOOLEAN/FLOAT/DOUBLE array
Bit Conversion	Converts bitN through bitM of the column data to integers and registers them in the column. For an integer scalar, bitN through bitM of the integer are treated as is. For a Boolean array, each element [N] through [M] of the array is treated as bitN through bitM. For INT8/UINT8 type arrays, each element, which has 8 bits, is converted to bit with 0 or !0, and each element [N] through [M] are treated as bitN through bitM. [Target column] Integer scalar BOOLEAN type array INT8/UINT8 type array (where each element is treated as 0 or 1)
Change Data Name	Registers the input data as is as separate data with a set name in a column. [Target column] No restrictions on input data

5.8.4.1 Basic Information

The basic information of the data conversion is as follows.

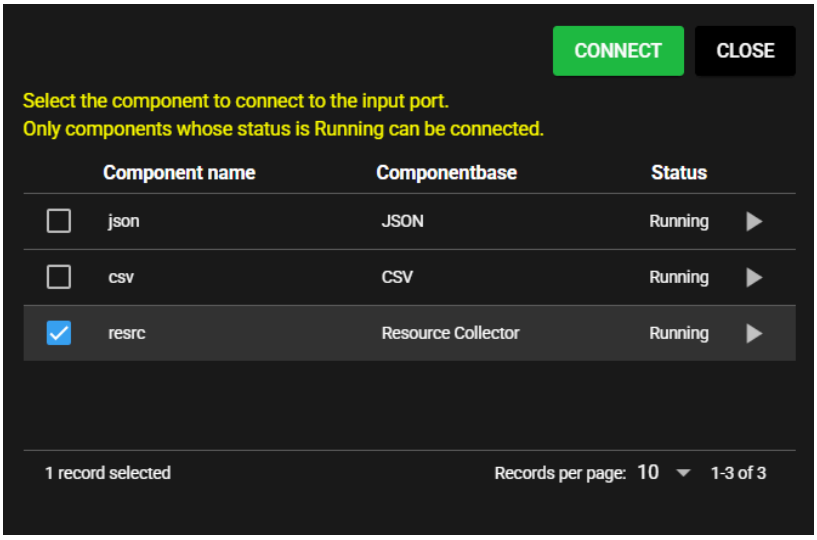
Item	Description
Base name	Data Conversion
Description	Retrieves various types of data and outputs it after various transformations.
Consumption point	2

Item	Description
Port	Input: 1 Receives data from other components. Output: 1 Outputs the converted values of the input data based on the settings specified by the Setting Method. System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.8.4.2 Setting Method (Common)

You can configure data conversion by following the steps below.

1. Add any component that has an output port. For more information on how to add components, see Add Components.
2. Start the added component to output data. For more information on how to start components, see Start/Stop Components.
3. Add data conversion. For more information on how to add components, see Add Components.
4. Select the component you added for the component to connect to the input port.

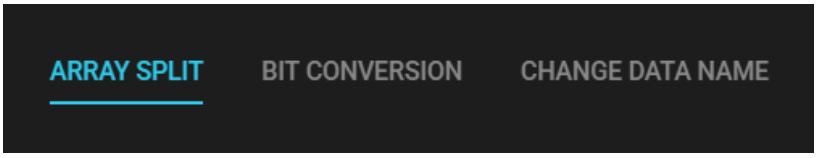


5. Enter the following component information.



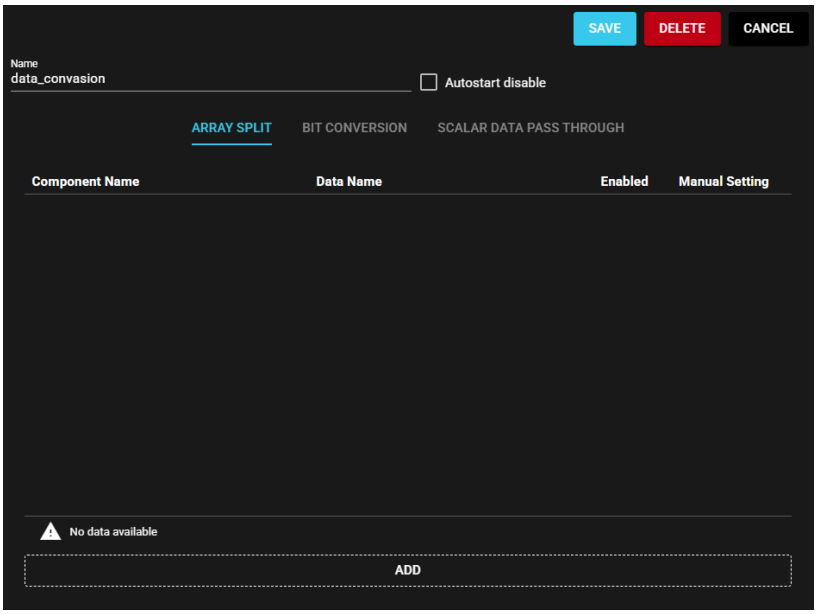
Item	Description
Name	Enter a name for the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.

6. Select the data conversion function you want to use.

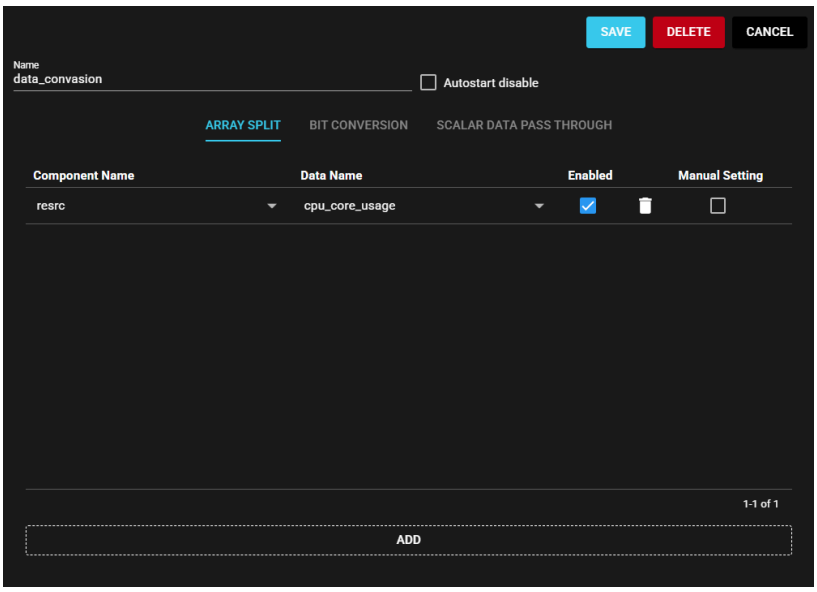


5.8.4.3 Setting Method (Array Split)

1. Press the Add button to add the input data field.

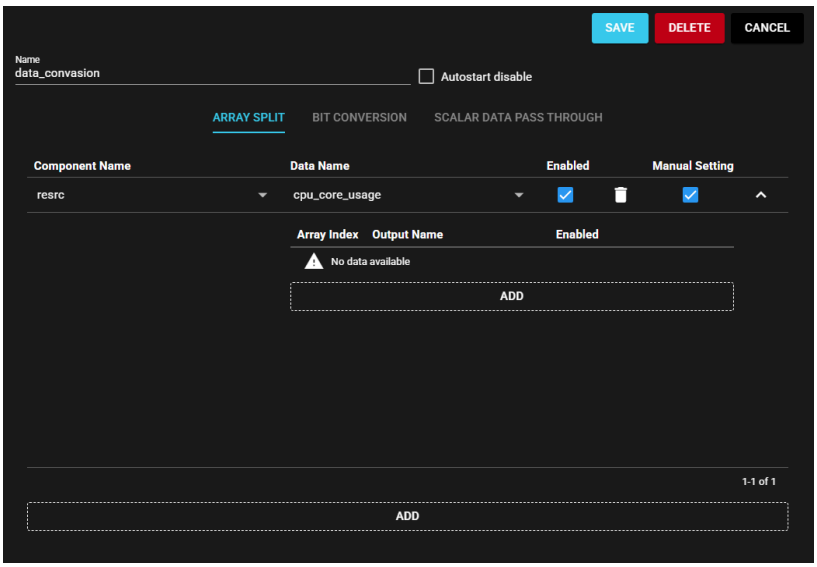


2. Enter the component name and data name to be input.
   (Drop-down selection or manual entry)

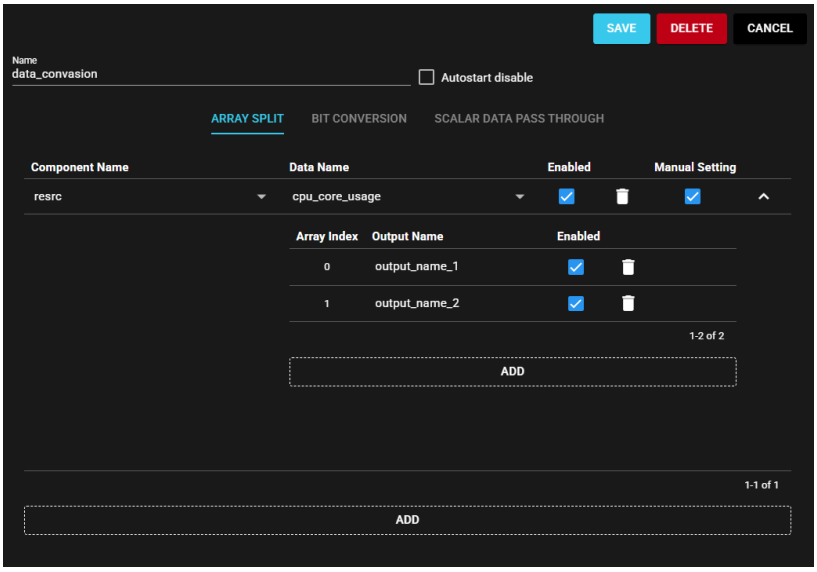


Item	Description
Component Name	Set the component name connected to the input port
Data Name	Set the data name of the component name connected to the input port (Only array columns can be set)
Enable	Set OFF to disable the set input data setting
Delete	Press this to delete the set input data setting
Manual Setting	Set ON when you want to manually set the output data name after array split (When set ON, the expand button of output data setting is enabled.)

3. (Optional) Press the Add button to add the output data field.
   (When the manual setting is set ON and the expand button of the output data setting is pressed, the Add button is displayed)

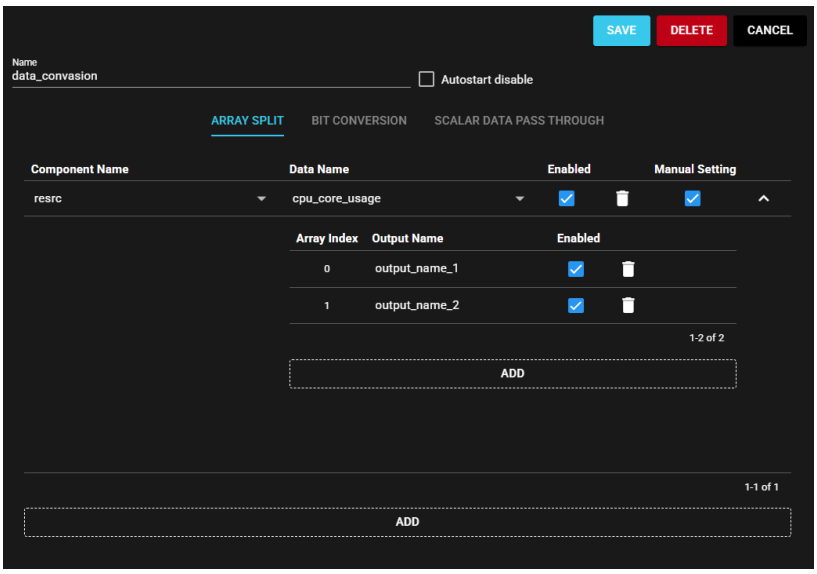


4. (Optional) Enter the data name to be output.



Item	Description
Array Index	The array index is automatically assigned and displayed (Not editable)
Output Name	Set the output data name
Enable	Set OFF to disable the set output data setting
Delete	Press this to delete the set output data setting

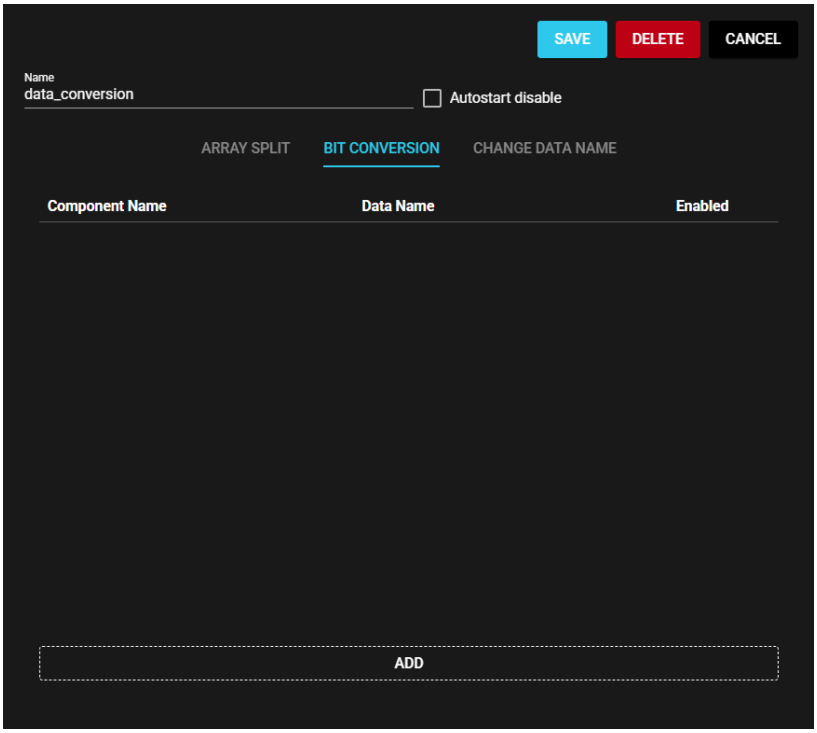
5. Press Save.



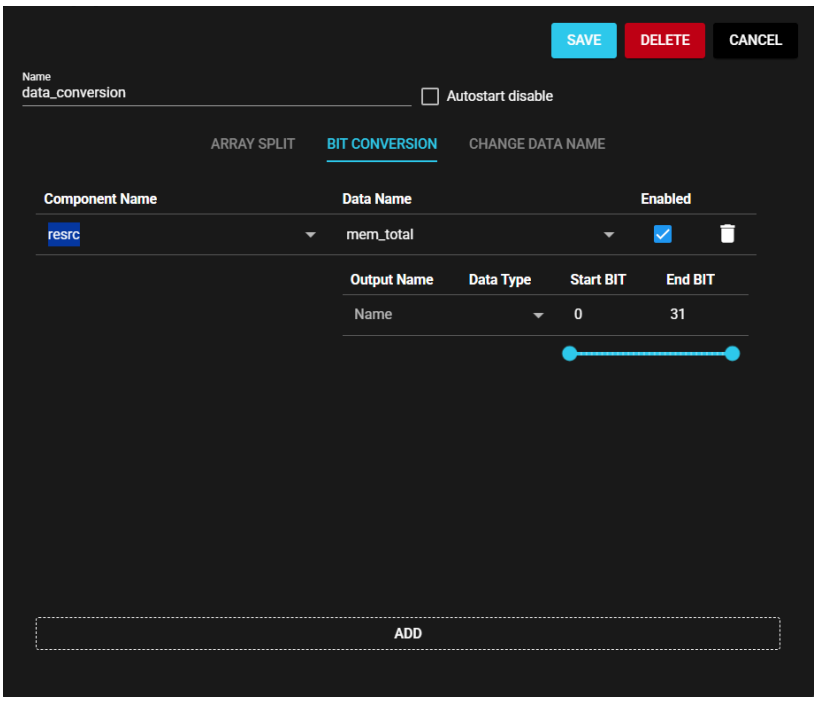
6. Start the component. For more information on how to start components, see Start/Stop Components.

5.8.4.4 Setting Method (Bit Conversion)

1. Press the Add button to add the input data field.

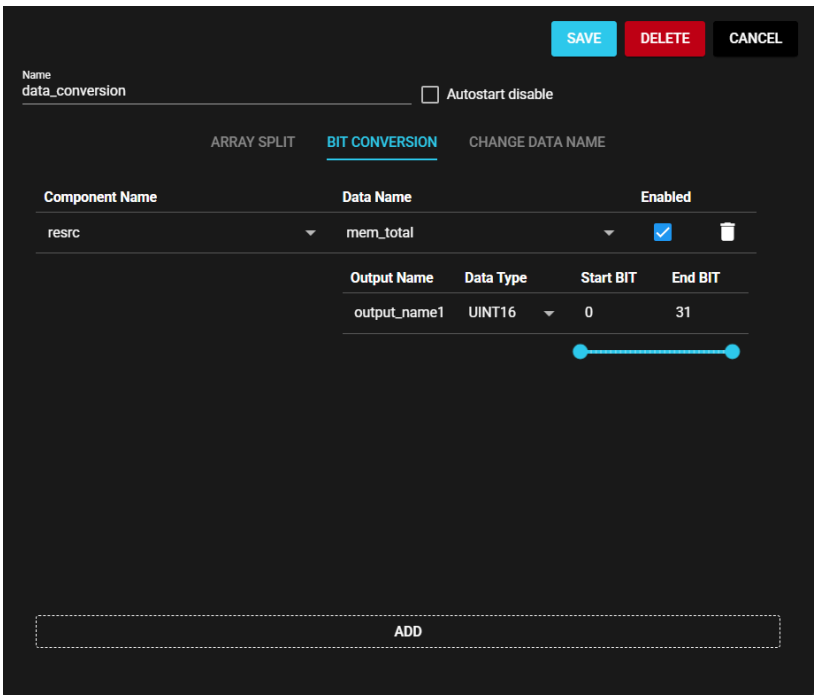


2. Enter the component name and data name to be input.
   (Drop-down selection or manual entry)



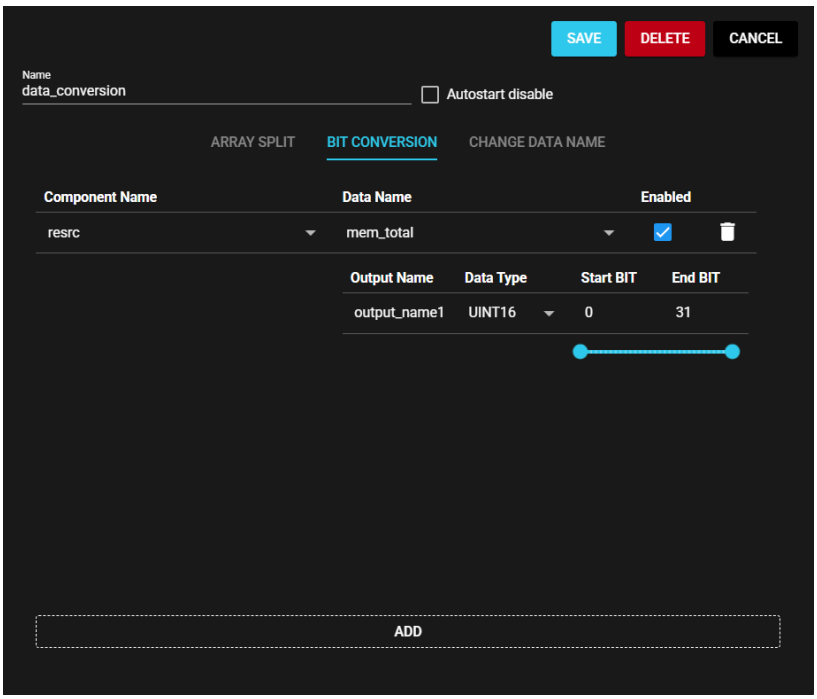
Item	Description
Component Name	Set the component name connected to the input port
Data Name	Set the data name of the component name connected to the input port
Enable	Set OFF to disable the set input data setting
Delete	Press this to delete the set input data setting

3. Enter the settings for the data to be output.



Item	Description
Output Name	Set the output data name
Data Type	Set the data type to be output
Start BIT	Set the start BIT of the data to be retrieved from the input data. Enter it manually or use the bit range setting. The least significant bit is 0, and the most significant bit is 7, 15, 31, or 63, depending on the bit width of the input data type.
End BIT	Set the end BIT of the data to be retrieved from the input data. Enter it manually or use the bit range setting. The least significant bit is 0, and the most significant bit is 7, 15, 31, or 63, depending on the bit width of the input data type.
Bit range setting	This is linked with the start BIT and end BIT, and can be entered by either method. The maximum value of the bit range depends on the input data type for scalar data, and on the number of input arrays for array data.

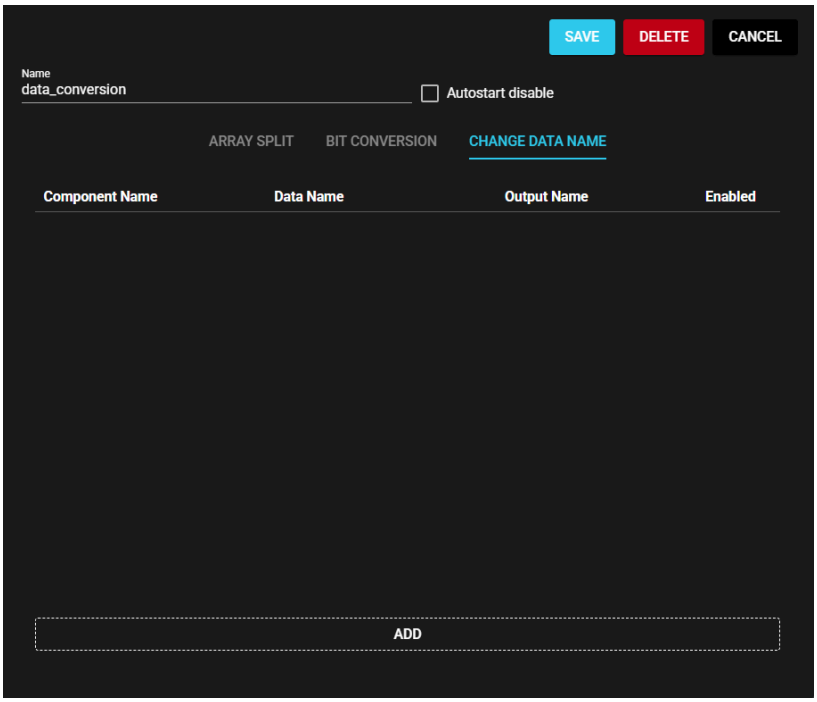
4. Press Save.



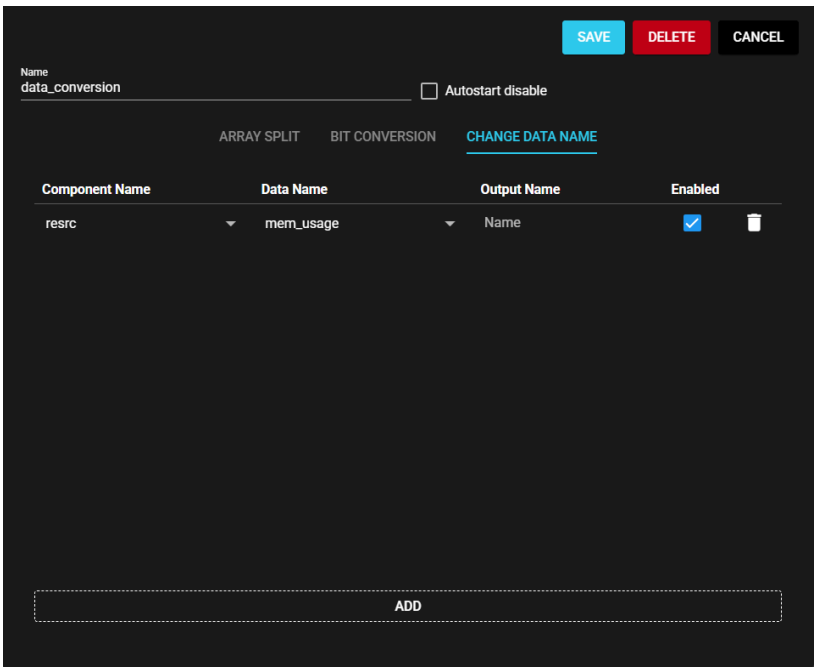
5. Start the component. For more information on how to start components, see Start/Stop Components.

5.8.4.5 Setting Method (Change Data Name)

1. Press the Add button to add the input data field.

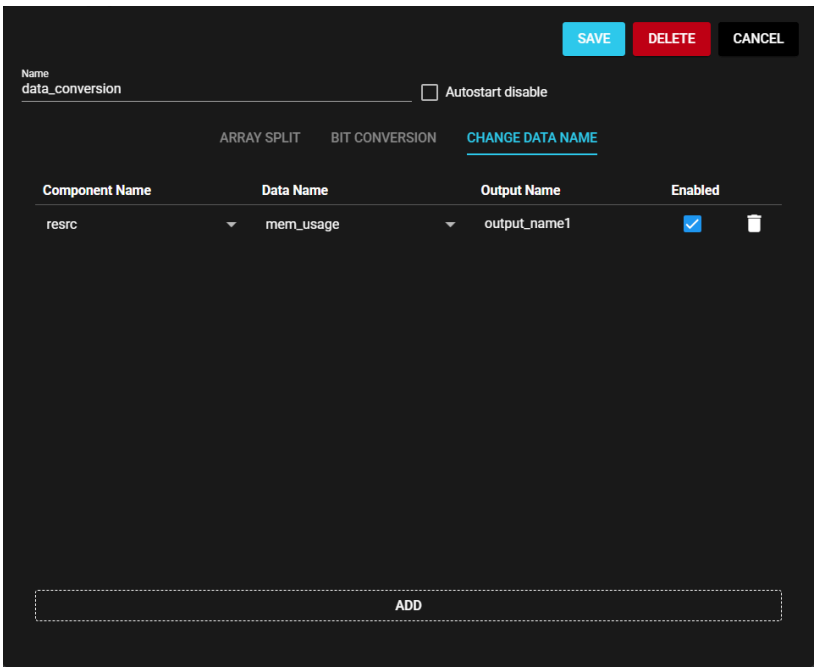


2. Enter the component name and data name to be input.
   (Drop-down selection or manual entry)

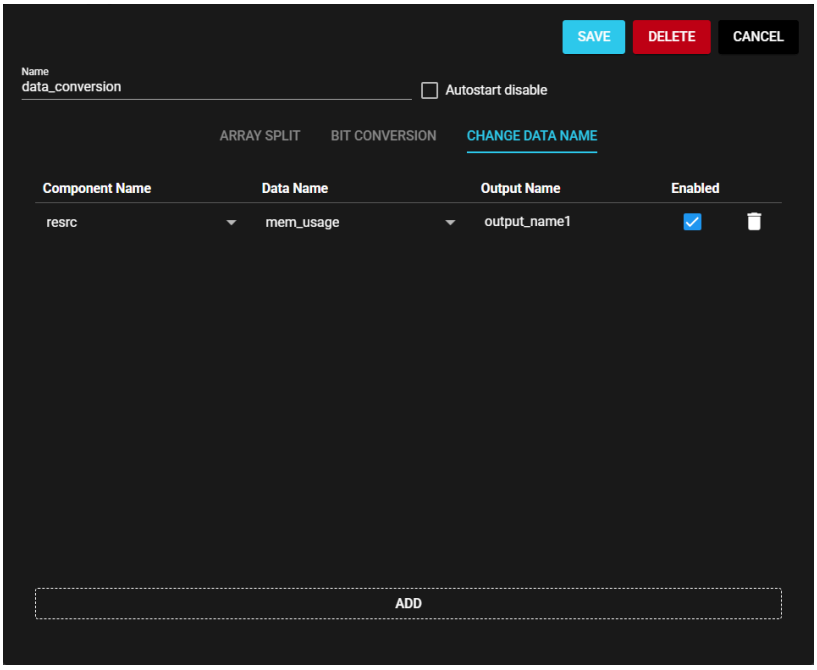


Item	Description
Component Name	Set the component name connected to the input port
Data Name	Set the data name of the component name connected to the input port
Output Name	Set the output data name
Enable	Set OFF to disable the set input data setting
Delete	Press this to delete the set input data setting

3. Enter the data name to be output.



4. Press Save.



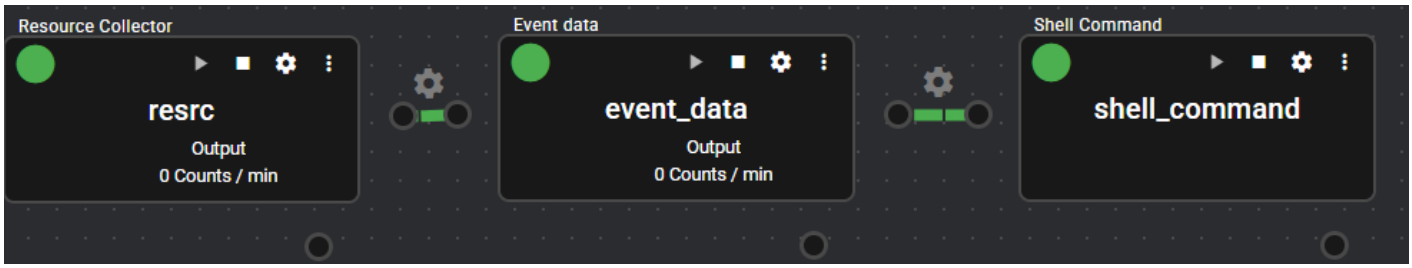
5. Start the component. For more information on how to start components, see Start/Stop Components.

5.9 Action

This page explains the action components.

5.9.1 Shell Command

This component executes any shell command by using collected data.
When the component receives data from the input port, it executes the shell command specified by a parameter.
For the string specifying the shell command, you can use the received data name, type, value, or timestamp.



In the product of OMRON environment, the use of shell commands for general-purpose and practical process integration is not recommended in SpeeDBee Synapse due to security restrictions.

5.9.1.1 Basic Information

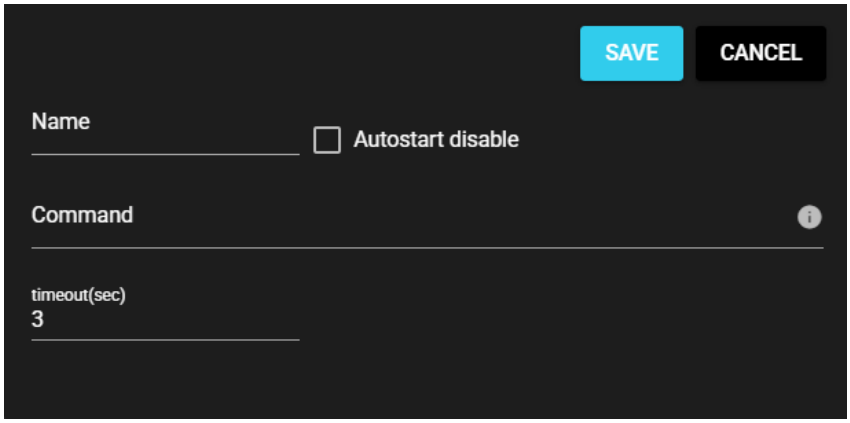
The basic information of Shell Command is as follows:

Item	Description
Base name	Shell Command
Description	Executes the shell command, triggered by data input.
Consumption points	2

Item	Description
Port	Input: 1 Receives data from other components. Output: 0 System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.

5.9.1.2 Settings

The Shell Command settings are as follows:

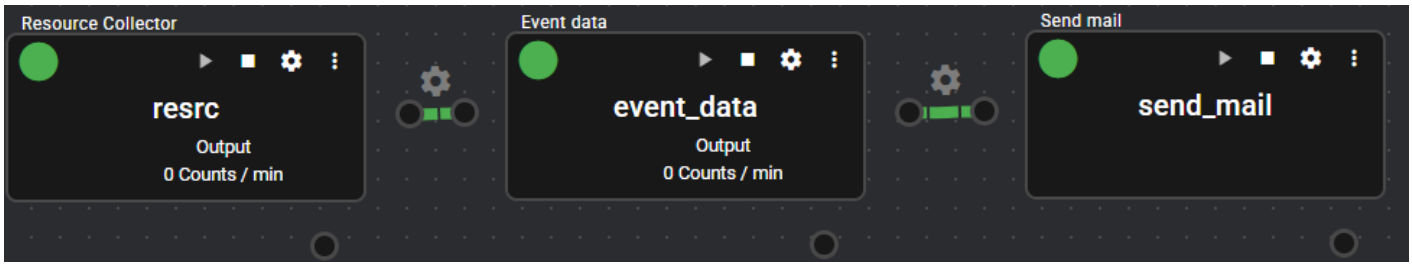


Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Command	Executed shell command Note: 1 Use variables to apply the corresponding information. Note: 2 How to specify the command differs depending on the OS. **Linux** $HIVE_COMPONENT: Source component name of data $HIVE_DATANAME: Data name $HIVE_DATATYPE: Data type $HIVE_DATAVALUE: Value $HIVE_TIMESTAMP: Data timestamp

Item	Description
	$HIVE_DATAARRAYSIZE: Data array size **Windows** %HIVE_COMPONENT%: Source component name of data %HIVE_DATANAME%: Data name %HIVE_DATATYPE%: Data type %HIVE_DATAVALUE%: Value %HIVE_TIMESTAMP%: Data timestamp %HIVE_DATAARRAYSIZE%: Data array size
Timeout(sec)	Enter the timeout duration for command execution.

5.9.2 Send Mail

This component sends e-mails using the collected data.
When the component receives data from the input port, it sends an e-mail with the given information to the specified address.



5.9.2.1 Basic Information

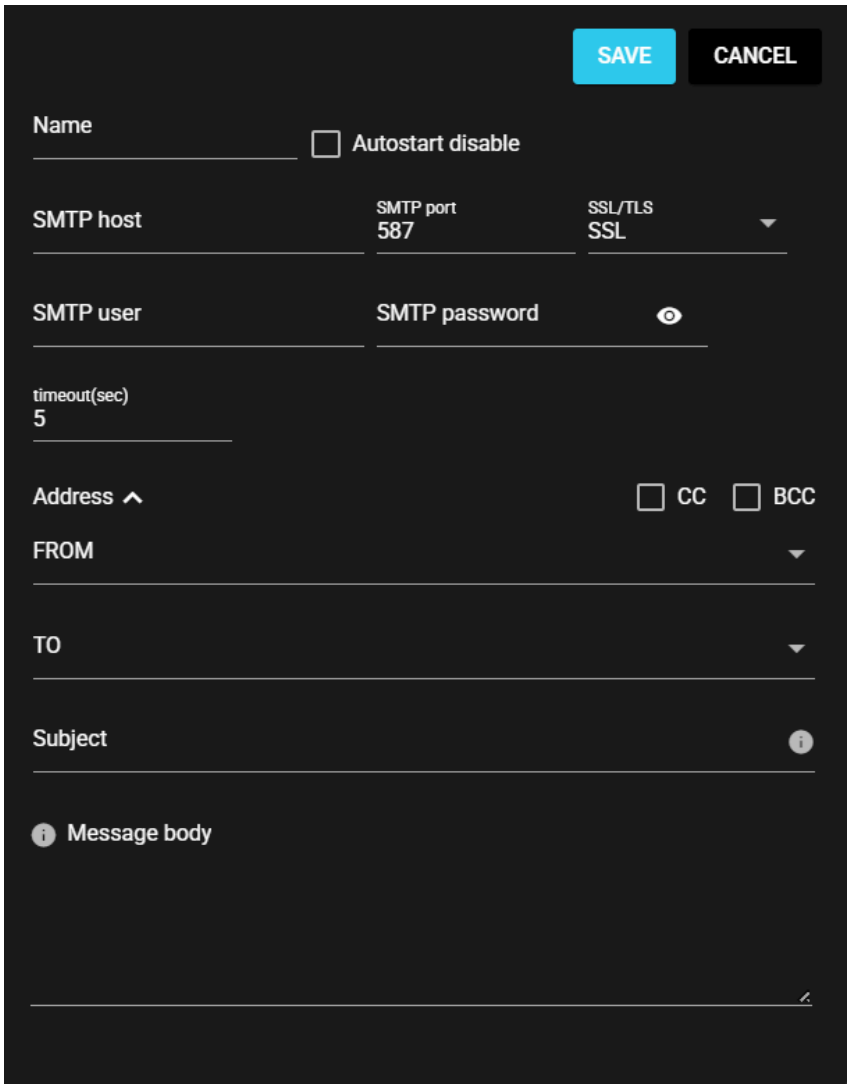
The basic information of Send mail is as follows:

Item	Description
Base name	Send email
Description	Sends an e-mail with the specified content and address, triggered by data input.
Consumption points	2
Port	Input: 1 Receives data from other components.

Item	Description
	Output: 0 System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	You can test connection with the e-mail server. For more information, see Test Connection.

5.9.2.2 Settings

The Send mail settings are as follows:



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
SMTP host	Host name or IP address of the SMTP server for sending e-mails.
SMTP port	Port for SMTP server used for sending e-mails
SSL/TLS	e-mail encryption protocol
SMTP user name	User name required for SMTP authentication for sending e-mails
SMTP password	Password required for SMTP authentication for sending e-mails
Timeout(sec)	Maximum time spent on sending an e-mail
FROM	e-mail address of the sender of the e-mail sent
TO	e-mail address of the recipient of the e-mail sent
CC	e-mail address of the recipient (CC) of the e-mail sent
BCC	e-mail address of the recipient (BCC) of the e-mail sent
Subject	Subject of the e-mail sent Note: Use variables to apply the corresponding information. For more information, see Available Variables for Send mail.
Message body	Message body of the e-mail sent Note: Use variables to apply the corresponding information. For more information, see Available Variables for Send mail.

5.9.2.2.1 Available Variables for Send mail

Variable name	Description	Example of converted string
$HIVE_COMPONENT	Source component name of data	plc-collector
$HIVE_DATANAME	Data name	D110
$HIVE_DATATYPE	Data type	double
$HIVE_DATAVALUE	Value	123.45
$HIVE_DATAARRAYSIZE	Data array size	0
$HIVE_TIMESTAMP	Data timestamp	1733356200123456789
$HIVE_TS	Data timestamp(in seconds)	1733356200
$HIVE_TS_MSEC	Data timestamp(in milliseconds)	1733356200123
$HIVE_TS_USEC	Data timestamp(in microseconds)	1733356200123456
$HIVE_TS_NSEC	Data timestamp(in nanoseconds)	1733356200123456789
$HIVE_TS_DATE_LOCAL	Date of data timestamp(Local Time Zone)	2024-12-05
$HIVE_TS_DATE_UTC	Date of data timestamp(UTC)	2024-12-04
$HIVE_TS_TIME_LOCAL	Time of data timestamp(Local Time Zone)	08:50:00
$HIVE_TS_TIME_UTC	Time of data timestamp(UTC)	23:50:00
$HIVE_TS_DATETIME_LOCAL	Date and time of data timestamp(Local Time Zone)	2024-12-05T08:50:00
$HIVE_TS_DATETIME_UTC	Date and time of data timestamp(UTC)	2024-12-04T23:50:00
$HIVE_TS_TZ_DIFF	Local time zone difference	+0900

Description of Variable Name

- The variable name HIVE_ can also be written as SYNAPSE_.
- Three formats can be used for describing variables: $HIVE_COMPONENT, ${HIVE_COMPONENT}, and %HIVE_COMPONENT%.

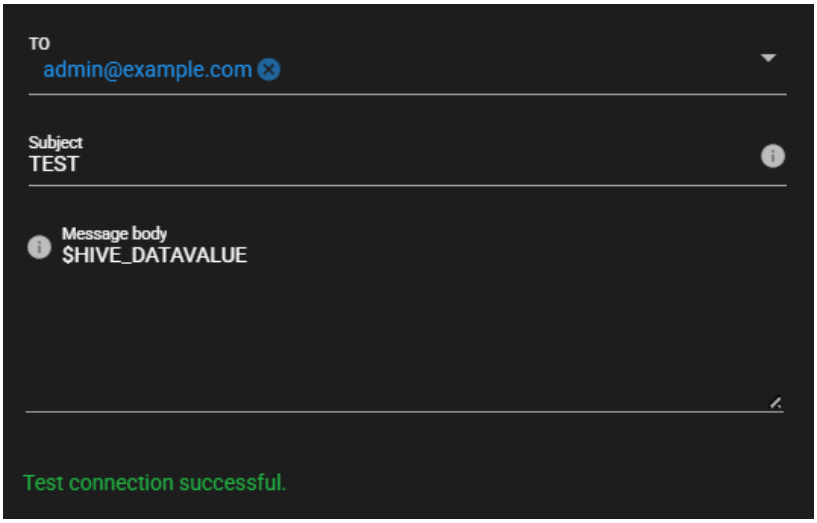
5.9.2.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered Send mail.
2. Press [Test].

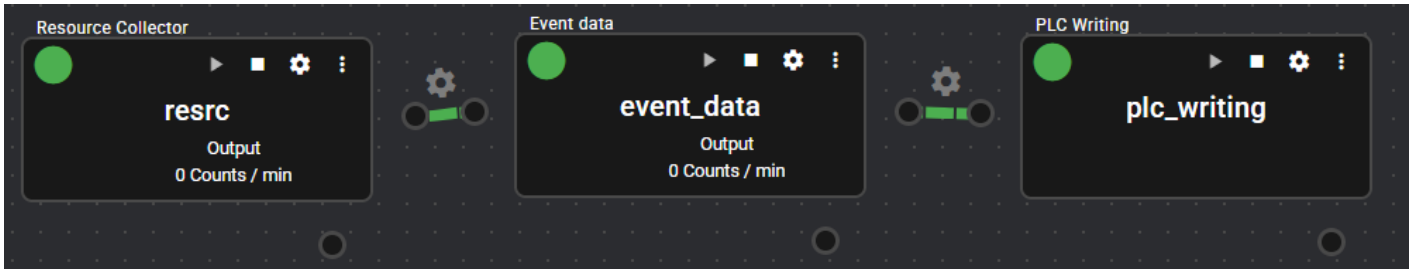


3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.9.3 PLC Writing

This component writes values to PLC registers.
When it receives trigger data from the input port, a value can be written to the specified register of the PLC.
Fix values, trigger data values, and data values of other components can be specified as values to be written.



5.9.3.1 Basic Information

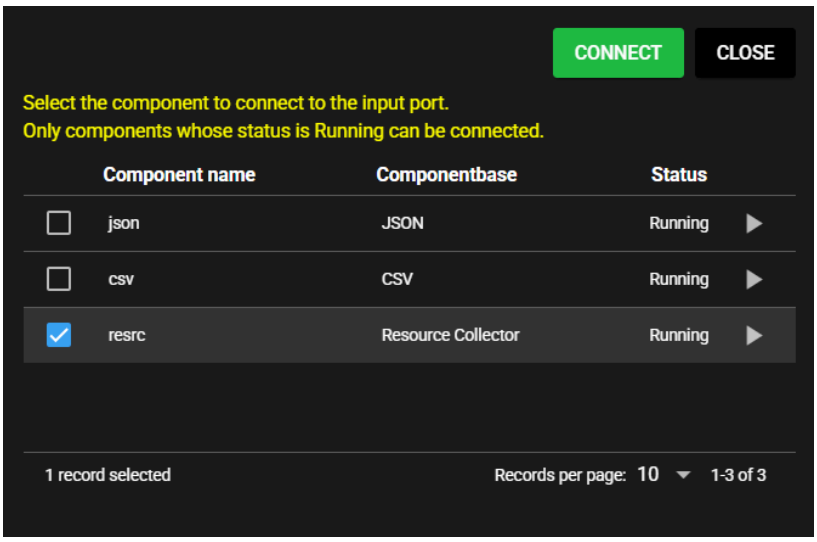
The basic information of PLC Writing is as follows:

Item	Description
Base name	PLC Writing
Description	Writes a value to the specified register of the PLC, triggered by data input.
Consumption points	2
Port	Input: 1 Receives data from other components. Output: 0 System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	You can test connection with PLCs. For more information, see Test Connection.

5.9.3.2 Setting Method

You can set up PLC Writing by following the steps below:

1. Add a component that has an output port. For instructions on how to add components, see Adding Components.
2. Start the added component and output data. For instructions on how to start components, see Starting/Stopping Components.
3. Add PLC Writing. For instructions on how to add components, see Adding Components.
4. Select the added component as the component to connect to the input port.



5. Enter the following component information.

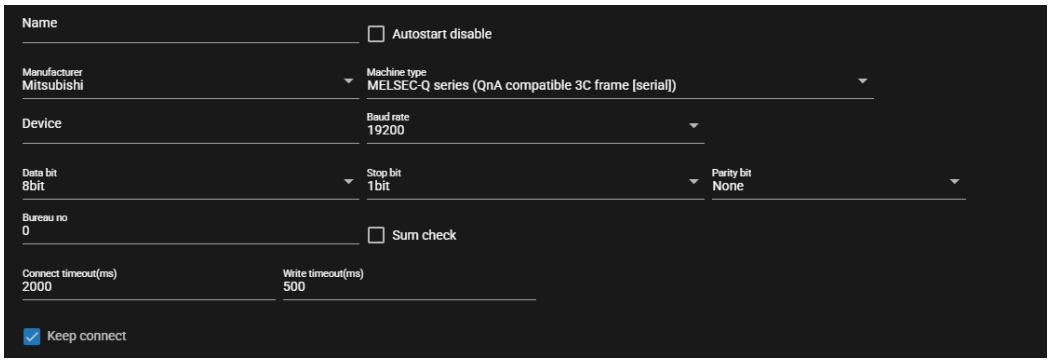
a. TCP/IP Connection



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Manufacturer	Select the manufacturer of the connecting PLC. Note: You can select from the following manufacturers: - Mitsubishi - Jtekt - Omron - Keyence - Panasonic
Machine type	Select the machine type of the connecting PLC. Note: Machine types that can be selected depend on the manufacturer.
Host	Host or IP address of the connecting PLC
Port	Port for connecting PLC

Item	Description
Connect timeout (ms)	Maximum time spent on connecting to a PLC
Write timeout (ms)	Maximum time spent on writing a value
Keep connect	Do not disconnect the connection to the PLC each time a query is made.

b. Serial Connection



Serial Connection

For Omron products, SpeeDBee Synapse does not support serial connections. Therefore, the following models cannot be selected.

- MELSEC-Q series (QnA compatible 3C frame [serial])

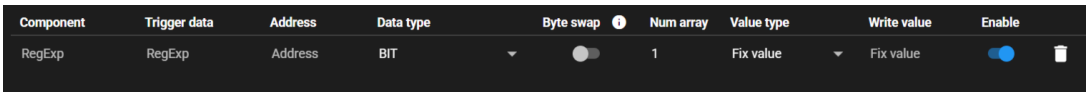
Item	Description
Device	Enter the serial communication device to use
Baud rate	Select the baud rate for serial communication
Data bit	Select the data bits for serial communication
Stop bit	Select the stop bits for serial communication
Parity bit	Select the parity bits for serial communication
Bureau no	Enter the bureau number for Mitsubishi 3C frame
Sum check	Set to ON to perform sum check for Mitsubishi 3C frame

6. Select a trigger specification method from the following:



Item	Description
Set After Receiving Data Definitions	From the list of data received by the input port, select the data that will trigger writing.
Set by regular expression	A regular expression is input, and among data received by the input port, the matching data is the trigger for writing.

7. Enter the following items for the data to be set:



Item	Description
(If the Setting Option is “Set by regular expression”)	
Component	From the data received by the input port, the regular expression that represents the component name for the data to be the trigger for writing
Trigger data	From the data received by the input port, the regular expression that represents the data name for the data to be the trigger for writing
(If the Setting Option is “Set After Receiving Data Definitions”)	
Component	From the data received by the input port, select the component to be the trigger for writing.
Trigger data	From the data received by the input port, select the data to be the trigger for writing.
(Common)	
Address	Address of register to which the value is written. Note: When an address is entered, `(DEC)` / `(HEX)` / `(DDDX)` appears to

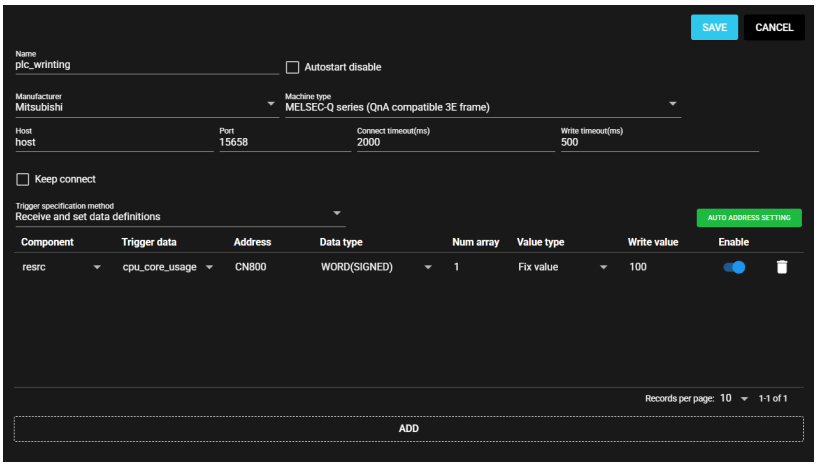
Item	Description
	the right of the address based on the protocol and device type. • Decimal representation: `D00 (DEC)` • Hexadecimal representation: `W00 (HEX)` • Mixed representation (upper digits in decimal, least significant digit in hexadecimal): `X001F (DDDX)` (Example: In `X001F`, the `001` portion is interpreted as decimal, and the least significant digit `F` is interpreted as hexadecimal.) This notation indicates which numeral system is applied to the numeric portion of the entered address.
Program	Program number of register to which the value is written Note: Available only when the manufacturer is Jtekt.
Data type	Data type of register to which the value is written
Byte swap	Set to ON when swapping the order of two bytes within 1WORD value. Note: This operation is only available when the data type is set to `STRING`.
Num array	Range to which value is written from among the consecutive registers starting from the input address
Value type	- Fix value: Enter a fixed value as the Write value. - Component: Select the data received by the input port as the Write value. - Event: Trigger data value is written.
Write value	Enter a value of the selected Value type. Note: Cannot be entered when the Value type is Event.
Enable	To write or not

Examples of Regular Expression Input

Here are examples of regular expression input:

- Matches only “data1”: data1
- Matches only data with “abc” in name: .*abc.*
- Matches only data whose name starts with “abc”: abc.*
- Matches only data whose name ends with “abc”: .*abc

8. Press [Save].



9. Start the component. For instructions on how to start components, see Starting/Stopping Components.

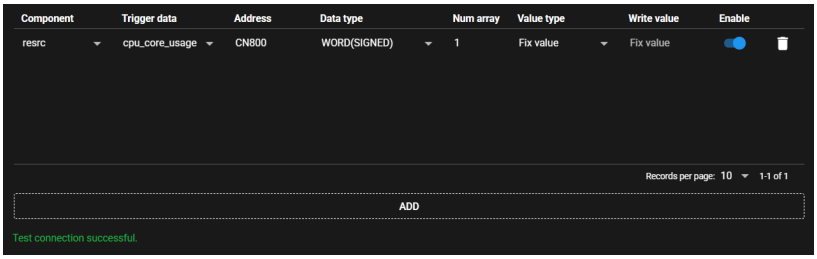
5.9.3.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered PLC Writing.
2. Press [Test].



3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.9.3.4 Auto address setting

With some PLCs, you can set addresses automatically at once by reading the file output by the ladder editing tool.

1. In the ladder editing tool, output the created ladder as a file. The output file format differs depending on the PLC manufacturer and machine type. Sample files are provided below:
    - Mitsubishi
        - MELSEC-iQ-R series (QnA compatible 3E frame)

- MELSEC-Q series (QnA compatible 3E frame)
- MELSEC-F series (A compatible 1E frame)
- MELSEC-Q series (QnA compatible 3C frame[serial])

Download sample file

- Omron
  - SYSMAC CS/CJ/CP/NSJ series (FINS)

Download sample file

- Keyence
  - KV-8000 series (QnA compatible 3E frame)

Download sample file

  - KV-8000 series (Upper Link)

Download sample file

- Panasonic
  - FP7 Series (MEWTOCOL COM)
  - FP7 Series (MEWTOCOL COM FP2 Compatible)

Download sample file

  - FP7 Series (QnA Compatible 3E Frame)

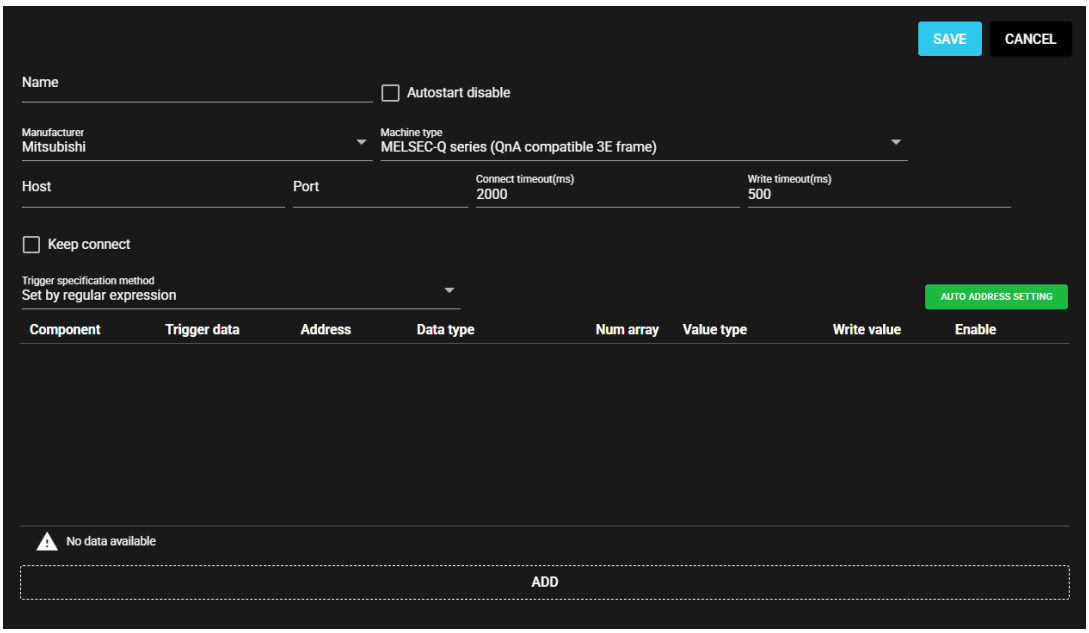
Download sample file

About the Address Auto-Configuration File for Certain Machine type

The following models support the same format as the address auto-configuration file for Mitsubishi. However, this format cannot be directly output from each manufacturer's ladder editing software. Therefore, it is necessary to prepare a file in the same format separately to enable this functionality.

- KV-8000 series (QnA compatible 3E frame)
- FP7 Series (MEWTOCOL COM)
- FP7 Series (MEWTOCOL COM FP2 Compatible)
- FP7 Series (QnA Compatible 3E Frame)

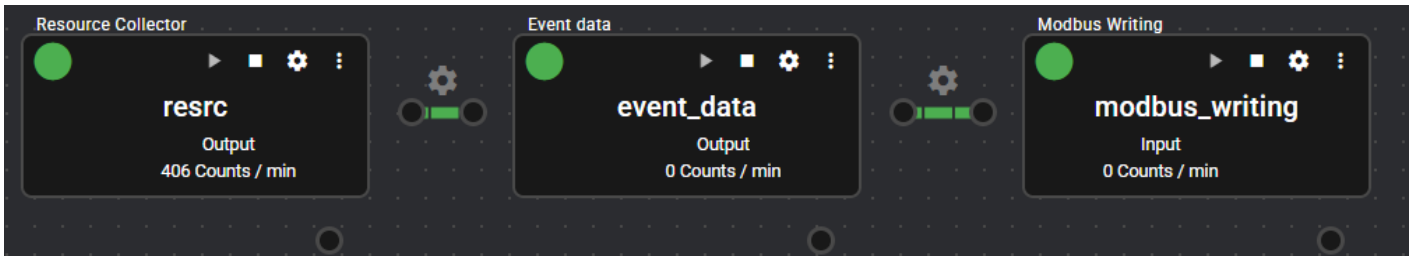
2. Press [Auto address setting].



3. Select the file that you output with the ladder editing tool.
4. The address is automatically set based on the file.

5.9.4 Modbus Writing

This component writes a value to the specified address of a Modbus compatible device.
When trigger data is received from the input port, this component acts as the master of Modbus and writes a value to the specified register of the Modbus slave device.
Fix values, trigger data values, and data values of other components can be specified as values to be written.



5.9.4.1 Basic Information

The basic information of Modbus Writing is as follows:

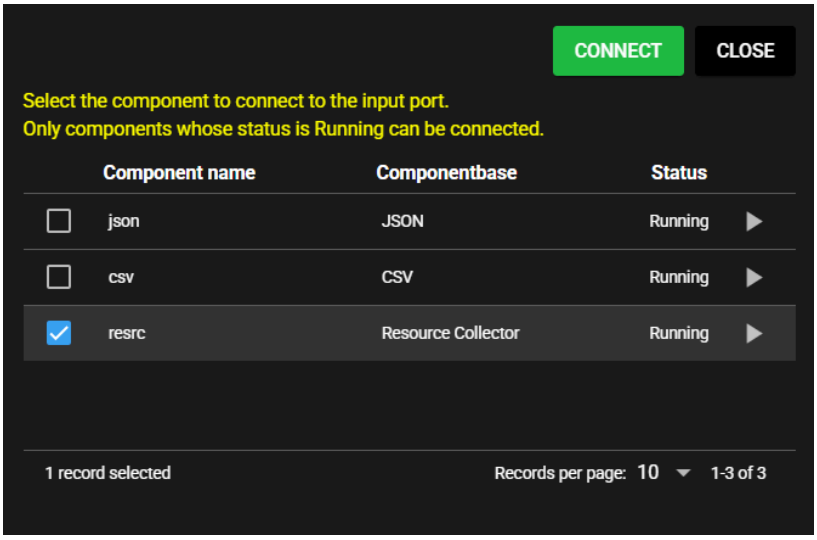
Item	Description
Base name	Modbus Writing
Description	Writes a value to the specified register of the Modbus compatible device, triggered by data input.

Item	Description
Consumption points	2
Port	Input: 1 Receives data from other components. Output: 0 System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	You can test connection with Modbus compatible devices. For more information, see Test Connection.

5.9.4.2 Setting Method

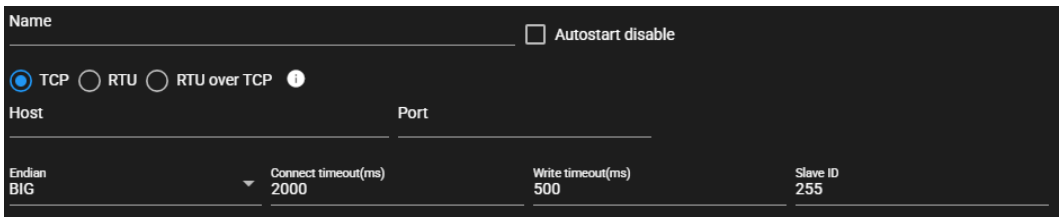
You can set up Modbus Writing by following the steps below:

1. Add a component that has an output port. For instructions on how to add components, see Adding Components.
2. Start the added component and output data. For instructions on how to start components, see Starting/Stopping Components.
3. Add Modbus Writing. For instructions on how to add components, see Adding Components.
4. Select the added component as the component to connect to the input port.



5. Enter the following component information.

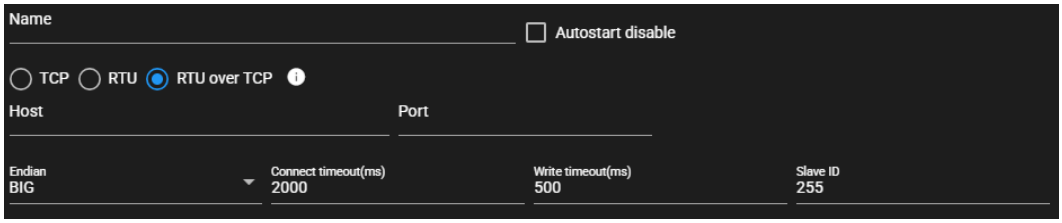
- When TCP is selected as the communication method



- When RTU is selected as the communication method



- When RTU over TCP is selected as the communication method



Communication Method

For Omron products, SpeeDBee Synapse does not support serial connections. Therefore, the following communication method cannot be selected.

- RTU

RTU over TCP Settings

With RTU over TCP, enter the pass-through unit connection information (host, port), and configure serial communication parameters such as communication speed and data bits on the pass-through unit side. Since the configuration screen and setting items differ depending on the pass-through unit product, please refer to the manual for each device.

※Verified pass-through units:

- ADAM-4571
- GW DEVICE SERVER 1E/1DB9

Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Communication method	Select TCP, RTU, or RTU over TCP
Host	Host name or IP address of the connecting device ※Available when communication method is TCP or RTU over TCP
Port	Port for the connecting device ※Available when communication method is TCP or RTU over TCP
Device	Serial communication device to use ※Available when communication method is RTU
Baud rate	Baud rate for serial communication ※Selectable when communication method is RTU
Serial mode	Serial communication mode ※Selectable when communication method is RTU
Data bit	Data bits for serial communication ※Selectable when communication method is RTU
Stop bit	Stop bits for serial communication ※Selectable when communication method is RTU
Parity bit	Parity bits for serial communication ※Selectable when communication method is RTU
Endian	Byte order of the value to be written
Connect timeout (ms)	Maximum time spent on connecting to a device
Write timeout (ms)	Maximum time spent on writing a value
Slave ID	Slave ID of connecting device

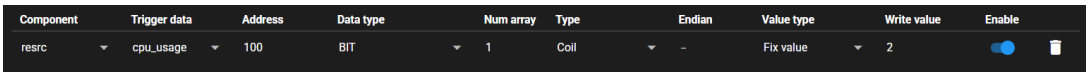
6. Select a trigger specification method from the following:

Trigger specification method

Receive and set data definitions

Item	Description
Set After Receiving Data Definitions	From the list of data received by the input port, select the data that will trigger writing.
Set by regular expression	A regular expression is input, and among data received by the input port, the matching data is the trigger for writing.

7. Enter the following items for the data to be set:



Item	Description
(If the Setting Option is “Set by regular expression”)	
Component	From the data received by the input port, the regular expression that represents the component name for the data to be the trigger for writing
Trigger data	From the data received by the input port, the regular expression that represents the data name for the data to be the trigger for writing
(If the Setting Option is “Set After Receiving Data Definitions”)	
Component	From the data received by the input port, select the component to be the trigger for writing.
Trigger data	From the data received by the input port, select the component to be the trigger for writing.
(Common)	
Type	Type of register to which the value is written

Item	Description
Address	Address of register to which the value is written Note: According to Modbus specifications, the address range is usually set to "1 to 65536," but when setting in SpeeDBee Synapse, it is necessary to set an address range shorter by 1, specifically, as "0 to 65535."
Data type	Data type of register to which the value is written
Num array	Range to which value is written from among the consecutive registers starting from the input address
Endian	Byte order of data to be written Note: If "-," the endianness of the entire data is applied.
Value type	- Fix value: Enter a fixed value as the Write value. - Component: Select the data received by the input port as the Write value. - Event: Trigger data value is written.
Write value	Enter a value of the selected Value type. Note: Cannot be entered when the Value type is Event.
Enable	To write or not

Examples of Regular Expression Input

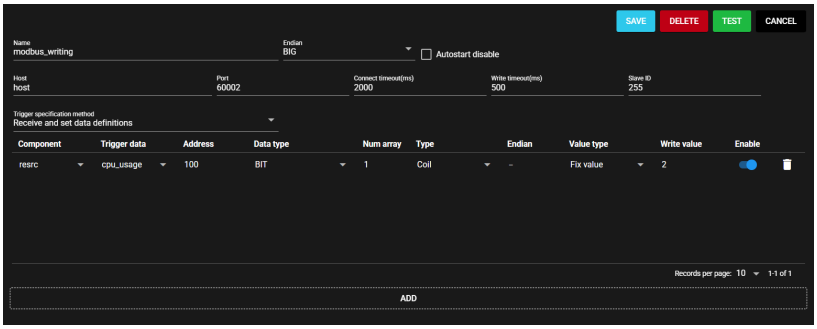
Here are examples of regular expression input:

- Matches only “data1”: data1
- Matches only data with “abc” in name: .*abc.*
- Matches only data whose name starts with “abc”: abc.*
- Matches only data whose name ends with “abc”: .*abc

Hexadecimal Address Notation

Address can be specified in hexadecimal format by prefixing with `0x`.
Example: 0x64 (decimal 100), 0xffff (decimal 65535)

8. Press [Save].



9. Start the component. For instructions on how to start components, see Starting/Stopping Components.

5.9.4.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered Modbus Writing.
2. Press [Test].

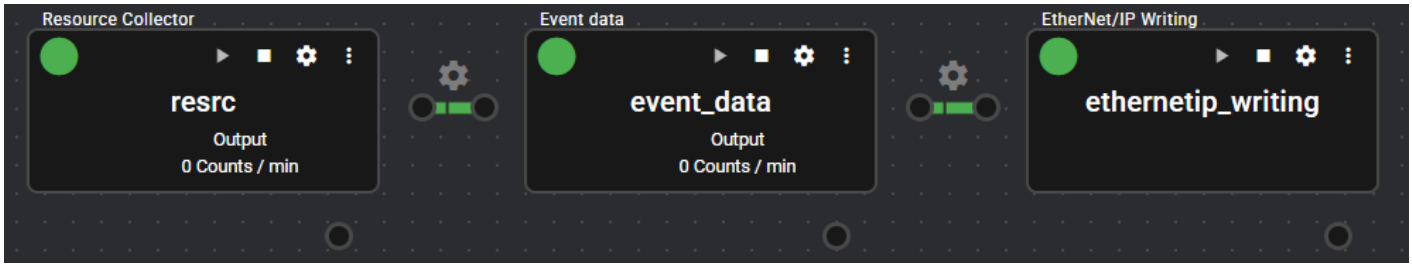


3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.9.5 EtherNet/IP Writing

This component writes a value to the specified address of an EtherNet/IP-enabled device.
When the component receives trigger data from the input port, it acts as an originator, connects to the EtherNet/IP-enabled device, and writes an arbitrary value to the specified variable.



5.9.5.1 Basic Information

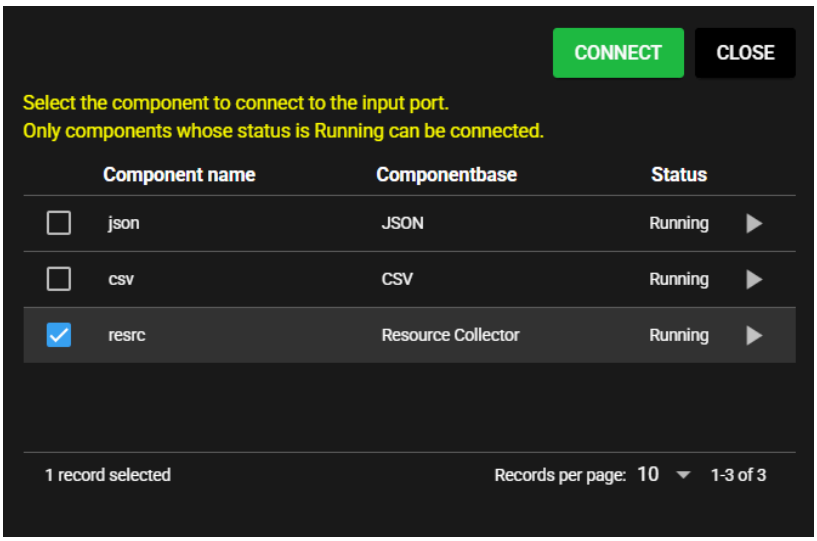
The basic information of EtherNet/IP Writing is as follows:

Item	Description
Base name	EtherNet/IP Writing
Description	Writes a value to the specified variable of the EtherNet/IP-enabled device, triggered by data input.
Consumption points	2
Port	Input: 1 Receives data from other components. Output: 0 System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.
Other	Variables can be set automatically. For more information, see Auto variable setting.

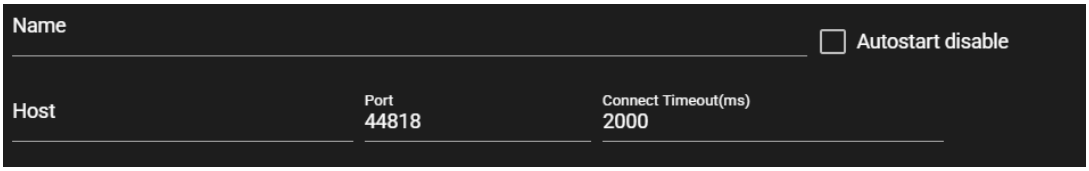
5.9.5.2 Setting Option

You can set up EtherNet/IP Writing by following the steps below:

1. Add a component that has an output port. For instructions on how to add components, see Adding Components.
2. Start the added component and output data. For instructions on how to start components, see Starting/Stopping Components.
3. Add EtherNet/IP Writing. For instructions on how to add components, see Adding Components.
4. Select the added component as the component to connect to the input port.

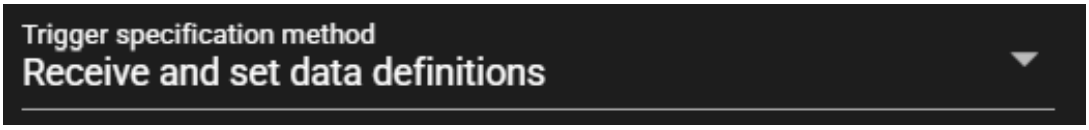


5. Enter the following component information.



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Host	Enter the host name or IP address of the EtherNet/IP-enabled device.
Port	Enter the port for the EtherNet/IP-enabled device.
Connect timeout(ms)	Enter the timeout duration for connecting.

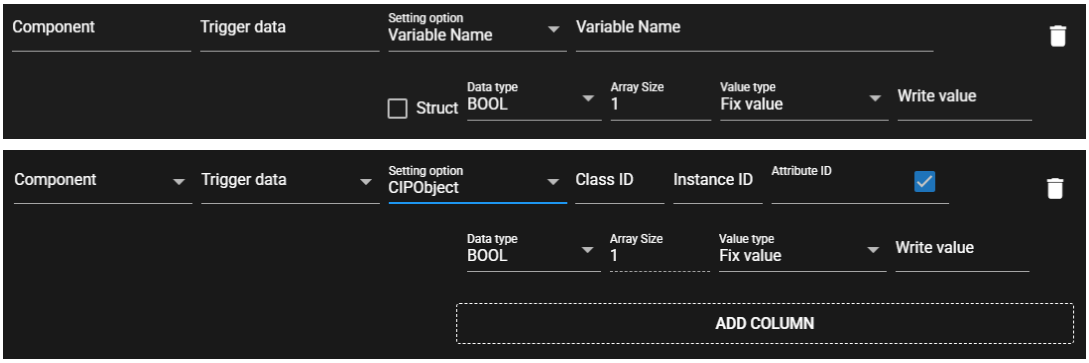
6. Select a trigger specification method from the following:



Item	Description
Set After Receiving Data Definitions	From the list of data received by the input port, select the data that will trigger writing.

Item	Description
Set by regular expression	A regular expression is input, and among data received by the input port, the matching data is the trigger for writing.

7. Enter the following items for the data to be set:



Item	Description
(If the Setting Option is “Set by regular expression”)	
Component	From the data received by the input port, the regular expression that represents the component name for the data to be the trigger for writing
Trigger data	From the data received by the input port, the regular expression that represents the data name for the data to be the trigger for writing
(If the Setting Option is “Set After Receiving Data Definitions”)	
Component	From the data received by the input port, select the component to be the trigger for writing.
Trigger data	From the data received by the input port, select the component to be the trigger for writing.
(Common)	
Setting Option	Select the setting option for the data to be acquired.
Variable name	Enter a tag name. Note: This item can be entered only when the Setting Option is Variable

Item	Description
	Name.
Struct	Set to ON to register multiple members. Note: This item can be set only when the Setting Option is Variable Name.
Class ID	Enter a Class ID. Note: This can be entered only when the Setting Option is CIPObject.
Instance ID	Enter an Instance ID. Note: This can be entered only when the Setting Option is CIPObject.
Attribute ID	Enter an Attribute ID. Note: This can be entered only when the Setting Option is CIPObject.
Attribute settings	Check when entering Attribute ID. Note: This can be manipulated only when the Setting Option is CIPObject.
Data type	Data type of member
Array size	Array size of member Note: This item can be manipulated only when the Setting Option is Variable Name.
Value type	- Fix value: Enter a fixed value as the Write value. - Component: Select the data received by the input port as the Write value. - Event: Trigger data value is written.
Write value	Enter a value of the selected Value type. Note: Cannot be entered when the Value type is Event.

Examples of Regular Expression Input

Here are examples of regular expression input:

- Matches only “data1”: data1
- Matches only data with “abc” in name: .*abc.*
- Matches only data whose name starts with “abc”: abc.*
- Matches only data whose name ends with “abc”: .*abc

Hexadecimal ID Notation

Class ID, Instance ID, and Attribute ID can be specified in hexadecimal format by prefixing with `0x`.
Example: 0x64 (decimal 100), 0xffff (decimal 65535)

8. Press [Save].



9. Start the component. For instructions on how to start components, see Starting/Stopping Components.

5.9.5.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered EtherNet/IP Writing.
2. Press [Test].



3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.



5.9.5.4 Auto variable setting

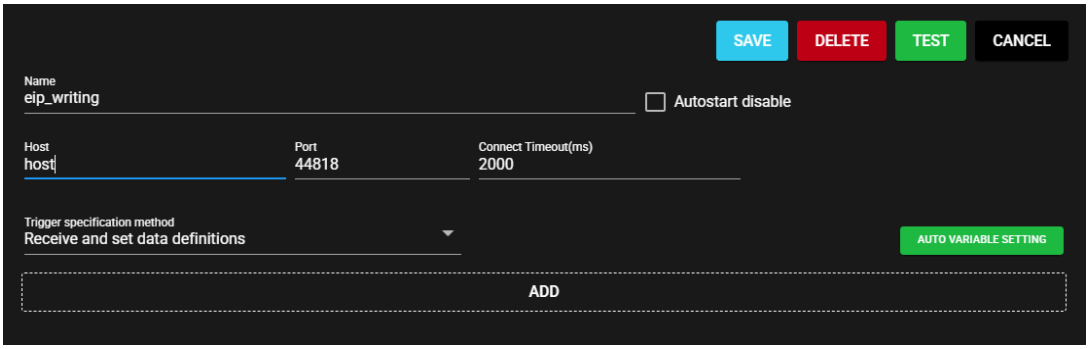
Variables can be automatically set by converting the global variable information exported by Sysmac Studio to a text file into a specified format and reading it.

1. Convert the global variable information exported by Sysmac Studio to a text file with the following column information:
   - HOST
   - NAME
   - DATATYPE
   - ADDRESS
   - COMMENT
   - TAGLINK
   - RW
   - POU

   Example

```
HOST    NAME    DATATYPE    ADDRESS COMMENT TAGLINK RW  POU
        b_test  BOOL        test1           TRUE    RW
        i_test  UINT        test2           TRUE    RW
        s_test  STRING(256) test3           TRUE    RW
```

2. Press [AUTO VARIABLE SETTING].

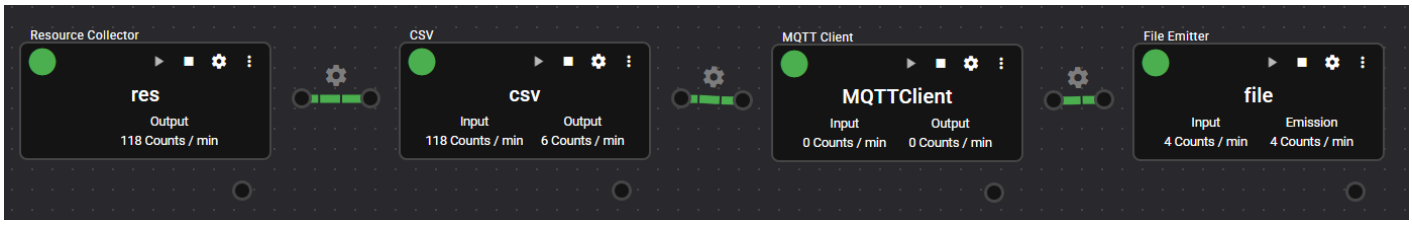


3. Select the prepared text file.

4. The variable is automatically set based on the file.

5.9.6 MQTT Client

This component performs publish/subscribe to the same MQTT broker.
When file type data is received from the input port, its contents are published to the MQTT broker.
Subscribed data is output from the output port as file type data.



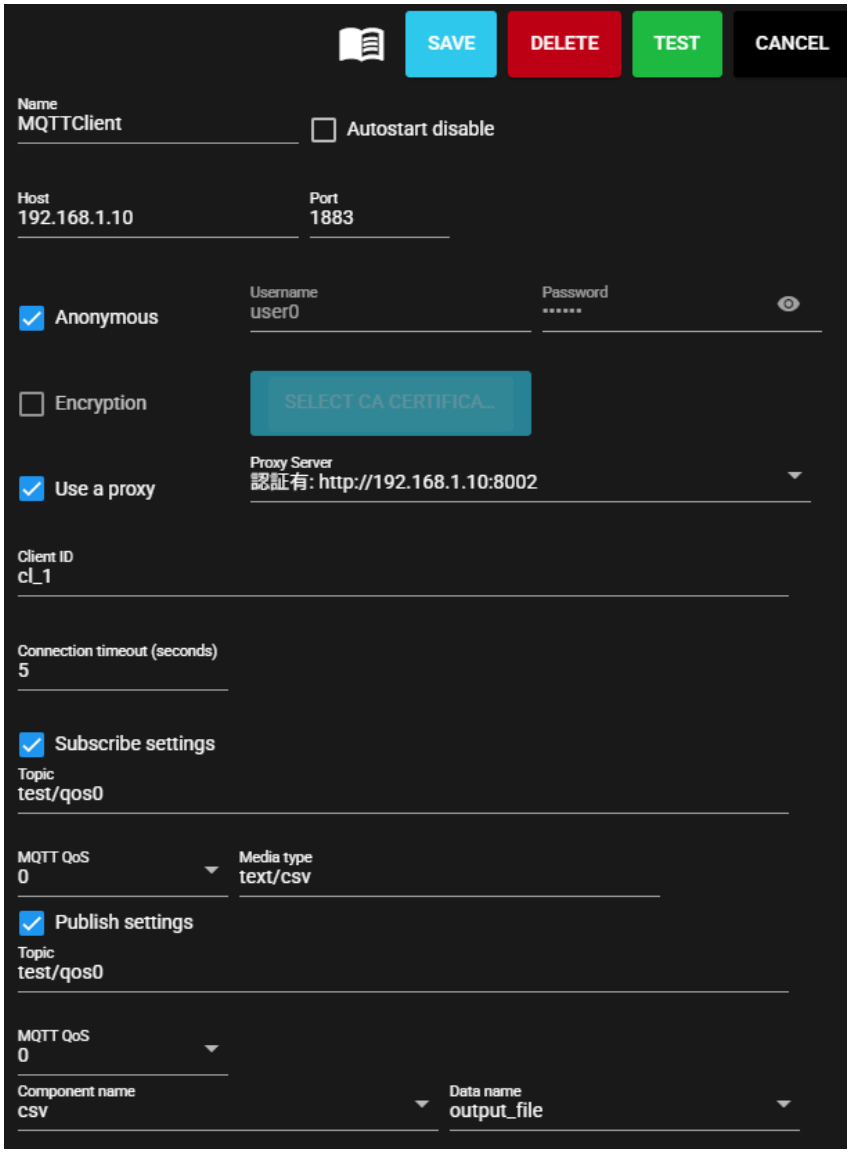
5.9.6.1 Basic Information

The basic information of MQTT Client is as follows:

Item	Description
Base name	MQTT Client
Description	A component that connects to an MQTT broker and performs bidirectional data transmission and reception. When data is received from the input port, the data is published with the specified topic name. Also, it subscribes to the specified topic name and outputs the received data from the output port.
Consumption points	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs subscribed data. System: 1 Outputs the component's operating status and error information. For details on the output error information, see List of Error Codes.

5.9.6.2 Settings

The MQTT Client settings are as follows:



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Host	Host name or IP address of the MQTT broker
Port	Port number of the MQTT broker
Anonymous	Set to ON when not performing user authentication

Item	Description
Username	Enter authentication user name Note: Can only be entered when Anonymous access is disabled.
Password	Enter authentication password Note: Can only be entered when Anonymous access is disabled.
Encryption	Set to ON to enable encryption.
CA Certificate	Upload the CA Certificate to be used for encrypted communication. If not specified, CA Certificates installed on the sytem will be used. Note: You can only upload it when encryption is enabled.
Use a proxy	Set to ON to communicate with the MQTT broker via a proxy server
Proxy server	Select proxy server Note: The server to be used must be set in advance in Proxy settings.
Client ID	MQTT Client ID Note: May need to be specified depending on the destination specifications. Note: When multiple components connect to the same MQTT broker, ensure that Client IDs do not overlap. Communication will not work properly.
Connection timeout(sec)	Enter the timeout seconds when connecting to the MQTT broker
Subscribe settings	Set to ON to perform subscription.
Topic	Enter the topic to collect
MQTT QoS	Select MQTT Quality of Service
Media type	Specify the media type of the output file.
Publish settings	Set to ON to perform publishing.
Topic	Enter the topic to publish
MQTT QoS	Select MQTT Quality of Service
Component name	Specify the name of the component to collect from the data received by the input port.

Item	Description
Data name	Specify the name of the data to collect from the data received by the input port.

5.9.6.3 Test Connection

You can test connection with the specified destination by following the steps below:

1. Open the settings screen for the registered MQTT Client.
2. Press [Test].



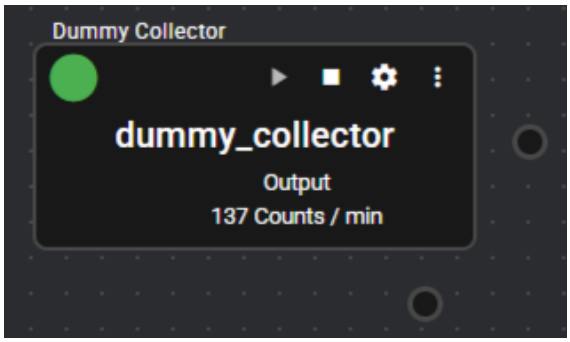
3. Test connection is attempted with the specified destination, and the result is displayed at the bottom of the settings screen.

5.10 Development Components

This section describes development components. The development component is available for use only for SCCDE Test Run.

5.10.1 Dummy Collector

This is a component for registering dummy data.
Based on the detail of designated Operation file (csv), a column is created, and dummy data is registered.



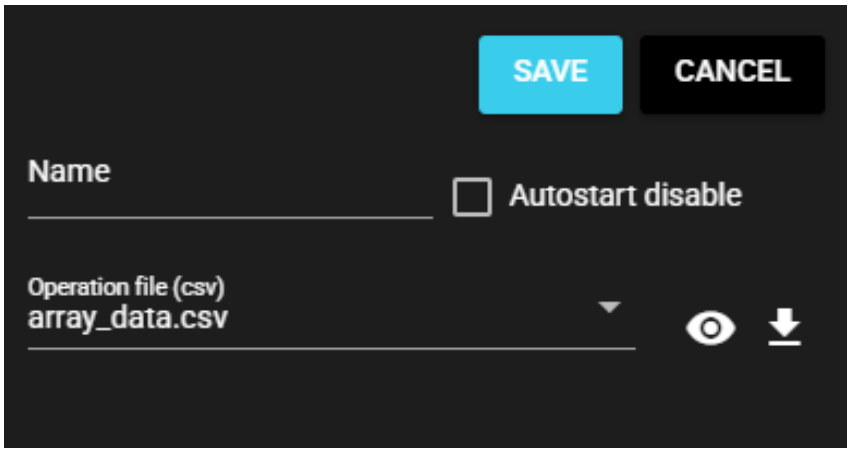
5.10.1.1 Basic Information

The basic information of the Dummy Collector is as follows.

Item	Description
Base name	Dummy Collector
Description	Dummy data is registered.
Consumption point	0
Port	Input: 0 Output: 1 This outputs dummy data based on the contents of Operation file (csv). System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.10.1.2 Setting item

Dummy Collector setting items are as follows.



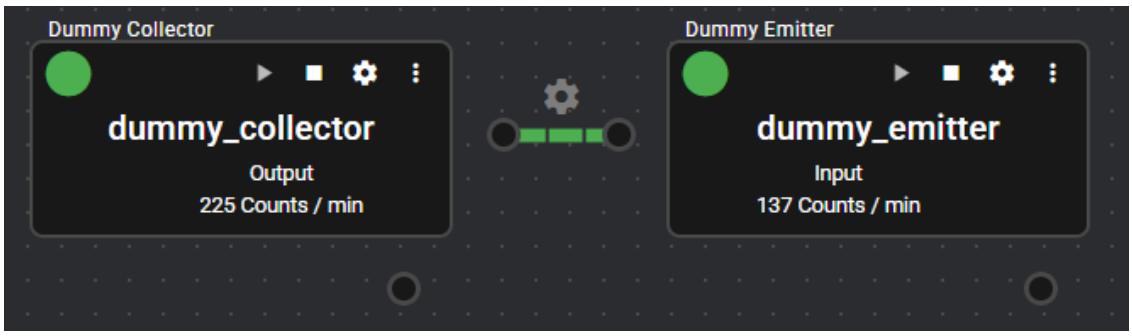
Item	Description
Name	Enter a component name. Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Operation file (csv)	Select the csv file that defines the columns to be created and the data to be registered.

The operation file is located in the `simulation_data` directory which is found in the sccde work directory. In the default setting, the following files are available for selection, but other files can also be added and selected. For the format of the operation file, see "Operation file format".

File name	Description
all_primitive_data_type.csv	This registers primitive type data.
array_data.csv	This registers array data.

5.10.2 Dummy Emitter

This component saves the collected data as a CSV file to storage.
This receives primitive type data from components connected to the IN Port and saves the data in a specified directory.



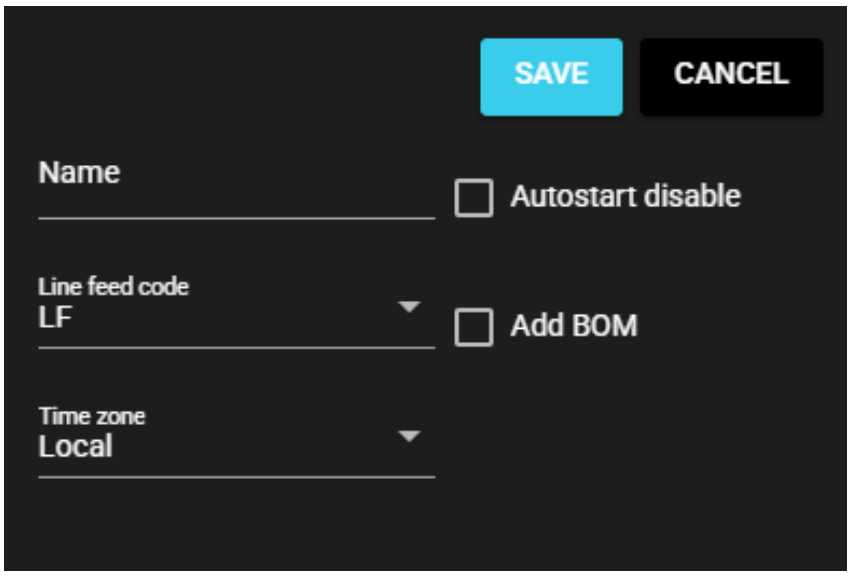
5.10.2.1 Basic Information

The basic information of the Dummy Emitter is as follows.

Item	Description
Base name	Dummy Emitter
Description	This saves the collected data in CSV format and stores it in the storage.
Consumption point	0
Port	Input: 1 This receives primitive type data from the component. The received data is saved in the storage in CSV format. Output: 0 System: 1 This outputs component operation status and error information. For details about the error information to be output, refer to Error Code List.

5.10.2.2 Setting item

Dummy Emitter setting items are as follows.



Item	Description
Name	Name to give the component Note: The same name as another component name cannot be used.
Autostart disable	Select ON to disable the Autostart of the component.
Line feed code	Select the line feed code for the CSV file.
Add BOM.	Select ON if File BOM is added to the CSV file.
Time zone	Select the time zone of the timestamp to be given to the file and CSV data

The output goes to the `.synapse/var-speedbeesynapse/projects/sccde/data` directory which is found in the sccde work directory. A directory is created for each registered Dummy Emitter, and a file is output.

5.10.2.3 Timing of file switching

Each time Dummy Emitter is started, the file is switched to a new one. After startup, output continues to be generated in the same file as a rule, but the file is switched to a new one only in the following cases.

- When the file size exceeds the maximum limit (10 MB).
- When the column definition (number of columns, column names, etc.) of the data received at the IN port changes.

Note: The output files are not automatically deleted, so delete them as necessary.

5.11 System Components

This section explains each system component. Unlike other components, system components cannot have multiple instances. You can configure and use the components arranged in the system panel.

5.11.1 DB Query Service

This component accepts queries to the data collected in SpeeDBee Synapse. No data collection or data referencing is performed by this component. This component is used to process SQL from external sources. For more information, see DB Query (SQL).
Before you execute a query, we recommend that you read through the notes on using the DB query service.

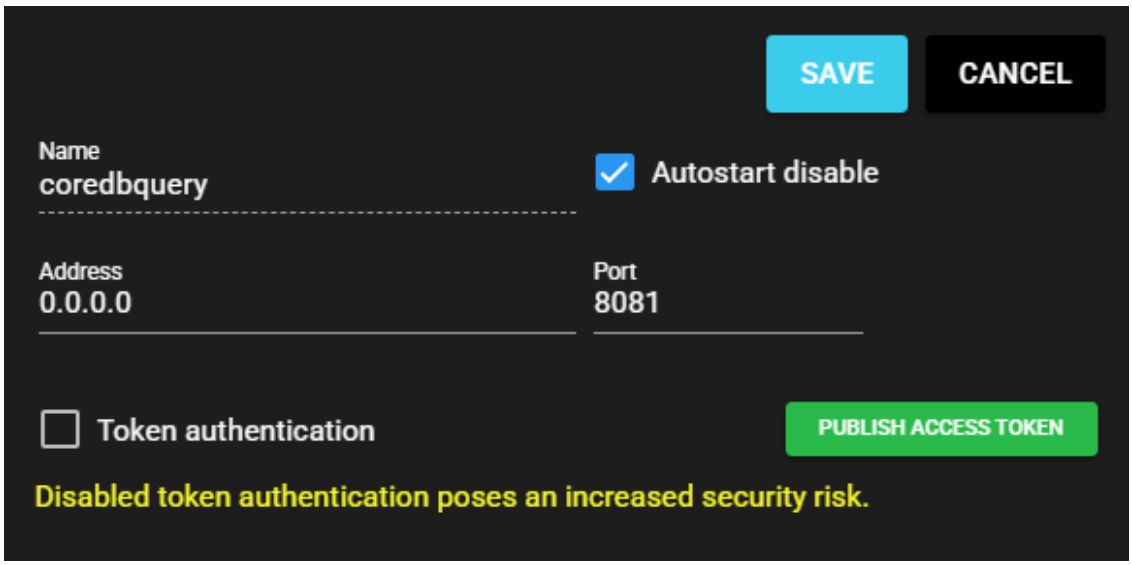
5.11.1.1 Basic Information

The basic information on the DB query service is as follows:

Item	Description
Base name	DB query service
Description	This component accepts queries to the database. Once started, you can publish queries to the DB over HTTP. For more information on DB queries, see DB Query (SQL). There is only one setting for this component in SpeeDBee Synapse. For details about the error information to be output, refer to Error Code List.
Consumption point	0
Port	Input: 0 Output: 0

5.11.1.2 Configuration Items

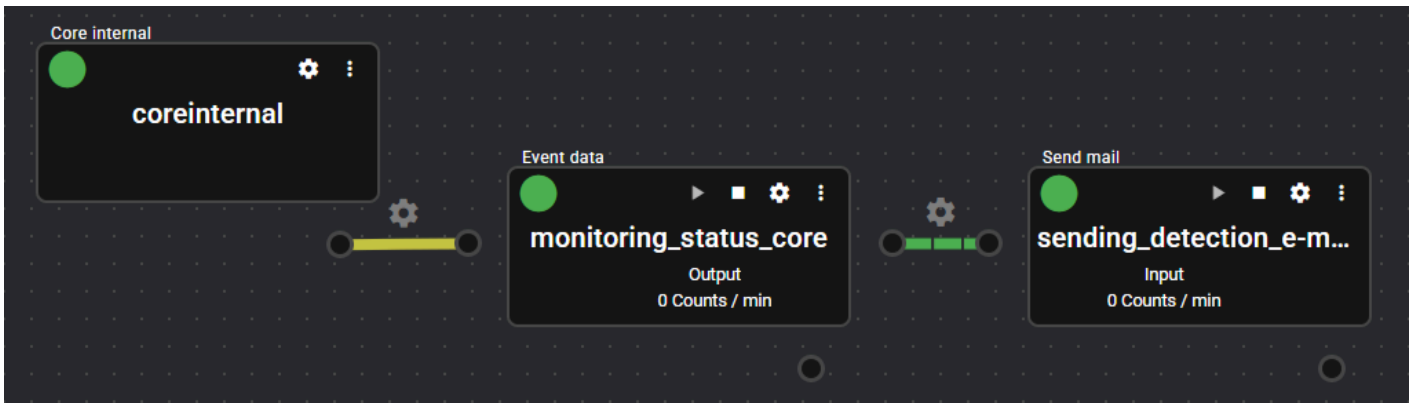
The DB query service configuration items are as follows:



Item	Description
Name	Component name (coredbquery fixed)
Autostart disable	Select ON to disable the Autostart of the component.
Address	You can input the IP address of the receiving side, and if it is 0.0.0.0, communication from all networks is allowed. Note: In some environments, this setting cannot be changed/edited.
Port	You can input the port that accepts DB queries. Note: In some environments, this setting cannot be changed/edited.
Token authentication	You can select an access token for authentication when making Web API requests.
Publish access token	Displays the Publish Access Token screen. Note: For more information on the access token screen, see Publish Access Token.

5.11.2 Core Internal

This component manages the internal status in SpeeDBee Synapse. The user cannot start or stop this component.
This component is a special component that starts automatically and runs constantly.
If data loss occurs due to a power interruption in Operating machine, the detected error can be notified by e-mail. For detailed instructions, see Troubleshoot.



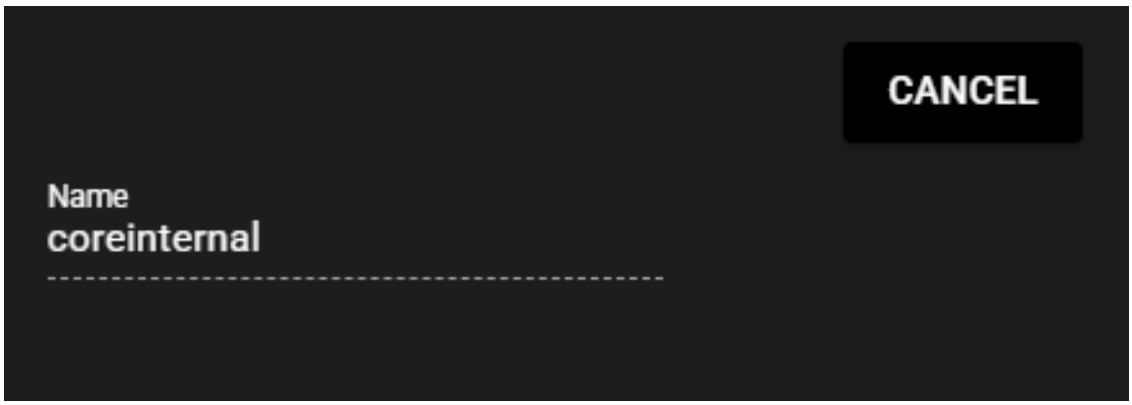
5.11.2.1 Basic Information

The basic information on the core internal is as follows:

Item	Description
Base name	Core internal
Description	This component performs processes that need to be executed periodically in the core. There is only one of this component in SpeeDBee Synapse. This component cannot be controlled by changing its settings or stopping it. For details about the error information to be output, refer to Error Code List.
Consumption point	0
Port	Input: 0 Output: 0

5.11.2.2 Configuration Items

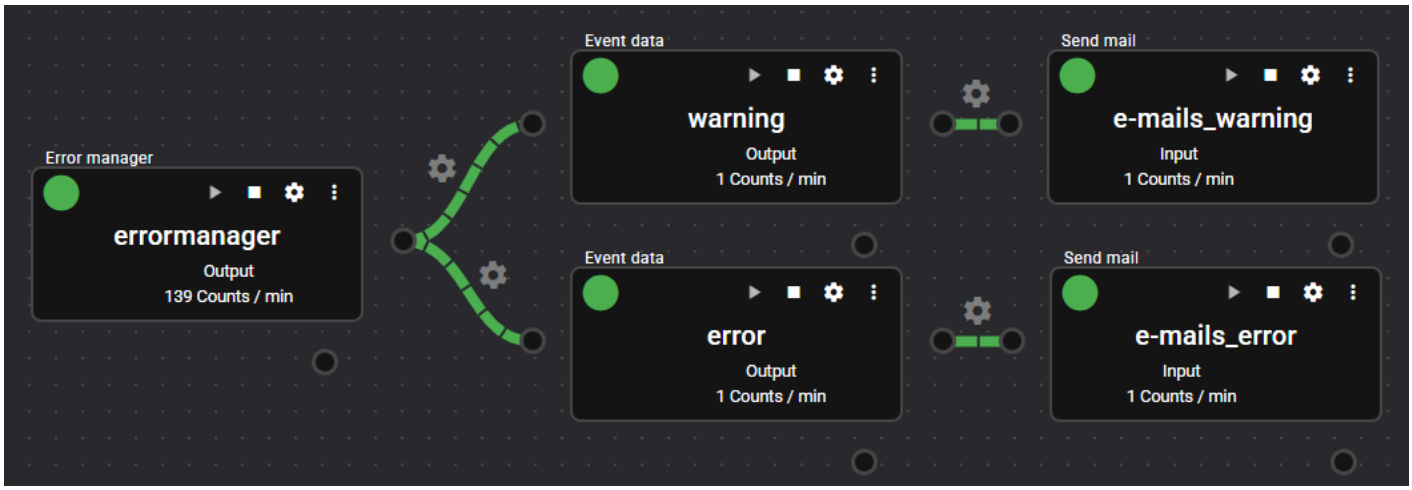
The core internal is a special component that cannot be configured.



Item	Description
Name	Component name (coreinternal fixed)

5.11.3 Error Manager

This component can detect errors for all components that occur in SpeeDBee Synapse.
Detected errors can be notified by e-mail. For detailed instructions, see Troubleshoot.



5.11.3.1 Basic Information

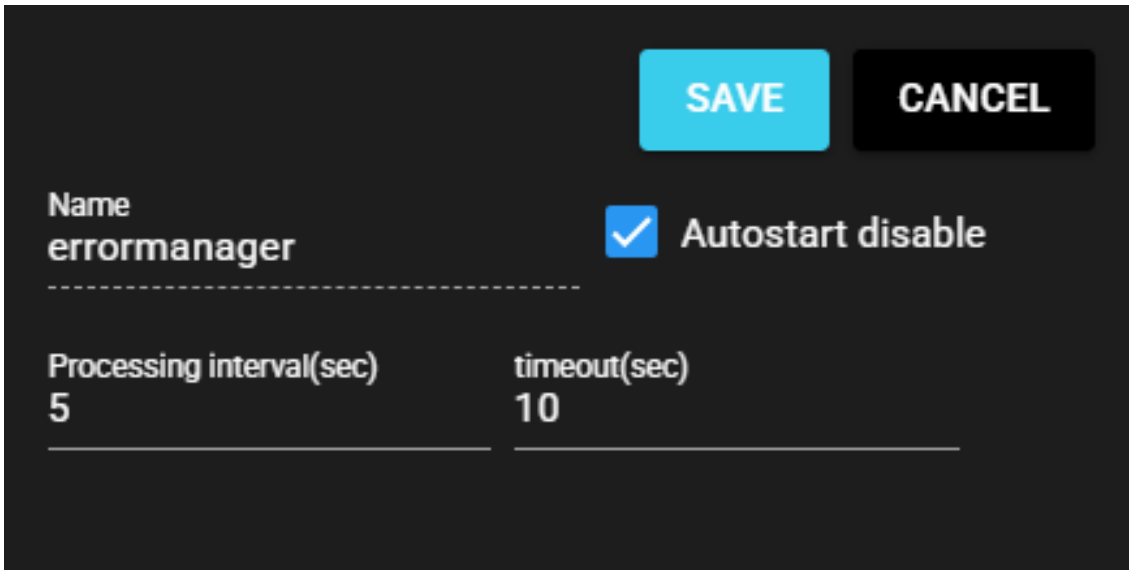
The basic information on the error manager is as follows:

Item	Description
Base name	Error manager
Description	This component detects all errors that occur in SpeeDBee Synapse. There is only one of this component in SpeeDBee Synapse. For details about the error

Item	Description
	information to be output, refer to Error Code List.
Consumption point	0
Port	Input: 0 Output: 1

5.11.3.2 Configuration Items

The error manager configuration items are as follows:



Item	Description
Name	Component name (errormanager fixed)
Autostart disable	Select ON to disable the Autostart of the component.
Processing interval	Processing interval to collect error information (seconds)
Timeout	Timeout period for processing to collect error information (seconds)

5.11.3.3 Output Data Items

The error manager outputs the following data:

Item	Column Name	Data Type	Description
Standby components	standby_component_count	uint16	Change in the number of standby components from the last time error information was collected to the current time
Running components	running_component_count	uint16	Change in the number of running components from the last time error information was collected to the current time
Warning components	warning_component_count	uint16	Change in the number of components running in a warning state from the last time error information was collected to the current time
Error components	error_component_count	uint16	Change in the number of components stopped in an error state from the last time error information was collected to the current time
Warning component name list	warning_components	string	Change in the list of names (comma separated) of components running in a warning state from the last time error information was collected to the current time
Error component name list	error_components	string	Change in the list of names (comma separated) of components stopped in an error state from the last time error information was collected to the current time
Latest standby components	latest_standby_component_count	uint32	Number of standby components when error information was collected
Latest running components	latest_running_component_count	uint32	Number of running components when error information was collected
Latest warning components	latest_warning_component_count	uint32	Number of components running in a warning state when error information

Item	Column Name	Data Type	Description
			was collected
Latest error components	latest_error_component_count	uint32	Number of components stopped in an error state when error information was collected
Latest warning component name list	latest_warning_components	string	List of names of components running in a warning state when error information was collected (comma separated)
Latest error component name list	latest_error_components	string	List of names of components stopped in an error state when error information was collected (comma separated)

5.11.4 Event Data Manager

This component is a special component that performs, for example, periodic deletion of event data recorded by i-Trigger. If you do not use i-Trigger, you do not need to start it.

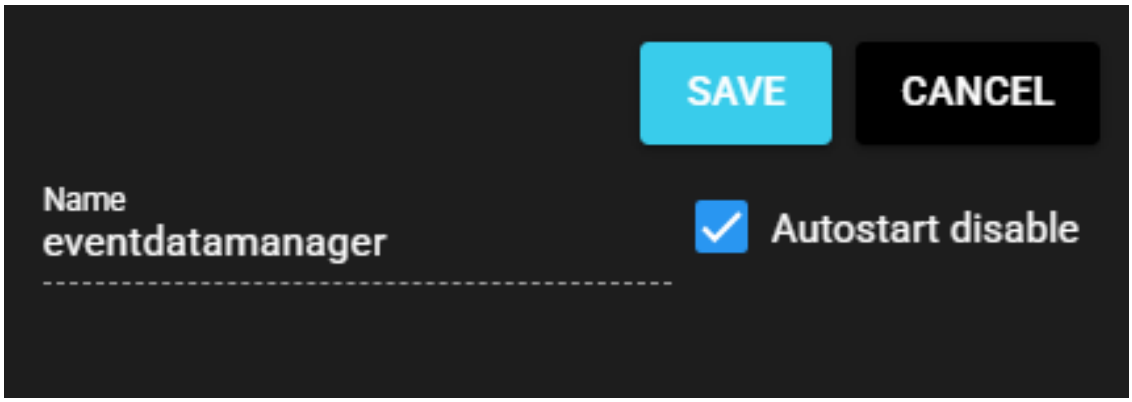
5.11.4.1 Basic Information

The basic information on the event data manager is as follows:

Item	Description
Base name	Event data manager
Description	It performs data management of event data. It periodically deletes various types of data saved by the Event Recorder. It also handles backup and restore of the internal database. There is only one of this component in SpeeDBee Synapse. For details about the error information to be output, refer to Error Code List.
Consumption point	0
Port	Input: 0 Output: 0

5.11.4.2 Configuration Items

The event data manager configuration items are as follows:



Item	Description
Name	Component name (eventdatamanager fixed)
Autostart disable	Select ON to disable the Autostart of the component.

5.11.4.3 Data Management Operational Specifications

The event data manager performs data management as follows. Please refer to this section only if you want to review the data management of the event data manager.

Periodic deletion of data saved by the Event Recorder

The Event Recorder configuration includes a Store Hour configuration item. This allows you to delete data (event data JSON files, video files) saved by the Event Recorder according to the specified Store Hour.

- If you change the Store Hour in the middle of the process, the deletion will be based on the most recent Store Hour.
- Even if you delete an instance of the Event Recorder, as long as the event data manager is running, it will perform the deletion of data that has exceeded the Store Hour.

Backup and restore of internal database files for Video Viewer

The Event Recorder maintains data in an internal database when events are recorded. This database is dedicated to i-Trigger. This database is backed up periodically and restored in case of corruption.

- The backup is processed on a daily basis (within 10 minutes of date/time transition).
- In case of corruption, the database is restored from the database of one day ago, so the latest data will be lost.
- When the disk is full, both backup and restore will be in error.

5.12 Component Default Settings

This section explains the configuration of component ports and the configuration of flowlink filters.

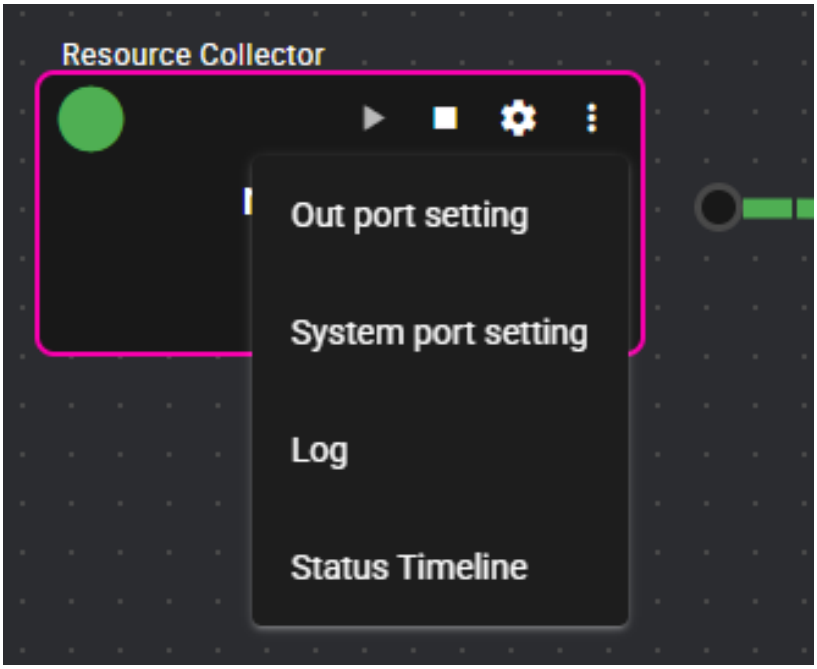
5.12.1 OUT Port

The OUT Port is the source port to which the flowlink is connected and to which the component outputs data. In the OUT Port, you can configure settings such as storage, registration conditions, and whether or not conversion is performed for the output data.

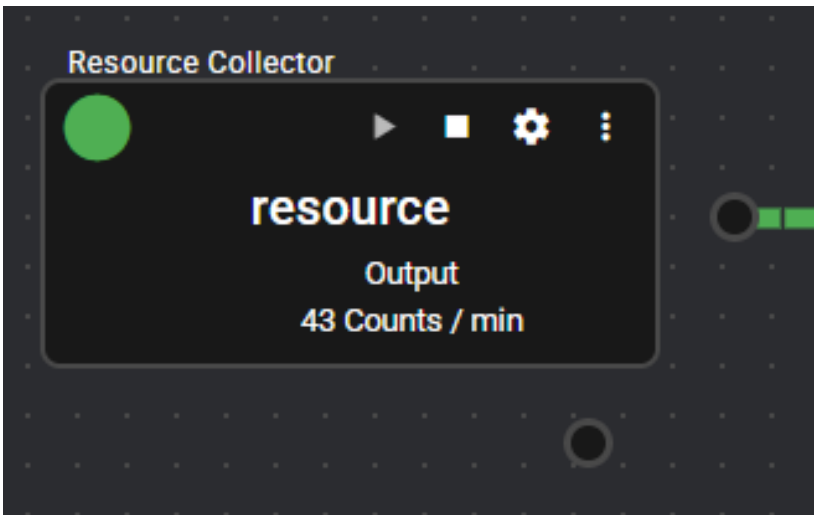
5.12.1.1 Storage Configuration

Data with persistence enabled will be saved in storage, and while it is saved, you will be able to reference it from SQL and through Grafana integration. By following the steps below, you can configure storage settings for the output data of the component.

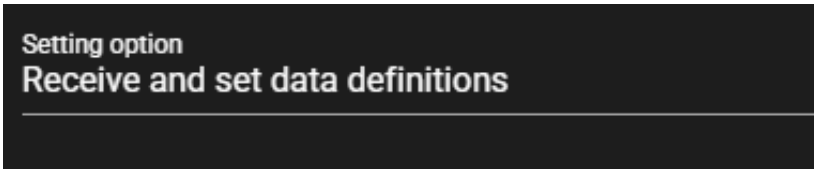
1. Open the OUT Port dialog using one of the following methods.
    - Left-click the menu icon and select "OUT port setting".



    - Right-click on the OUT Port.

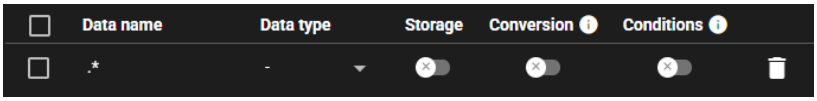


2. Select a Setting Option from the following.



Item	Description
Set after receiving data definitions	Output data is listed and storage configuration is performed individually.
Set by regular expression	A regular expression is input, and storage configuration is performed on any matches in the output data.

3. Only when the Setting Option is Set by Regular Expression, input the following items for the data to be configured.



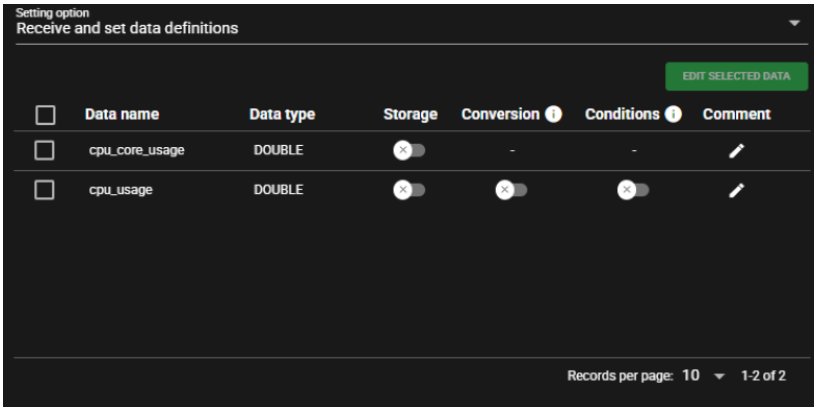
Item	Description
Data Name	Regular expression indicating the data name of the output data to be configured
Data Type	Data type of the output data to be configured

Regular Expression Input Examples

The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

4. Do one of the following to open the Storage Configuration dialog.
   - Press the Storage icon for the data for which you want to configure storage.



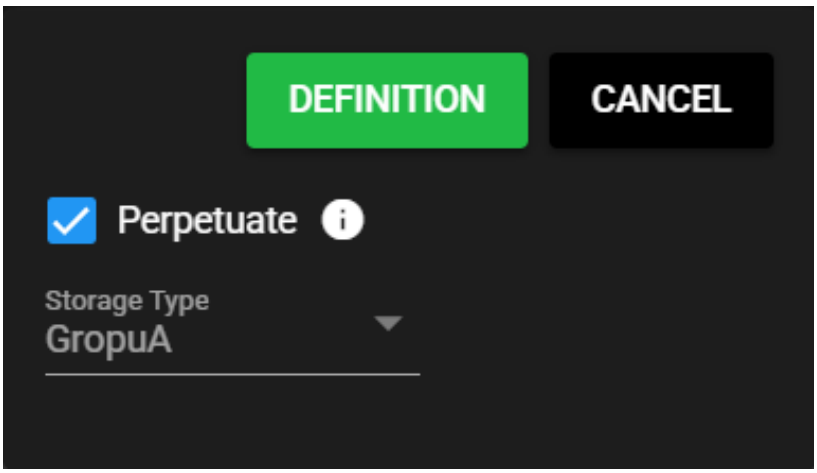
   - Place a check mark on the data for which you want to configure storage, press Edit Selected Data and select Storage. The same configuration will be applied to the data with a check mark.



5. Set Perpetuate to ON.

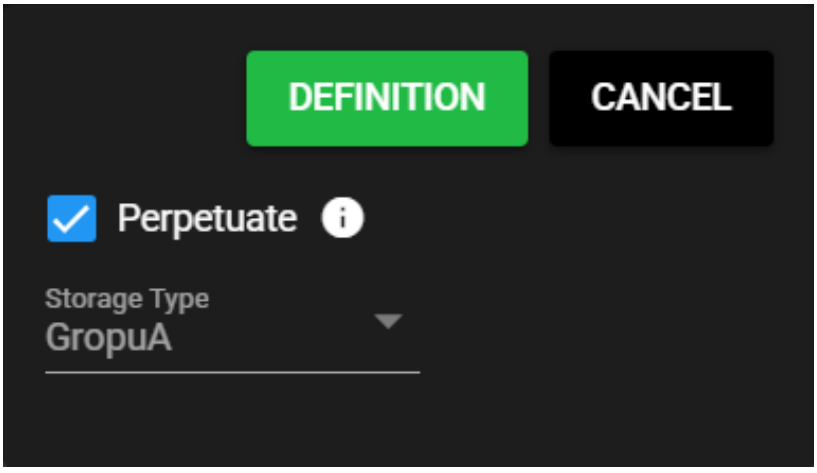


6. Input the following items.

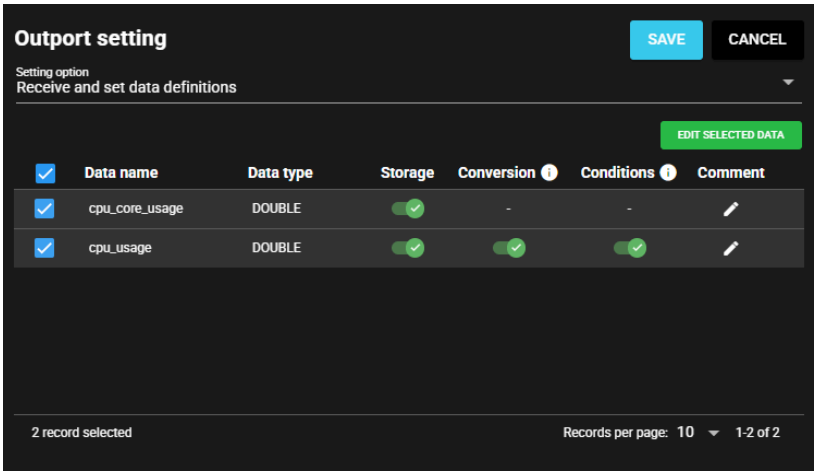


Item	Description
Storage Type	Storage period in the time series database. You can check and change the storage period for GroupA, GroupB, and GroupC in Specify the Storage Keep Time by Group in the Server Settings. You can only make changes when Specify the Storage Keep Time by Group is ON. Default Value • GroupA : 30 minutes • GroupB : 24 hours • GroupC : 7 days

7. Press Definition.



8. Press "Save".

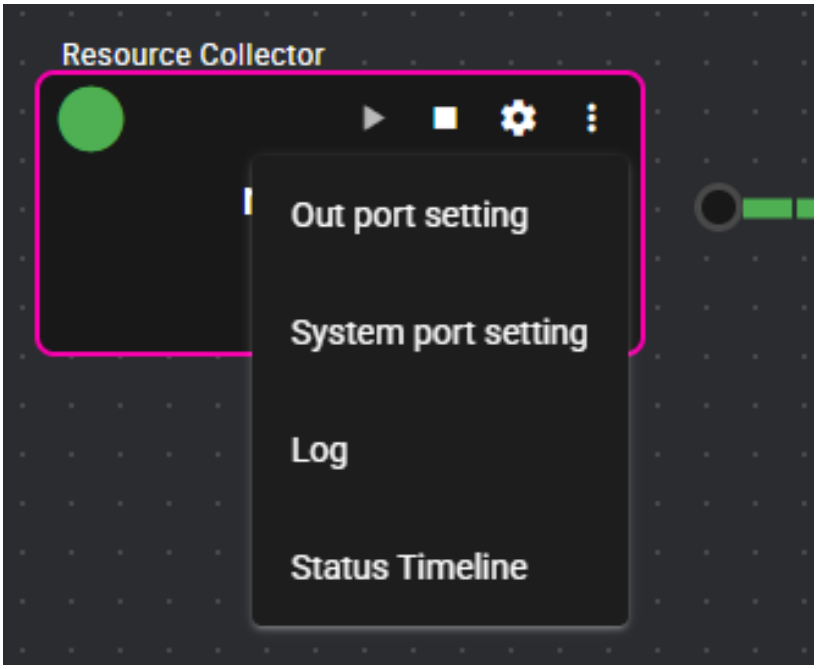


9. The storage configuration will be applied.

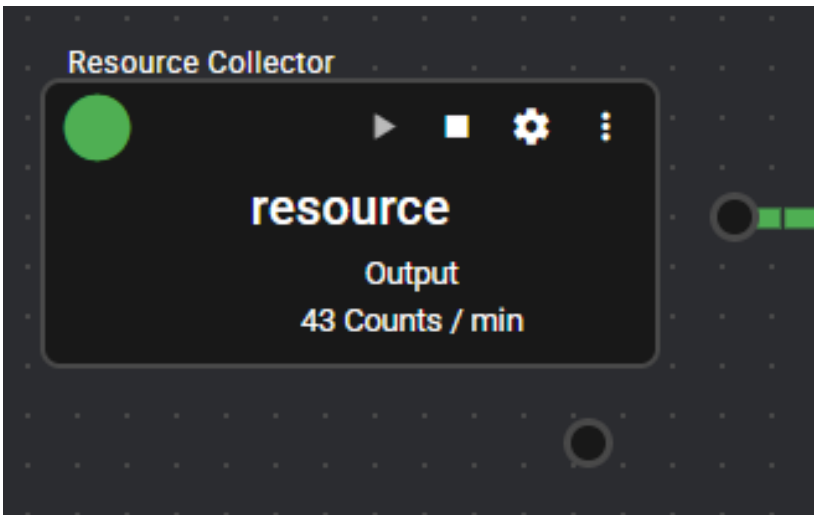
5.12.1.2 Conversion Configuration

Conversion is a function for registering data obtained from the results of calculations, rather than registering the original data as is. By following the steps below, you can configure conversion settings for the output data of the component.

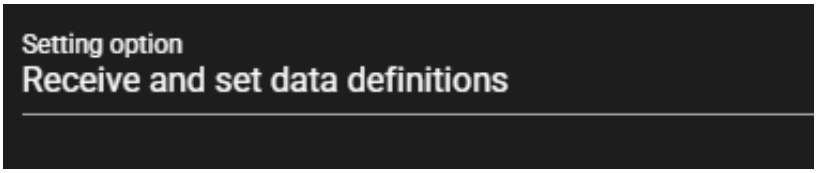
1. Open the OUT Port dialog using one of the following methods.
   - Left-click the menu icon and select "OUT port setting".



   - Right-click on the OUT Port.



2. Select a Setting Option from the following.

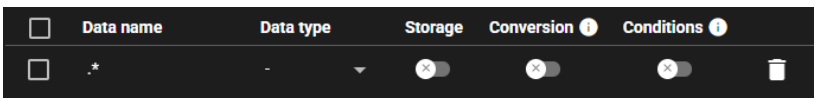


Item	Description
Set after receiving data definitions	Output data is listed and conversion configuration is performed individually.
Set by regular expression	A regular expression is input, and conversion configuration is performed on any matches in the output data.

Array type output data is not applicable

Array type output data is not applicable for conversion.
Even if the data name is applicable in the regular expression condition, if the data is an array type, the conversion will not be processed.
An example of this is the Resource Collector "cpu_core_usage".

3. Only when the Setting Option is Set by Regular Expression, input the following items for the data to be configured.



Item	Description
Data Name	Regular expression indicating the data name of the output data to be configured
Data Type	Data type of the output data to be configured

Regular Expression Input Examples

The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

4. Do one of the following to open the Conversion Configuration dialog.
   - Press the Conversion icon for the data for which you want to configure the conversion settings.



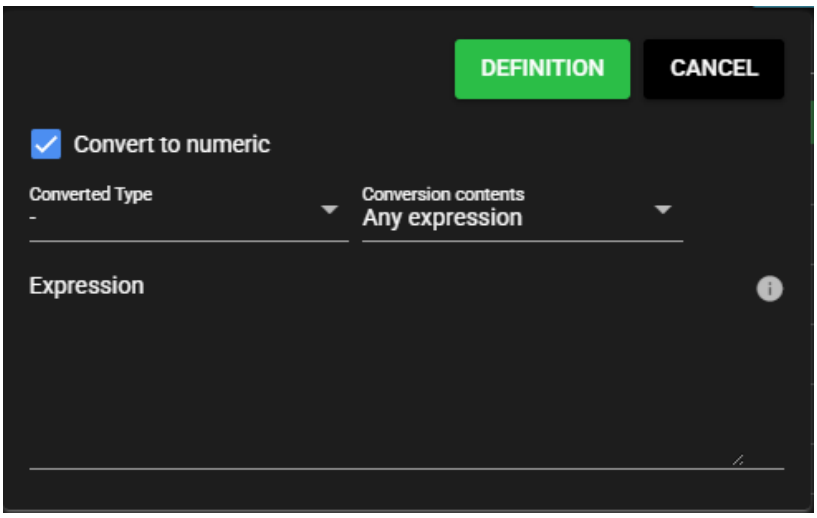
   - Place a check mark on the data for which you want to configure the conversion settings, press Edit Selected Data and select Conversion. The same configuration will be applied to the data with a check mark.



5. Set Convert to numeric to ON.

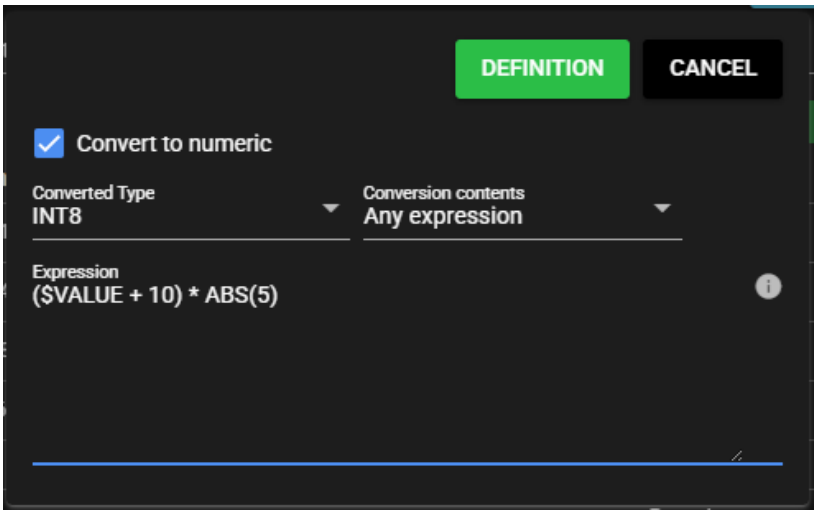


6. Input the following items.

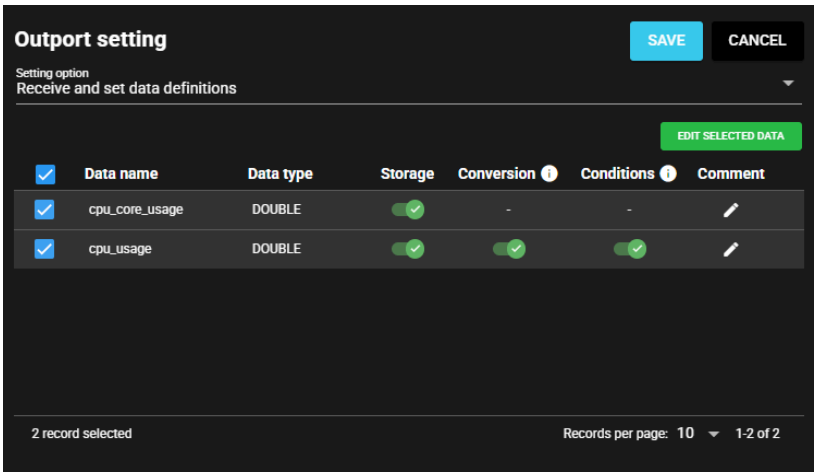


Item	Description
Converted Type	Converted data type
Conversion Type	Select the type of numerical conversion formula to be applied The following conversion types are available: ・Custom formula ・BCD (4 digits) → INT ・BCD (8 digits) → INT ・BCD (16 digits) → INT Note: Selecting Custom formula allows you to enter a user-defined numerical conversion formula. Note: Selecting BCD (4 digits) → INT, BCD (8 digits) → INT, or BCD (16 digits) → INT will automatically insert a predefined formula for converting a BCD value to an integer. You cannot edit the formula in these modes. To edit the formula manually, select Custom formula. Note: Data converted from BCD to an integer will be treated as an integer type. If you write the converted value to an address that expects BCD—such as for PLC writes or EtherNet/IP writes—the value will not be correct.
Expression	Expression to be executed in the conversion Note: Click on the icon at the right end of the input area to display a list of input examples and elements that can be used in the expression.

7. Press Definition.



8. Press Save.

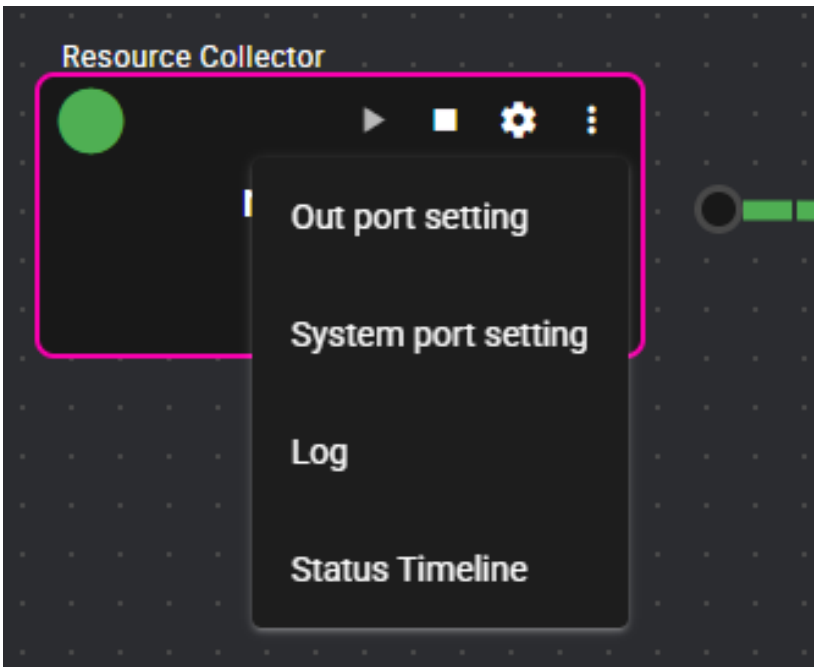


9. The conversion configuration will be applied.

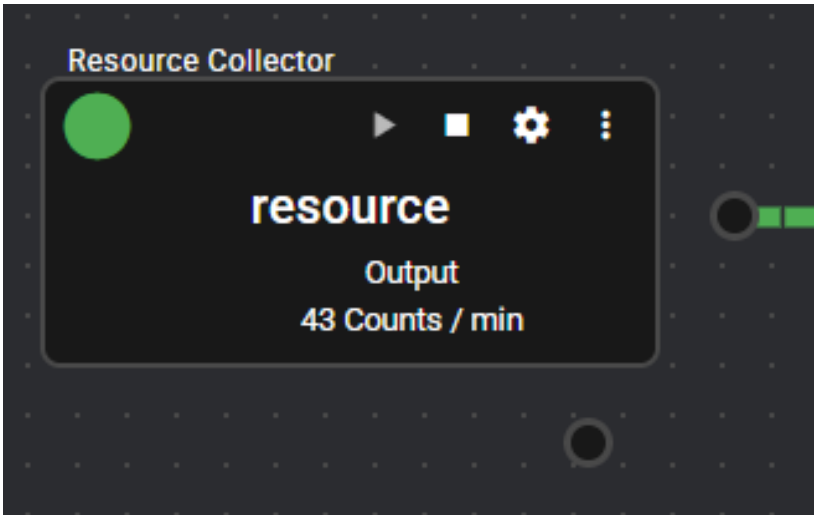
5.12.1.3 Set Registration Conditions

Registration Conditions are a function for registering only the data that meets the conditions. Data that does not meet the conditions will be treated as not applicable and will not be registered. By following the steps below, you can configure the settings of the registration conditions for the output data of the component.

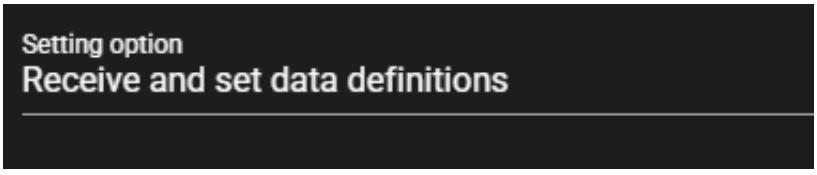
1. Open the OUT Port dialog using one of the following methods.
    - Left-click the menu icon and select "OUT port setting".



- Right-click on the OUT Port.



2. Select a Setting Option from the following.

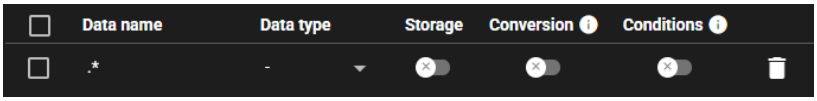


Item	Description
Set after receiving data definitions	Output data is listed and the configuration of registration conditions is performed individually.
Set by regular expression	A regular expression is input, and the configuration of registration conditions is performed on any matches in the output data.

Array type output data is not applicable

Array type output data is not applicable for registration conditions.
Even if the data name is applicable in the regular expression condition, if the data is an array type, the registration conditions will not be processed.
An example of this is the Resource Collector "cpu_core_usage".

3. Only when the Setting Option is Set by Regular Expression, input the following items for the data to be configured.



Item	Description
Data Name	Regular expression indicating the data name of the output data to be configured
Data Type	Data type of the output data to be configured

Regular Expression Input Examples

The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

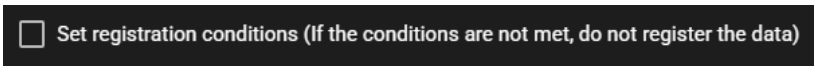
4. Do one of the following to open the Set Registration Conditions dialog.
   - Press the Registration Conditions icon for the data for which you want to configure the settings of the registration conditions.



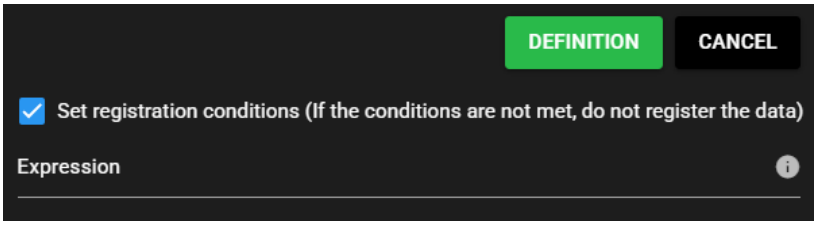
- Place a check mark on the data for which you want to configure the settings of the registration conditions, press Edit Selected Data and select Registration Conditions. The same configuration will be applied to the data with a check mark.



5. Select ON for Set Registration Conditions.

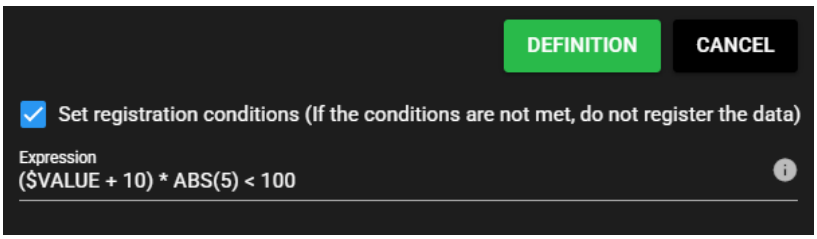


6. Input the following items.

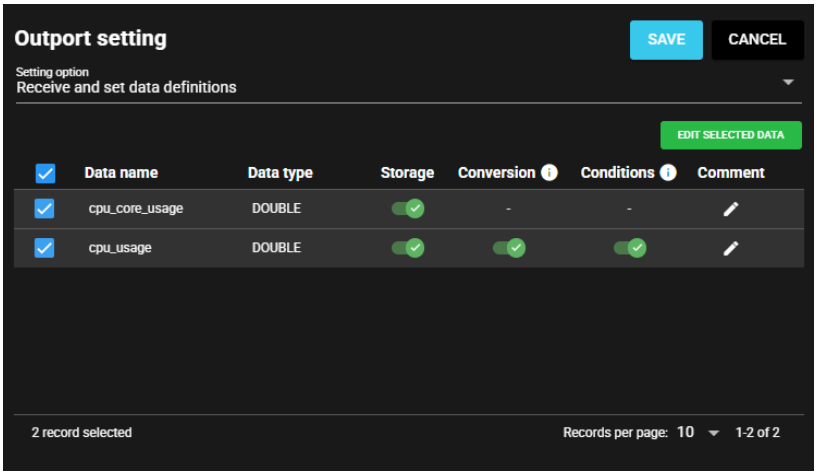


Item	Description
Conditional expression	Conditional expression for determining the data to be registered Note: Click on the icon at the right end of the input area to display a list of input examples and elements that can be used in the expression.

7. Press Definition.



8. Press Save.



9. The configuration of registration conditions will be applied.

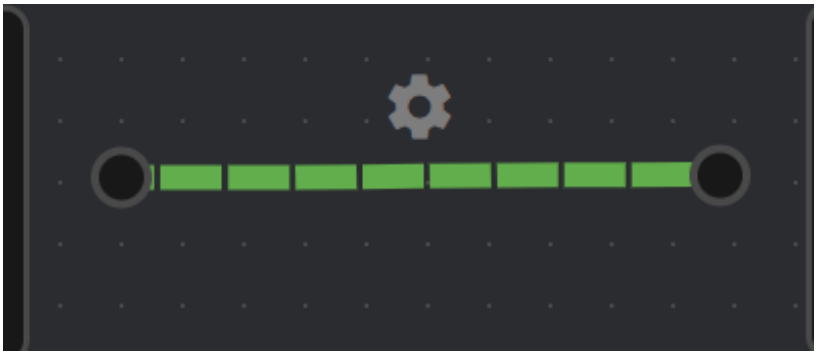
5.12.2 Filter

The filter function allows you to determine which data is allowed to pass over the flowlink.

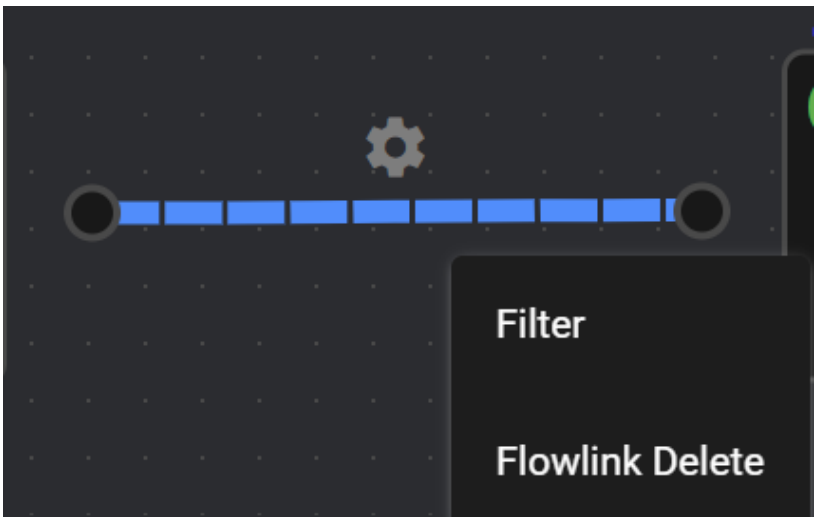
5.12.2.1 Setting Option

You can configure filter settings by following the steps below.

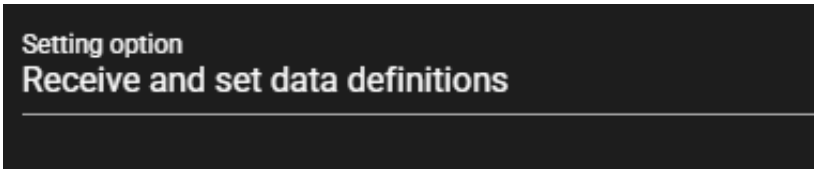
1. Right-click on the flowlink.



2. Select Filter.

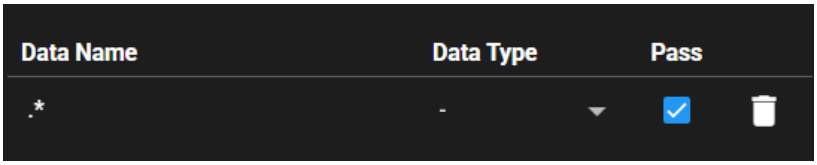


3. Select a Setting Option from the following.



Item	Description
Set after receiving data definitions	Data from the source to which the flowlink is connected is listed, and the configuration of filter settings is performed individually.
Set by regular expression	A regular expression is input, and the configuration of filter settings is performed on any matched data from the source to which the flowlink is connected.

4. Only when the Setting Option is Set by Regular Expression, input the following items for the data to be configured.



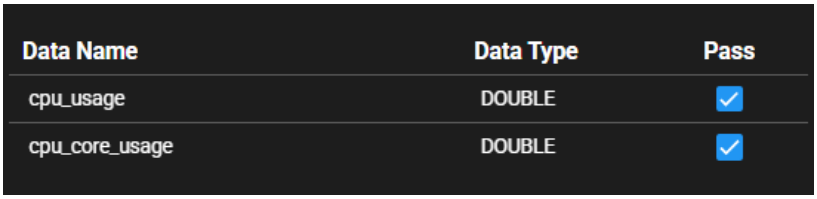
Item	Description
Data Name	Regular expression indicating the data name of the data to be configured from the source to which the flowlink is connected
Data Type	Data type of the data to be configured from the source to which the flowlink is connected

Regular Expression Input Examples

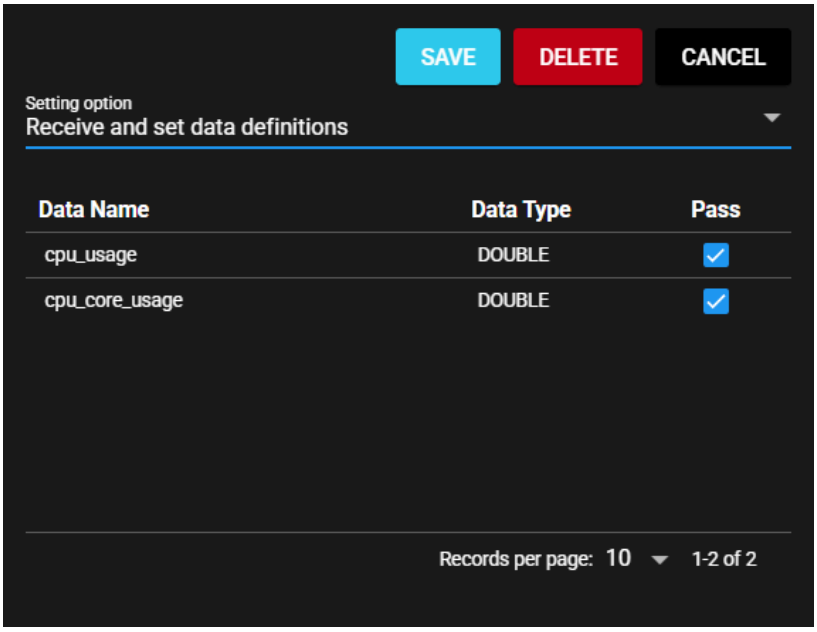
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

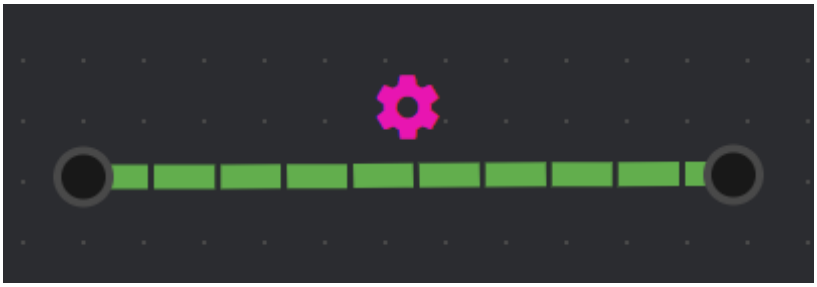
5. Set Pass Through to ON for the data you want to pass through to the destination to which the flowlink is connected.



6. Press Save.



7. The configuration of filter settings will be applied. The icon of the flow link with the applied settings will change color.



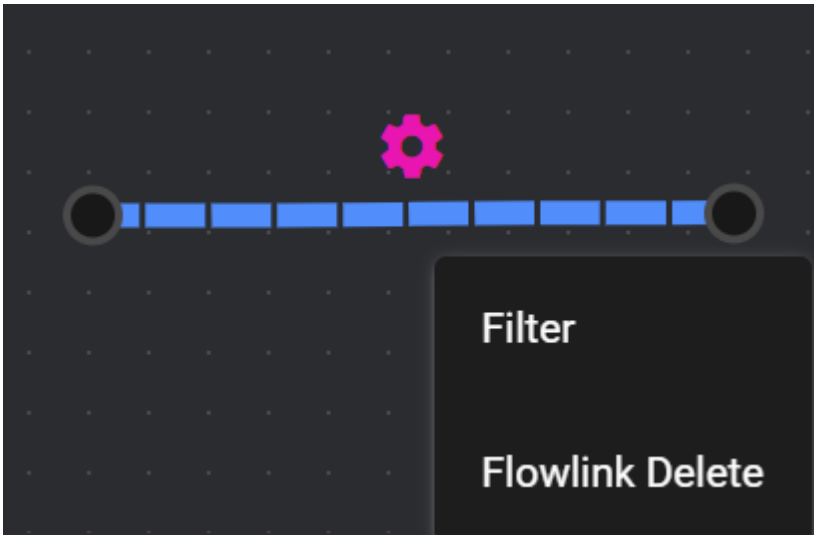
5.12.2.2 Delete

You can delete a filter by following the steps below.

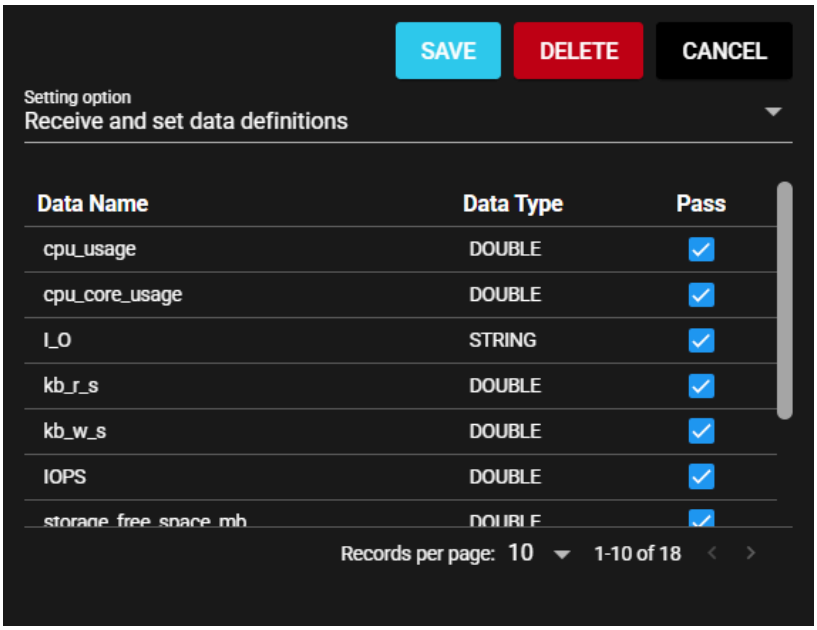
1. Right-click on the filtered flowlink.



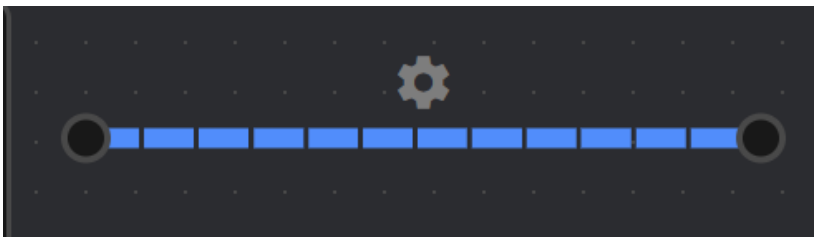
2. Select Filter.



3. Press Delete and select Yes in the confirmation dialog.

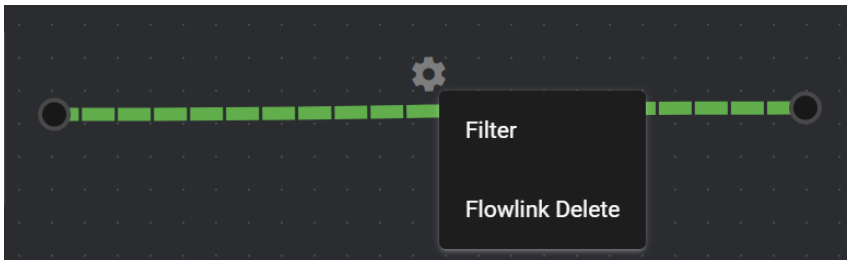


4. The configuration of filter settings will be deleted. The icon color changes.



5.12.3 Flowlink Icon

When the icon that appears upon connecting a flow link is left-clicked, the same operations as right-clicking the flow link can be performed. For more information about filter settings, see "Filters". For more information about deleting flow links, see "Delete".



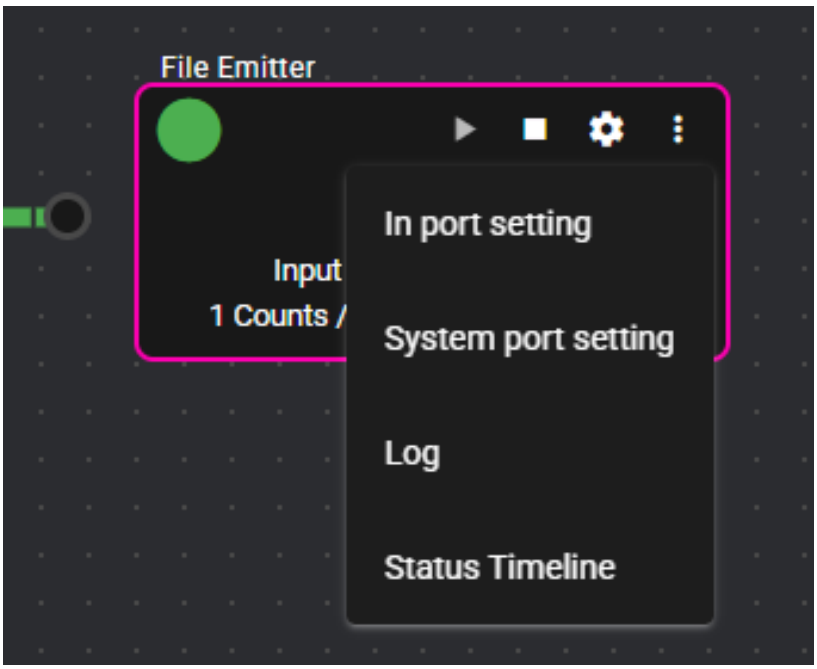
5.12.4 IN Port

The IN Port is the port to which the flowlink is connected and the component receives data. In the IN Port, you can configure settings such as the use of events and whether or not to aggregate the received data.

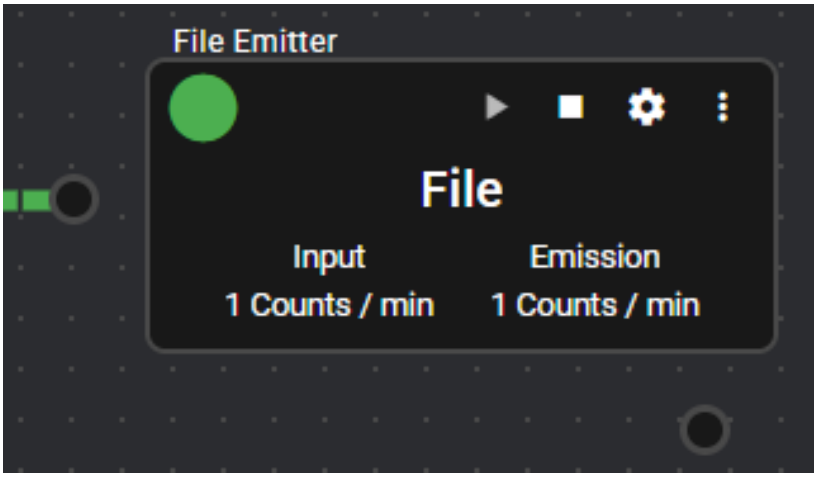
5.12.4.1 Set Delay Time

By following the steps below, you can configure delay time settings for the data received by the component.

1. Open the IN Port dialog using one of the following methods.
    - Left-click the menu icon and select "In port setting".



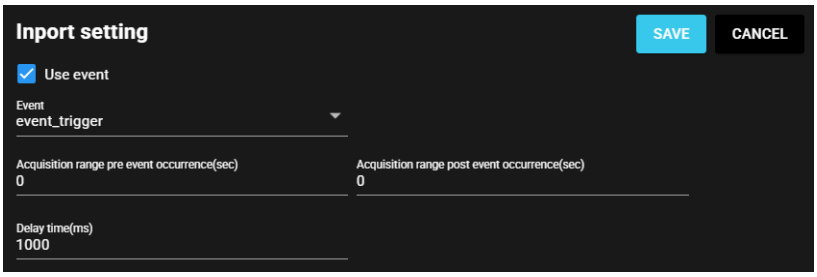
- Right-click on the IN Port.



2. Input the data collection delay time in milliseconds in the Delay Time(ms) field.
   For more information, see Delay Time for Data Reference in Operational Specifications Related to Data Collection.



3. Press Save.

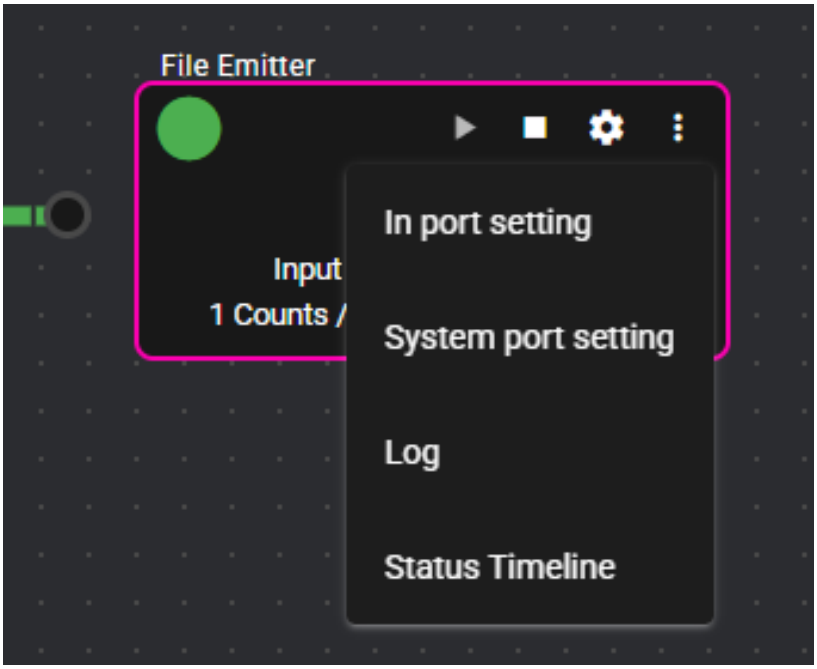


4. The configuration of delay time settings will be applied.

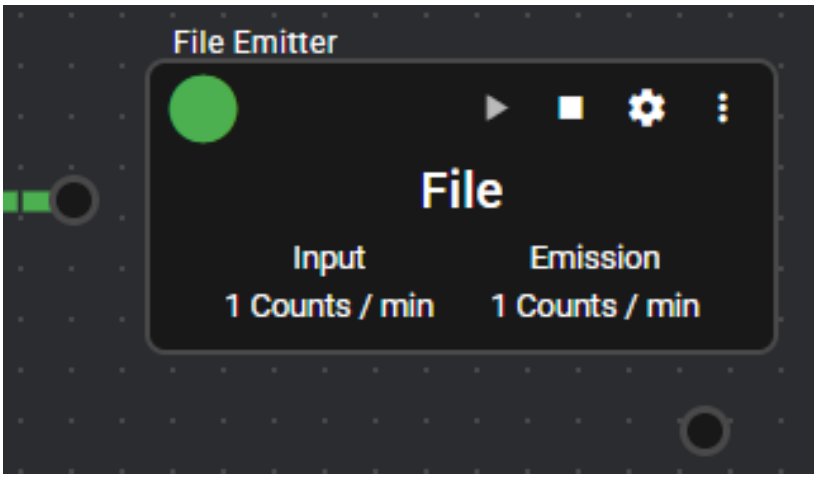
5.12.4.2 Configure Events

By following the steps below, you can configure event settings for the data received by the component.

1. Register an event trigger. For more information, see Event Trigger.
2. Connect a flowlink from the event trigger to the component. For more information, see Add a Flowlink.
3. Open the IN Port dialog using one of the following methods.
   - Left-click the menu icon and select "In port setting".



   - Right-click on the IN Port.



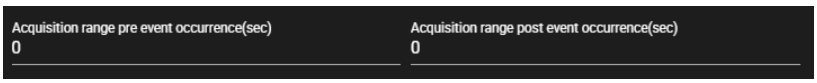
4. Set Use Event to ON.



5. Select the registered event trigger in the Event field.

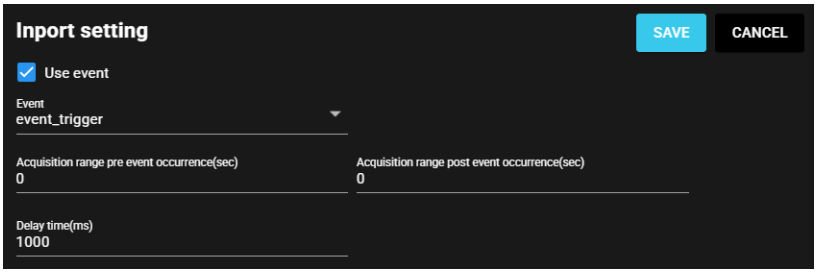


6. Input the following items.



Item	Description
Acquisition range pre event occurrence(sec)	Range of acquisition of data before the event occurred
Acquisition range post event occurrence(sec)	Range of acquisition of data after the event occurrence

7. Press Save.

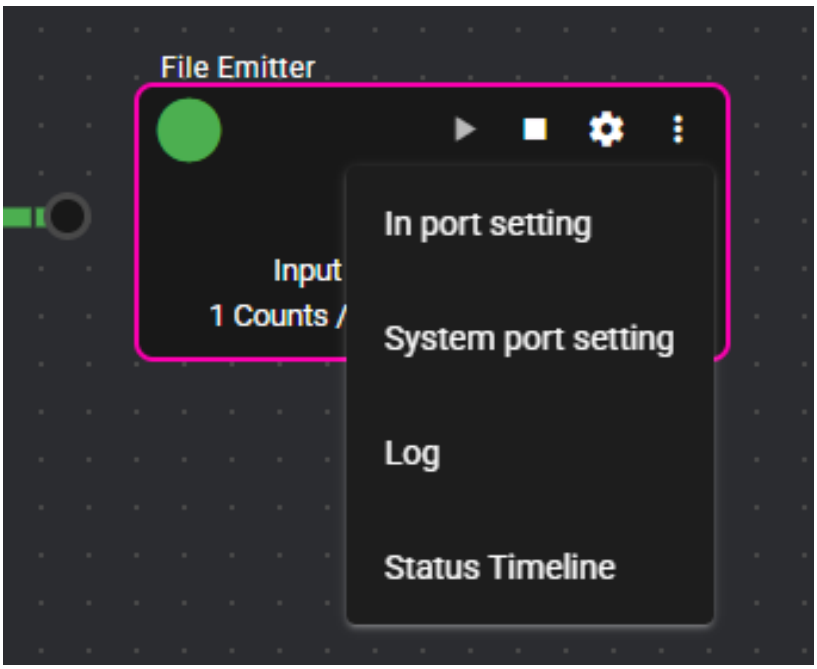


8. The configuration of event settings will be applied.

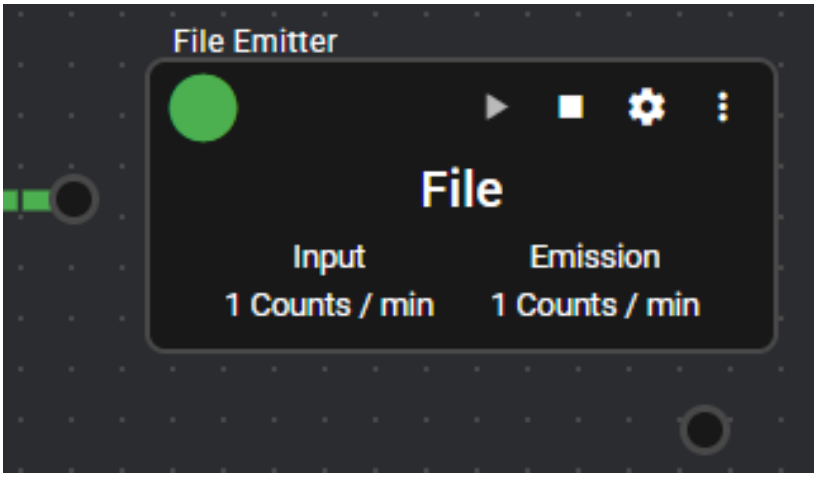
5.12.4.3 Aggregation Configuration

You can configure aggregation settings by following the steps below.

1. Open the IN Port dialog using one of the following methods.
   - Left-click the menu icon and select "In port setting".



- Right-click on the IN Port.



2. Set Aggregation to ON.

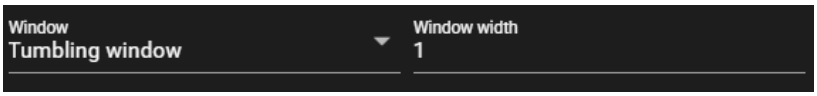


3. Select a Window from the following.



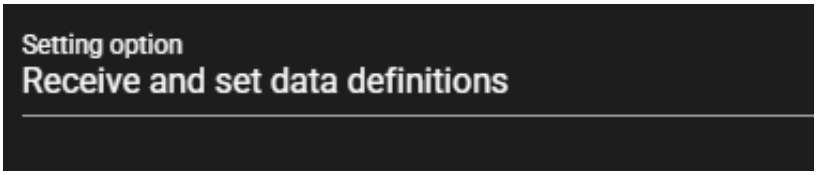
Item	Description
Tumbling window	The range of data to be aggregated is delimited by the specified time span.

4. Specify the window width. The input items will vary depending on the window you have selected.



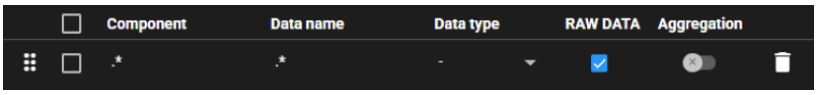
Item		Description
(Tumbling window selected)	Window width	Input time span to be aggregated (in seconds)

5. Select a Setting Option from the following.



Item	Description
Set after receiving data definitions	Data received at the IN Port is listed and the configuration of aggregation settings is performed individually.
Set by regular expression	A regular expression is input, and the configuration of aggregation settings is performed on any matched data received at the IN Port.

6. Input the following items for the data to be configured.



Item	Description
Component name	Regular expression indicating the component name of the data to be configured that is received at the IN Port Note: You can input this item only when the Setting Option is Set by Regular Expression.
Data Name	Regular expression indicating the data name of the data to be configured that is received at the IN Port Note: You can input this item only when the Setting Option is Set by Regular Expression.

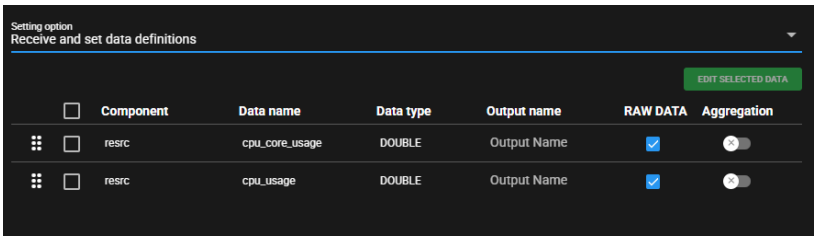
Item	Description
Data Type	Data type of the data to be configured that is received at the IN Port Note: You can select this item only when the Setting Option is Set by Regular Expression.
Output Name	New name to overwrite existing data name Note: You can select this item only when the Setting Option is Set After Receiving Data Definitions.
RAW DATA	You get unprocessed data.

Regular Expression Input Examples

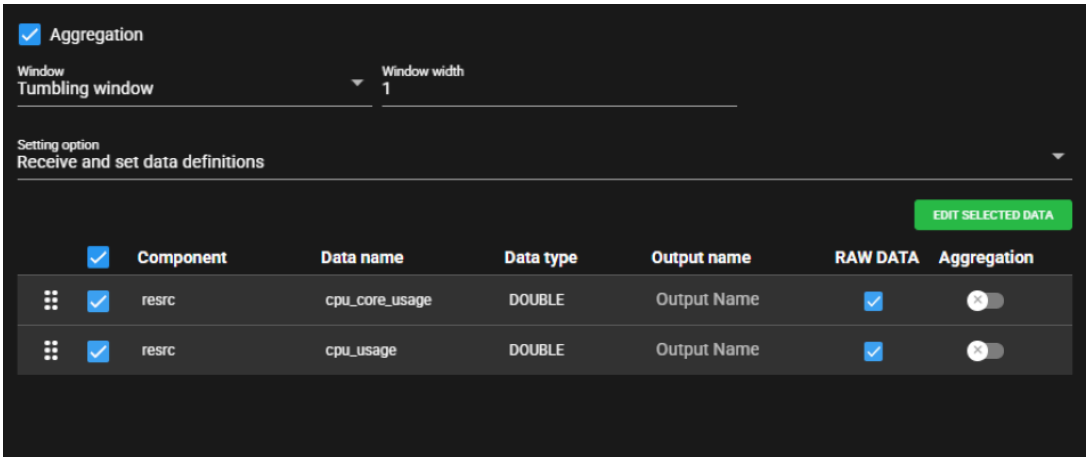
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

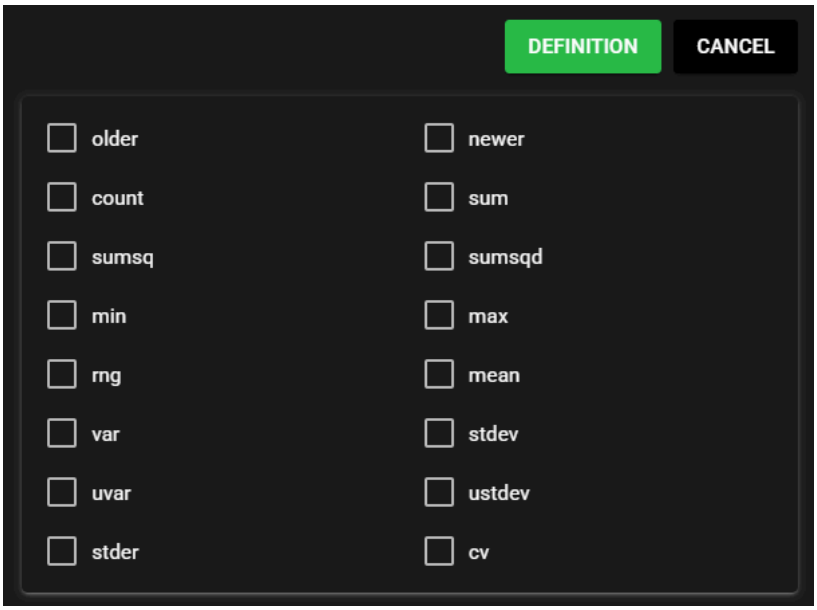
7. Do one of the following to open the Aggregation Configuration dialog.
   - Press the Aggregation icon for the data for which you want to configure the aggregation settings.



   - Place a check mark on the data for which you want to configure the aggregation settings and press Edit Selected Data. The same configuration will be applied to the data with a check mark.



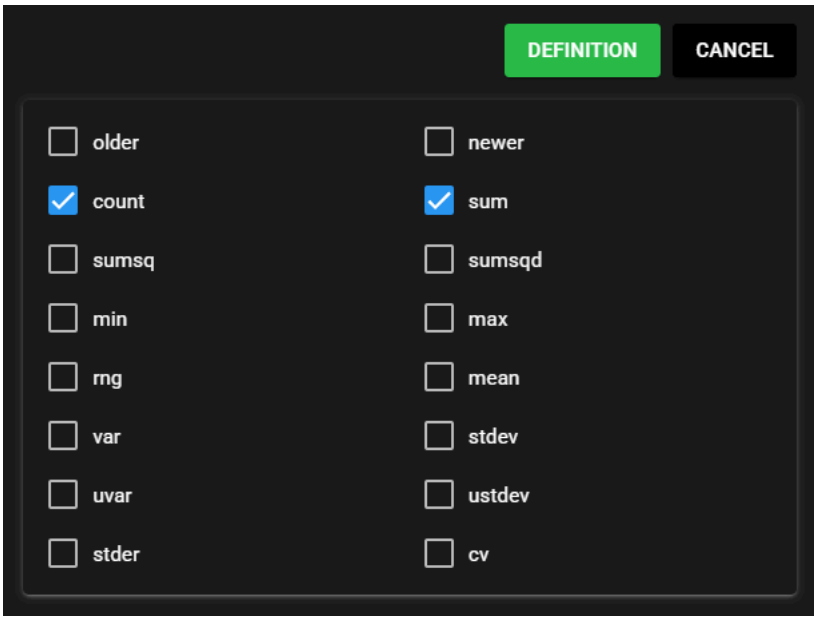
8. Place a check mark in the aggregation value you want to retrieve.



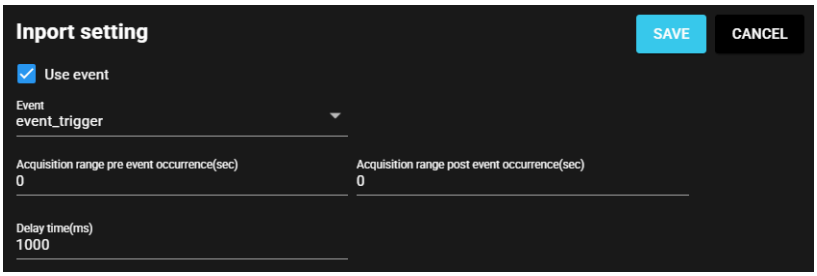
Item	Description
Older	Oldest value in the target data
Latest value	Most recent value in the target data
Count	Number of counts of target data
Sum	Total value of the target data
Sumsq	Sum of squares of each target data
Sumsqd	Sum of the squares of the deviations of each target data from the mean
Min	Smallest value in the target data
Max	Largest value in the target data
rng	Difference between the maximum and minimum values of the target data
Mean	Average of the target data
Var	Square of the average of the deviation of each target data from the mean
Stdev	Square roots of variance

Item	Description
Uvar	Variance value divided by the number of counts minus one
Ustdev	Standard deviation of the sample
Stder	Standard deviation of the sample mean
CV	Standard deviation divided by the mean

9. Press Definition.



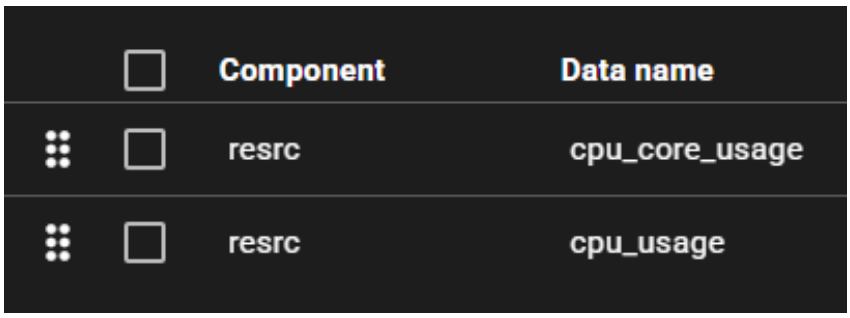
10. Press "Save".



11. The configuration of aggregation settings will be applied.

5.12.4.4 Sorting Data

You can change the sort order of the data by clicking on the icon at the left end of the data and moving it.



5.12.5 SYSTEM Port

The SYSTEM Port is the source port to which the flowlink is connected and outputs system information (error information, operating information) of the component. In the SYSTEM Port, you can configure settings such as whether or not storage is performed for system information.

Changing the configuration of the SYSTEM Port

Do not change the settings that can be configured for this SYSTEM Port unless necessary, because they affect settings such as the error logging duration.

A component with unmodifiable settings

The Core Internal component is a special component for which System Port settings cannot be modified, ensuring persistence remains enabled.

5.12.5.1 Configurable System Information

The following is the system information for the component output by the SYSTEM Port.

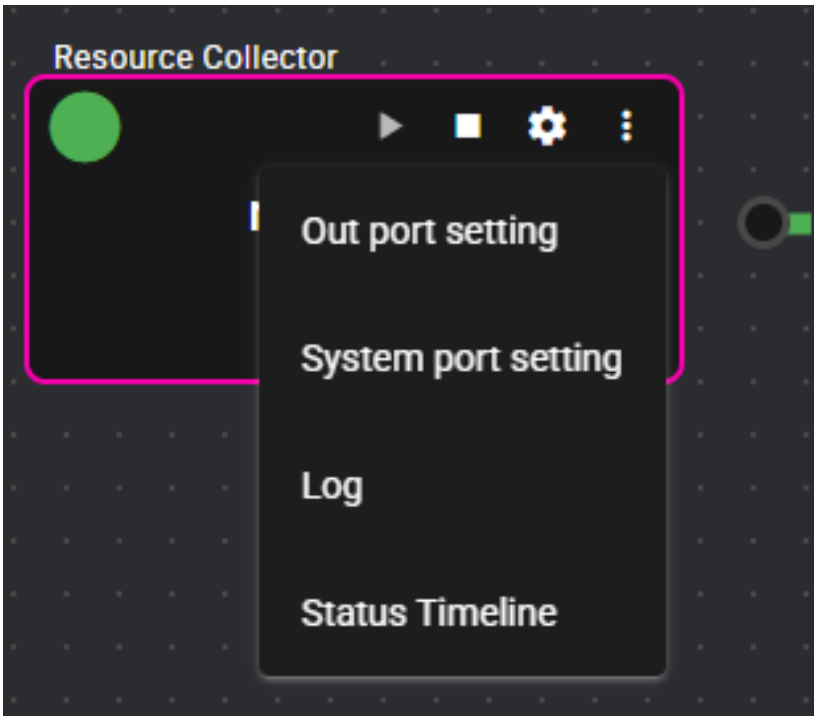
Item	Data Type	Description
errors	COMPONENT_STATUS	Error information
errors_run_status	UINT8	Operating status by integer value 0: Stop 1: Running (no errors) 2: Running (error occurred) 3: Error stop
total_input_data_count	UINT64	Number of counts of input data
total_input_data_bytes	UINT64	Number of bytes of input data

Item	Data Type	Description
total_output_data_count	UINT64	Number of counts of output data
total_output_data_bytes	UINT64	Number of bytes of output data
total_received_count	UINT64	Number of counts of data received by the component
total_received_bytes	UINT64	Number of bytes of data received by the component
total_emission_count	UINT64	Number of counts of data emitted by the component
total_emission_bytes	UINT64	Number of bytes of data emitted by the component

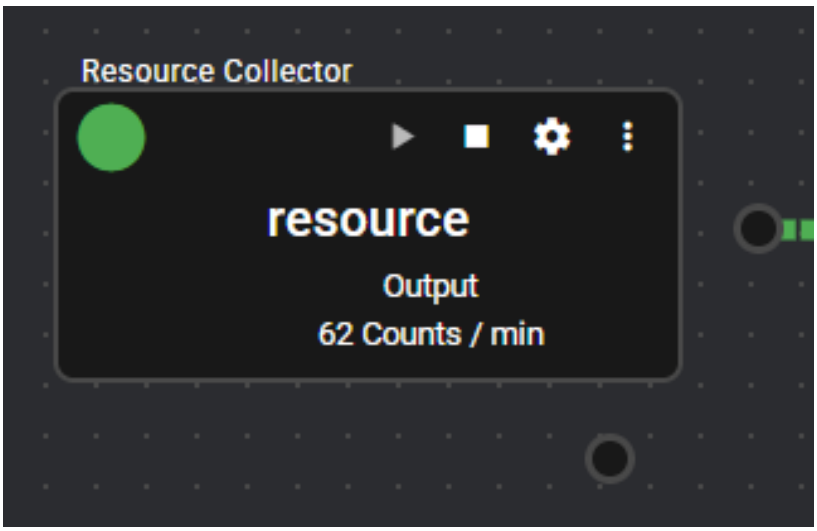
5.12.5.2 Storage Configuration

Data with persistence enabled will be saved in storage, and while it is saved, you will be able to reference it from SQL and through Grafana integration. You can configure storage settings by following the steps below.

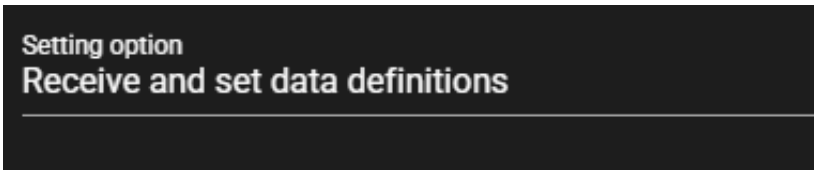
1. Open the SYSTEM Port dialog using one of the following methods.
   - Left-click the menu icon and select "System port setting".



   - Right-click on the SYSTEM Port.

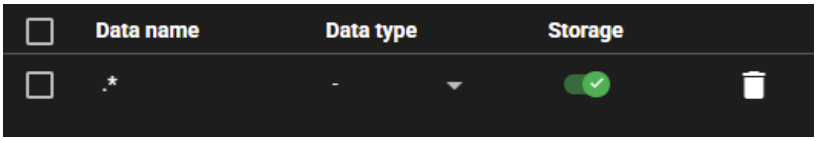


2. Select a Setting Option from the following.



Item	Description
Set after receiving data definitions	System information is listed and storage configuration is performed individually.
Set by regular expression	A regular expression is input, and storage configuration is performed on any matched data in the system information.

3. Only when the Setting Option is Set by Regular Expression, input the following items for the data to be configured.



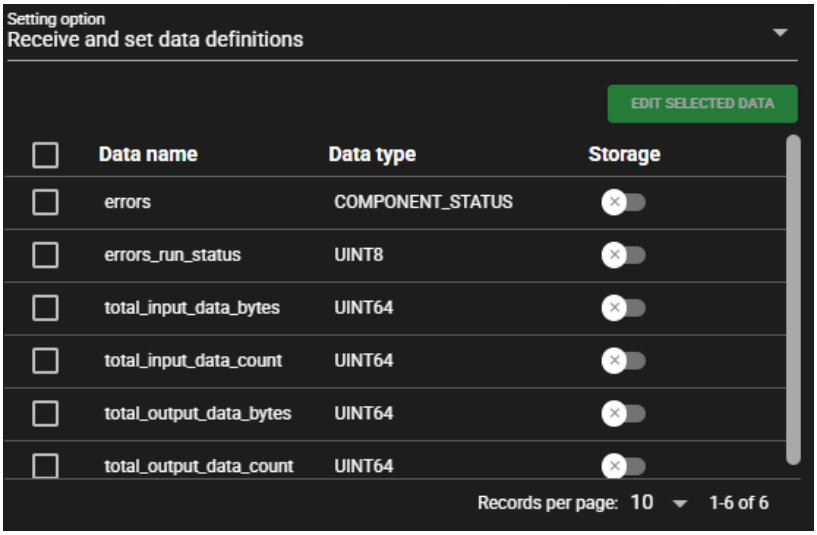
Item	Description
Data Name	Regular expression indicating the data name of the data to be configured in the system information
Data Type	Data type of the data to be configured in the system information

Regular Expression Input Examples

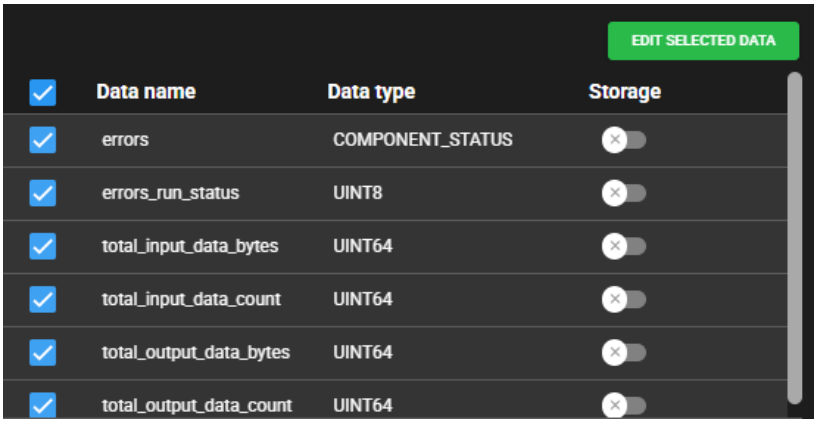
The following are input examples of regular expressions.

- Match only "data1": data1
- Match data whose name includes "abc": .*abc.*
- Match data whose name starts with "abc": abc.*
- Match data whose name ends with "abc": .*abc

4. Do one of the following to open the Storage Configuration dialog.
   - Press the Storage icon for the data for which you want to configure storage.



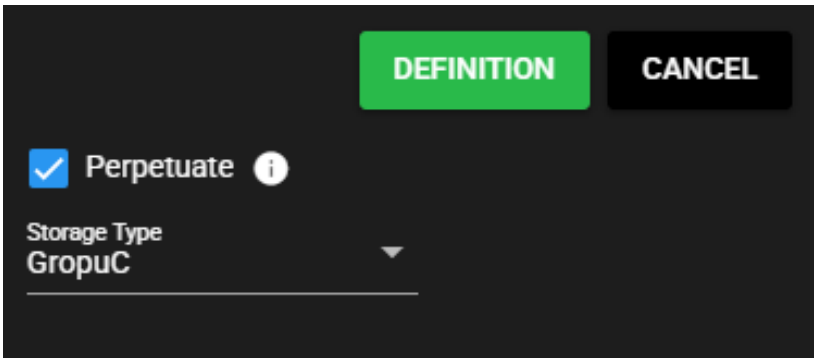
   - Place a check mark on the data for which you want to configure storage and press Edit Selected Data. The same configuration will be applied to the data with a check mark.



5. Set Perpetuate to ON.

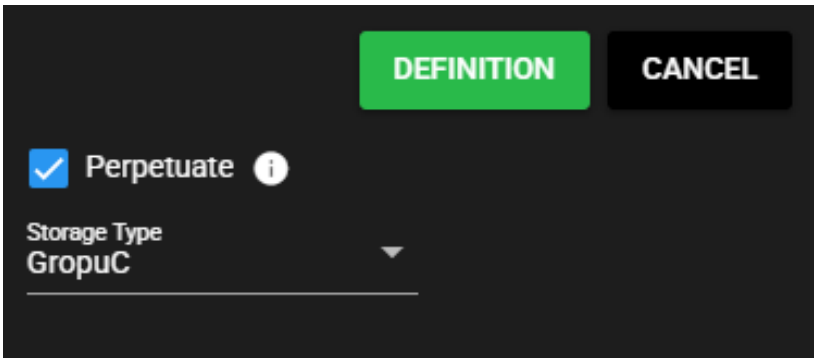


6. Input the following items.

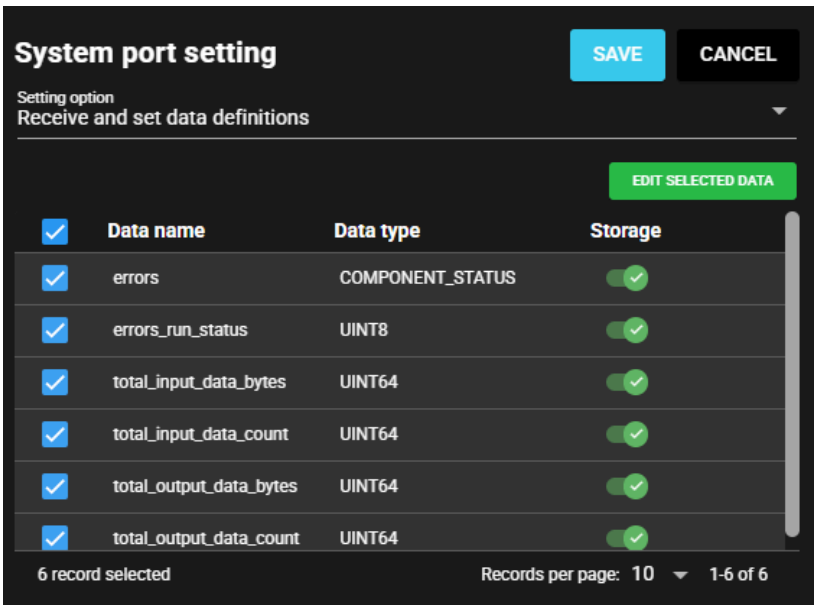


Item	Description
Storage Type	Storage period in the time series database. You can check and change the storage period for GroupA, GroupB, and GroupC in Specify the Storage Keep Time by Group in the Server Settings. You can only make changes when Specify the Storage Keep Time by Group is ON.

7. Press Definition.



8. Press "Save".



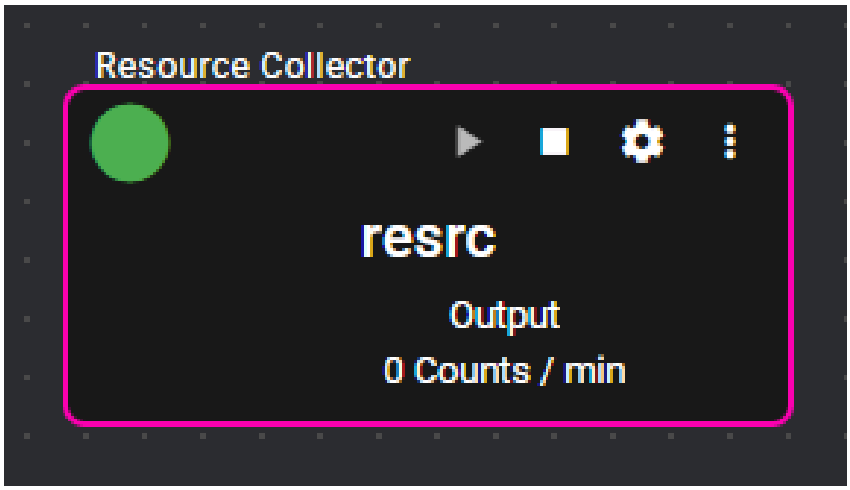
9. The storage configuration will be applied.

5.13 Status Timeline

This page explains the status timeline, which is a list of operating statuses.

5.13.1 Status list

You can check the operating status by the component's status lamp.



There are the following types of operating statuses:

Lamp	Status name	Description
	Normal	Component is running normally.
	Warning	A problem has occurred, but the component is still running.
	Error	A problem has occurred, and the component has stopped.
	Stop	Component has stopped.
	Stopping	Component is transitioning from running state to a stop state.

Lamp	Status name	Description
▬	Load failed	No custom component is registered, or loading has failed, making it unable to use a component.

5.13.2 Status timeline

You can view the history of the components' operating statuses. If a component has a Warning or Error status, you can check the error code.

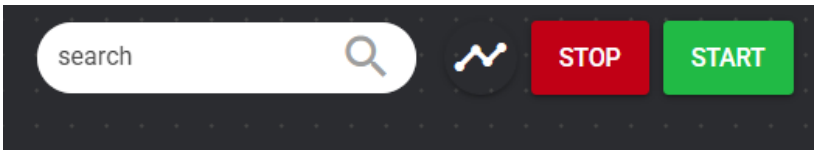
5.13.2.1 View

This section explains how to view the status timeline. You can either display the information of all components or the information of a specific component in the status timeline.

5.13.2.1.1 All components

You can view a status timeline displaying the information of all components by following the steps below:

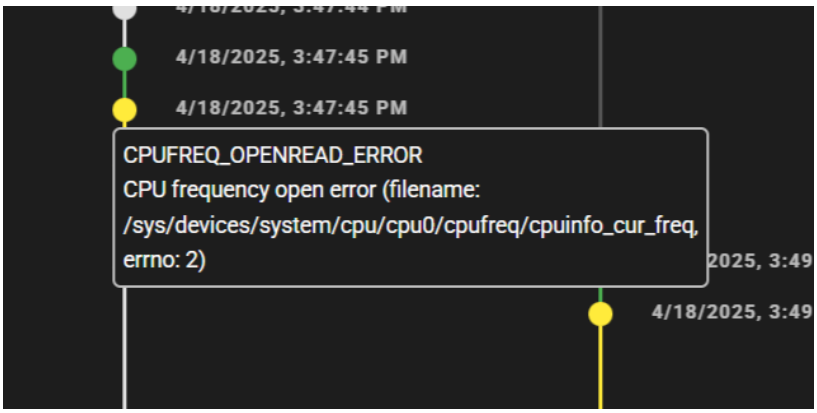
1. Press the Status Timeline icon.



2. The status history of all components is displayed.



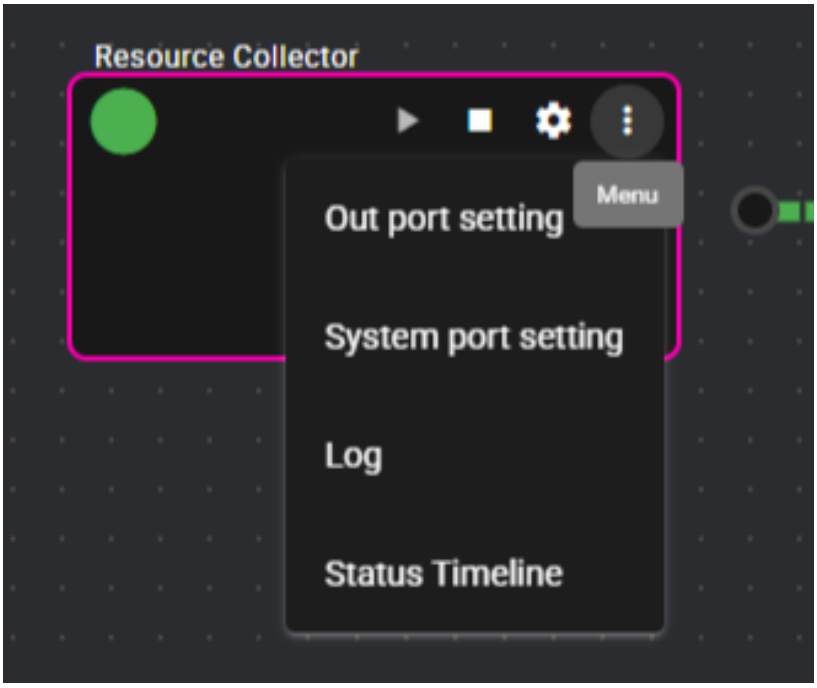
3. Place the cursor over the Warning or Error status to view the error code.



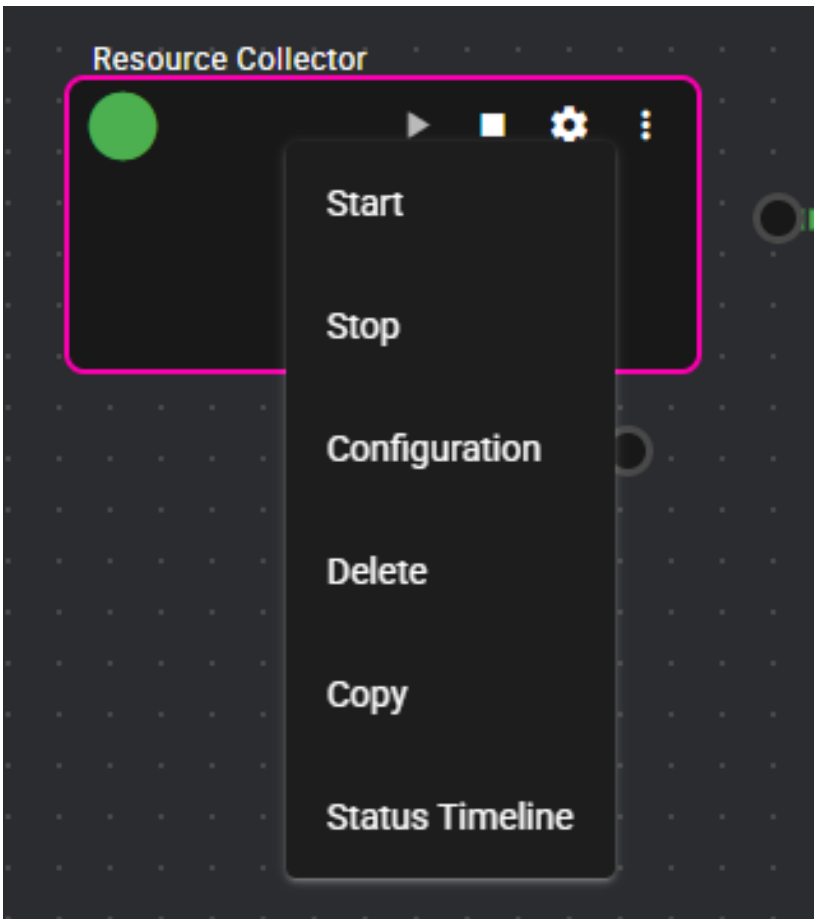
5.13.2.1.2 Individual component

You can view a status timeline displaying a specific component by following the steps below:

1. Open the Status Timeline dialog using one of the following methods.
   - Left-click the menu icon and select [Status Timeline].



   - Right-click the component and select [Status Timeline].



2. The status history of the individual components is displayed. An error code is displayed for Warning or Error status.



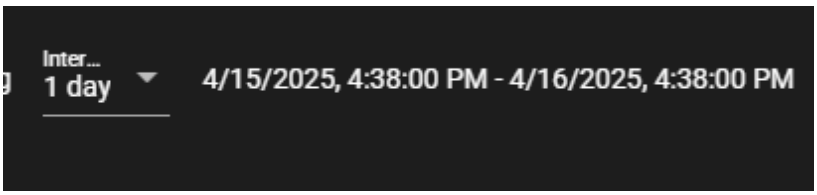
5.13.2.2 Refining the view

This section explains how to refine the information to be displayed in the status timeline.

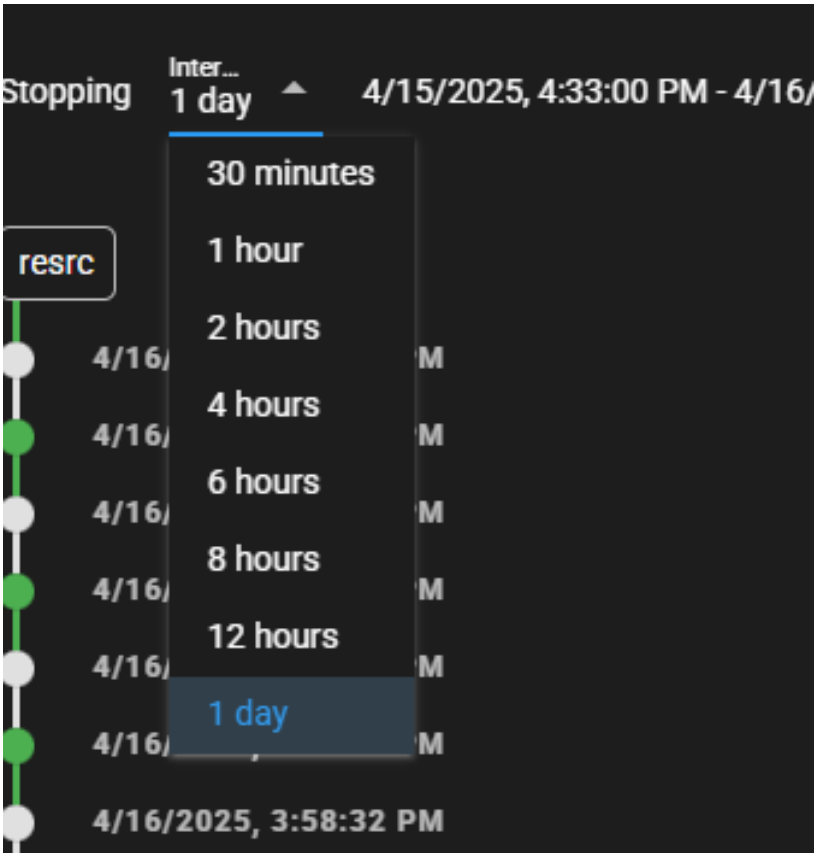
Display interval

You can change the display interval of the status timeline by following the steps below:

1. Press [Interval].



2. Select a time unit.



3. The status history is displayed within the range of the selected time unit.



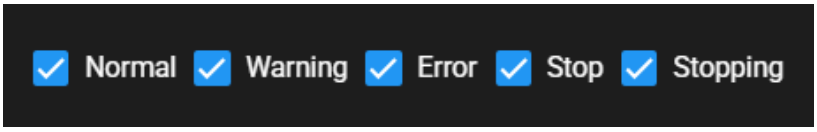
4. Press the up arrow icon to view older history and the down arrow icon to view newer history.



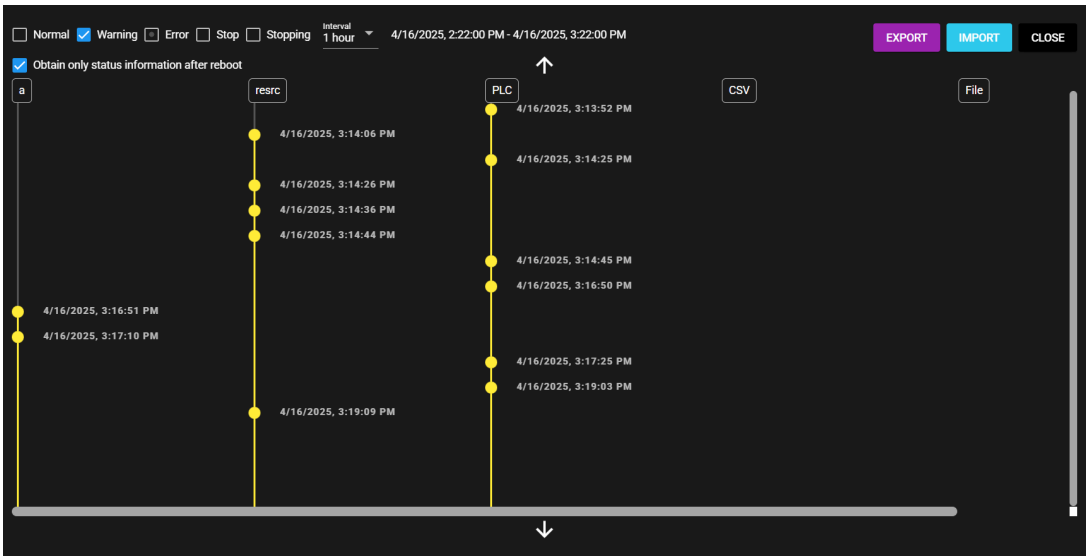
Status

You can display only the specified status(es) in the status timeline by following the steps below:

1. Select or deselect status(es).



2. Of all statuses in the history, only the status types selected are displayed.

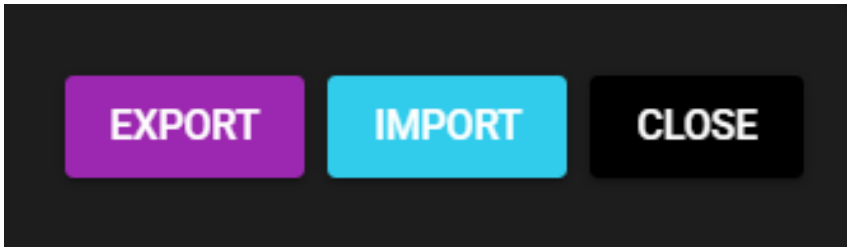


5.13.2.3 Export/Import

You can output the displayed status timeline information to a file. By importing and displaying the output file in SpeeDBee Synapse on a different environment, you can share operating status information.

Export

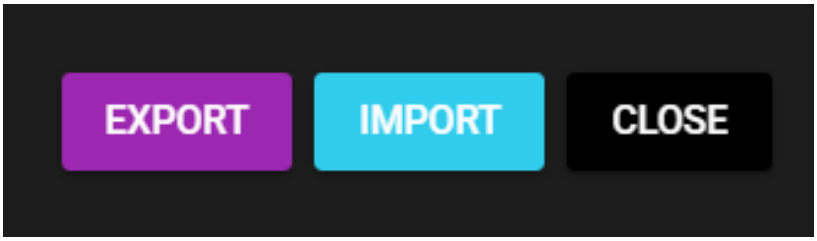
Press [Export] to output the displayed status timeline information to a file.



Import

You can retrieve information from an exported file and view the status timeline by following the steps below:

1. Press [Import].



2. Select an exported file.
3. Information is retrieved from the file, and the status timeline is displayed.

5.14 Various Logs

This section explains various logs of SpeeDBee Synapse.

5.14.1 Log Types

SpeeDBee Synapse handles the following log types.

Type	Description	Viewing on the screen	Download
SpeeDBee Synapse operation log	A history of SpeeDBee Synapse operations performed by users, etc. is recorded. For more information, see Operation Log.	○	○
WEBUI system log	The system information of WEBUI is recorded.	×	○
Core, component, and database system log	The system information of the core, components, and database is recorded.	×	○
Individual component system log	The system information of components is recorded.	○	○

5.14.2 Operation Log

All operations performed by users on SpeeDBee Synapse are recorded in the operation log. The operation log also records operations performed by SpeeDBee Synapse and external APIs.

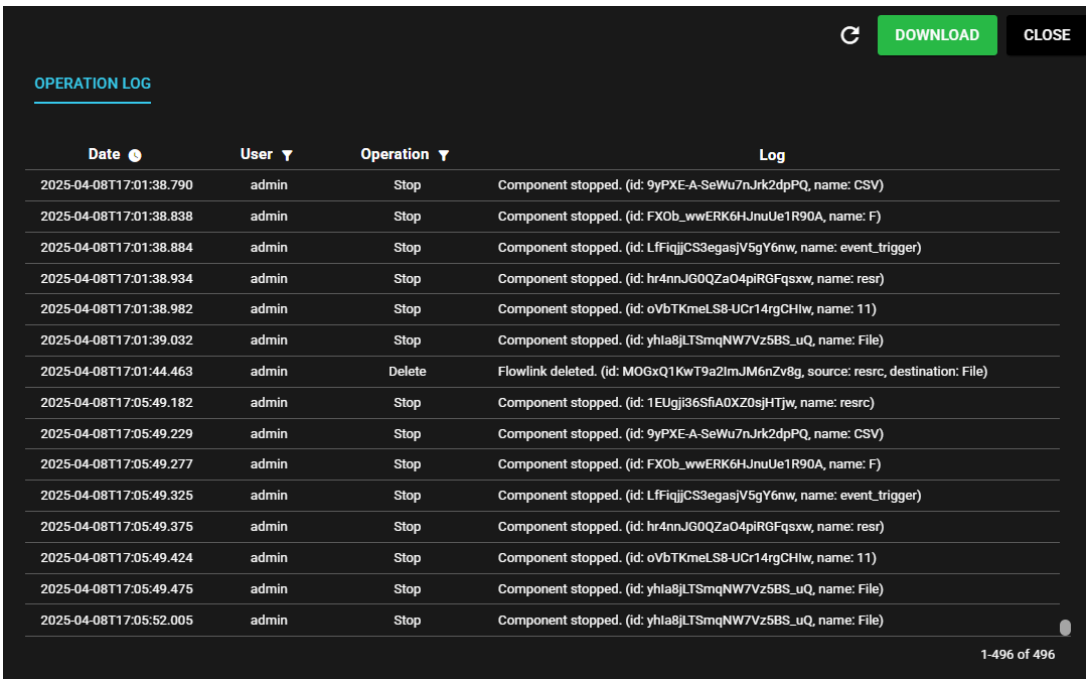
5.14.2.1 View

You can view the operation log by following the steps below.

1. Press the Log icon in the header.



2. The operation log is displayed.



The operation log displays the following information.

Item	Description
Time	Time when the operation was performed
User	User who performed the operation Note 1: "System" refers to an operation performed by SpeeDBee Synapse Note 2: "External API" refers to an operation performed by an external API.
Operation	Operation type
Log	Details of the operation

5.14.2.2 Refresh

You can get the latest log by pressing the refresh button.



5.14.2.3 Narrowing Down

You can narrow down the operation log to be viewed by time, user, and operation type.

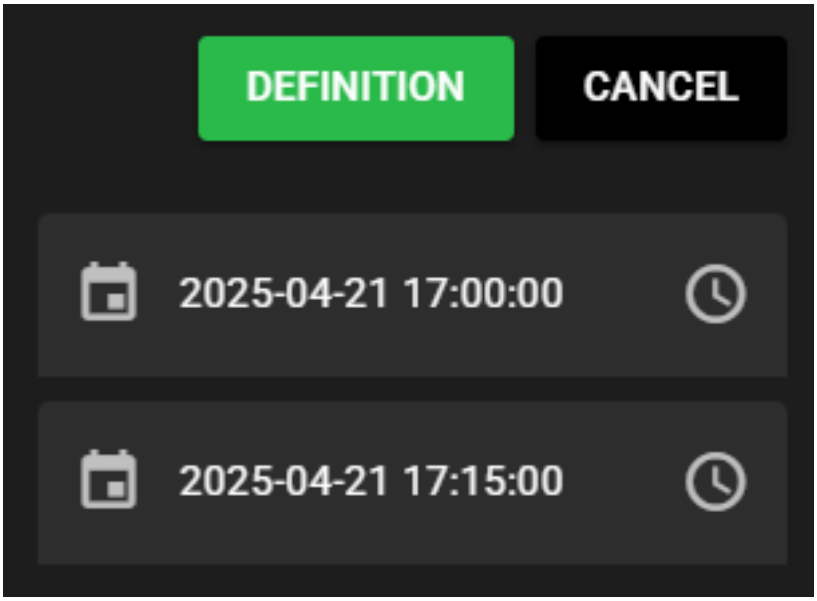
Time

You can view the operation log only for the specified time range by following the steps below.

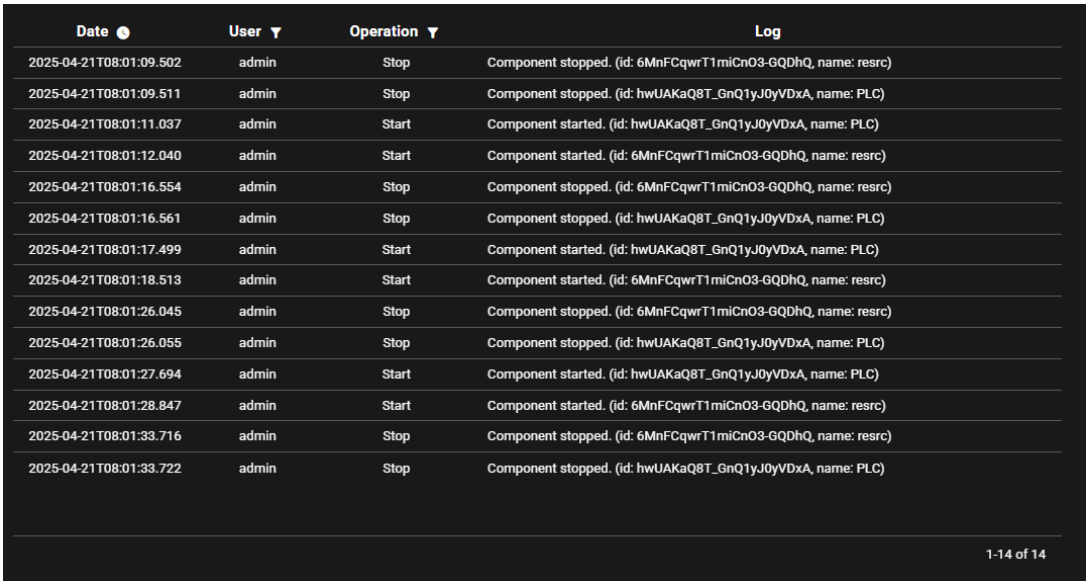
1. Press the icon next to Time.



2. Enter a time range and press Definition.



3. The log for the time range you entered is displayed.



Date	User	Operation	Log
2025-04-21T08:01:09.502	admin	Stop	Component stopped. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:09.511	admin	Stop	Component stopped. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:11.037	admin	Start	Component started. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:12.040	admin	Start	Component started. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:16.554	admin	Stop	Component stopped. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:16.561	admin	Stop	Component stopped. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:17.499	admin	Start	Component started. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:18.513	admin	Start	Component started. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:26.045	admin	Stop	Component stopped. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:26.055	admin	Stop	Component stopped. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:27.694	admin	Start	Component started. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)
2025-04-21T08:01:28.847	admin	Start	Component started. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:33.716	admin	Stop	Component stopped. (id: 6MnFCqwrT1miCnO3-GQDhQ, name: resrc)
2025-04-21T08:01:33.722	admin	Stop	Component stopped. (id: hwUAKaQ8T_GnQ1yJ0yVDxA, name: PLC)

1-14 of 14

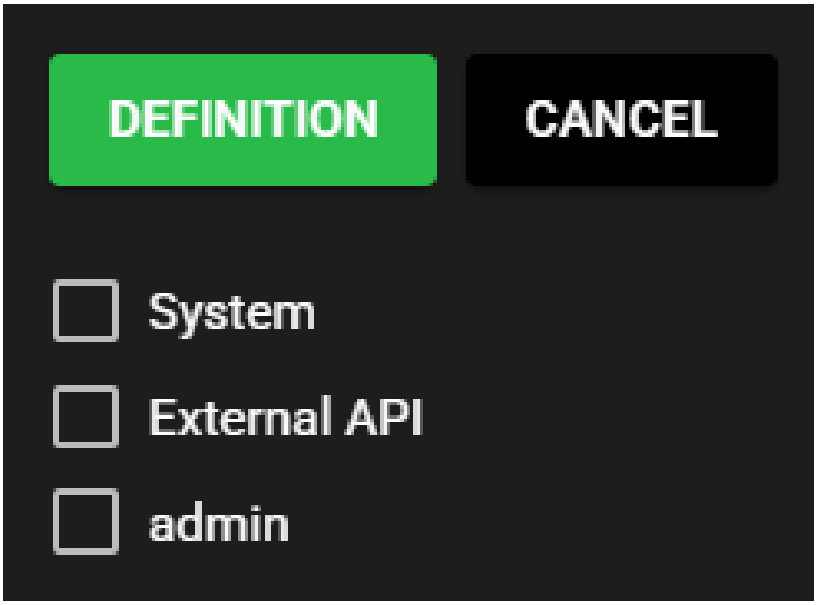
User

You can view the operation log only for the specified user by following the steps below.

1. Press the icon next to User.



2. Select a user and press Definition.



3. A log of operations performed by the selected user is displayed.

Date	User	Operation	Log
2025-04-08T17:05:49.229	admin	Stop	Component stopped. (id: 9yPXE-A-SeWu7nJrk2dpPQ, name: CSV)
2025-04-08T17:05:49.277	admin	Stop	Component stopped. (id: FXOb_wwERK6HJnuUe1R90A, name: F)
2025-04-08T17:05:49.325	admin	Stop	Component stopped. (id: LfFiqjjCS3egasjV5gY6nw, name: event_trigger)
2025-04-08T17:05:49.375	admin	Stop	Component stopped. (id: hr4nnJG0QZaO4piRGFqsxw, name: resr)
2025-04-08T17:05:49.424	admin	Stop	Component stopped. (id: oVbTKmeLS8-UCr14rgCHlw, name: 11)
2025-04-08T17:05:49.475	admin	Stop	Component stopped. (id: yhla8jLTSmqNW7Vz5BS_uQ, name: File)
2025-04-08T17:05:52.005	admin	Stop	Component stopped. (id: yhla8jLTSmqNW7Vz5BS_uQ, name: File)
2025-04-09T09:43:15.578	admin	Delete	Flowlink deleted. (id: KKtS1LrCSJGlyFoX_VaK_g, destination: CSV)
2025-04-09T09:43:15.596	admin	Delete	Component deleted. (id: hr4nnJG0QZaO4piRGFqsxw, name: resr)
2025-04-09T09:43:19.426	admin	Delete	Flowlink deleted. (id: NVEsjdQ9SFe5Lvgnw85hVw, destination: F)
2025-04-09T09:43:19.438	admin	Delete	Component deleted. (id: 9yPXE-A-SeWu7nJrk2dpPQ, name: CSV)
2025-04-09T09:43:23.761	admin	Delete	Component deleted. (id: FXOb_wwERK6HJnuUe1R90A, name: F)
2025-04-09T09:43:30.751	admin	Delete	Component deleted. (id: yhla8jLTSmqNW7Vz5BS_uQ, name: File)
2025-04-09T09:43:34.682	admin	Delete	Component deleted. (id: oVbTKmeLS8-UCr14rgCHlw, name: 11)
2025-04-09T09:43:38.023	admin	Delete	Component deleted. (id: 1EUgji36SfiA0XZ0sjHTjw, name: resrc)

1-504 of 504

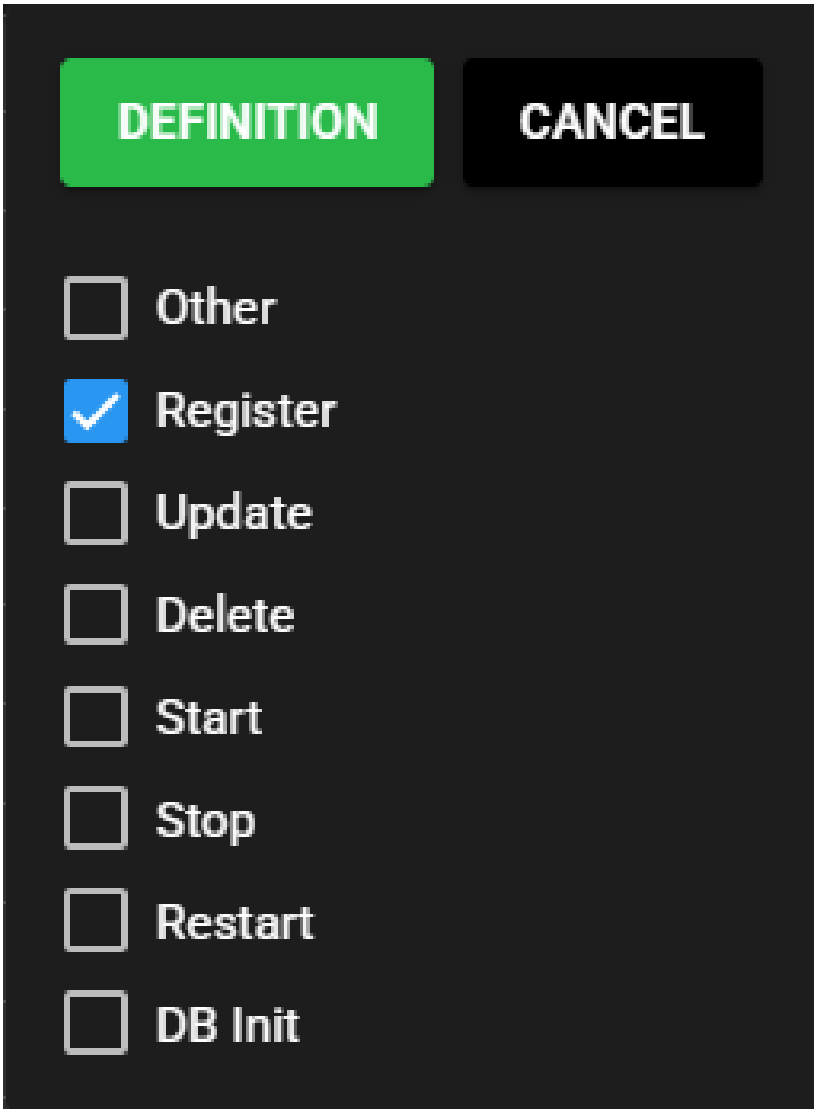
Operation

You can view the operation log only for the specified type of operation by following the steps below.

1. Press the icon next to Operation.



2. Select an operation log type and press Definition.



3. The selected type of operation log is displayed.

Date	User	Operation	Log
2025-04-08T15:14:48.913	admin	Register	Component registered. (id: TEOgjI36SIIA0XZ0sjH1jw, name: 1)
2025-04-08T15:14:53.796	admin	Register	Flowlink registered. (id: 5qPmeMl_Qwy3E12Alo3rrA, source: 1, destination: File3)
2025-04-08T15:17:39.920	admin	Register	Component registered. (id: oVbTKmeLS8-UCr14rgCHlw, name: 11)
2025-04-08T15:17:53.888	admin	Register	Flowlink registered. (id: Q2vPNDcVRLe-sRVgiw1Frw, source: 1, destination: 11)
2025-04-08T15:17:56.155	admin	Register	Flowlink registered. (id: Dp0IWZUKS423yxbU7lXukw, source: 11, destination: File3)
2025-04-08T15:38:19.393	admin	Register	Flowlink registered. (id: PGp7h-uoRnyCRWt5GwfGCQ, source: resrc, destination: File3)
2025-04-08T16:05:25.790	admin	Register	Flowlink registered. (id: sucRils0R12vMxzUF-n_NA, source: resrc, destination: 11)
2025-04-08T16:05:28.144	admin	Register	Flowlink registered. (id: Ad9i-IZHS9eOqlyhlg6gtw, source: 11, destination: File3)
2025-04-08T16:19:11.067	admin	Register	Component registered. (id: LfFiqjjCS3egasjV5gY6nw, name: eventtriger)
2025-04-08T16:19:12.745	admin	Register	Flowlink registered. (id: _T6L8__fSUm_tzYenV76fA, source: resrc, destination: eventtriger)
2025-04-08T16:19:17.411	admin	Register	Flowlink registered. (id: rYzcPCYZSGm96Jw2crIwwQ, source: eventtriger, destination: File)
2025-04-08T16:47:35.041	admin	Register	Flowlink registered. (id: NW3wPC7fRNORsk1Ku4TMAw, source: resrc, destination: File)
2025-04-08T16:53:21.293	admin	Register	Flowlink registered. (id: u1AF1PNyTe29bRKZECFN1g, source: resrc, destination: event_trigger)
2025-04-08T16:53:23.686	admin	Register	Flowlink registered. (id: K-LXcNUmThWmT5VBuimakQ, source: event_trigger, destination: File)
2025-04-08T16:57:41.850	admin	Register	Flowlink registered. (id: MOGxQ1KwT9a2ImJM6nZv8g, source: resrc, destination: File)

1-60 of 60

5.14.3 Component Individual Log

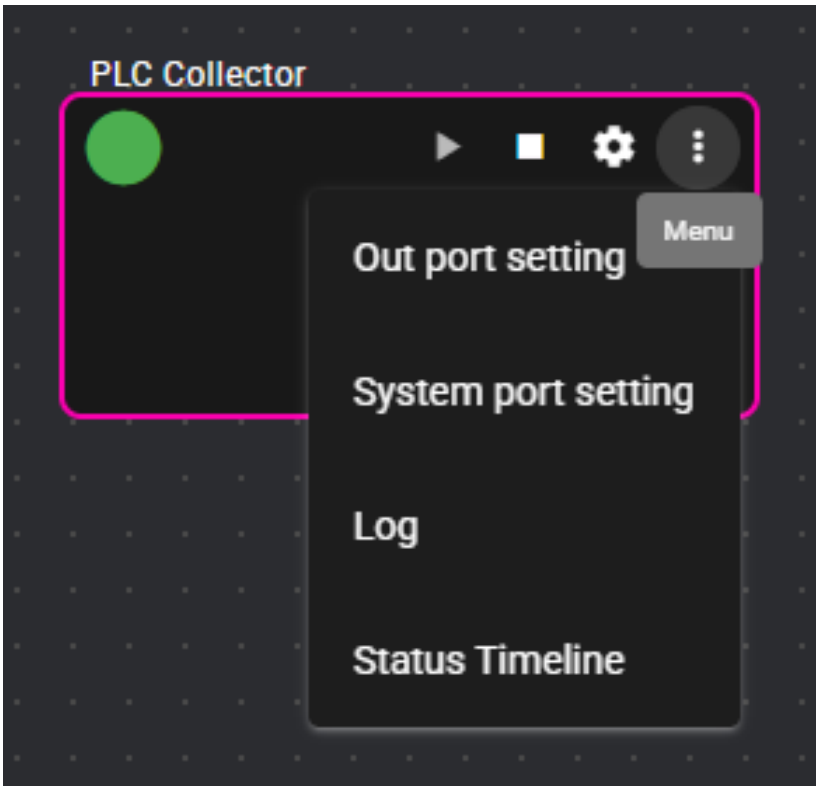
A log output by a component is recorded.

You can also set the level of log output for each component from the log viewing screen of this function.
For more information on the setting method, see Individual Log Level Setting.

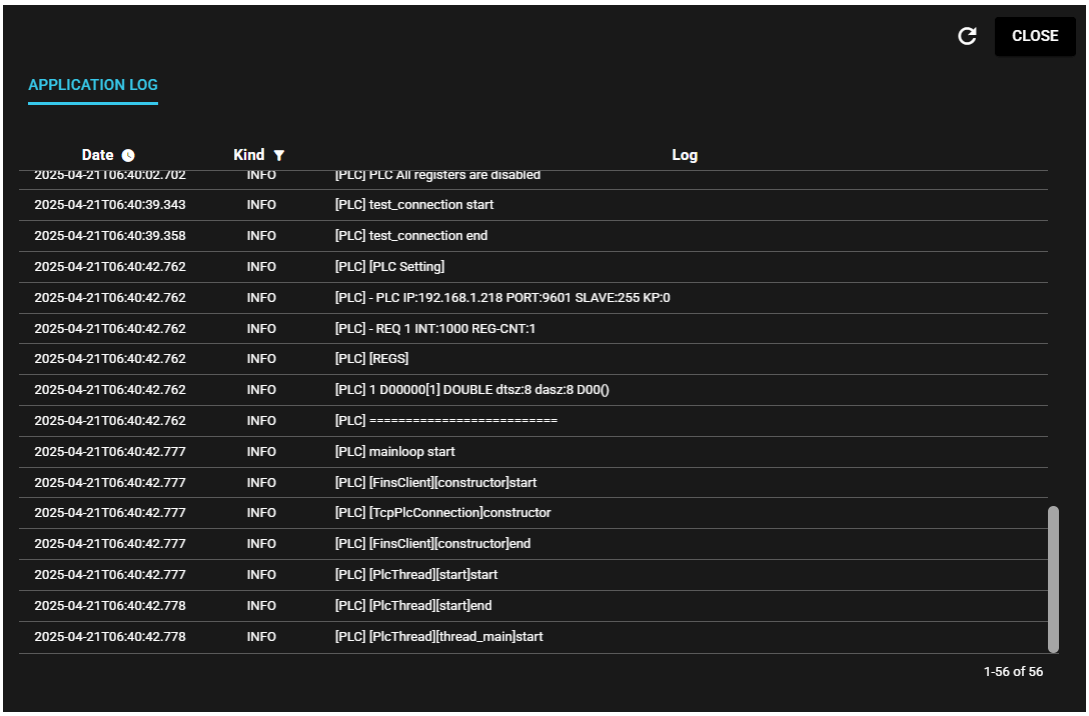
5.14.3.1 View

You can view the component individual log by following the steps below.

1. Left-click the menu icon and select [Log].



2. The component individual log is displayed.



The component individual log displays the following information.

Item	Description
Time	Time when the component output the log entry
Kind	Level of log output
Log	Log details

5.14.3.2 Refresh

You can get the latest log by pressing the refresh button.



5.14.3.3 Narrowing Down

You can narrow down the component individual log to be viewed by time and kind.

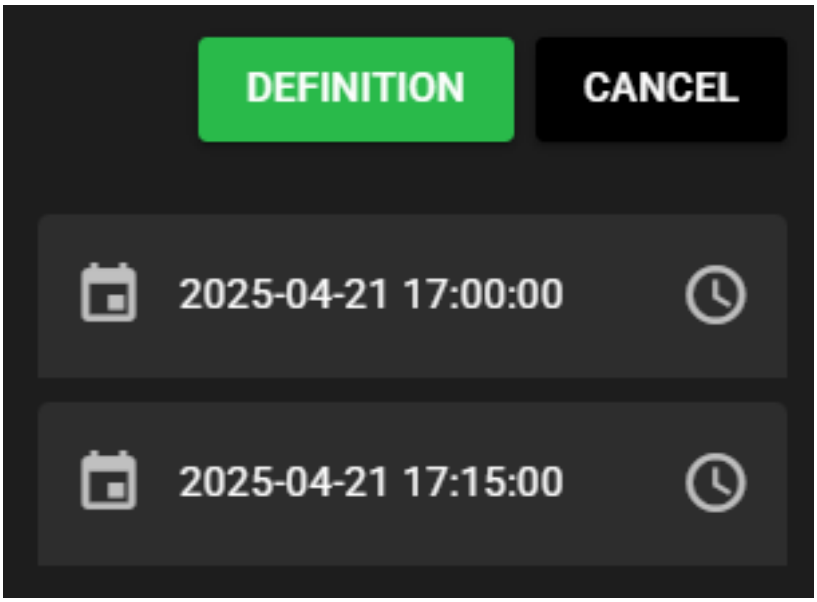
Time

You can view the component individual log only for the specified time range by following the steps below.

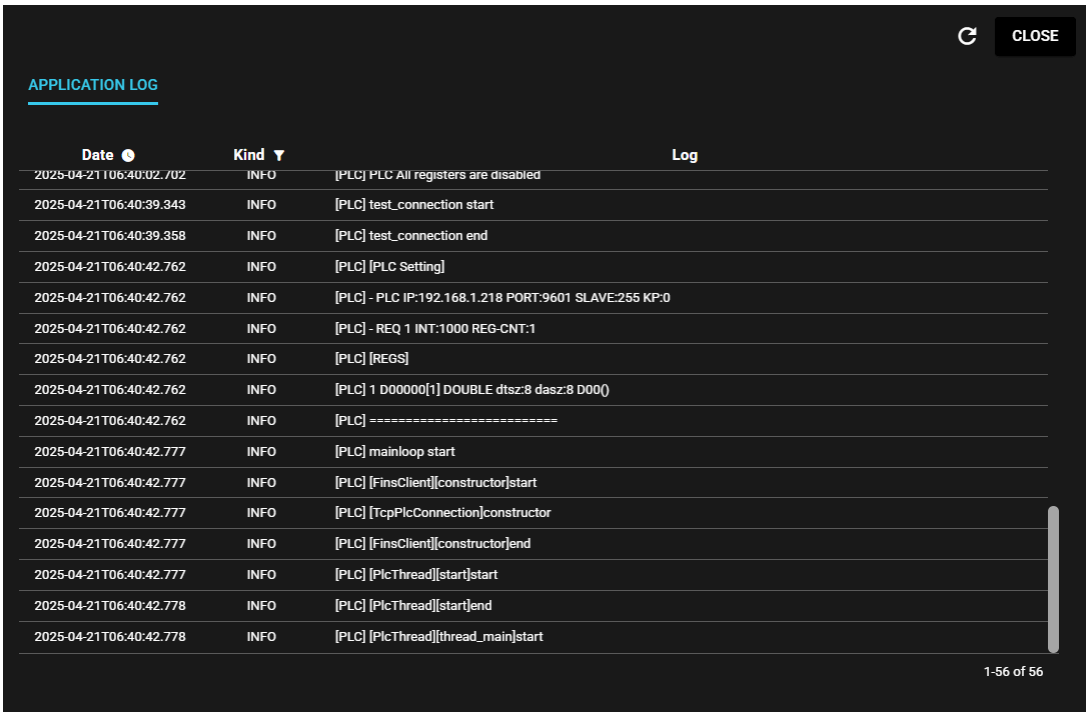
1. Press the icon next to Time.



2. Enter a time range and press Definition.



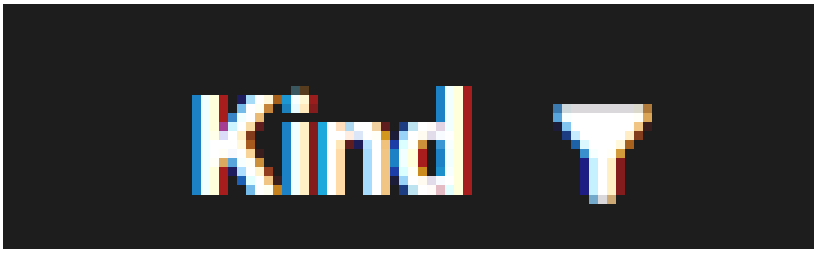
3. The log for the time range you entered is displayed.



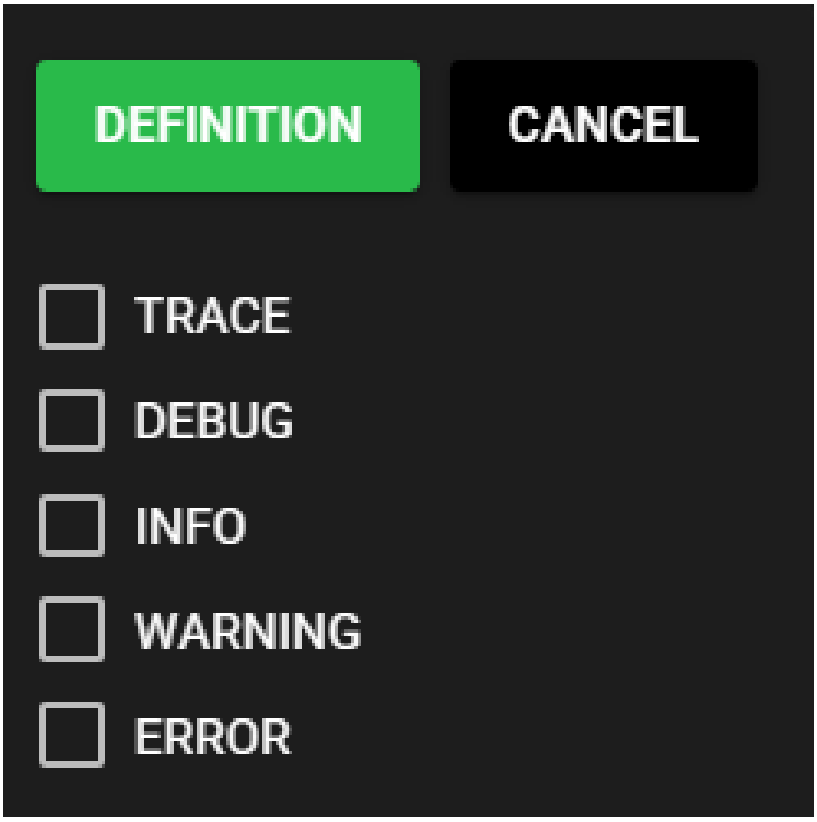
Kind

You can view the component individual log only for the specified kind (log level) by following the steps below.

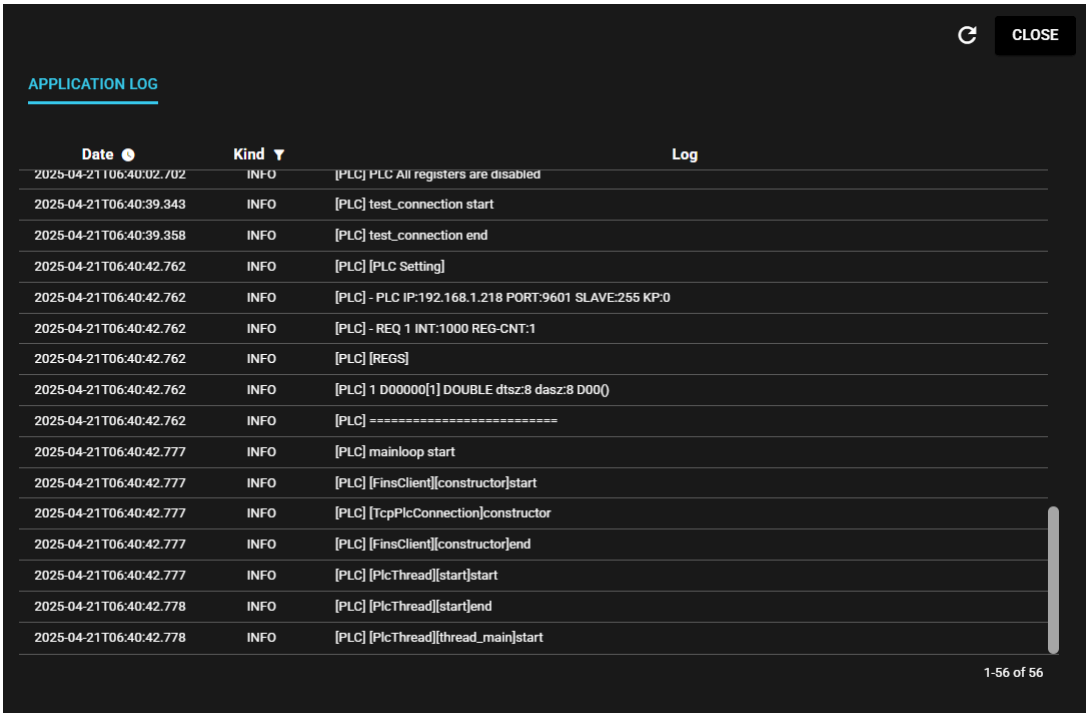
1. Press the icon next to Kind.



2. Select a kind (log level) and press Definition.



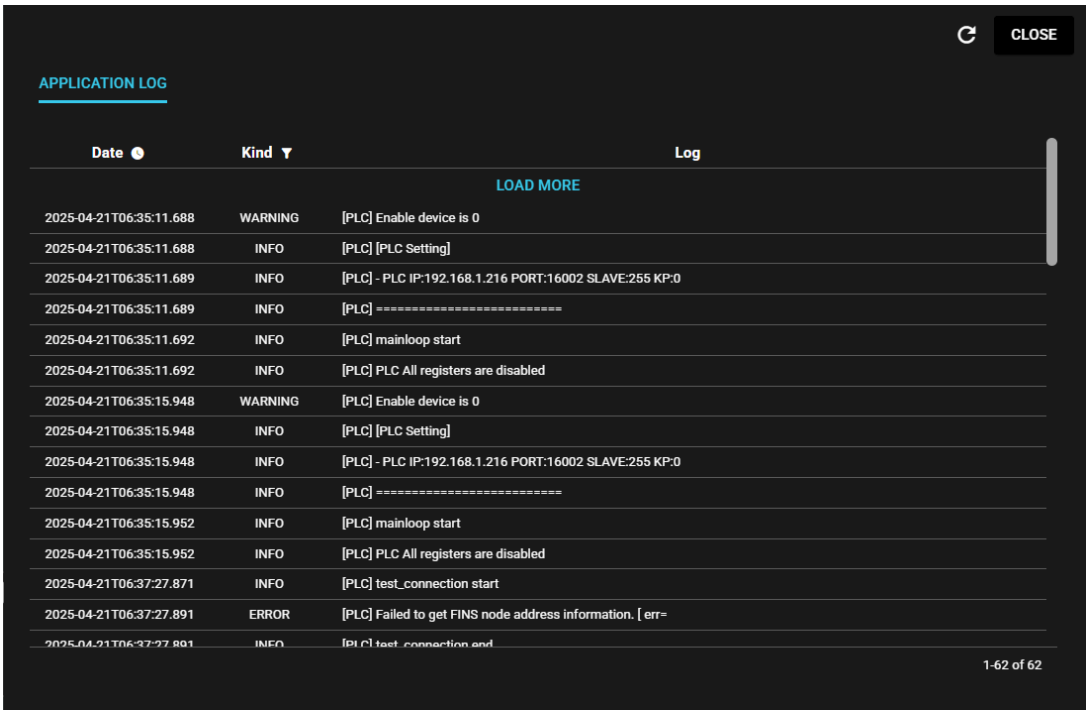
3. The component individual log for the selected kind (log level) is displayed.



5.14.3.4 Load More

You can view older log entries of the component individual log currently displayed by following the steps below.

1. When you scroll the component individual log to the top, Load more is displayed. Press this.



Note: If the oldest log entry is already displayed, Load more is not displayed.

5.14.3.5 Individual Log Level Setting

You can set the level of log output for each component.

When this setting is enabled, the log level set in Log Setting in General Settings has no effect, and log information below the log level set individually is not output.

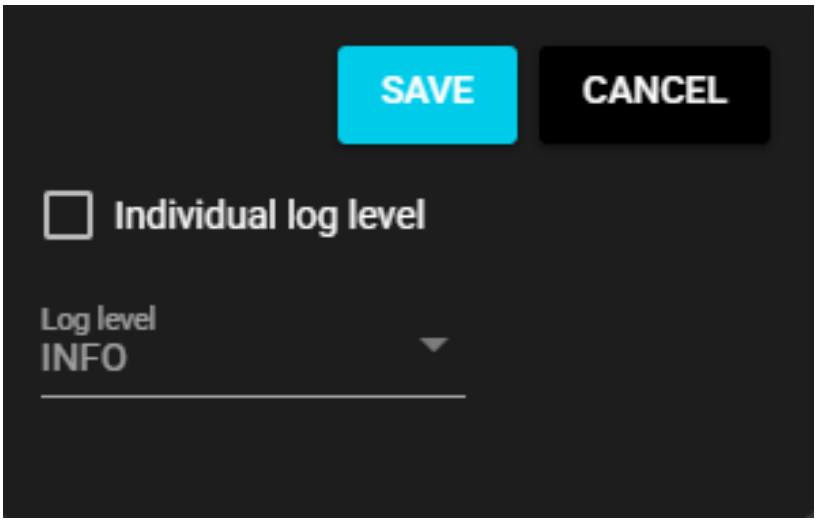
When this setting is disabled, the log is output according to the log level set in Log Setting in General Settings.

You can set the log level for the individual component currently displayed by following the steps below.

1. Press the Configuration icon.



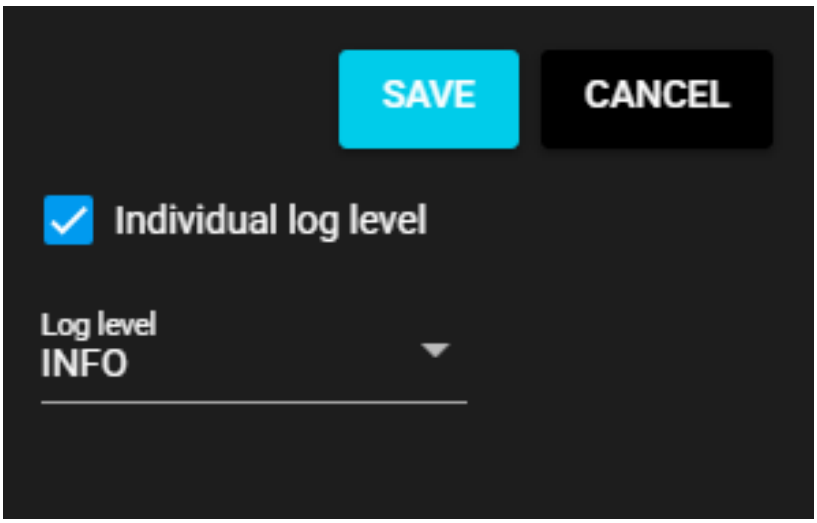
2. The individual log level setting is displayed.



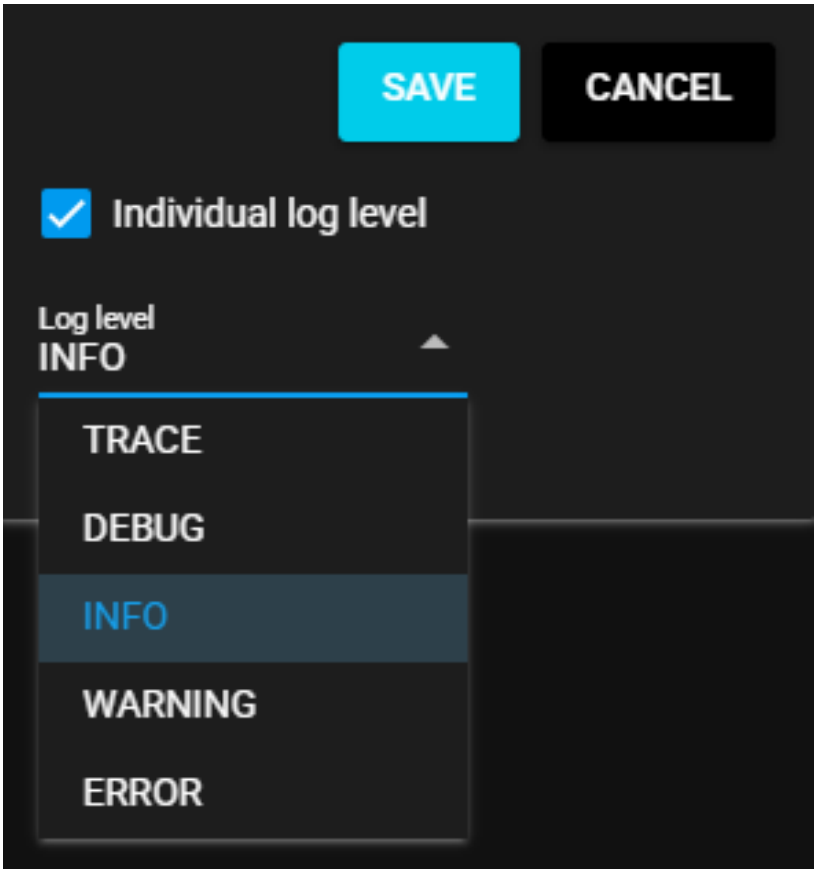
The individual log level setting displays the following information.

Item	Description
Save	Saves the settings
Cancel	Discards the settings
Individual log level	Enables/disables individual log level
Log level	Level of log output

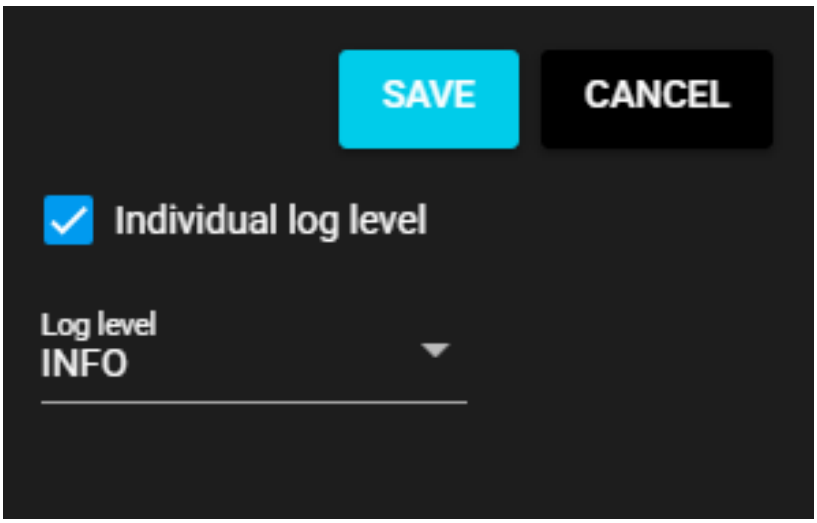
3. Enable Individual log level.



4. Select a Log level.



5. Press Save.



Press Cancel if you don't want to change the settings.

> This operation is only available in Edit Mode.

5.14.4 Download

You can download the various logs of SpeeDBee Synapse by following the steps below. Component Individual logs are included in the download target, but operation logs of all components and SpeeDBee Synapse are also included in the file.

1. Press the Download button.



2. A ZIP file is downloaded. The ZIP file stores log files in the following structure.

ZIP File Configuration

```
├─webui
│  ├─operation_log.YYYYMM.csv (SpeeDBee Synapse operation log)
│  └─speedbeesynapse-webui-backend.log (WEBUI system log)
├─ speedbeesynapse.log (Core, component, and database system log)
└─ speedbeesynapse-YYYYMMDDHHMMSS.log (Old version of speedbeesynapse.log)
```

5.15 Database Query (SQL)

This section explains DB queries (SQL).

5.15.1 DB Query Overview

When you start the DB query service, you can query the time series database using Structured Query Language (SQL) via Web-API (HTTP).

The syntax is unique to SpeeDBee Synapse, but is similar to SQL, which is used for common RDBs.

5.15.2 DB Query Detail

This section explains the Web-API specification and syntax.

5.15.2.1 Web-API Specification

For the Web-API specification of the DB query service, see here.

5.15.2.2 Syntax Specifications

5.15.2.2.1 SELECT

The SELECT clause is used to reference data registered in the time series database.

1. Example statement: Retrieve five cpu_usage records of resource collector r1 for a range up to the current time

```
SELECT "r1:cpu_usage" FROM SPDB WHERE _ts < NOW LIMIT 5;
```

2. Example statement: Retrieve the cpu_usage data of resource collector r1 for a range up to the current time where cpu_usage is 10 or more

```
SELECT "r1:cpu_usage" FROM SPDB WHERE _TS < NOW AND "r1:cpu_usage" >= 10
```

3. Example statement: Retrieve the max and min values of dataA of PLC collector plc1 with an analysis time width of 5 seconds

```
SELECT MAX("plc1:dataA"),MIN("plc1:dataA") FROM SPDB WHERE _ts < NOW GROUP BY TUMBLING(5,sec)
LIMIT 10;
```

Syntax Specification

```
SELECT <select_list> FROM <table> [WHERE <condition>] [GROUP BY <window_condition> [HAVING
<condition>]] [ORDER BY _TS [ASC/DESC]] [LIMIT <number> [OFFSET <start>]|LIMIT <start>,<row_count>]
```

Reserved words are not case sensitive.

- <select_list>

Specify the column to extract or expression. You can specify multiple values separated by commas. Note that the column name must be enclosed in double quotation marks (") when specifying a column used in SpeeDBee Synapse. Column names enclosed in double quotation marks are case sensitive.

Example 1: Column name

```
"plcA:data1", "computeB:DATAa"
```

Example 2: Integer, floating decimal, string

```
3, -5, 1.52, 0.32e2, 'TEST-STR'
```

Integers in SQL are treated as int64. Floating decimals are treated as double and can be represented in exponential notation. Enclose strings in single quotation marks. Double quotation marks are treated as identifiers for column names, etc.

Example 3: Expression

```
"plcA:data2" + 3
```

Available operators	Description
+	Addition

Available operators	Description
-	Subtraction
*	Multiplication
/	Division
%	Remainder

Example 4: Function

```
MAX("compute1:dataA"), AVG("compute2:dataB")
```

Available mathematical functions	Return value	Description
ABS	INT64	Absolute value (integer)
ACOS	DOUBLE	Arc cosine
ASIN	DOUBLE	Arc sine
ATAN	DOUBLE	Arc tangent
CEIL	DOUBLE	Round up after the decimal point
COS	DOUBLE	Cosine
COSH	DOUBLE	Hyperbolic cosine
EXP	DOUBLE	Exponential
FABS	DOUBLE	Absolute value (float)
FLOOR	DOUBLE	Round down after the decimal point
FMOD	DOUBLE	Surplus
LOG	DOUBLE	Natural logarithms
LOG10	DOUBLE	Logarithm with base 10

Available mathematical functions	Return value	Description
POW	DOUBLE	Power
ROUND	DOUBLE	Round to after the decimal point
SIN	DOUBLE	Sine
SINH	DOUBLE	Hyperbolic sine
SQRT	DOUBLE	Square roots
TAN	DOUBLE	Tangent
TANH	DOUBLE	Hyperbolic tangent

Available aggregation and statistical functions	Return value	Description
AVEDEV	DOUBLE	mdev
AVG	DOUBLE	mean
COUNT	INT64	count
CV	DOUBLE	cv
DOF	INT64	degree of freedom
MAX	DOUBLE	max
MIN	DOUBLE	min
SE	DOUBLE	stder
STDEV	DOUBLE	ustdev (unbiased standard deviation)
STDEVP	DOUBLE	stdev (standard deviation)
SUM	DOUBLE	sum

Available aggregation and statistical functions	Return value	Description
SUMSQ	DOUBLE	sumsq (sum of squares)
VAR	DOUBLE	uvar
VARP	DOUBLE	sample var

Available representative values	Return value	Description
LAST	Data type of the value	Window last value
TOP	Data type of the value	Window top value

Example 5: Constant

```
M_PI
```

Available constants	Return value	Description
M_E	DOUBLE	Napier number, base of natural logarithm (e)
M_LOG2E	DOUBLE	log_e(e)
M_LOG10E	DOUBLE	log_10(e)
M_LN2	DOUBLE	log_e(2)
M_LN10	DOUBLE	log_e(10)
M_PI	DOUBLE	Pi
M_PI_2	DOUBLE	Pi/2
M_PI_4	DOUBLE	Pi/4
M_1_PI	DOUBLE	1/Pi
M_2_SQRTPI	DOUBLE	2/Pi

Available constants	Return value	Description
M_SQRT2	DOUBLE	sqrt(2)
M_SQRT1_2	DOUBLE	1/sqrt(2)

Example 6: Pseudocolumn

```
_WID, _TS
```

Available pseudocolumns	Description
_WID	Window ID (ID that changes in analysis width units when GROUP BY is used)
_TS	Timestamp (nanoseconds)

- <table>

Specify what you want to search for time series data.

To search both memory and files

```
SPDB
```

To search memory only

```
SPDB.MEM
```

To search files only

```
SPDB.FILE
```

- <condition>

Specify a conditional expression.

Example 1: Memory usage is less than 50 or CPU usage is 30 or less

```
"r1:mem_usage" < 50 OR "r1:cpu_usage" <= 30
```

Available operators	Description
=	Equivalent

Available operators	Description
!=	Not match
<>	Not match
<	Less than
>	More than
<=	Less than or equal to
>=	More than or equal to
IS [NOT] NULL	NULL, not NULL
BETWEEN AND	In range
BETWEEN EXCLUSIVE AND EXCLUSIVE	Do not include the specified range

Available logical operators	Description
AND	Logical conjunction
OR	Logical disjunction
NOT	Negation

Operators available for _TS	Description
_TS = 100000000	Equal to specified nanoseconds
_TS > 100000000	Exceeding specified nanoseconds
_TS >= 100000000	More than or equal to specified nanoseconds
_TS < 100000000	Less than specified nanoseconds
_TS <= 100000000	Less than or equal to specified nanoseconds

Operators available for _TS	Description
_TS BETWEEN 1000000000 AND 9000000000	In the specified nanosecond range
_TS BETWEEN 1000000000 EXCLUSIVE AND 9000000000 EXCLUSIVE	Range excluding specified nanoseconds
_TS > '20230808T01:23:45.111222333'	Comparison by string specification in ISO8601 format
_TS < NOW	Comparison with the current time
_TS < NOW + 1 hour	Set the current time + 1 hour as the range
_TS < NOW + 10 minutes	Set the current time + 10 minutes as the range
_TS < NOW + 5 sec	Set the current time + 5 seconds as the range
_TS < NOW - 100 msec	Set the current time - 100 milliseconds as the range
_TS < NOW - 100 usec	Set the current time - 100 microseconds as the range
_TS < NOW - 100 nsec	Set the current time - 100 nanoseconds as the range

Note: _TS can only be used on the left side.

You cannot specify more than one time condition in the same SELECT clause.

Use AND to connect with other conditions. OR connection is not allowed.

- <window_condition>

Specify the condition for window analysis. There are two types of windows: the tumbling window, which separates windows to be analyzed by time width, and the value change window, which separates windows by value change.

Example 1: Tumbling window (Split analysis window in 5 seconds)

```
GROUP BY TUMBLING(5,sec)
```

Example 2: Value change window (Separate analysis window when plcA:pressure changes)

```
GROUP BY VCHG("plcA:pressure")
```

Example 3: Value change window (Separate analysis window when plcA:pressure changes or after 30 seconds)

```
GROUP BY VCHG("plcA:pressure", 30, sec)
```

Example 4: Tumbling window + HAVING (Split analysis window in 5 seconds, and extract records whose plcA:pressure max is 10 or more)

```
GROUP BY TUMBLING(10,sec) HAVING MAX("plcA:pressure") >= 10
```

Example 5: Value change window + HAVING (Separate analysis window when plcA:pressure changes or after 30 seconds, and extract records whose plcA:pressure min is less than 5)

```
GROUP BY VCHG("plcA:pressure", 30, sec) HAVING MIN("plcA:pressure") < 5
```

- <row_count>

Specify the number of output rows as an integer.

To set the number of data records to be retrieved to 3

```
LIMIT 3
```

- <start>

Specify the number of rows to skip for the start position as an integer.

To retrieve the first three records after the fifth record

```
LIMIT 5,3
```

- About ORDER BY

You can specify only _TS for ORDER BY. If ORDER BY is not specified, then it is considered that ascending order retrieval with ORDER BY _TS ASC is specified.

Descending order specification

```
ORDER BY _TS DESC
```

- About specifying array type columns

Example 1: If com1:a is an array type, all array elements are referenced.

```
SELECT "com1:a" FROM SPDB;
```

Example 2: If com1:a is an array type, the first and second elements are referenced.
An array element is also specified in the conditional expression.

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE "com1:a"[0] > 1;
```

Example 3: If com1:a is an array type, an array is specified in the conditional expression.
This is implicitly equivalent to com1:a[0].

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE "com1:a" > 1;
```

Example 4: This is an example of an array search ANY. Assume that com1:a has two elements. The following two example statements are equivalent.
ANY specifies that all elements are compared and true is returned if any of them is true.

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE 1 < ANY("com1:a");
```

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE 1 < "com1:a"[0] OR 1 < "com1:a"[1];
```

Note: ANY can only be used on the right side.

Example 5: This is an example of array search ALL. Assume that com1:a has two elements. The following two example statements are equivalent.
ALL specifies that all elements are compared and true is returned if all of them are true.

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE 1 < ALL("com1:a");
```

```
SELECT "com1:a"[0], "com1:a"[1] FROM SPDB WHERE 1 < "com1:a"[0] AND 1 < "com1:a"[1];
```

Note: ALL can only be used on the right side.

5.15.2.2.2 SHOW

The SHOW clause is used to reference meta information in the time series database. Currently, only alias names (which correspond to the column names created by SpeeDBee Synapse) can be referenced.
If ORDER BY is omitted, no sort order is guaranteed.

- SHOW ALIAS

Example 1: Retrieve a list of alias names (ascending)

```
SHOW ALIAS ORDER BY ALIAS_NAME ASC
```

Example 2: Retrieve a list of alias names (descending)

```
SHOW ALIAS ORDER BY ALIAS_NAME DESC
```

Syntax Specification

```
SHOW ALIAS [ORDER BY ALIAS_NAME [ASC/DESC]]
```

5.15.2.2.3 SET

This syntax is used to assign a value to a global variable. In SQL, global variables must always be prefixed with $.

Example 1: Assigning an integer

```
SET $GVAR0 TO 10;
```

Example 2: Assigning a negative value

```
SET $PVAR0 -111;
```

Example 3: Assigning a floating decimal

```
SET $GVAR1 1.235;
```

Example 4: Assigning in exponential expression

```
SET $PVAR1 -1.234E-2
```

Example 5: Assigning a string

```
SET $PVAR2 'abcd'
```

Example of referencing the assigned global variables

```
SELECT \"com1:a\",$GVAR1, $PVAR2 FROM SPDB WHERE _ts BETWEEN NOW - 1 sec AND NOW ORDER BY _TS DESC
LIMIT 1;
```

Syntax Specification

```
SET global variable name [TO] literal value
```

5.15.2.2.4 INSERT

The INSERT clause is used to register data into the time series database. Data can be registered into columns of existing components. You can use SHOW to retrieve a list of column names into which you can register data. The table specified immediately after [INTO] in the INSERT statement is fixed to SPDB.

Example 1: Insert the value "2" into the column "cpu_usage" of the resource collector "resource1 (resource collector)"

```
INSERT INTO SPDB (\"resource1:cpu_usage\") VALUES (2);
```

Note: If the timestamp is omitted, the system time of Operating machine is used as the timestamp to be registered.

Example 2: Insert the value "10" into the column "cpu_usage" of the resource collector "resource1 (resource collector)" with the specified time "April 3, 2025 13:02:09"

Specifying in ISO8606 format

```
INSERT INTO SPDB (_TS, \"resource1:cpu_usage\") VALUES ('2025-04-03T13:02:09', 10);
```

Specifying in epoch nanoseconds

```
INSERT INTO SPDB (_TS, \"resource1:cpu_usage\") VALUES (1743685329000000000, 10);
```

Example 3: Insert the integer value "10"

```
INSERT INTO SPDB (\"customcomp1:columnA\") VALUES (10);
```

Example 4: Insert the floating decimal value "3.14"

```
INSERT INTO SPDB (\"customcomp1:columnB\") VALUES (3.14);
```

Example 5: Insert the string value "ABCDE"

```
INSERT INTO SPDB (\"customcomp1:columnC\") VALUES ('ABCDE');
```

Example 6: Insert the array values {3,4}

```
INSERT INTO SPDB (\"customcomp1:columnD\") VALUES ({3,4});
```

Example 7: Insert multiple data into multiple columns of one collector

```
INSERT INTO SPDB (\"customcomp1:columnA\",
\"customcomp1:columnB\",\"customcomp1:columnC\",\"customcomp1:columnD\",\"customcomp1:columnE\")
VALUES (10,3.14,'ABCDE',{3,4},'00000000000000000000000000')
```

Example 8: Insert multiple data into multiple columns of multiple collectors

```
INSERT INTO SPDB (\"resource1:cpu_usage\", \"customcomp1:columnA\",
\"customcomp1:columnB\",\"customcomp1:columnC\",\"customcomp1:columnD\",\"customcomp1:columnE\")
VALUES (99,10,3.14,'ABCDE',{3,4},'00000000000000000000000000');
```

Syntax Specification

```
INSERT [INTO] SPDB (_TS, <column_name>,...) VALUES (timestamp value, value, ...)
```

5.15.3 Cautions in using the DB query service

The following provides cautions in using the DB query service.

The DB query service of SpeeDBee Synapse, in its design, aims to maintain stable operation even under heavy load. However, the execution of DB queries is highly dependent on the system load condition.
In particular, if the system is running under heavy load, DB query response performance may be temporarily degraded.

System load from query condition

The broader the scope of data that you specify in a query to the database, the heavier the load on the system is.
For example, a query covering a long period or a query with few narrowing-down conditions must scan a large amount of data, which not only increases response time but can affect other processing.

For stable system operation, note the following points when executing queries.

Notes on query execution

- Set an access token
  By publishing an access token and applying it to the DB query service, you can restrict the users who can use DB queries.
- Execute queries to the minimum extent necessary
  Narrow down to the required data and set the period and conditions as specifically as possible.
- Avoid high load times
  If possible, you should avoid executing queries during periods of heavy system activity.

5.16 Arithmetic and Conditional Expressions

This section explains the expressions and conditional expressions that can be used in SpeeDBee Synapse.

5.16.1 Overview of expressions and conditional expressions

Functions that can use expressions and conditional expressions are shown in SpeeDBee Synapse.

Item	Expression	Conditional expression	Description
Compute Collector	○	-	This registers any computation results as Collector data.
Event Data	-	○	When the conditional expression is true, it is registered as data that match the condition.
Event Trigger	-	○	When the conditional expression is true, its event section information is output.
OUT Port Conversion	○	-	This registers any computation results as Collector data.
OUT Port Conditions	-	○	When a conditional expression is true, applicable data are registered.

5.16.2 Expression and conditional expression (common)

This section describes expressions, conditional expressions, and formulas that can be used in both.

Item	
Example	($VALUE + 10) * ABS(5)
Available comparison operations	=,<>,!=,<,>,<=,>=
Available arithmetic operations	(),+,-,*,/,%
Available logical operations	AND,OR,NOT

Item	
Control statement	IF

Available variables	Description
$VALUE	Value received from collector or calculation result by conversion formula
$PVALUE	The value registered in the DB last time
$GVAR0-63	Global variable
$PVAR0-63	Persistence global variable
	Note: Please refer to here for global variables.

Available constants	Description
M_E	Napier number, base of natural logarithm (e)
M_LOG2E	log_e(e)
M_LOG10E	log_10(e)
M_LN2	log_e(2)
M_LN10	log_10(e)
M_PI	Pi
M_PI_2	Pi/2
M_PI_4	Pi/4
M_1_PI	1/Pi
M_2_SQRTPI	2/Pi
M_SQRT2	sqrt(2)

Available constants	Description
M_SQRT1_2	1/sqrt(2)

Available functions	Description
ABS	Absolute value (integer)
ACOS	Trigonometric function: arc cosine
ASIN	Trigonometric function: arc sine
ATAN	Trigonometric function: arc tangent
CEIL	Round up after the decimal point
COS	Trigonometric function: cosine
COSH	Hyperbolic cosine
EXP	Exponential function with base Napier's number e
FABS	Absolute value (float)
FLOOR	Round down after the decimal point
FMOD	Surplus
LOG	Natural logarithms
LOG10	Logarithm with base 10
POW	Power
ROUND	Round to after the decimal point
SIN	Trigonometric function: sine
SINH	Hyperbolic sine

Available functions	Description
SQRT	Square roots
TAN	Trigonometric function: tangent
TANH	Hyperbolic tangent

5.16.3 Expression

This section describes expressions. An expression is a formula used to compute any given operation result. This is used when an operation needs to be performed on a value received from the collector.

Item	
Example	($VALUE + 10) * ABS(5)

About assign of global variable (Reference)

GVAR1 := (expression)

- The "$" is not necessary at the beginning of a global variable to be assigned.
- When an assignment such as "GVAR1 := $VALUE * 2" is specified, the returned calculation result is the value of $VALUE multiplied by 2.

Example
GVAR1 := ($VALUE * 3)
(If $VALUE is 2.)
① 2 * 3 = 6 will be assigned to GVAR1.
② The operation result is the same as the value assigned to GVAR1 (6).

Assignment using the IF statement

IF (condition expression, value if true, value if false)

GVAR1 := IF($VALUE = 1,$GVAR1 + 1, $GVAR1)

Assignment using the IF statement

In the above case, if $VALUE is 1 and it is true, the result of $GVAR1+1 is returned.
If false, $GVAR1 is assigned with the original value.

A caution is that if an assignment is made in "value if true" and "value if false", it is evaluated each time, regardless of whether the condition is true or false.
IF($VALUE=1,GVAR1 := $GVAR1 + 1, $GVAR1)
If specified this way, the "value if true" is evaluated as a formula and assigned every time, regardless of whether the condition is true or false. This may result in receiving an unintended value. Therefore, the assignment must be specified using a statement other than the IF statement.

5.16.4 Conditional expression

This section describes conditional expressions. A conditional expression is an expression used to determine the truth value of a given condition. When a truth value is obtained, it can be used to control subsequent outputs.

Item	
Example	($VALUE + 10) * ABS(5) < 100

About assign of global variable (Reference)

1 < (PVAR5 := (expression))

- The "$" is not necessary at the beginning of a global variable to be assigned.
- As in "PVAR5 := ($VALUE * 2)", it is not allowed to only specify an assignment. It must be specified as a conditional expression.

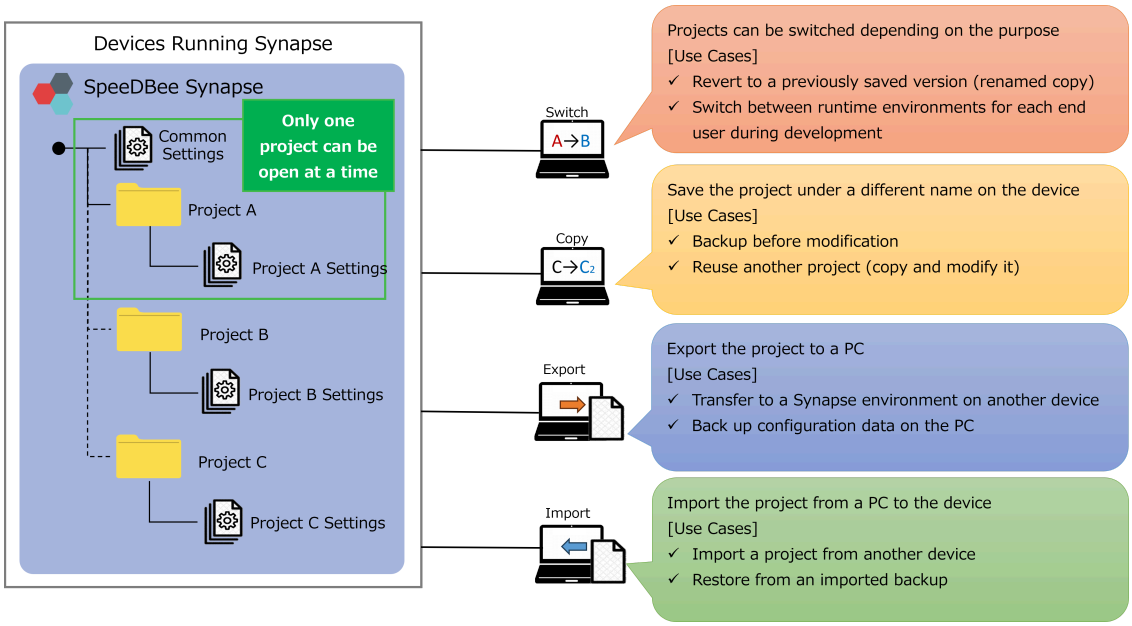
Example
GVAR1 := ($VALUE * 3)
(If $VALUE is 2.)
① 2 * 3 = 6 will be assigned to GVAR1.
② The operation result is the same as the value assigned to GVAR1 (6).

5.17 Managing Configuration Information (Project)

5.17.1 Project Overview

Projects are the mechanism for managing the configuration in SpeeDBee Synapse. The configuration in SpeeDBee Synapse is divided into common configuration and project-specific configuration. By switching projects, you can manage multiple configurations on the same device. You can also use features such as exporting project settings and importing them to other devices.

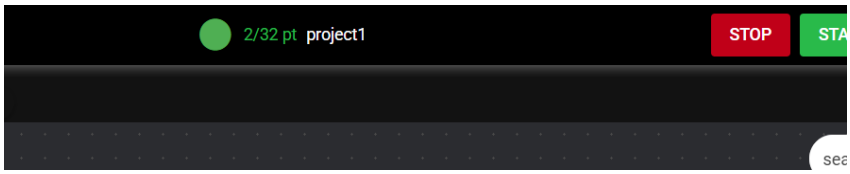
The following illustration shows the project's features.



The following is a list of common and project-specific configurations.

Common/Project-specific	Configuration Details
Common	- License Information - User Management/Authentication - Encrypt Communication - Server Settings - Log - Database - Access Token
Project Specific	- Configured Data Flow (Panel, Component, Flowlink) - Registered Custom Components - Registered Dependencies (Python) - Data Output by Components - Grafana Settings

While SpeeDBee Synapse is running, one project is always selected. The currently selected project is displayed in the header.



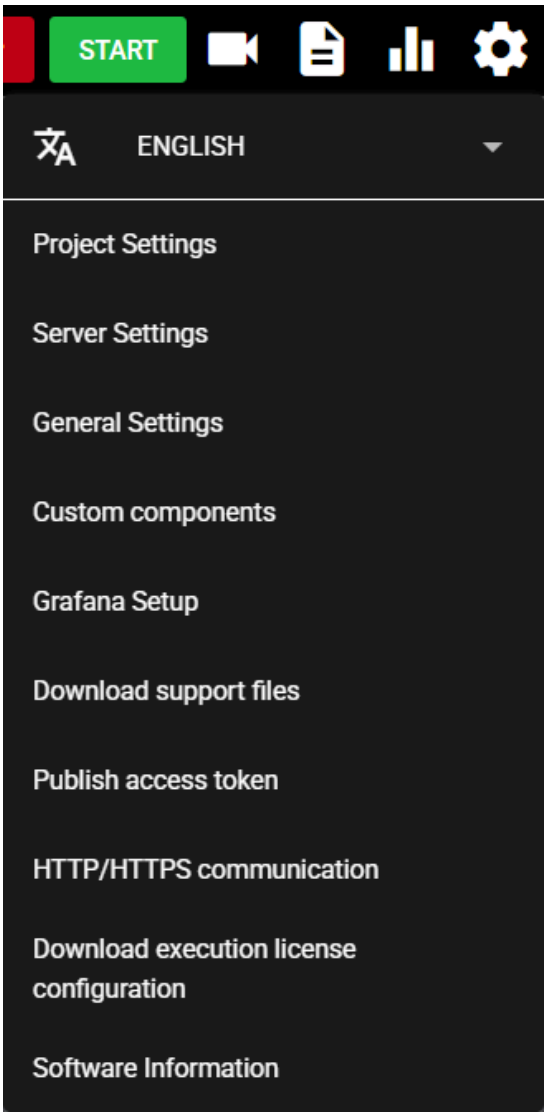
You cannot perform the following operation to the currently selected project.

- Delete

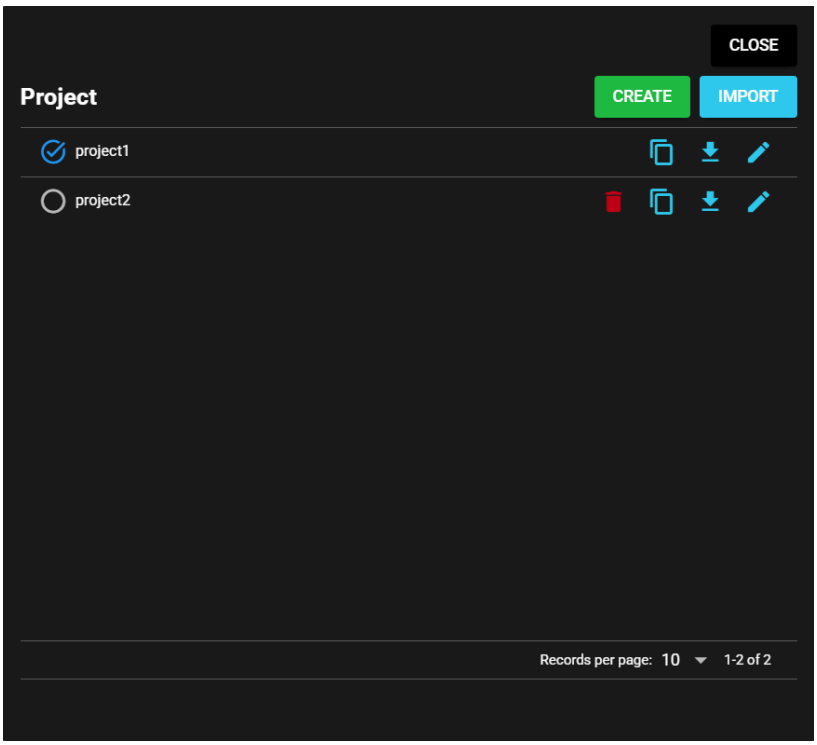
5.17.2 Create project

You can create a project by following the steps below.

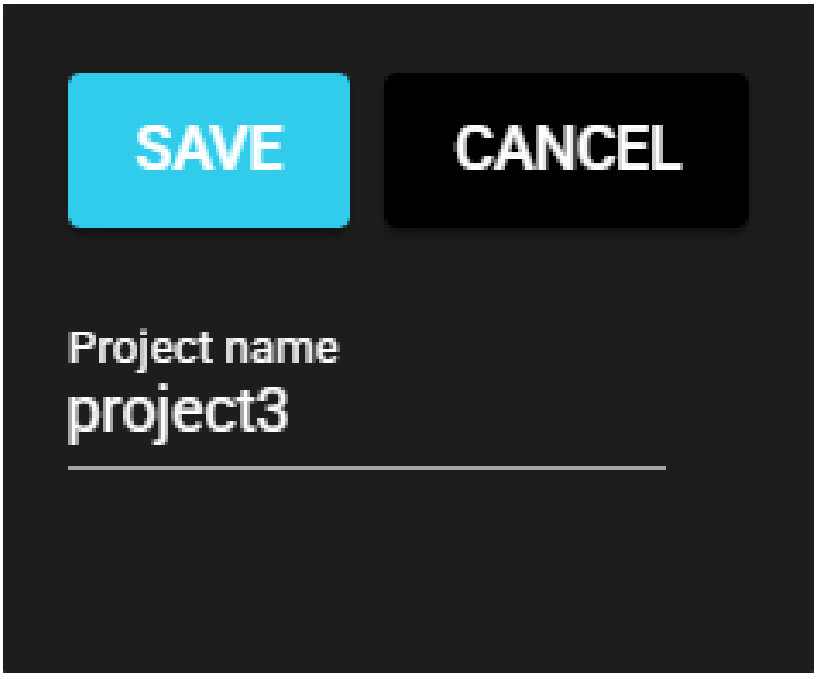
1. Press the Setting Menu icon and select Project Settings.



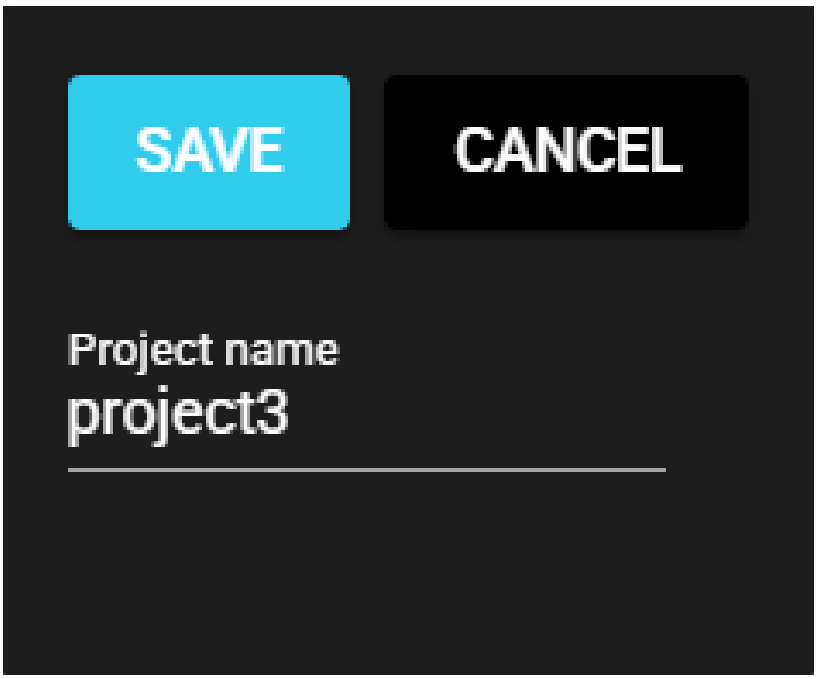
2. Press the Create button.



3. Input a name for your newly created project.



4. Press Save.

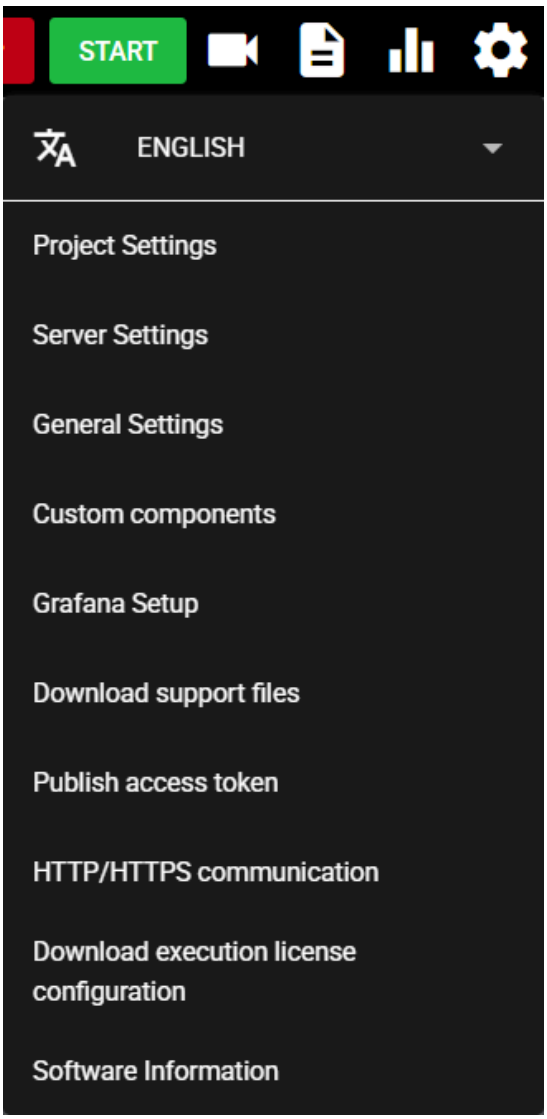


5. A new project will be created.

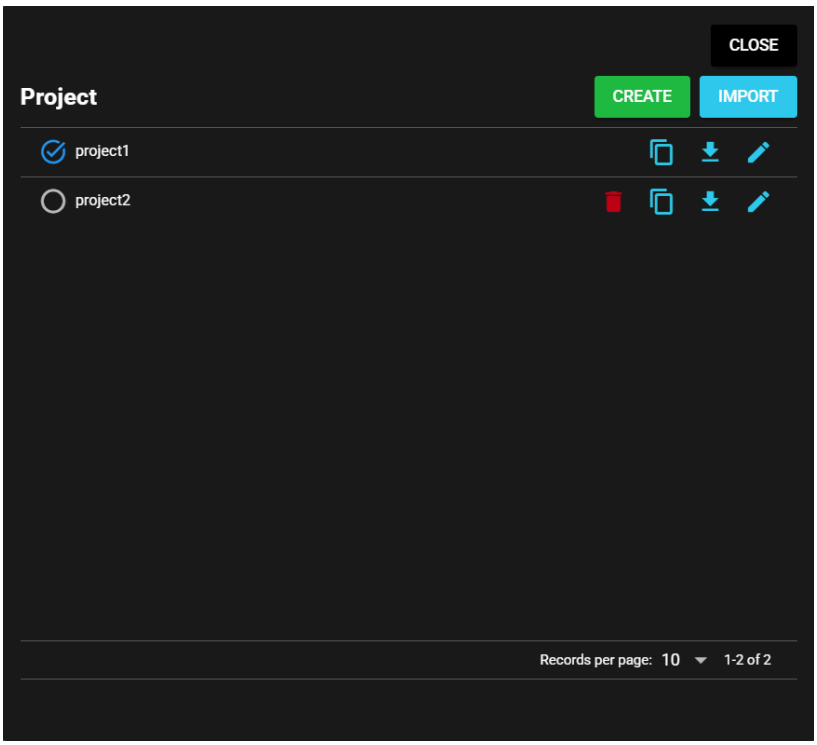
5.17.3 Copy project

You can make a copy of the created project by following the steps below.

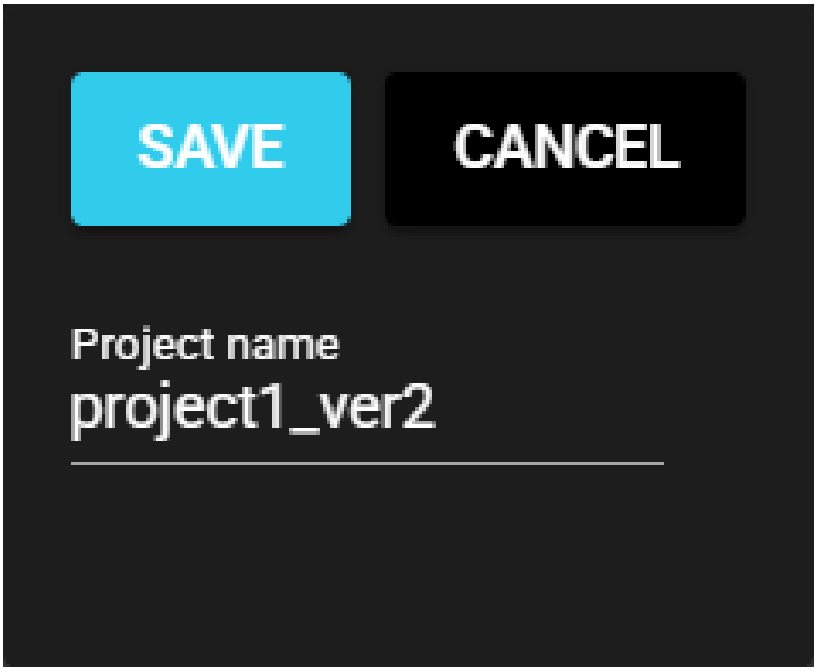
1. Press the Setting Menu icon and select Project Settings.



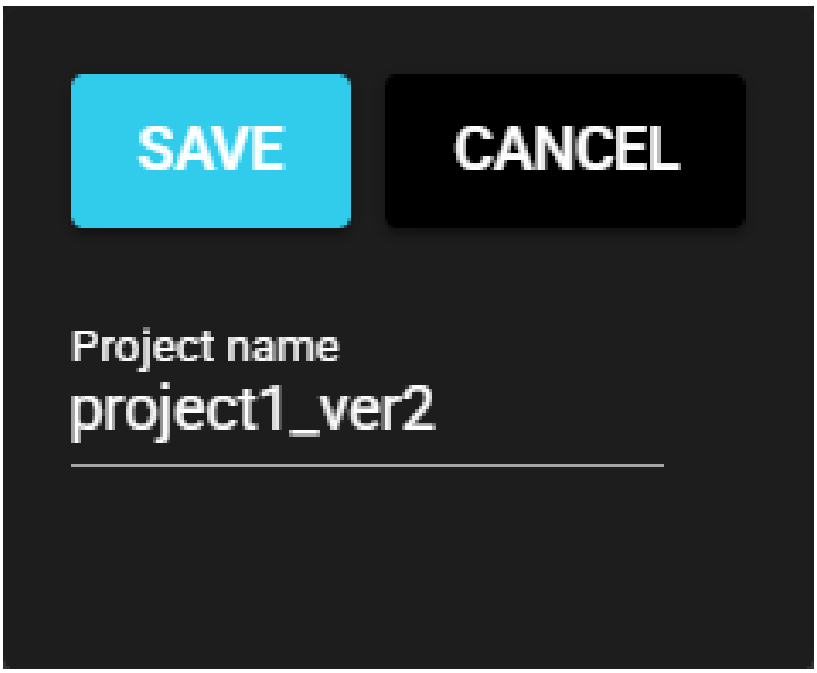
2. Press the Copy Icon (Replicate) for the project you want to replicate.



3. Input a replicated project name.



4. Press Save.

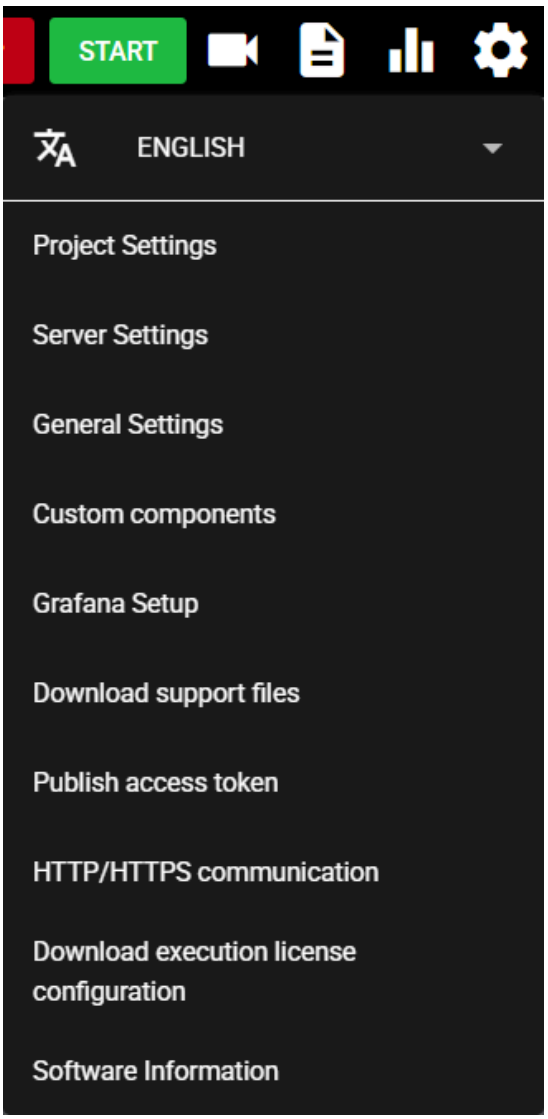


5. A replicate copy of the project will be created.

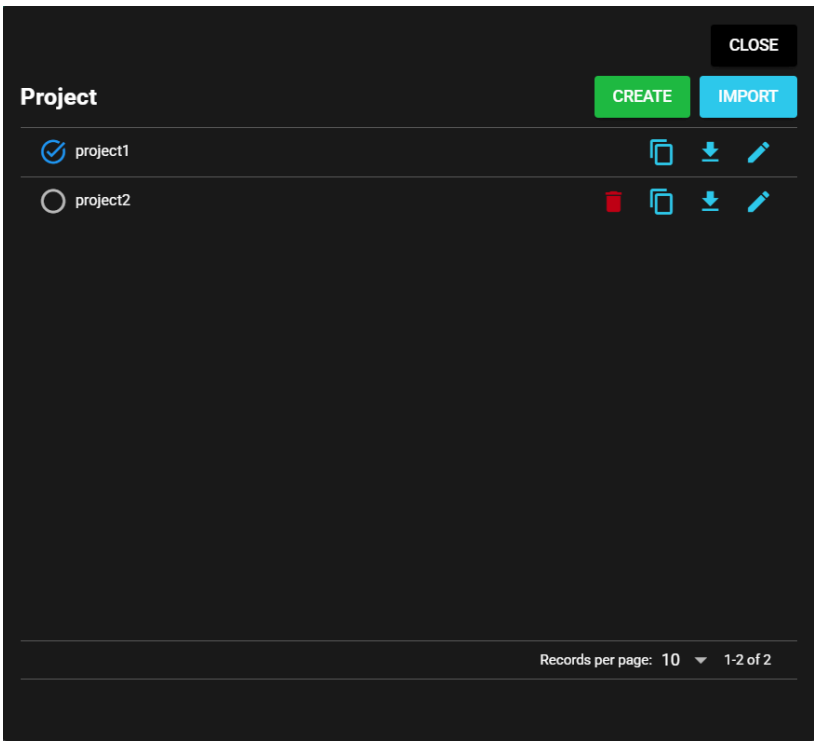
5.17.4 Import project

You can import a project by following the steps below.

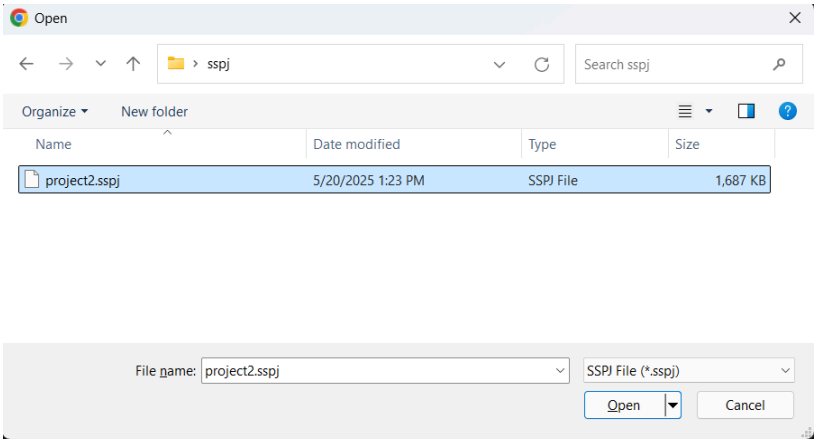
1. Press the Setting Menu icon and select Project Settings.



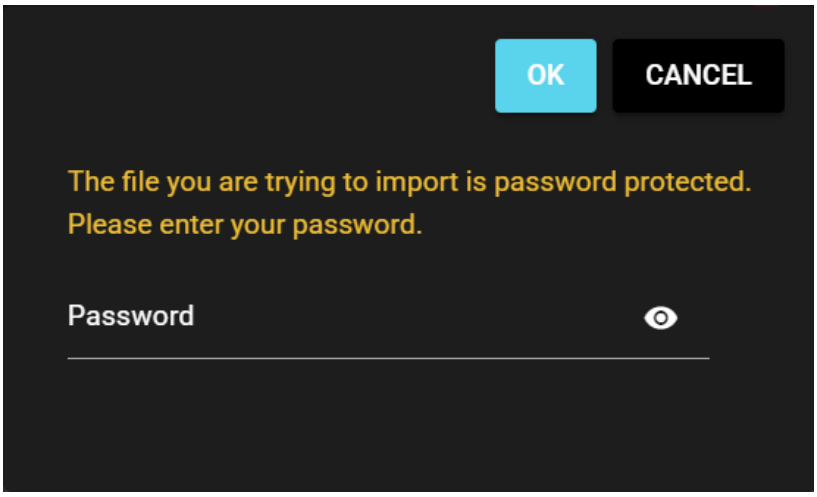
2. Press the Import button.



3. Follow the Export steps and select the exported file.



4. If the file is protected by a password, input the password and select OK.



5. The project will be imported.

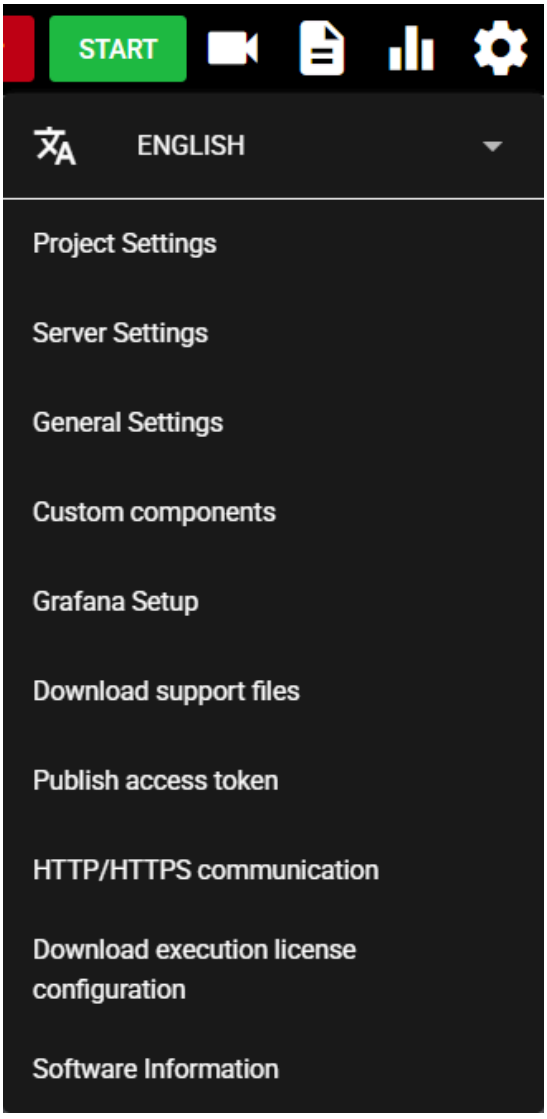
Caution

- The dependencies of Python custom components are installed over the Internet. Therefore, if there is no Internet connection, the Python Modules will not be installed.
- Custom components in C programming language will not be imported if the import target platform is different.
- When migrating projects between different platforms, dependency modules may need to be reinstalled.
- Python modules added directly from wheel files (.whl files) will not be migrated. Please manually reinstall them after import if necessary.

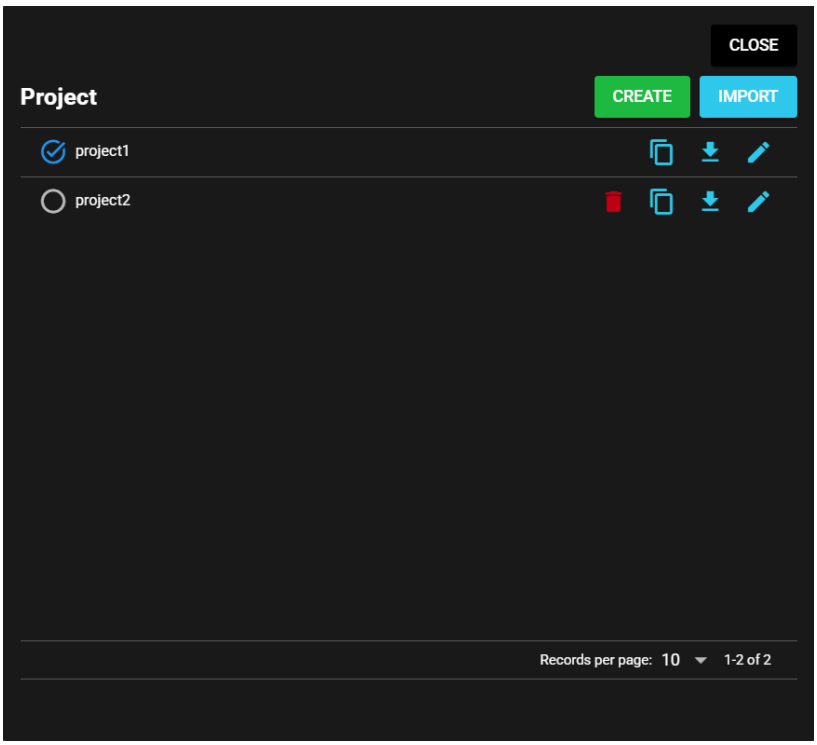
5.17.5 Export project

You can export a project by following the steps below.

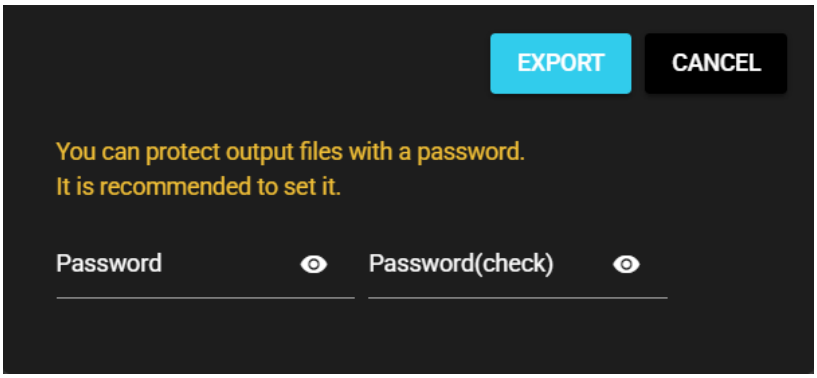
1. Press the Setting Menu icon and select Project Settings.



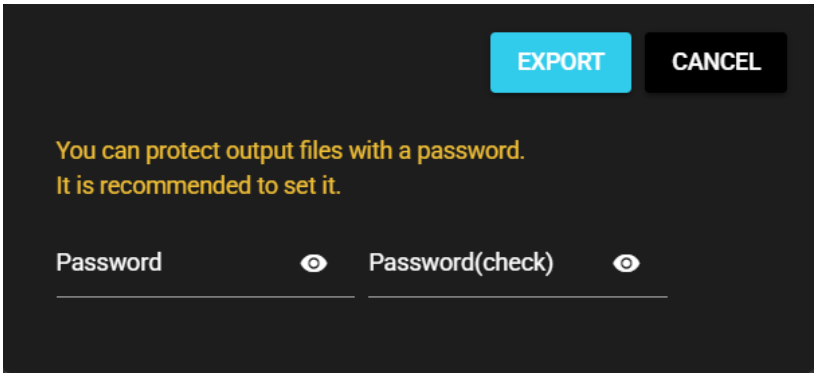
2. Press the Export Icon (Export) for the project you want to export.



3. If you want to protect the exported project with a password, input any password you wish.
   Note: This step can be skipped.



4. Select Export.

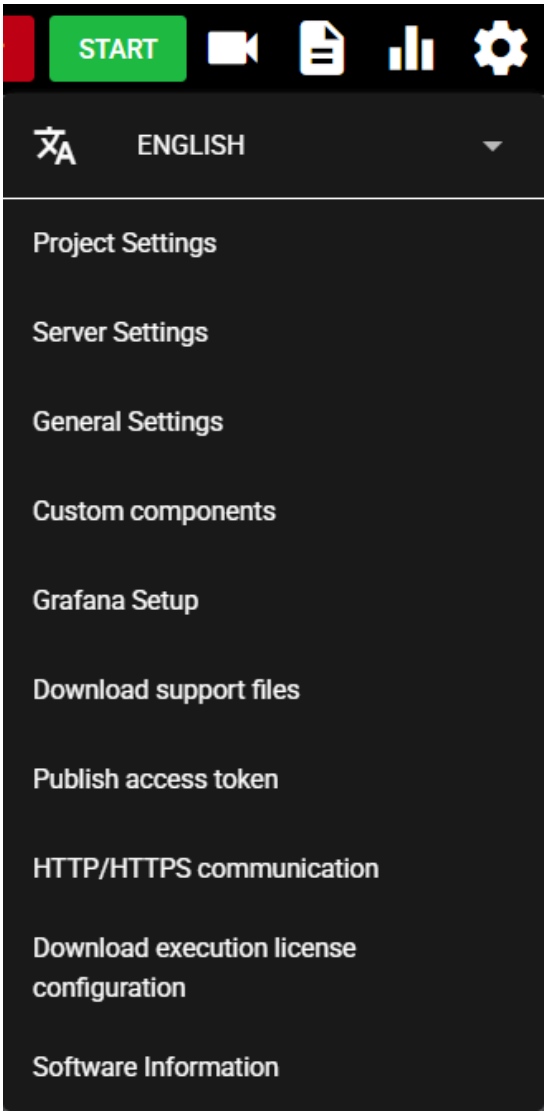


5. The project will be exported.

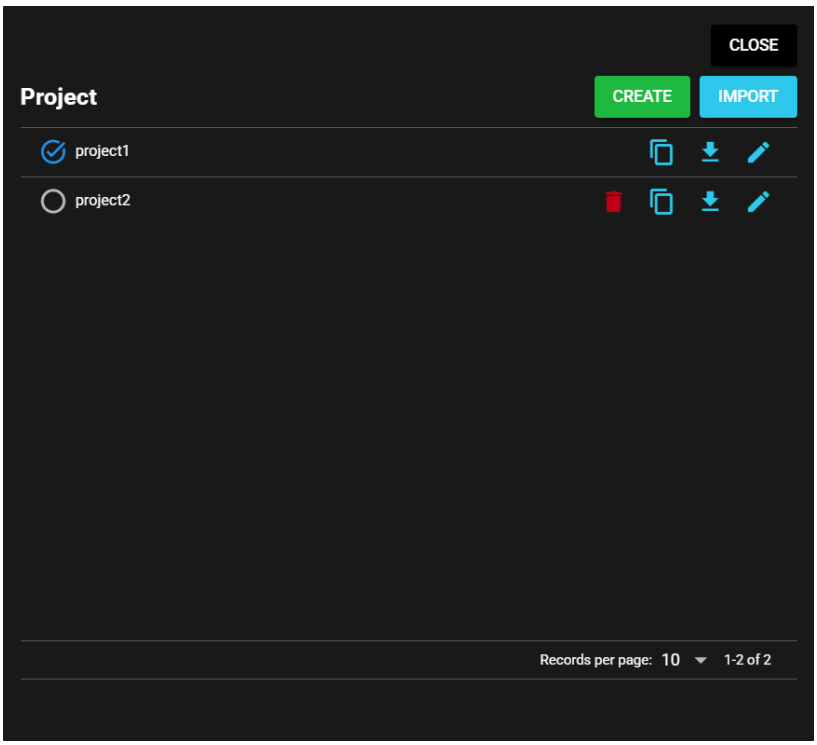
5.17.6 Switch project

You can switch projects by following the steps below.

1. Press the Setting Menu icon and select Project Settings.



2. Select the radio button for the project you want to switch to.

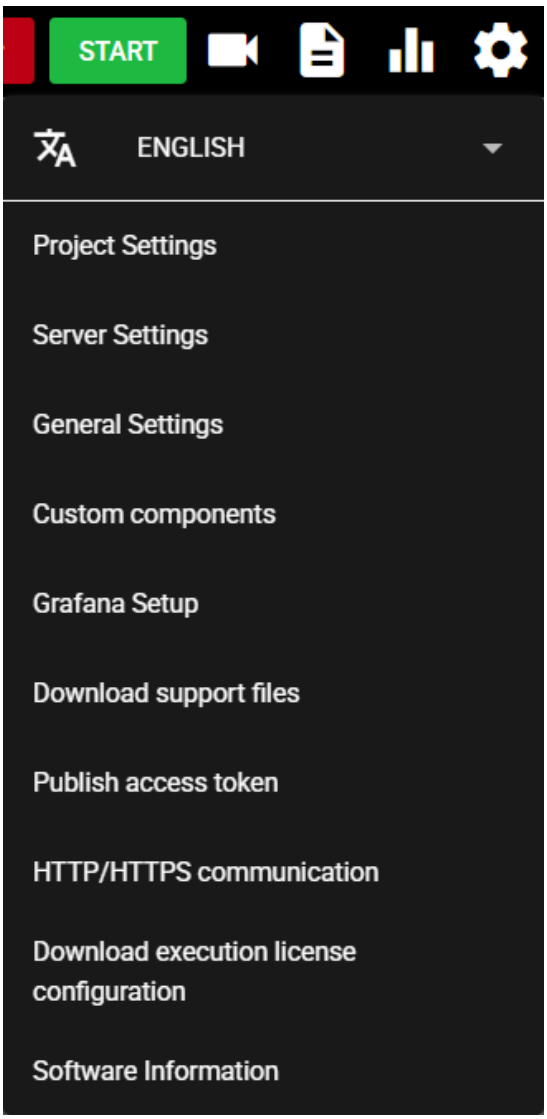


3. Select Yes in the confirmation dialog.
4. The project will be switched.

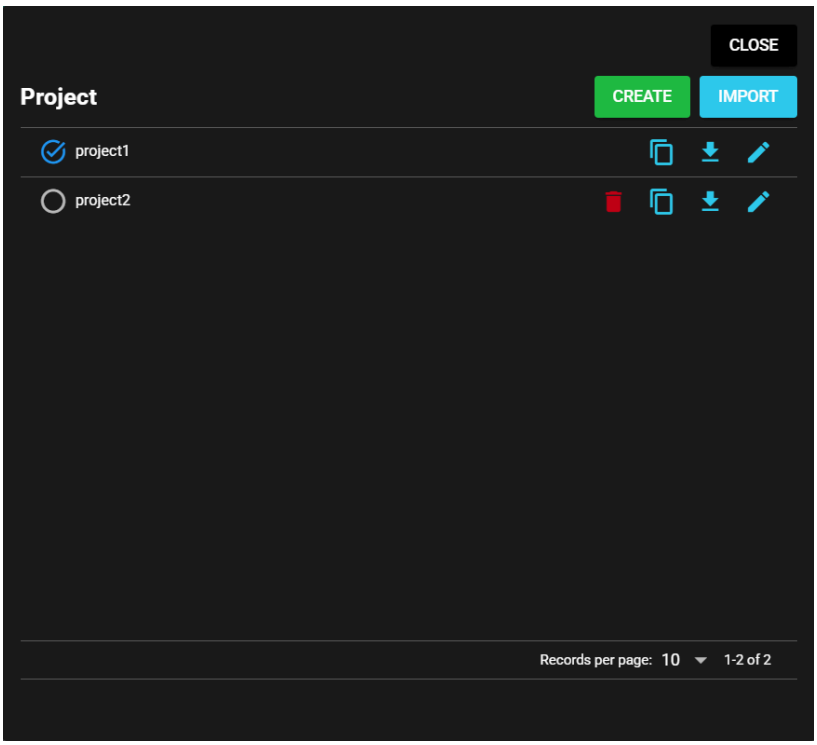
5.17.7 Information Update

You can update the project information by following the steps below. The only information you can update is the project name.

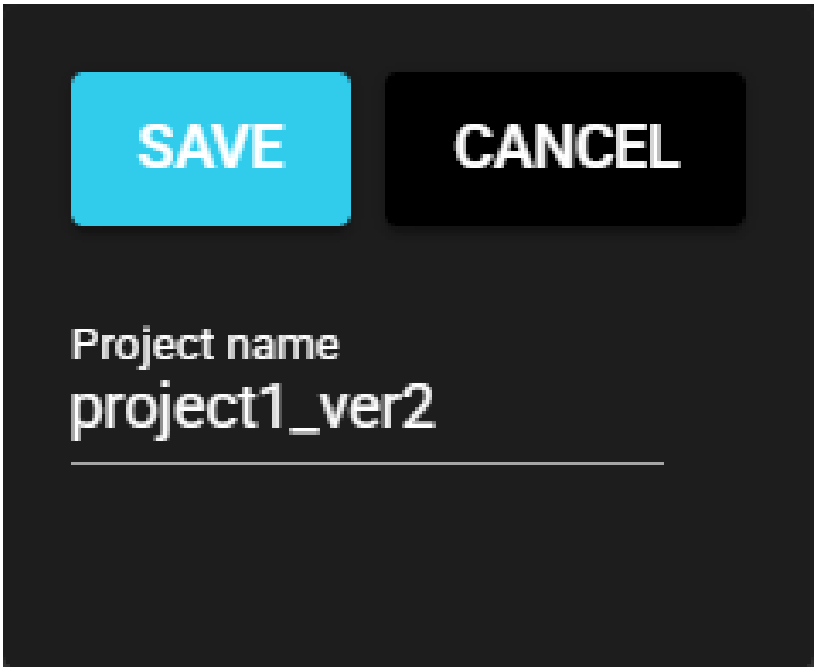
1. Press the Setting Menu icon and select Project Settings.



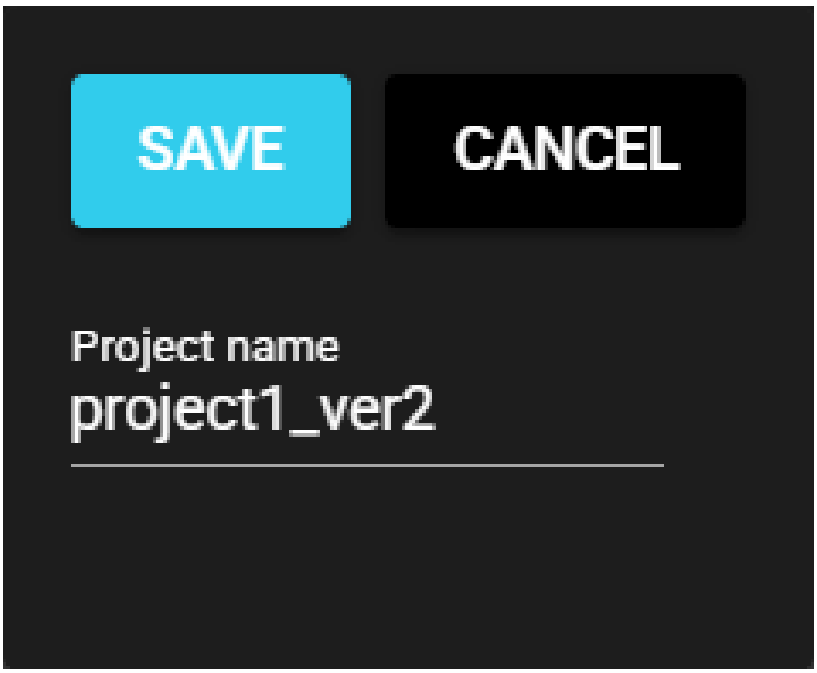
2. Press the Edit Icon (Edit) for the project whose information you want to update.



3. Input a new project name.



4. Press Save.

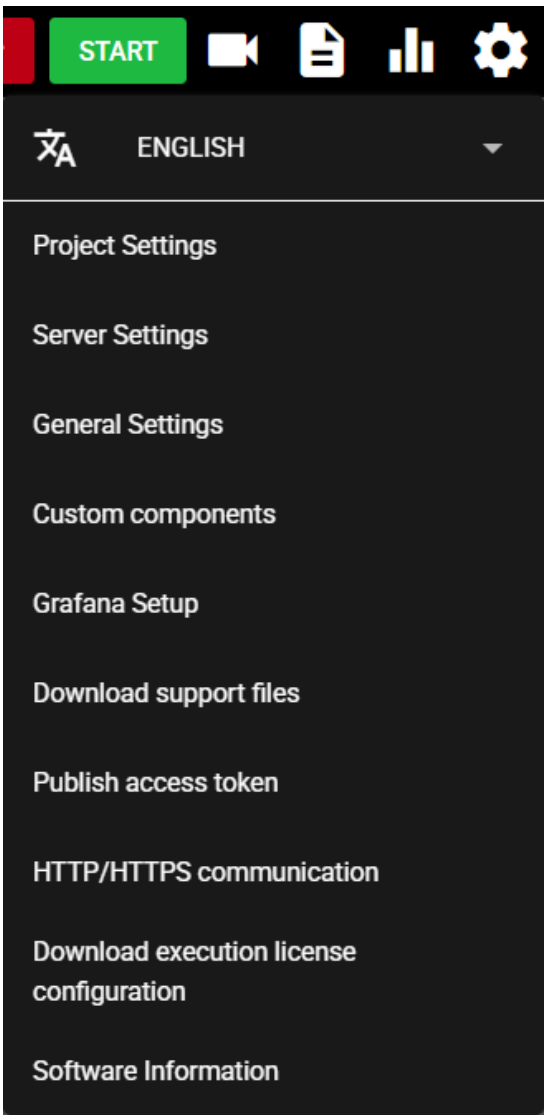


5. The project name will be updated.

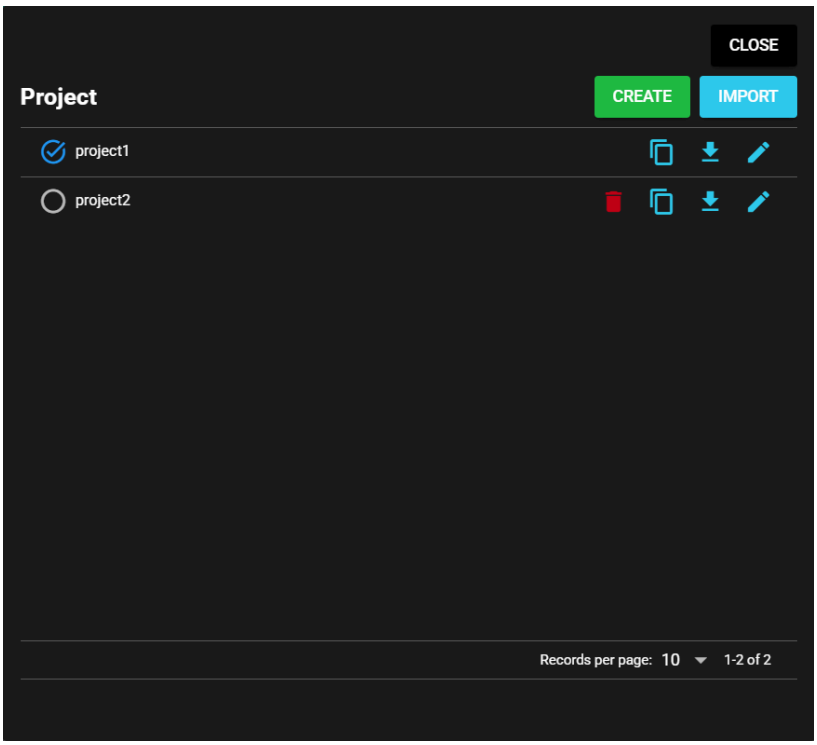
5.17.8 Delete project

You can delete a project by following the steps below.

1. Press the Setting Menu icon and select Project Settings.



2. Press the Trash Icon (Delete) for the project you want to delete.



Note: You cannot delete the currently selected project.

3. Select Yes in the confirmation dialog.
4. The project will be deleted.

6 Advanced Use

6.1 Advanced Use

This chapter explains the advanced use of SpeeDBee Synapse.

The information provided here is useful for building a more advanced system. The following topics are covered:

1. Creating and Using Custom Components
2. Linking with Control Programs (WebAPI)
3. Linking with Grafana
4. Global Variables

6.2 Creating and Using Custom Components

6.2.1 Creating and Using Custom Components

This section explains how to create and use custom components.

By using these custom components, you can add your own functions to SpeeDBee Synapse. The following topics are covered:

1. Overview of Custom Components
2. Creating Custom Components in Python
3. Creating Custom Components in C
4. Creating Custom UI
5. Custom Component Development Environment (SCCDE)

6.2.2 Overview

6.2.2.1 Custom Component

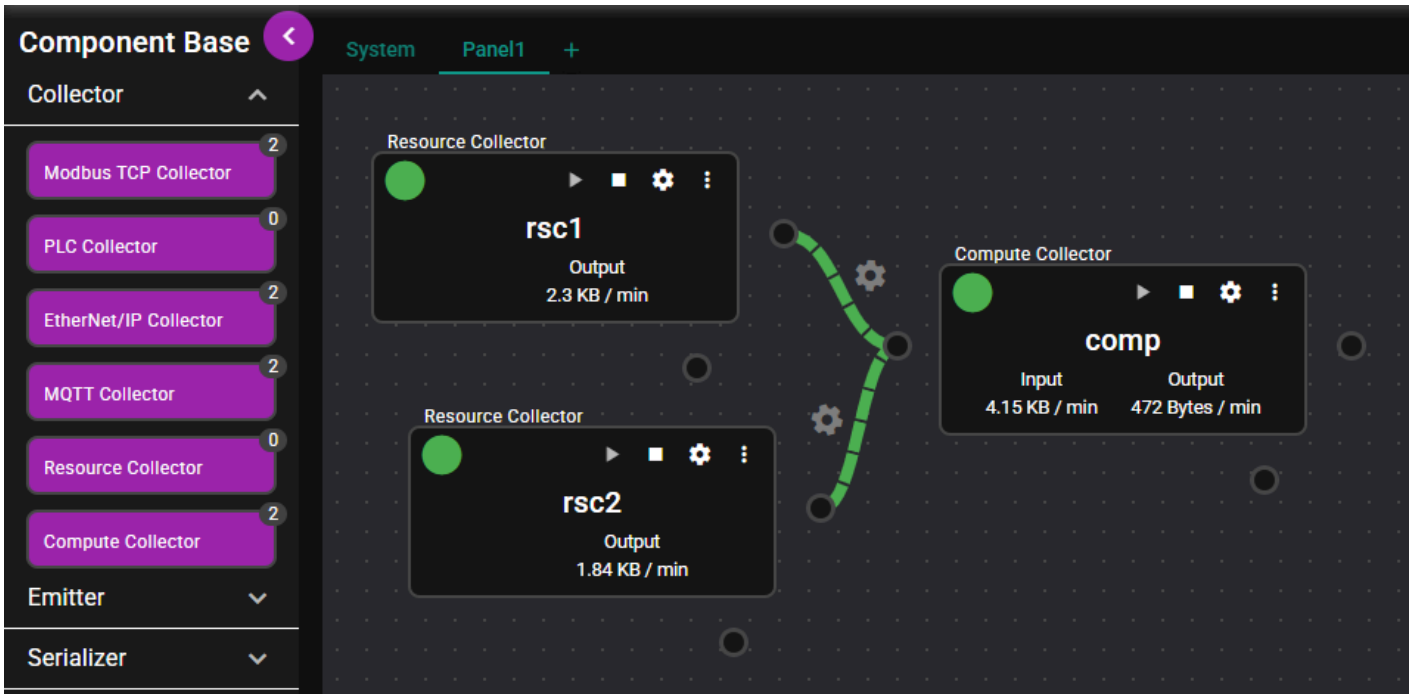
In SpeeDBee Synapse, a module that collects and transmits data is called a component.

The product comes standard equipped with components for collecting data from PLCs and other devices and components for sending data to the cloud. It is also possible for users to develop and operate their own components.

Although you can use C or Python to develop components, this chapter describes the basic concepts that are commonly needed to develop custom components regardless of language.

6.2.2.2 Component Bases and Component Instances

Implementing a custom component requires understanding of the difference between a component base and a component instance while they are both generally referred to as "components."



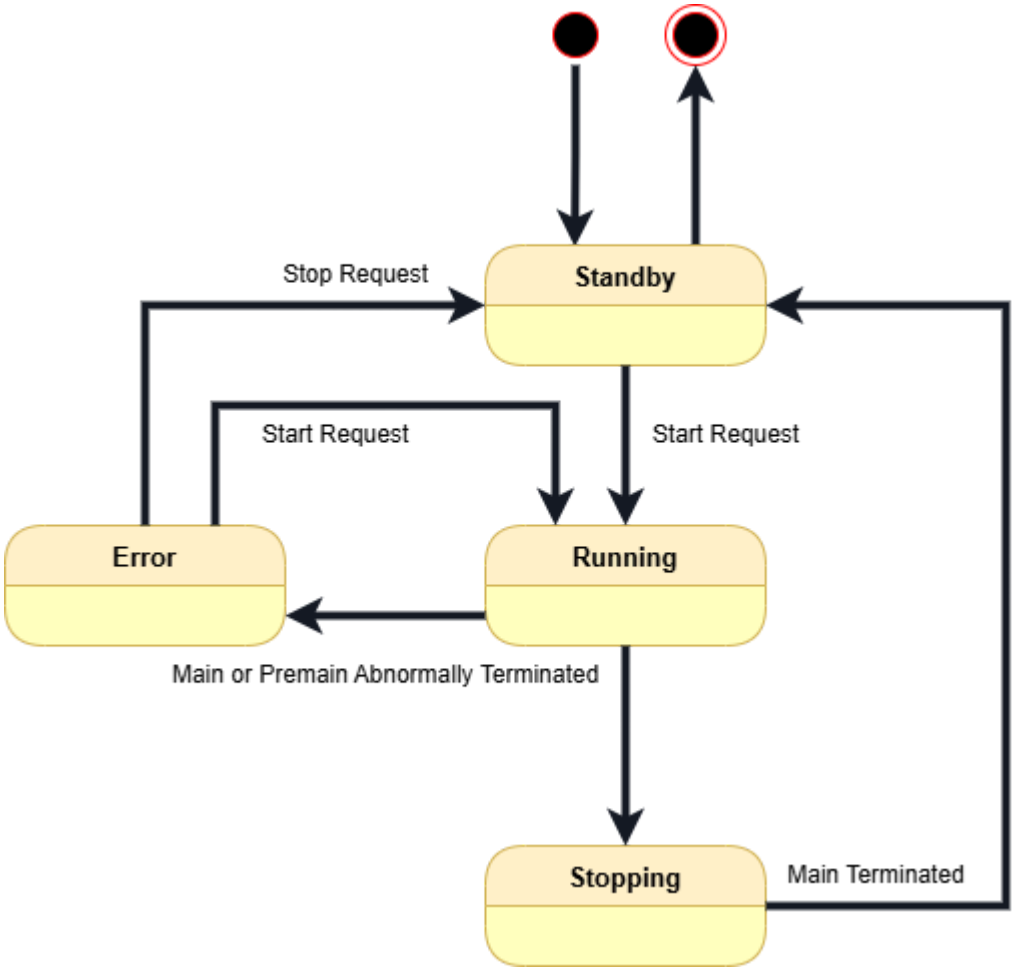
The items aligned in a column on the left side are the component bases. The three elements placed in the center and the right side are the component instances.

A component instance is created by dragging and dropping one of the component bases. Except for special cases, you can create multiple component instances from a single component base.

The same applies when implementing custom components. When you create a custom component, a new component base is added to the left column of the screen. You can create multiple component instances by dragging and dropping the added component base.

6.2.2.3 Lifecycle of Component Instance

All component instances operate along the lifecycle shown below from the time they are created until they are deleted.

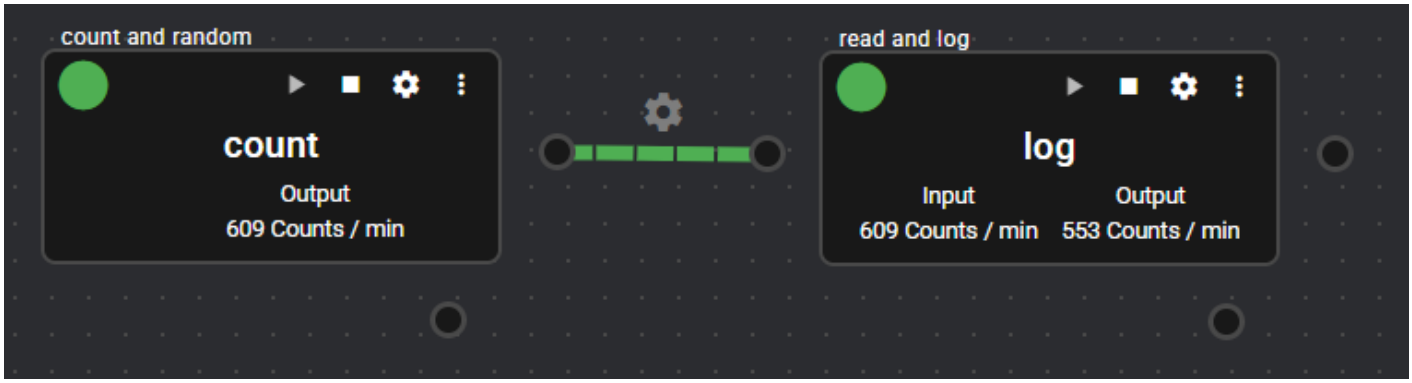


- Immediately after a component instance is created, it is in `Standby` state.
  In this state, the component is not running at all. No data is output and no data is input.
- Clicking the start icon of a component instance transitions it to the `Running` state.
  In this state, the component instance can continuously output data or receive data from other components.
- When you click the stop icon of a component instance, it transitions to the `Stopping` state, and then returns to the `Standby` state once all operations are complete.
- If an error or any other event that prevents continuation of the processing of the component instance occurs during the `Running` state, the status changes to `Error` and processing stops.

When developing custom components, you can program various operations in this lifecycle. The following chapters explain how to implement components by programming language.

6.2.2.4 Input and Output Ports of Components

Each component has output and input ports. An output port is an element that outputs data generated by that component or collected through external access to the SpeeDBee Synapse. An input port is an element that accepts data output by another component and uses it for that component.



When implementing custom components, you must specify the number of ports for each component. For components that output data, set the number of output ports to 1. For components that accept data, set the number of input ports to 1. If a component does not need either or both of the ports, set the number of the relevant port(s) to 0. Future updates may enable components to handle multiple ports, but as of current version, each component is limited to having a maximum of one input port and one output port.

6.2.3 Creating Python custom components

This section explains how to create a custom components using Python.

6.2.3.1 Prerequisite Knowledge

As a prerequisite for developing custom components, you should have a good understanding of the Python language tutorials, as well as with class definitions, method definitions, and class inheritance. For entry-level information, see the official Python documentation below.

Official Python documentation: https://docs.python.org/3/

6.2.3.2 Sample Custom Components

To demonstrate how to implement custom components in Python, simple custom components are provided as samples.

Download sample custom components

Alternatively, if you have installed SpeeDBee Synapse, the sample custom components are already installed in the following locations:

```
$ ls /usr/local/speedbeesynapse/share/sample_component_py/
sin_and_random.py  read_and_log.py
```

6.2.3.2.1 Sample Description

This chapter explains the following sample as an example of custom component.

- `sin_and_random.py`

  This component creates the following two columns for output ports and continues to output data.

Column name	Data type	Description
sin	DOUBLE	Ten 1 Hz SIN wave values are registered every second. The period of SIN wave can be changed by a parameter.
rand	INT32	A random value in the range [0, 100) is registered once every second.

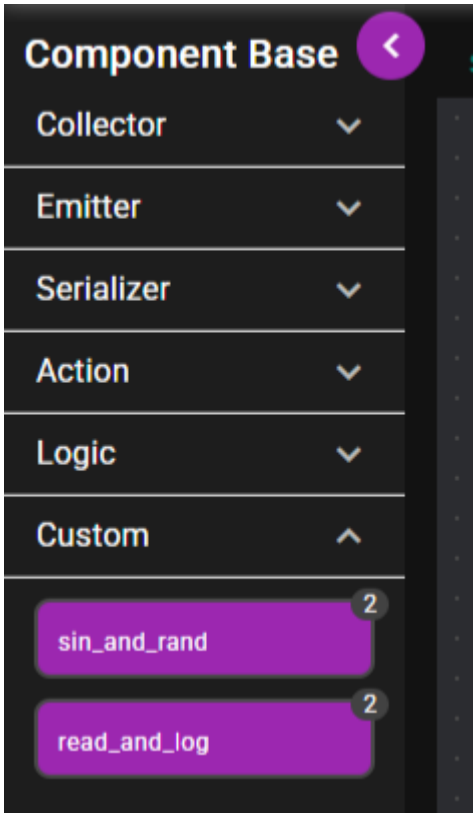
- `read_and_log.py`

This component receives data from the input port and outputs it to string columns and logs.

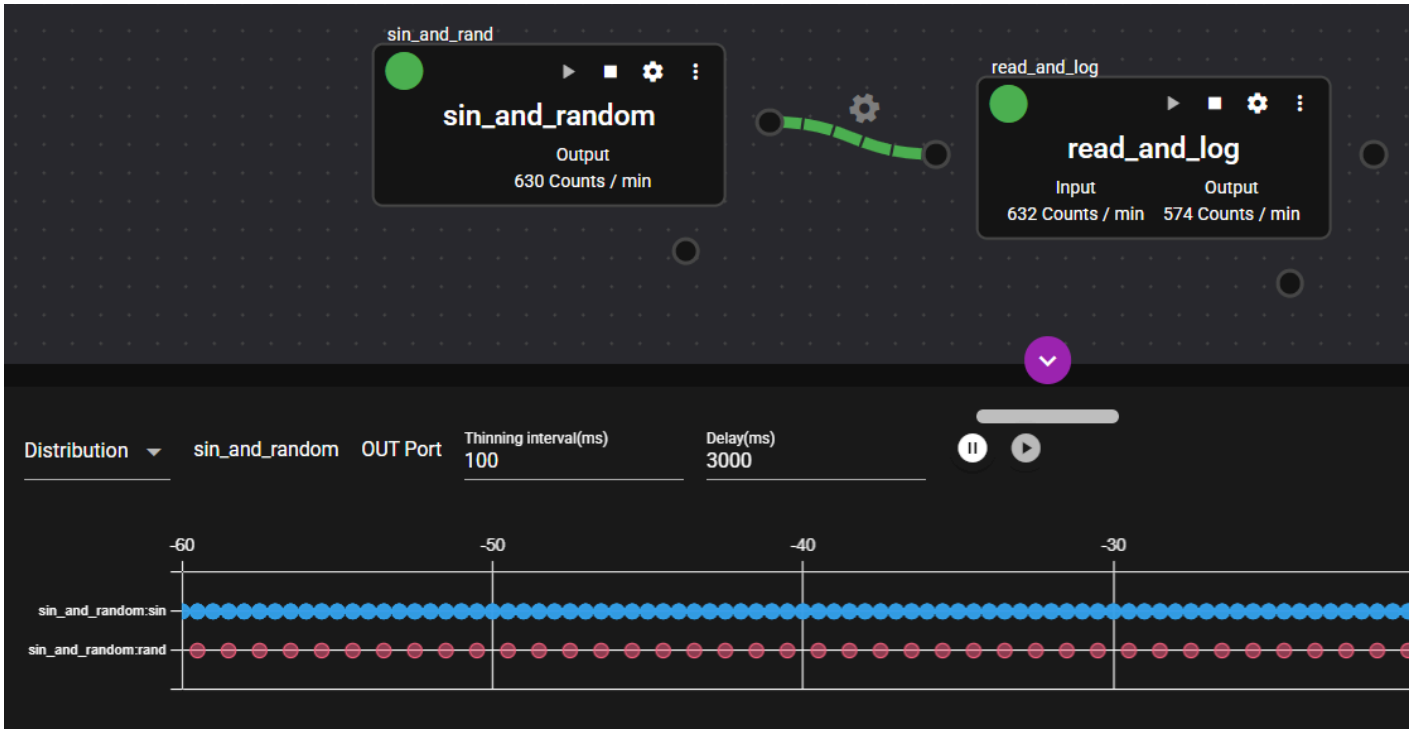
6.2.3.2.2 Execution

A component base can be added by registering the `*.py` file directly from the screen, provided that there are no errors. For instructions on how to register files, see Using Custom Components.

Once registered, you can generate component instances by dragging and dropping just as you would a normal component.



You can concatenate `sin_and_random` and `read_and_log` and execute them to display their output columns. The following monitors the output ports of `sin_and_random`. To see a change in value, change the thinning interval to 100 milliseconds.



6.2.3.3 Custom Component Implementation Details

The following sections describe the implementation of custom components using as examples source codes that use the sample custom components `sin_and_random` and `read_and_log`.

6.2.3.3.1 Custom Component Class Definition

With custom components in Python, you can define the name, identity, and runtime processing of their component base by defining a new class that inherits from the `HiveComponentBase` class as follows:

```
from speedbeesynapse.component.base import HiveComponentBase, HiveComponentInfo, DataType
  :

@HiveComponentInfo(uuid='00000000-0000-0000-0000-000000000000', name='Component Name', inports=0,
outports=1)
class HiveComponent(HiveComponentBase):
    def main(self, param):
        :
```

The class name must be `HiveComponent`. You must also wrap the component's information in the `@HiveComponentInfo` decorator. The information defined here enables you to work with component base information on the screen. Refer to the following explanation to set the members.

Member name	Definition information	Description
`uuid`	UUID	The UUID is used to identify the component. Generate a random ID and set it here.
`name`	Component name	Set the name for the component. This name will be displayed on the screen. Use a name that is easy to understand and that can be distinguished from other components.
`tag`	Tag	This adds a tag to the component. Use "collector," "emitter," "serializer," "action," or "logic" to specify under which component base to place the component. If this member is omitted, the component is placed under Custom.
`outports`	Number of output ports	Set the number of output ports for this component. To pass data to an input port of another component, set this to 1.
`inports`	Number of input ports	Set the number of input ports for this component. To receive data from an output port of another component, set this to 1.

Always generate random IDs for UUIDs. If you use the same UUID as another component, you may not be able to use either component.

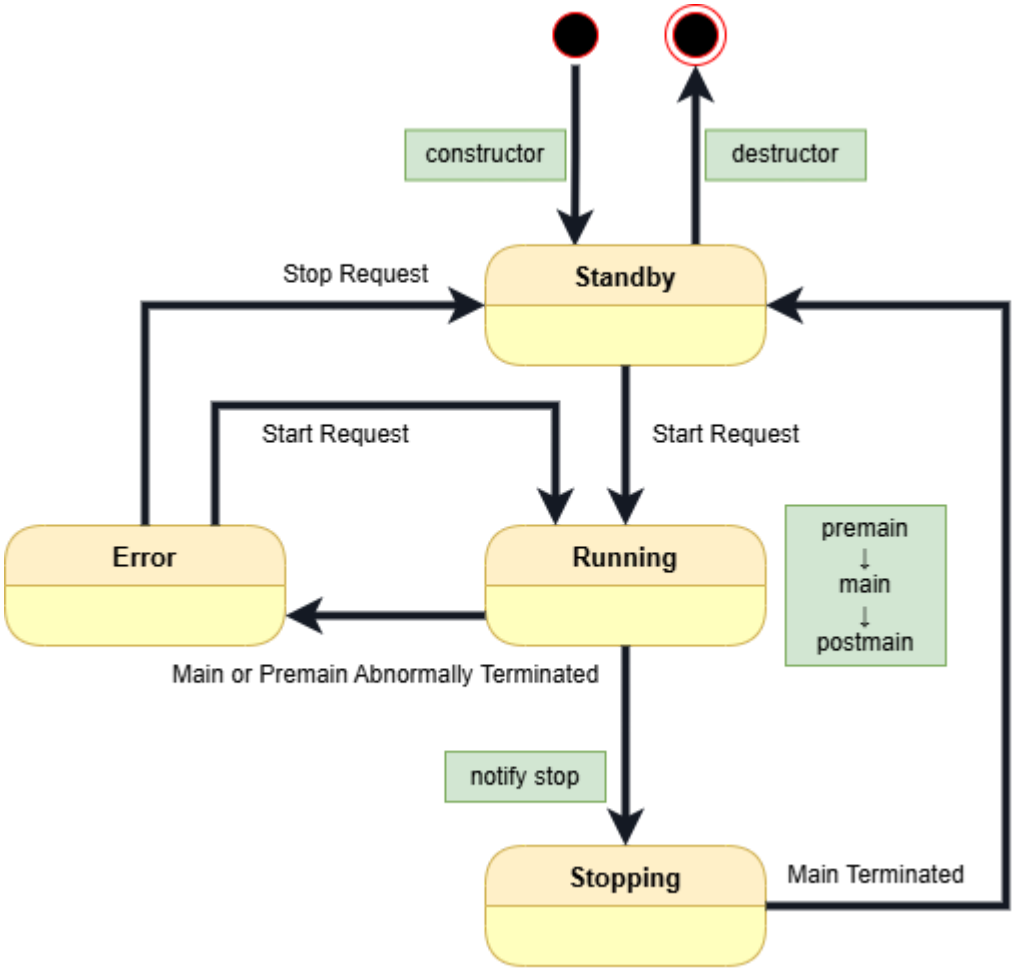
6.2.3.3.2 Lifecycle of `HiveComponent` Class

The `HiveComponent` class defined here is a component base. There is no need to create an instance from this class in the Python script file. Instead, when you create a component instance by dragging and dropping a component base on the screen, an instance of the `HiveComponent` class defined here is automatically created. When a component instance is deleted on the screen, the instance of this class is destroyed.

Therefore, if you define a `__init__()` method or `__del__()` method in the `HiveComponent` class, the method will be executed when the component instance is generated or deleted on the screen.

```
@HiveComponentInfo(uuid='00000000-0000-0000-0000-000000000000', name='Sample Component',
inports=0, outports=0)
class HiveComponent(HiveComponentBase):
    def __init__(self):
        # called at creating component instance
        pass
    def __del__(self):
        # called at deleting component instance
        pass
    def premain(self, param):
        # called before main()
        pass
    def postmain(self, param):
        # called after main() finished
        pass
    def main(self, param):
        # called at starting component instance
        while self.is_runnable():
          pass
    def notify_stop(self):
        # called before the component received stop request from system
        pass
```

There are other methods that are called by manipulating components from the screen. Refer to the figure and table below for such methods and when they are called.



Callback function	Description
`__init__`	This function is called when the component instance is generated.
`__del__`	This function is called when the component instance is destroyed.
`premain`	This function is called before the "main" function when starting the component.
`main`	This function implements the main processing of the component. This function must continue to loop without exiting until the component is requested to terminate.
`postmain`	This function is called when a component exits, just after the "main" function ends.
`notify_stop`	This function is executed when a component is requested to stop during the execution of the "main" function. Note that this method is called during the execution of `main()`, so it runs on a separate thread.

The `main` function must always be defined as it is the body of the component's processing. The following section explains how to implement the `main` function in the sample program.

6.2.3.3.3 Creating Output Port Columns and Registering Data (`sin_and_random`)

The following is the main processing of the custom component sample `sin_and_random`. The procedure for outputting data is explained based on this operation.

```
def main(self, param):
     :
    count = 0

    # Create columns.
    self.sin = self.out_port1.Column('sin', DataType.DOUBLE)
    self.rand = self.out_port1.Column('rand', DataType.INT32)

    # Repeat at 100msec intervals.
    for [ts, skip] in self.interval_iteration(100000000):

        # Compute sin wave value and insert it into 'sin' column.
        val = math.sin(2*math.pi*sin_hz*count/10)
        self.sin.insert(val, ts)

        # Insert random value into 'rand' column only once in 10.
        if count % 10 == 0:
            self.rand.insert(random.randrange(0, 100), ts)

        count += 1
```

1. Creating a column

   Start by creating a column.

   ```
   self.sin = self.out_port1.Column('sin', DataType.DOUBLE)
   self.rand = self.out_port1.Column('rand', DataType.INT32)
   ```

   This component uses the double-precision floating point column `sin` and the 32-bit integer column `rand`. The `Column()` method in `self.out_port1` is used to specify the column name, data type, etc. and create the column object. Store the return value in a variable, because you will use it later to populate the column.

   All data output by the component must be registered in this generated column.

   > **About Column Names That Cannot Be Used**
   >
   > The following characters cannot be used in column names
   >
   > - `\` - backslash
   > - `/` - slash
   > - `*` - asterisk
   > - `?` - question mark
   > - `"` - double quote
   > - `< >` - angle brackets
   > - `|` - pipe
   > - `'` - single quote
   > - ` ` - space

2. Repeating periodic operation

   To periodically register data, the for statement is used to implement repeated operation.

   ```
   for [ts, skip] in self.interval_iteration(100000000):
       :
   ```

   The loop condition calls `self.interval_iteration()`, which is a generator that returns repeated values at the time interval specified in the argument. The code inside the for statement is then executed at regular intervals.

   The time interval specified in the argument is in nanosecond units, so in this example the loop runs once every 0.1 seconds (100 million nanoseconds).

   This for statement loop does not end unless you stop the component from the screen.

3. Registering data in the column

   Data is registered in the column in an iterative process.

```
val = math.sin(2*math.pi*sin_hz*count/10)
self.sin.insert(val, ts)

if count % 10 == 0:
    self.rand.insert(random.randrange(0, 100), ts)
```

Data can be registered by calling the `insert()` method of a previously created column object. Specify the value to be registered in the first argument and the time at which the value is to be registered in the second argument. If the second argument is omitted, the current time is used as default.

> **Maximum size that can be registered with insert**
>
> The maximum size of data that can be registered with insert() is 512 bytes (STRING is 511 bytes, not including null). Registering a value that exceeds the maximum size will throw an exception.

6.2.3.3.4 Reading Data from Input Port (`read_and_log`)

The following is the main processing of the custom component sample `read_and_log`. Using this sample, the procedure for receiving data output by another component is explained below.

```
def main(self, param):
    # Create a column for logging received data.
    self.logclm = self.out_port1.Column('log', DataType.STRING)

    # Logging column information received from 'in_port1'.
    columns = self.in_port1.get_columns()
    for column in columns:
        self.log.info(column)

    # Start receving column values from 'in_port1'.
    with self.in_port1.ContinuousReader(start=self.get_timestamp()) as reader:
        # Loop until the component received 'stop' request.
        while self.is_runnable():
            # Get a group of records of column data (called as window_data).
            window_data = reader.read()
            if not window_data:
                # window_data may be None.
                self.log.info("no data yet")
                continue

            # Process all records in a window_data one by one.
            for record in window_data.records:
                # Convert timestamp as a string.
                ts = datetime.datetime.fromtimestamp(record.timestamp / 1000000000)

                # Process all column's value in a record one by one.
                for columnvalue in record.data:
                    if columnvalue.value:
                        # Make string from columnvalue, logging it and insert it into the column
                        logmsg = f"{columnvalue.column}[{str(ts)}]: {columnvalue.value}"
                        self.log.info(logmsg)
                        self.logclm.insert(logmsg)
                    else:
                        pass # ignore None
```

The `log` column created at the beginning is intended to output information as a string. It is not required in cases where data is merely received from another component, so it is omitted.

1. Getting input port column information

   The `get_columns()` method in `in_port1` allows you to retrieve column information for data being output by another component connected to the component's input port.

   ```
   columns = self.in_port1.get_columns()
   for column in columns:
       self.log.info(column)
   ```

   The above example outputs logs of the retrieved columns using an iteration loop.

   Changing the connection to a component or stopping a connected component changes the column information that can be retrieved with `get_columns()`. If you constantly need the latest information, retrieve it periodically and check the information.

2. Starting the `ContinuousReader` loop

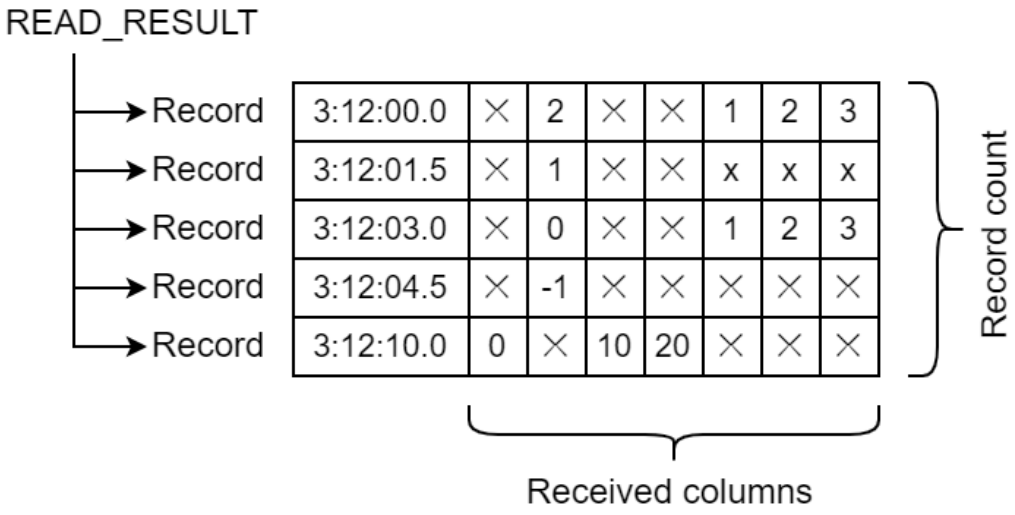
If you want to continuously retrieve data from the input port, use a `ContinuousReader` object. You can use it in the with clause as shown below and then call `read()` repeatedly to retrieve the data arriving at the input port.

```
with self.in_port1.ContinuousReader(start=self.get_timestamp()) as reader:
    while self.is_runnable():
        window_data = reader.read()
        if not window_data:
            self.log.info("no data yet")
            continue
        :
```

The return value of `read()` can be None, in which case it should be skipped.

3. Reading the records of retrieved data

The `read()` method of `ContinuousReader` returns a `HiveWindowData` object. This object is a collection of data coming into the input port for a certain period of time. As shown in the figure below, it consists of multiple records that contain data from the same time. One record contains data from multiple columns.



This enables access to all the data with a double for loop statement:

```
for record in window_data.records:
    ts = datetime.datetime.fromtimestamp(record.timestamp / 1000000000)

    for columnvalue in record.data:
        if columnvalue.value:
            logmsg = f"{columnvalue.column}[{str(ts)}]: {columnvalue.value}"
            self.log.info(logmsg)
            self.logclm.insert(logmsg)
        else:
            pass # ignore None
```

See the `HiveRecordData` Objects for details of data.

6.2.3.3.5 Log Output

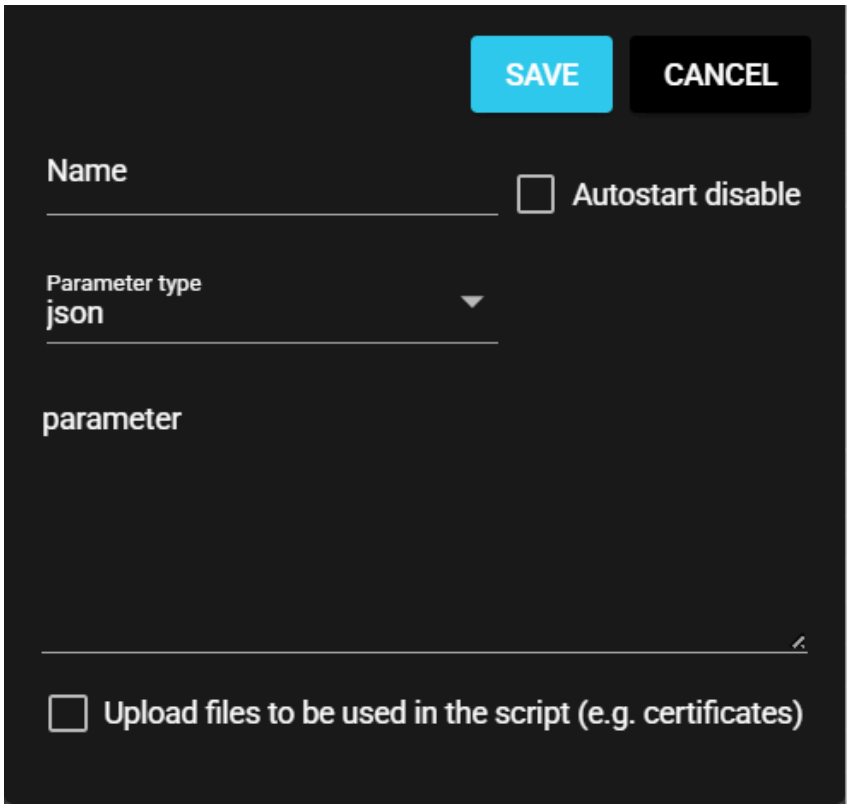
Custom components can log at any time using `self.log` methods as shown below. See `HiveLog` for details.

```
self.log.info('create columns')
self.log.info(f'insert value={val} into {column_name}')
self.log.error('invalid data received')
```

The output log can be downloaded from the screen. See Various Logs.

6.2.3.3.6 Component Parameters

When you generate a component instance from the screen, you can set the component's runtime parameters as its settings. By changing the parameter settings for each instance, you can vary the behavior of a component even when the component base implementation remains the same.



When specifying the parameter type on the screen, STRING or JSON can be selected as parameter type. Choose a type that is easy to handle during implementation.

Parameter type	Description
STRING	You can set any character string. It can be referenced as `str` type in the second argument of each method of the component.
JSON	You can set a JSON string. It can be referenced as `dict` type in the second argument of each method of the component.

The parameters set for the component instance can be acquired as the second argument of the `premain()`, `main()`, or `postmain()` method of the `HiveComponent` class you defined.

```
@HiveComponentInfo(uuid='00000000-0000-0000-0000-000000000000', name='Sample Component',
inports=0, outports=0)
class HiveComponent(HiveComponentBase):
    def premain(self, param):
        // use 'param' here.
        pass
    def postmain(self, param):
        // use 'param' here.
        pass
    def main(self, param):
        // use 'param' here.
        while ...
```

How these methods are used depends on the implementer of the custom component. For example, `sin_and_random` uses a numeric string as a parameter for the waveform frequency of the `sin` column.

6.2.3.4 API for Custom Components

Many API functions are provided in addition to the functions introduced in this section. For more information, see the Custom Component Python-API Reference in the appendix.

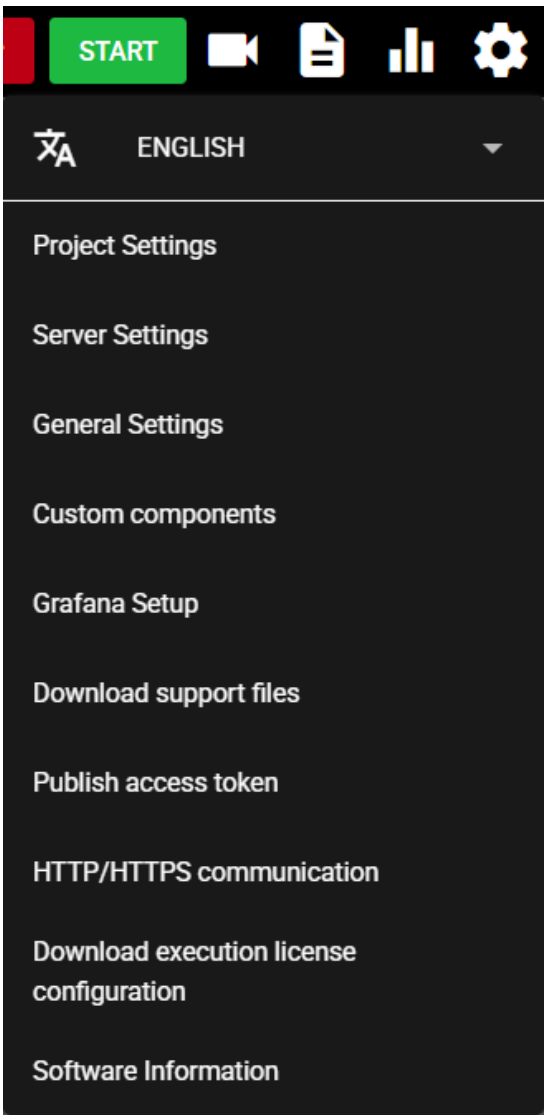
6.2.3.5 Using Custom Components

This section explains how to use custom components created in SpeeDBee Synapse.

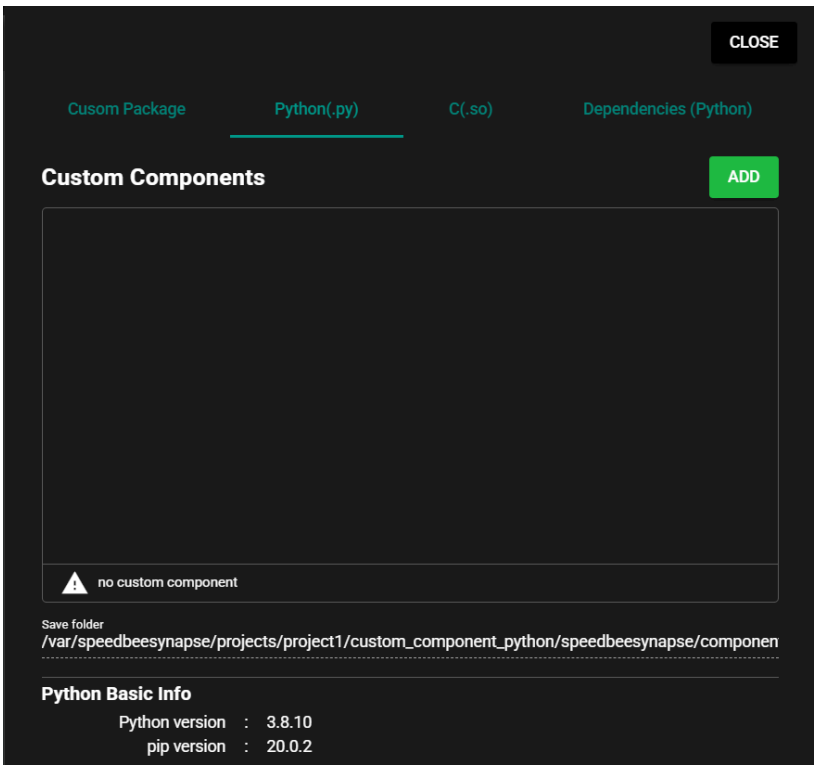
6.2.3.5.1 Add

You can register a created custom component in SpeeDBee Synapse using the following procedure.

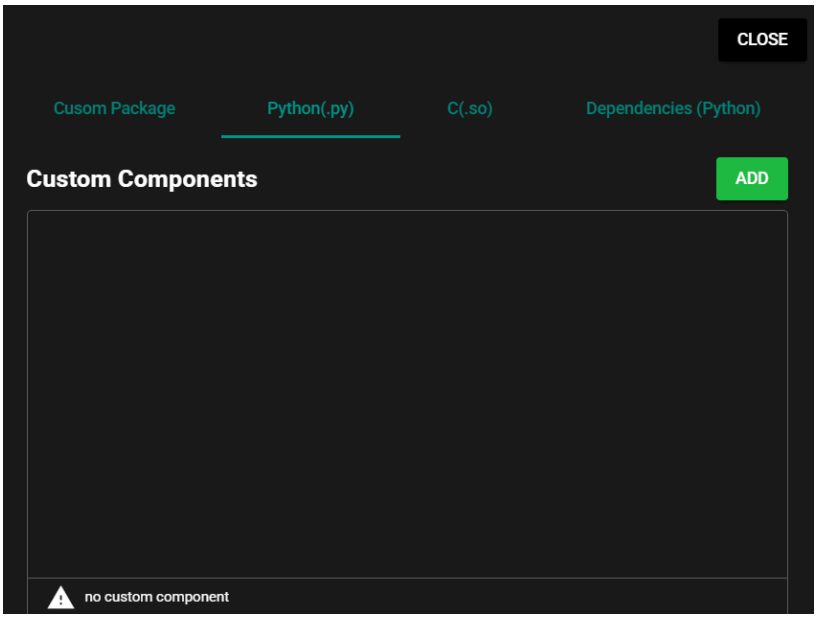
1. Press the Setting Menu icon and choose [Custom Components].



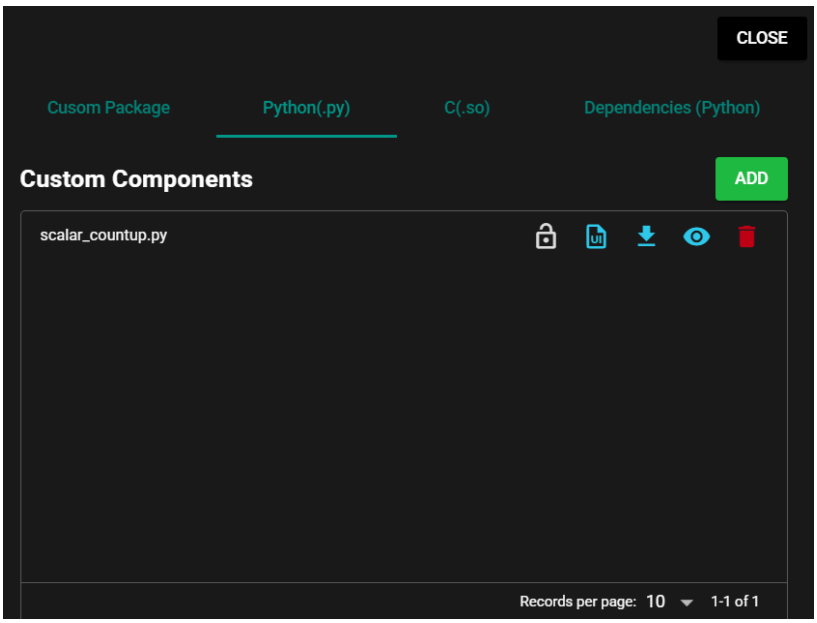
2. Select [Python(.py)] in the tab at the top of the dialog.



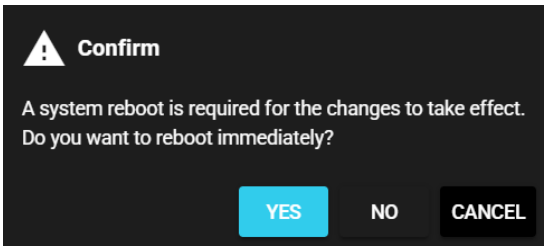
3. Press [Add] to register the file for the custom component.



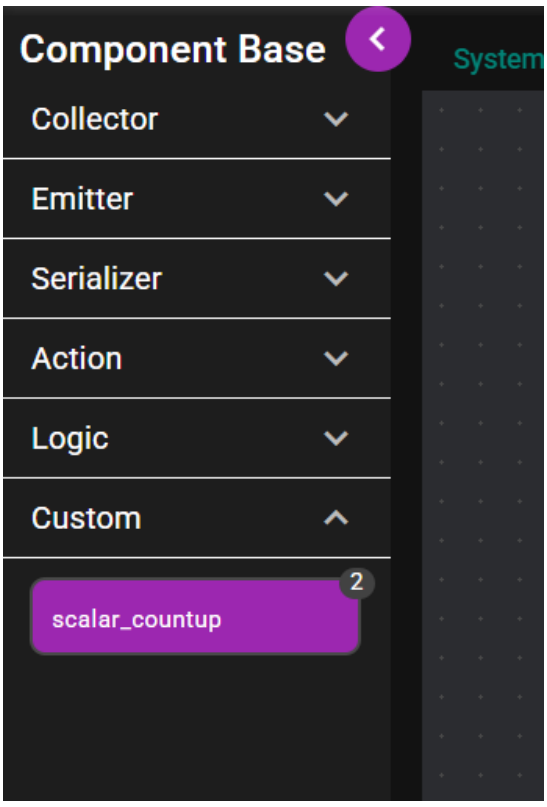
4. Press [Close].



5. Select [Yes] in the confirmation dialog to restart the system to apply the changes.



6. The left menu shows the registered custom component.



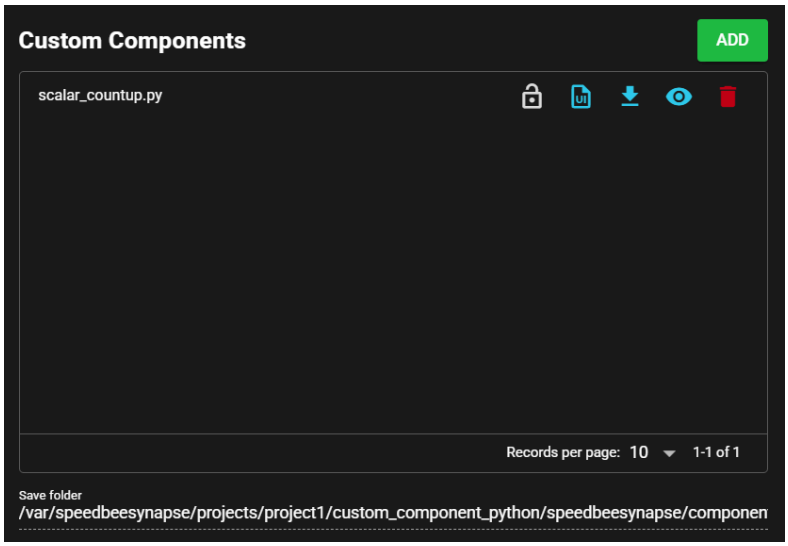
File Save Location

Registered custom component files are saved in the following directory:

Linux

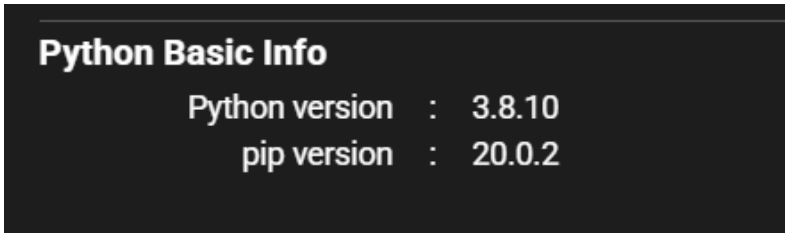
/var/speedbeesynapse/custom_component_python/speedbeesynapse/component/custom

The save directory is displayed in the [Save Location] screen item.



Python version

The version of Python that runs the custom component is displayed in [Python version] under [Python Basic Info].



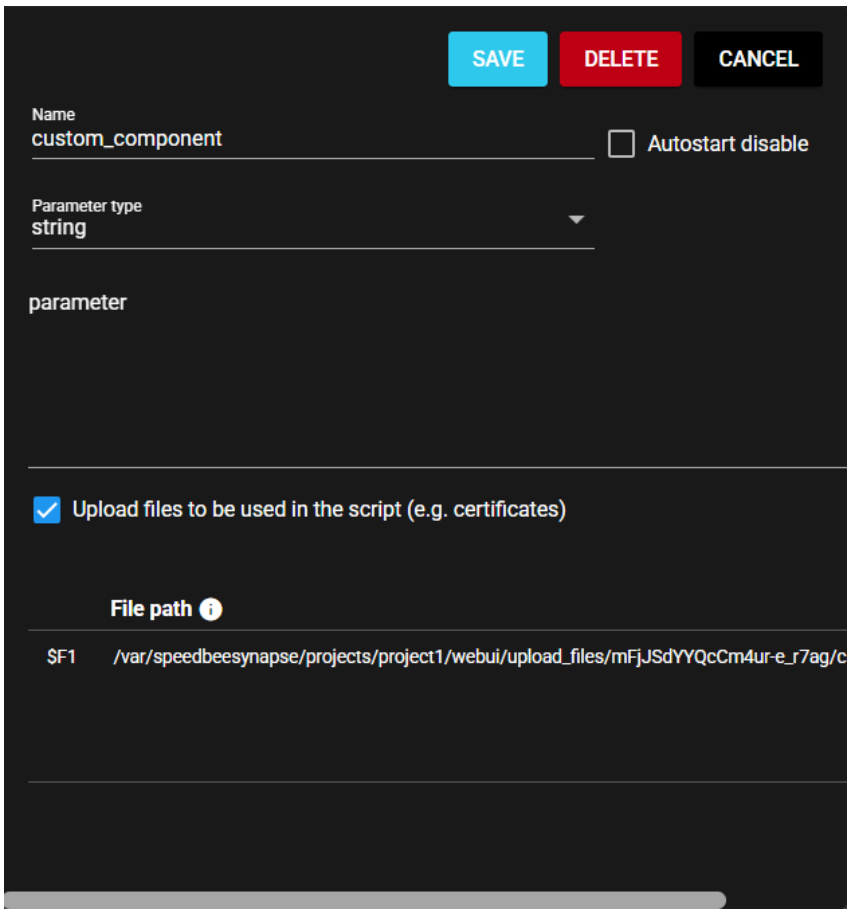
Python file name

Custom components are treated as modules in SpeeDBee Synapse, and the file name is equivalent to the module name. Therefore, "-" in a file name prevents it from being handled correctly.

In line with Python's naming conventions, use short file names with all lowercase, using _ as separators.

6.2.3.5.2 Settings Screen

The items on the custom component settings screen are as follows.



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Parameter type	Select a parameter type from the following: • STRING: String • JSON: JSON string
Parameter	Enter component parameters.
Upload files to be used in the script (e.g. certificates)	Set to ON to upload a file, such as a certificate, for use in custom component processing.
Add file	Press this button to upload the file to be used in the script. Note: Newly added files are uploaded when you save the component settings.
File path	Save location for uploaded files Note: When using a file path in a parameter, you can use a variable. For more

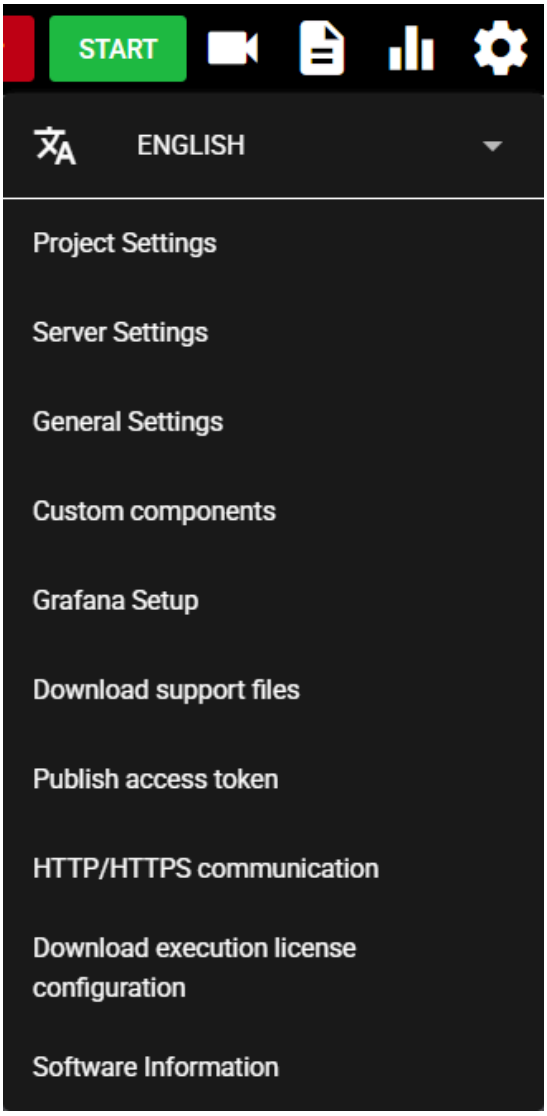
Item	Description
	information, click the icon to the right of [File Path] in the title.

You can change the parameter input area of the settings screen to display the screen items defined by the user. For more information, see Creating Custom UI.

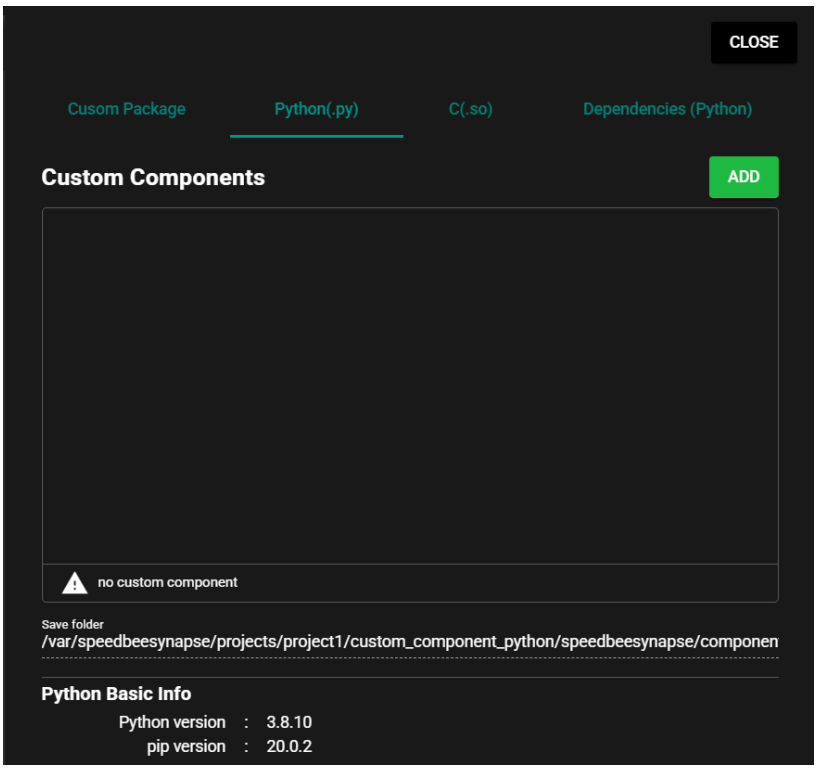
6.2.3.5.3 Delete

You can delete a registered custom component from SpeeDBee Synapse using the following procedure.

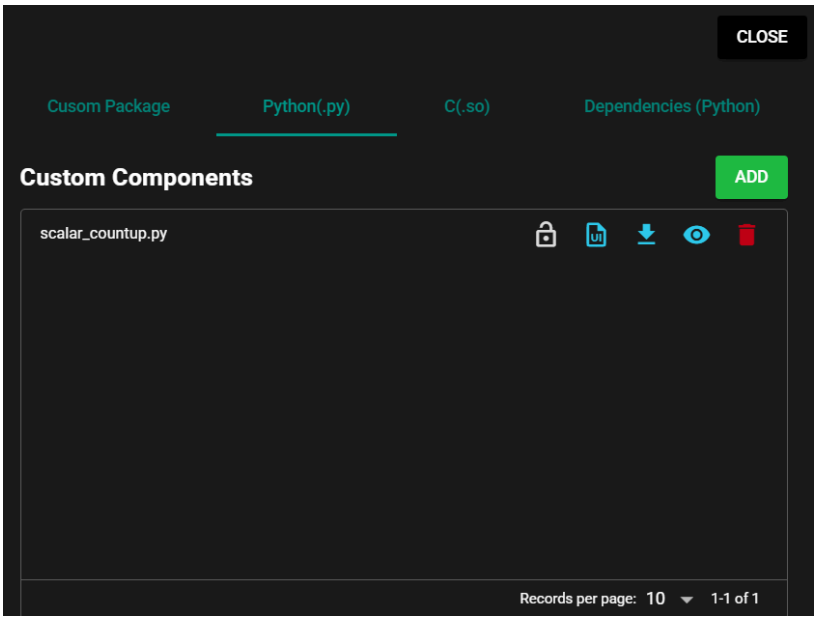
1. Press the Setting Menu icon and choose [Custom Components].



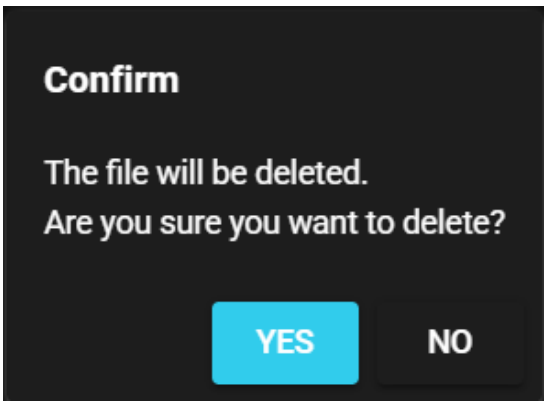
2. Select [Python(.py)] in the tab at the top of the dialog.



3. Press the trash can icon (Delete) for the custom component.



4. Select [Yes] in the confirmation dialog.

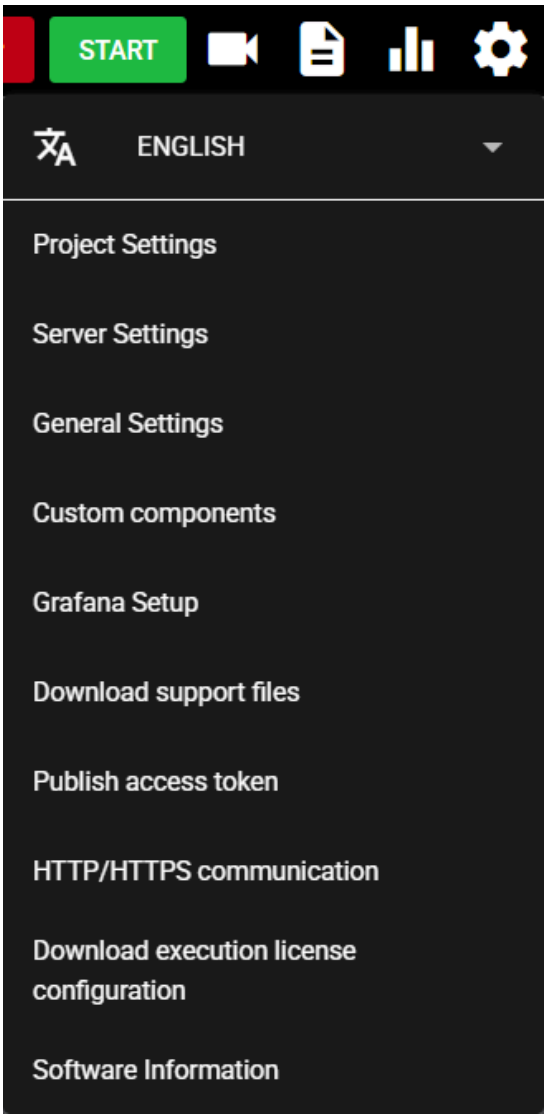


5. The custom component is deleted.

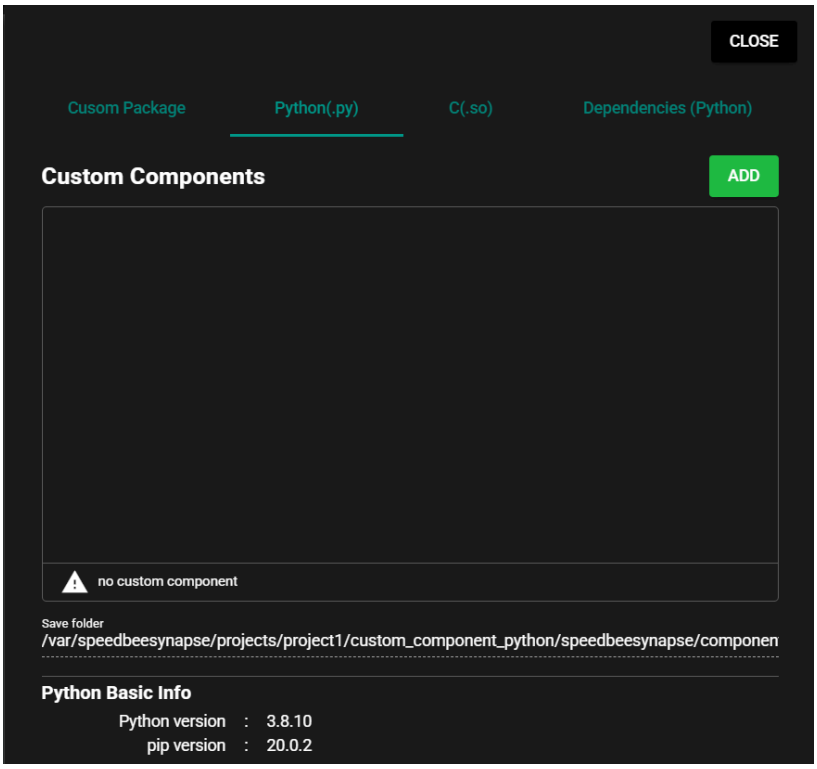
6.2.3.5.4 Display

You can display the information of a registered custom component using the following procedure.

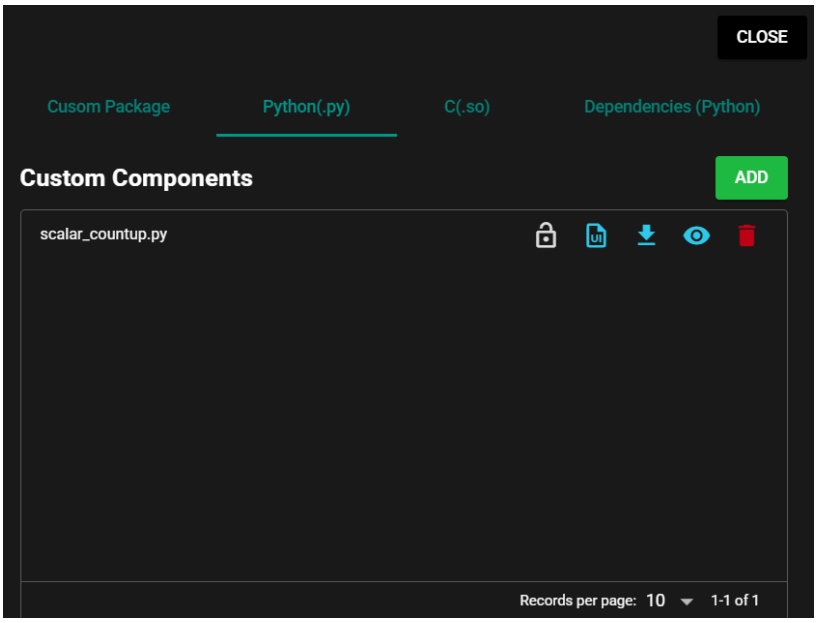
1. Press the Setting Menu icon and choose [Custom Components].



2. Select [Python(.py)] in the tab at the top of the dialog.



3. Press the visibility icon (Display) for the custom component.



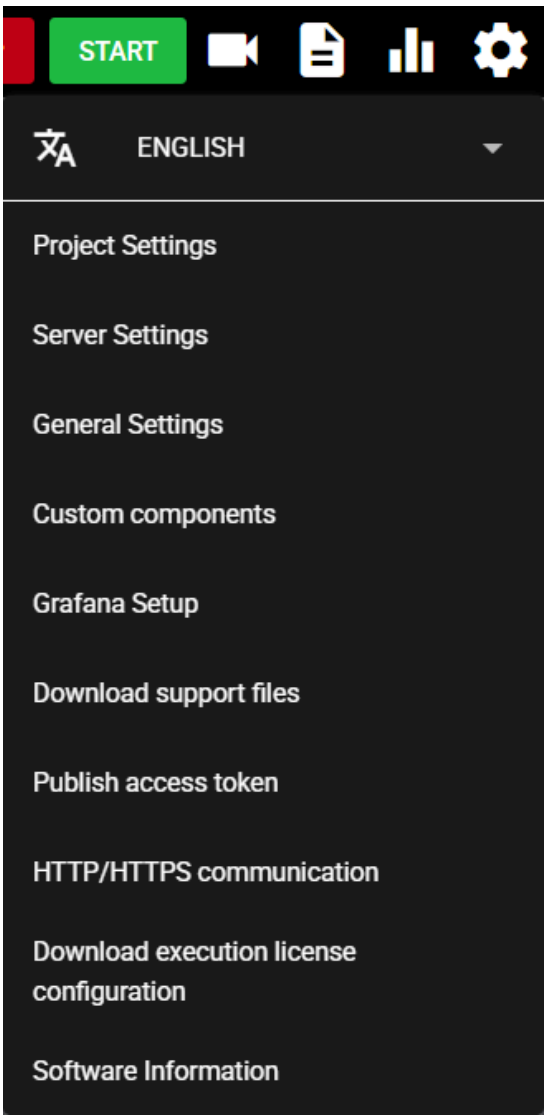
4. The source code of the custom component is displayed on the screen.



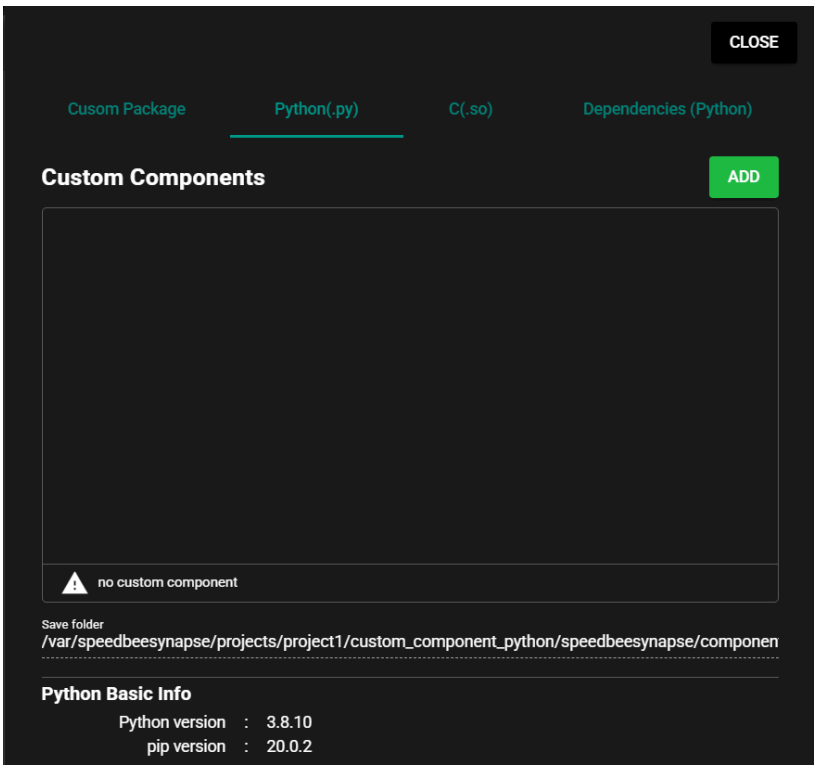
6.2.3.5.5 Download

You can download a registered custom component as follows:

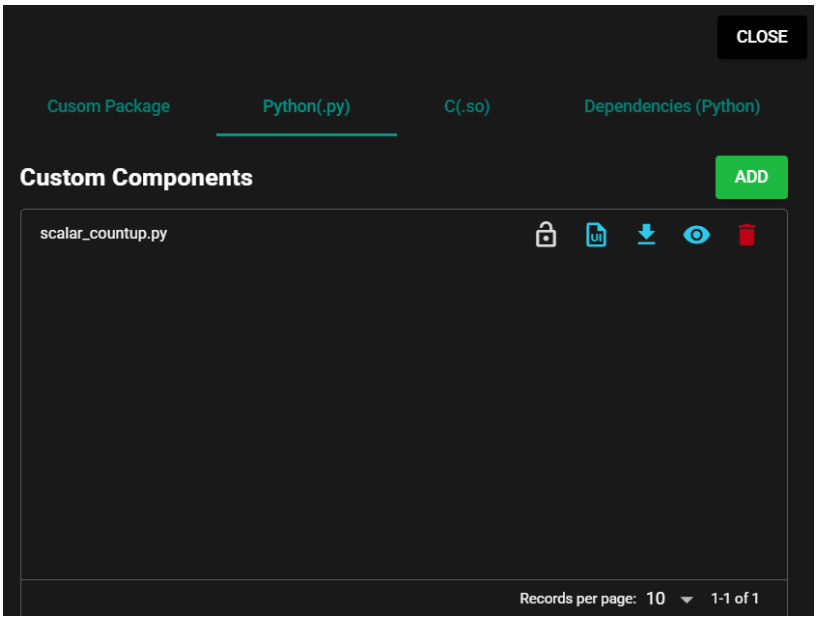
1. Press the Setting Menu icon and choose [Custom Components].



2. Select [Python(.py)] in the tab at the top of the dialog.



3. Press the download icon (Get) for the custom component.



4. The custom component file is downloaded.

6.2.3.5.6 Protect

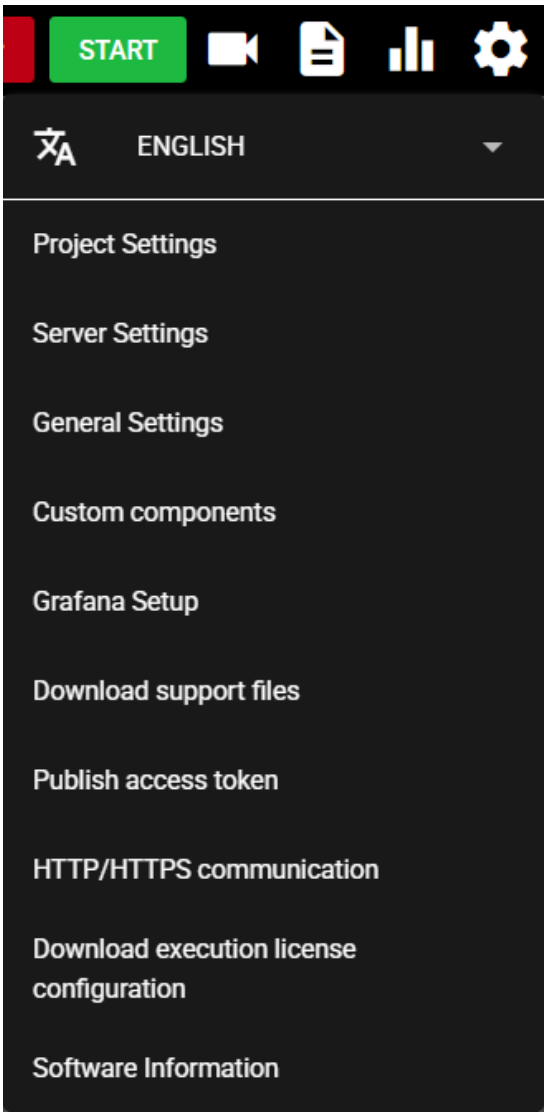
Registered components can be protected.
Protected components are not subject to the following operations and therefore are protected from unintended updates:

- Register

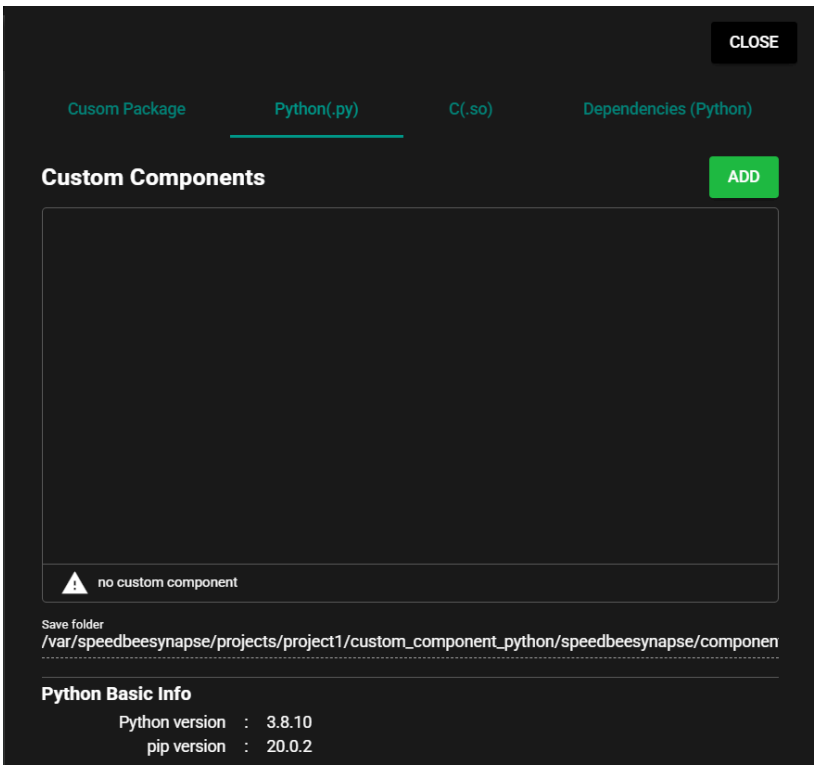
- Delete
- Display
- Download
- Set Custom UI

You can protect a custom component using the following procedure.

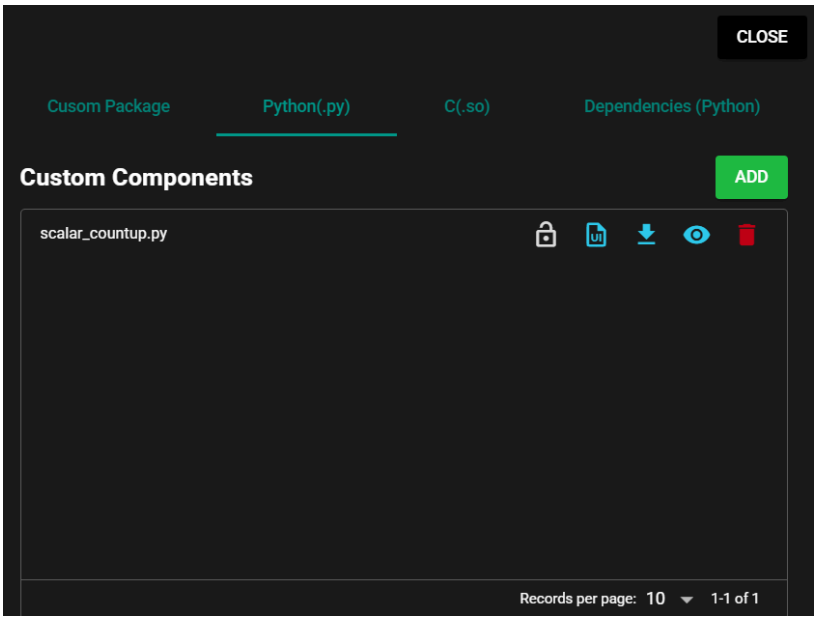
1. Press the Setting Menu icon and choose [Custom Components].



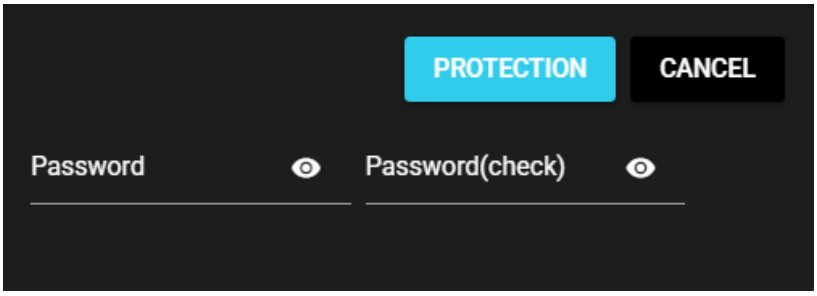
2. Select [Python(.py)] in the tab at the top of the dialog.



3. Press the key icon (Protect) for the custom component.

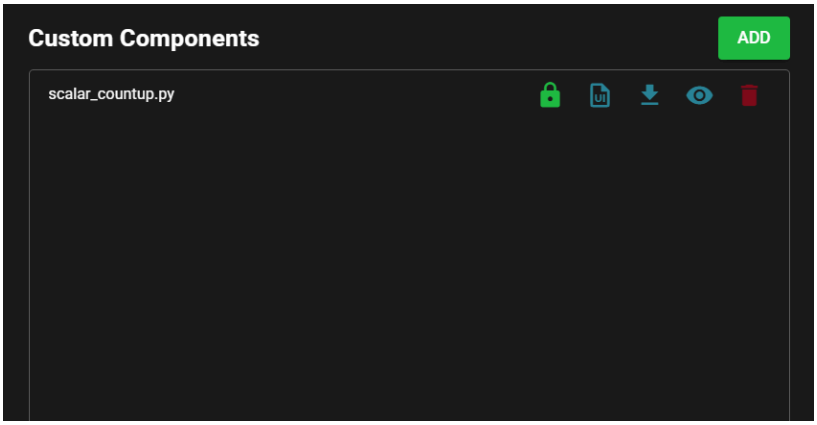


4. Enter the password required to remove protection and press [Protection].



> **Warning**
>
> - The password is stored in a undecodable format and cannot be reset until protection is removed.
> - If you forget the password for protection, the protected custom component becomes inoperable.
>
> Be careful when handling passwords.

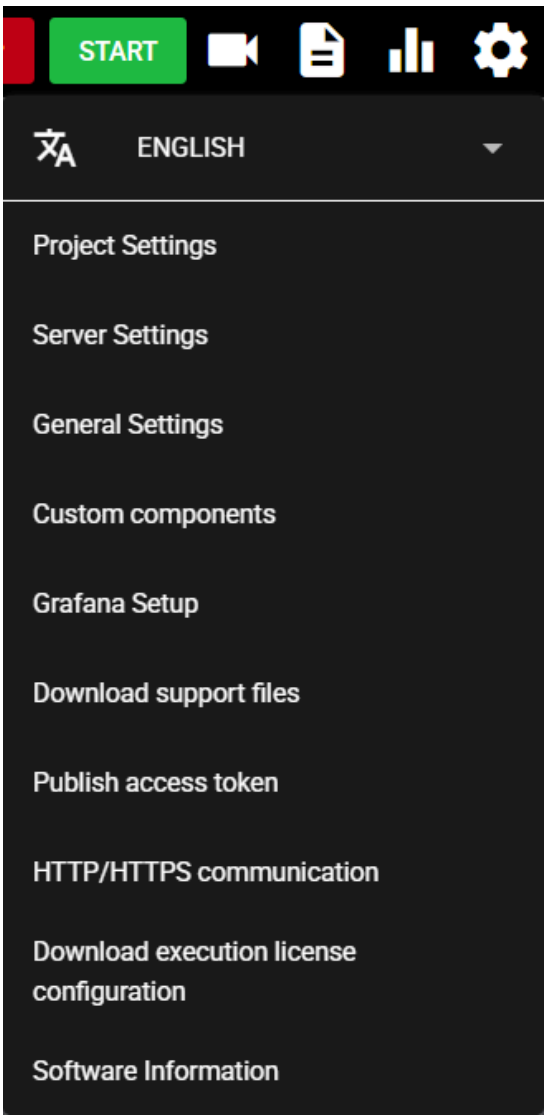
5. The custom component is now protected.



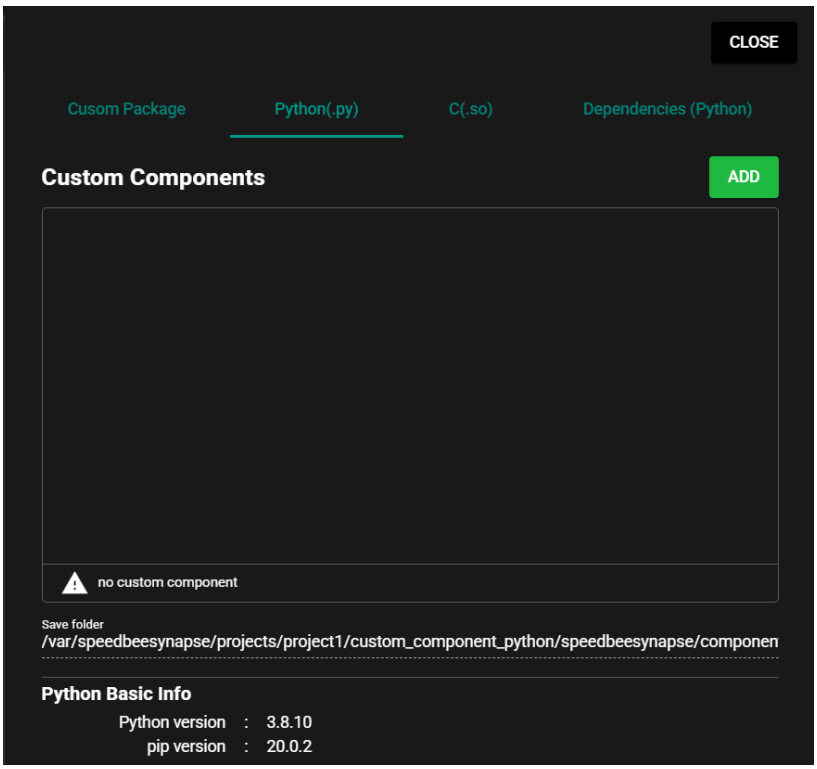
6.2.3.5.7 Unprotect

Follow the procedure below to remove the protection from a protected custom component:

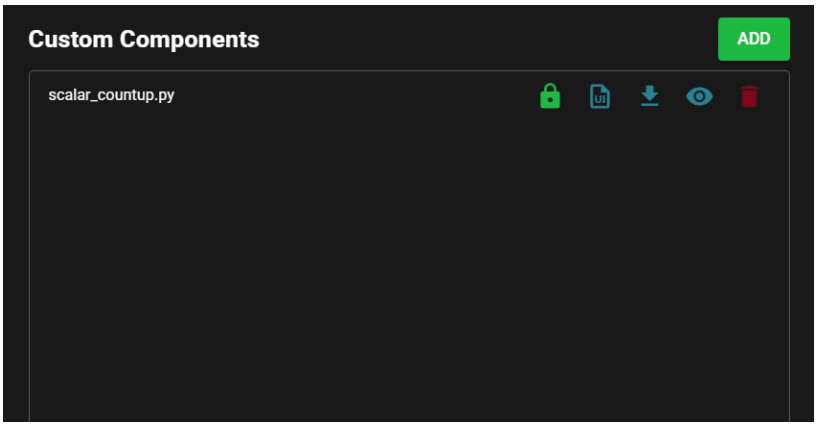
1. Press the Setting Menu icon and choose [Custom Components].



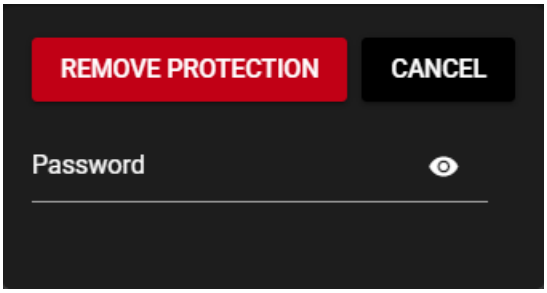
2. Select [Python(.py)] in the tab at the top of the dialog.



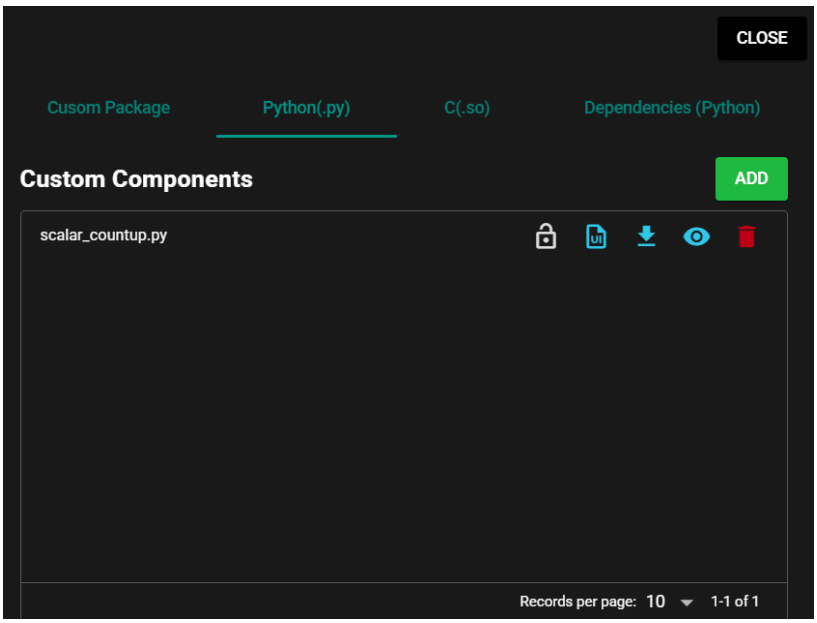
3. Press the key icon (Remove protection) for the custom component.



4. Enter the password set when applying protection and press [Remove protection].



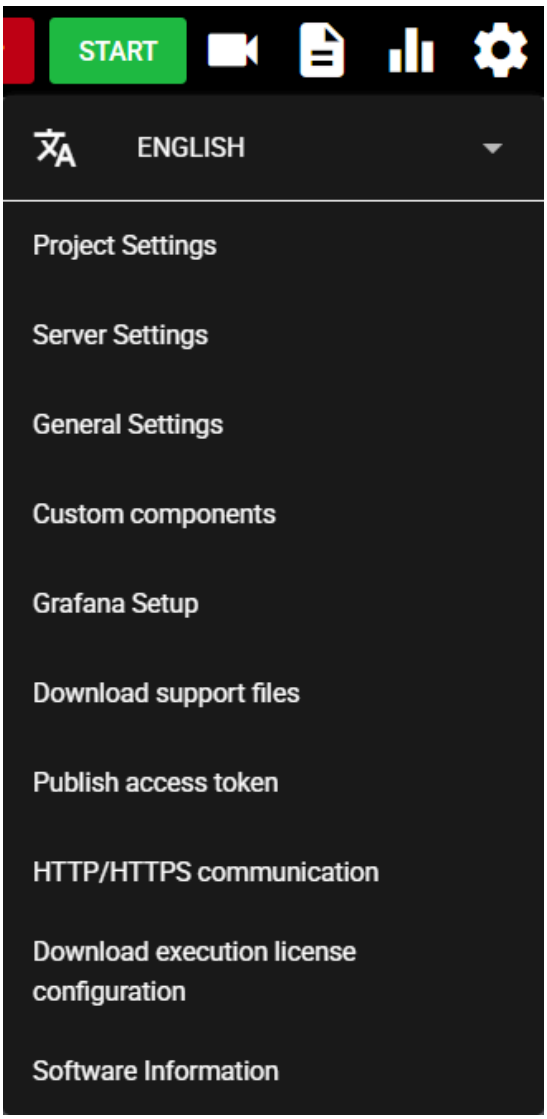
5. The protection is removed from the custom component.



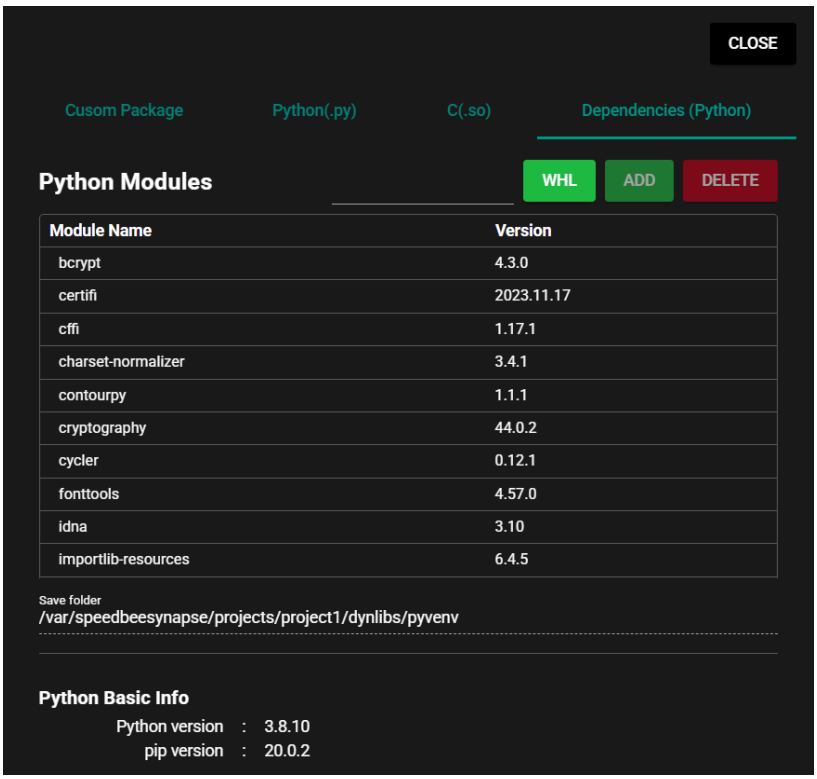
6.2.3.5.8 Adding External Libraries

You can add an external library used by a custom component to SpeeDBee Synapse using the following procedure.

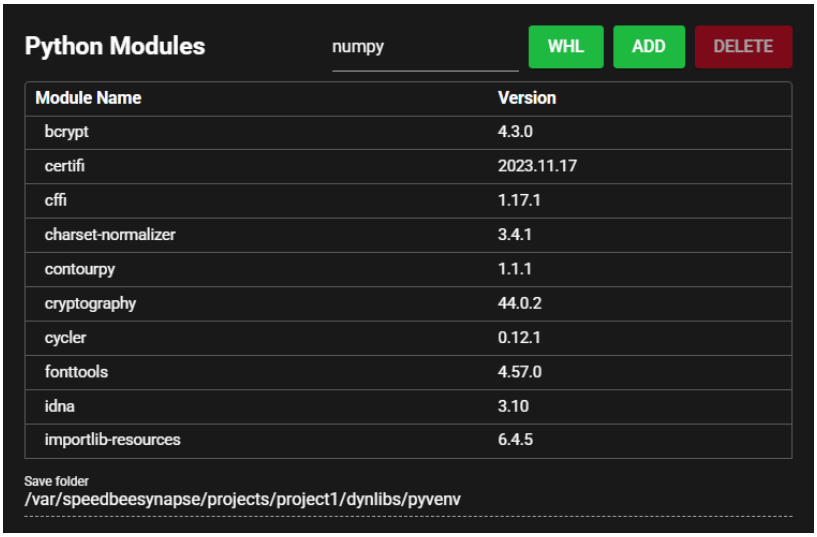
1. Press the Setting Menu icon and choose [Custom Components].



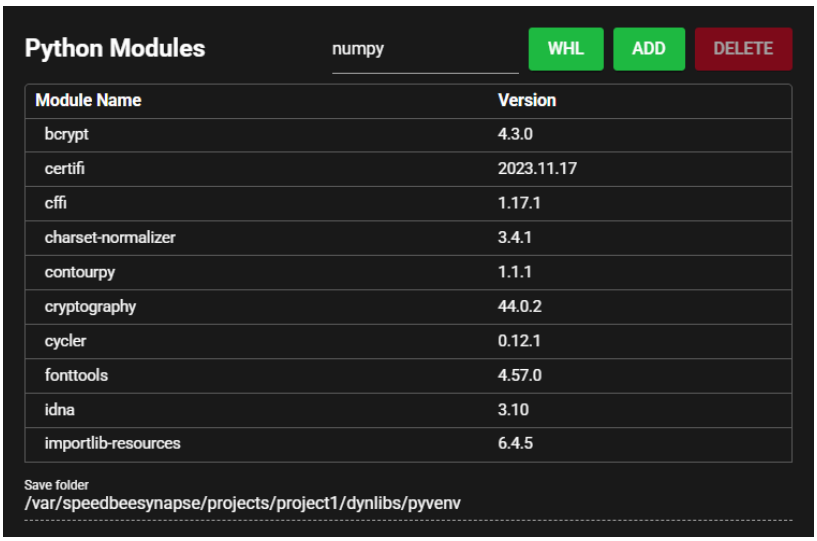
2. Select [Dependencies (Python)] in the tab at the top of the dialog.



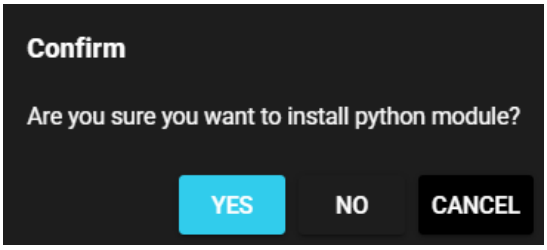
3. In the Python module field, enter the name of the library you want to register.



4. Press [Add] for the Python module.



5. Select [Yes] in the confirmation dialog.



6. The library is registered.

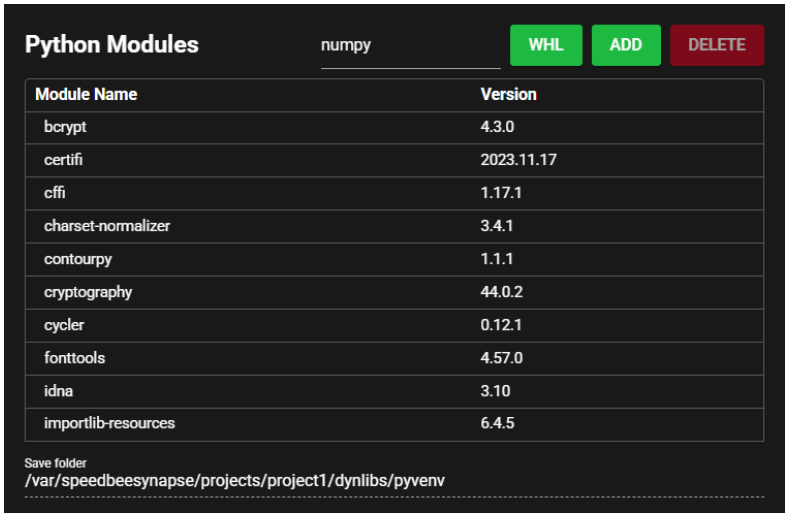
External Library Save Location

Registered external libraries are saved in the following directory:

Linux

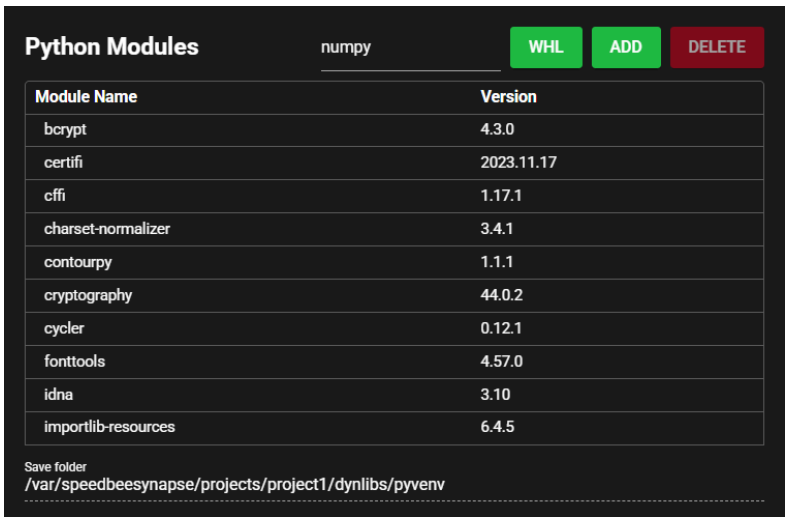
/var/speedbeesynapse/dynlibs/pyvenv

The save directory is displayed in the [Save Location] screen item.



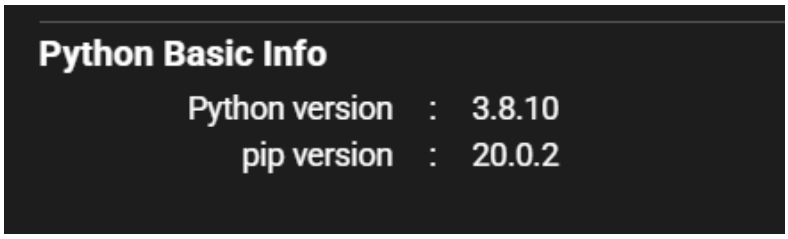
Add from .whl file

Press [WHL] for the Python module and select the .whl file.



pip version

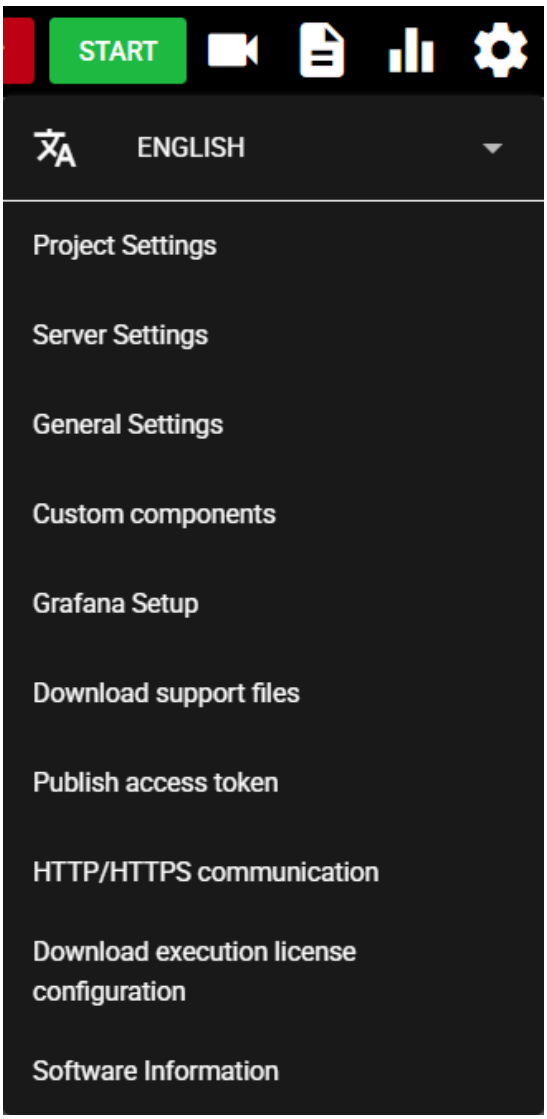
The version of pip used to manage external libraries is displayed in [pip version] under [Python Basic Info].



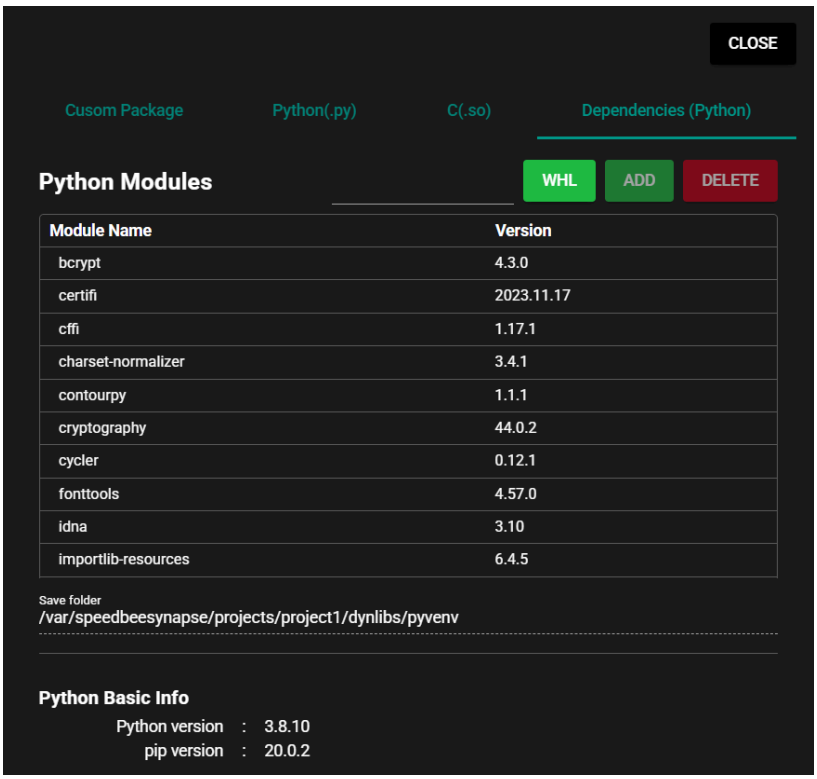
6.2.3.5.9 Deleting External Libraries

You can delete an added external library from SpeeDBee Synapse using the following procedure.

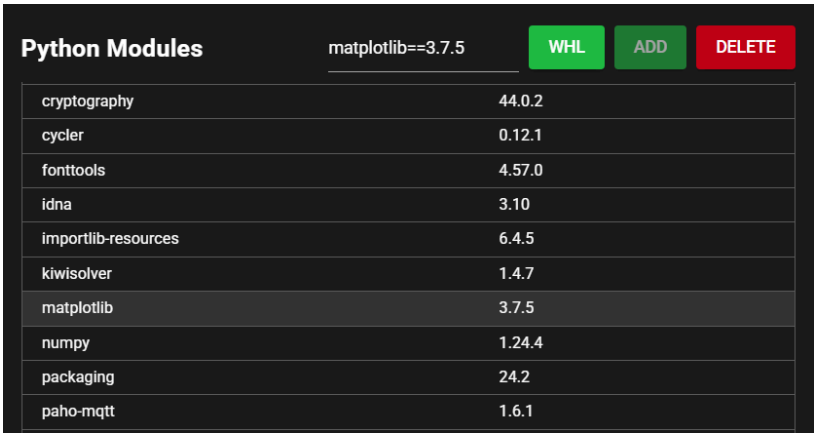
1. Press the Setting Menu icon and choose [Custom Components].



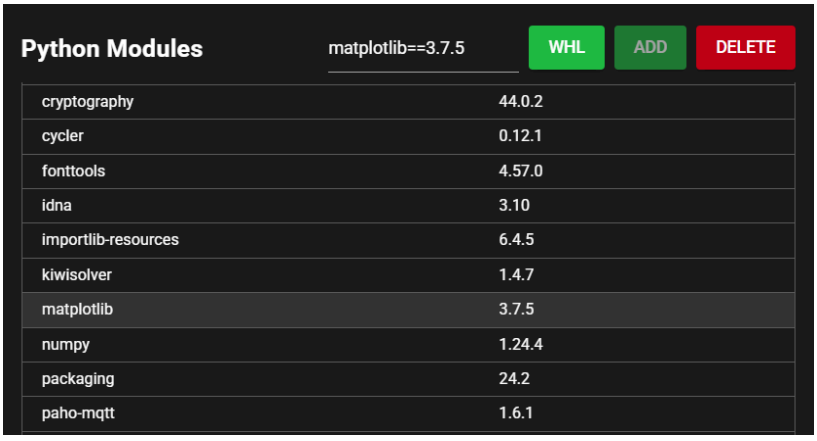
2. Select [Dependencies (Python)] in the tab at the top of the dialog.



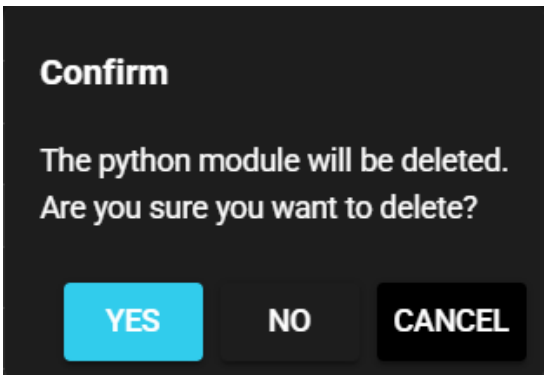
3. Click the library you want to delete.



4. Press [Delete] for the Python module.



5. Select [Yes] in the confirmation dialog.



6. The library is deleted.

6.2.4 Creating C language custom components

This section explains how to create a custom components using C.

6.2.4.1 Prerequisite Knowledge

The intended readers of this section should have the following knowledge and/or experience:

- Experience in general program development using C
- Preparation of tools necessary for development in C
- Execution of build process using a Makefile

This section also assumes that C programs are built in an environment where SpeeDBee Synapse is running. If you want to build programs in a different environment than the one in which SpeeDBee Synapse runs and later move them to the same environment, you will need to cross-build for that environment. For how to cross-build, refer to the manual or other reference document for the operating environment.

6.2.4.2 Sample Custom Components

To demonstrate how to implement custom components in C, simple custom components are provided as samples.

Download sample custom components

Alternatively, if you have installed SpeeDBee Synapse, the sample custom components are already installed in the following locations:

```
$ ls /usr/local/speedbeesynapse/share/sample_component_c/*
/usr/local/speedbeesynapse/share/sample_component_c/cmake_project:
CMakeLists.txt  README.md  countandrandom.c  readandlog.c

/usr/local/speedbeesynapse/share/sample_component_c/makefile_project_for_linux:
Makefile  README.md  countandrandom.c  readandlog.c

/usr/local/speedbeesynapse/share/sample_component_c/makefile_project_for_windows:
Makefile  README.md  countandrandom.c  readandlog.c
```

The following three sample custom component directories are provided to suit your build environment:

- `cmake_project`

  Project directory for building with CMake. The same steps can be used to build on both Linux and Windows.
- `makefile_project_for_linux`

  Linux environment project directory for building with Make.
- `makefile_project_for_windows`

Project directory for building with NMAKE on Windows.

6.2.4.2.1 Building Sample Components

See the respective README.md file for instructions on how to build in each project directory. The components built and generated are the same with all project directories.

If you are running SpeeDBee Synapse on Windows, choose either `cmake_project` or `makefile_project_for_windows`.
If you are running SpeeDBee Synapse on Linux, choose either `cmake_project` or `makefile_project_for_linux`.

6.2.4.2.2 Sample Description

This section explains the following sample as an example of custom component.

- `count_and_random` (countandrandom.c)

  This component creates the following two columns for output ports and continues to output data.

Column name	Description
count	A value counting up from 1 is registered once every 3 seconds.
random	A random value in the range [0.0, 100.0) is registered once every 3 seconds.

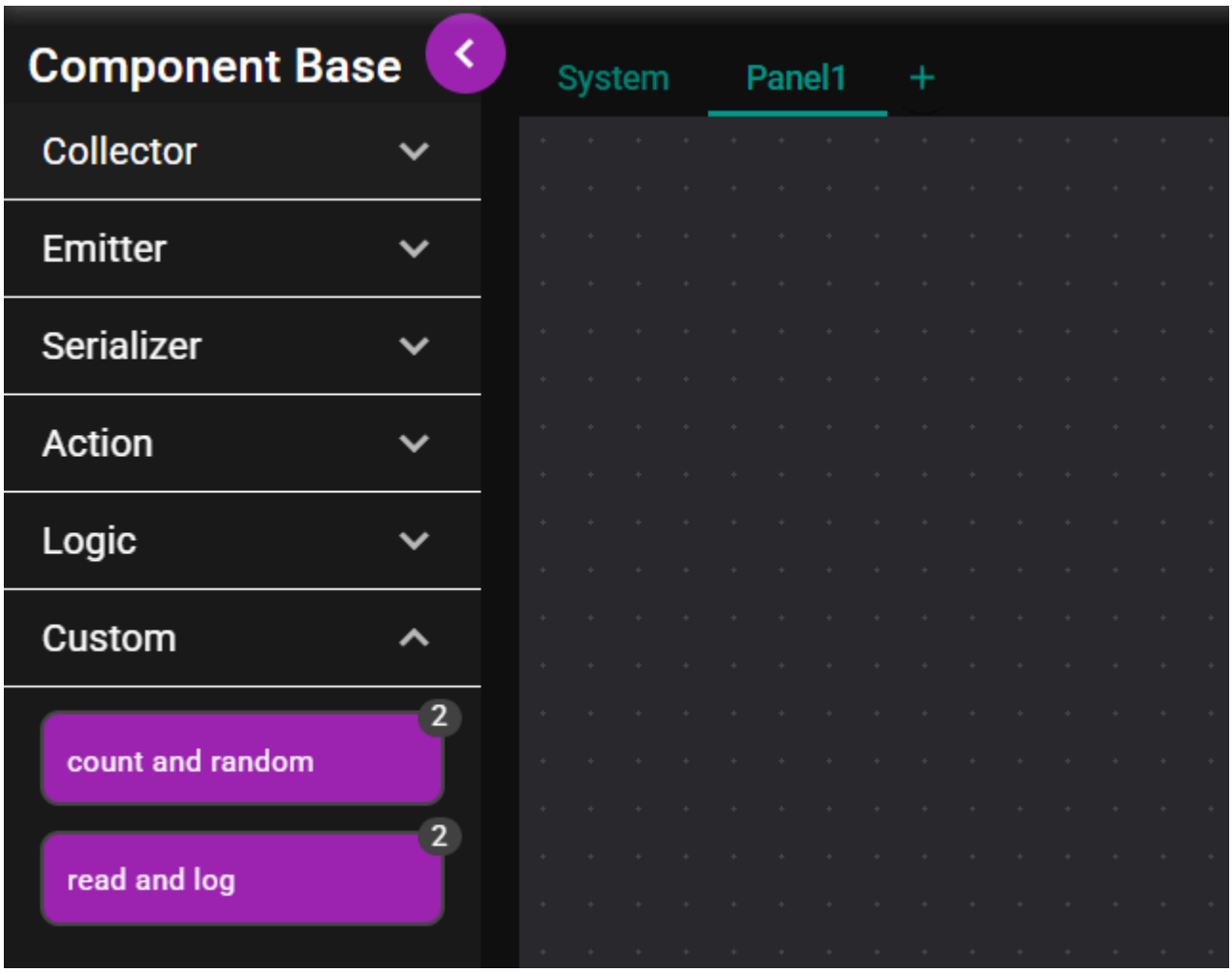
- `read_and_log` (readandlog.c)

  This component receives data from the input port and outputs it to string columns and logs.

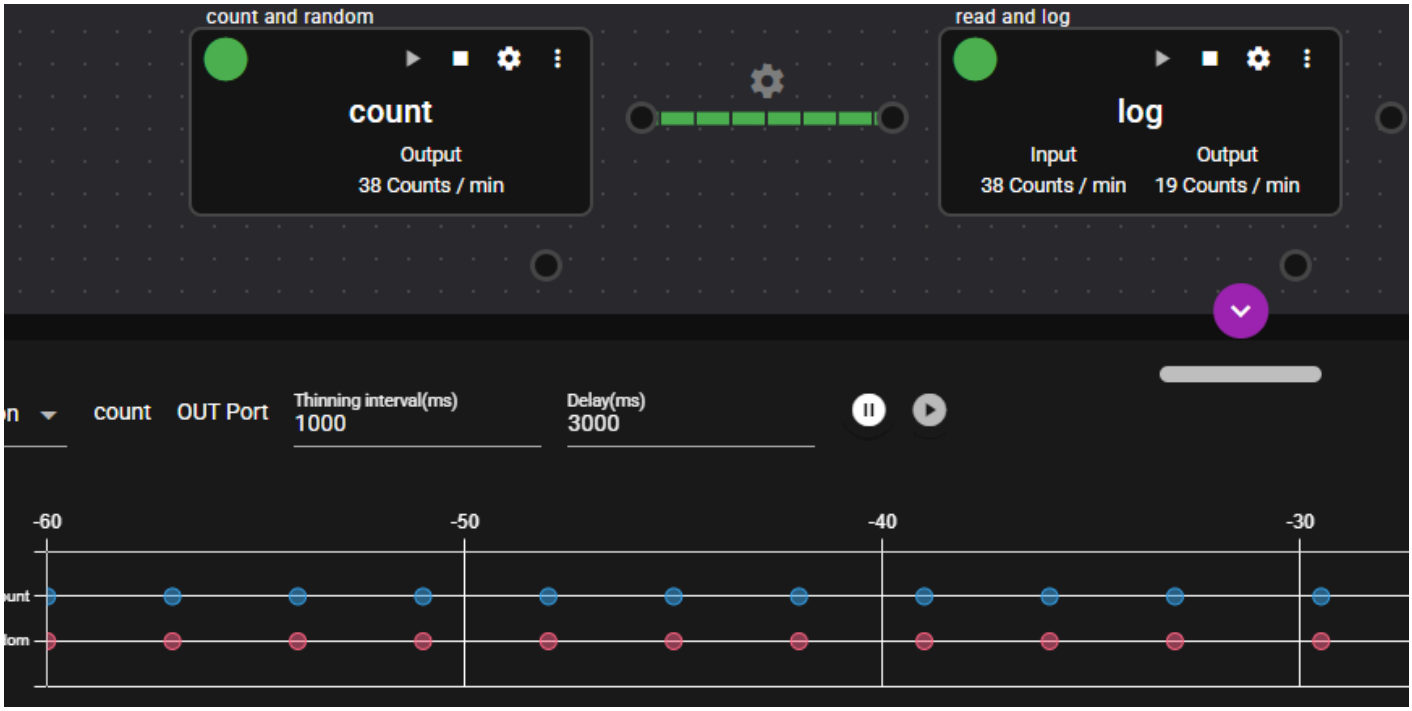
6.2.4.2.3 Execution

A successful build of a sample component generates a `.so` file on Linux or a `.dll` file on Windows for each component. The generated `.so` or `.dll` file can be registered from the WEBUI to be used as a component from SpeeDBee Synapse. For instructions on how to register these files, see Using Custom Components.

Once the files are registered and the core is restarted, the new custom components are available from the SpeeDBee Synapse screen just like any other component.



You can concatenate `count_and_random` and `read_and_log` and execute them to display their output columns. The following monitors the output ports of `count_and_random`.



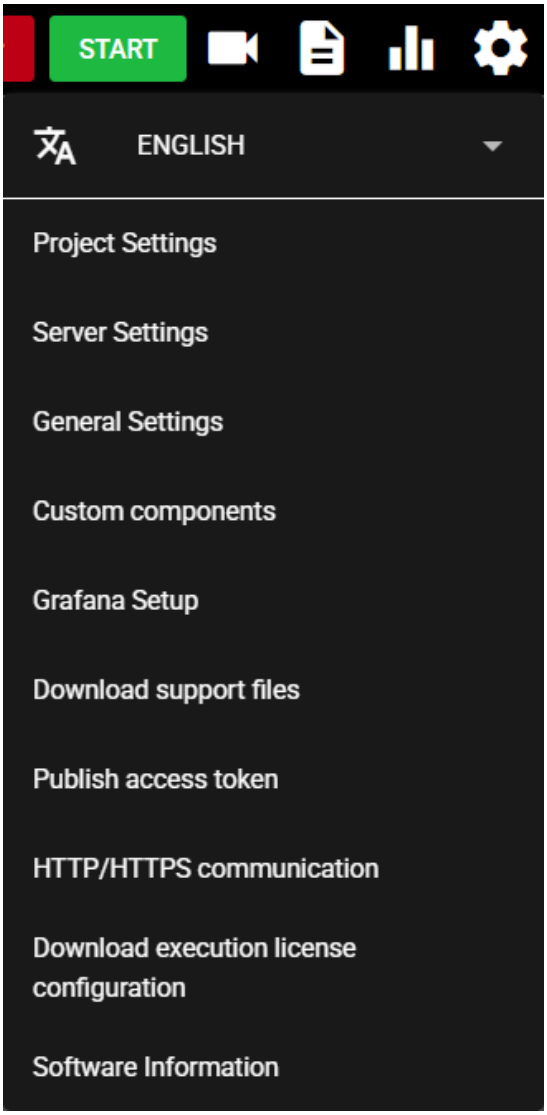
6.2.4.3 Development files

If you want to obtain the header files and library files necessary for custom component development, you can get them from the WebUI.

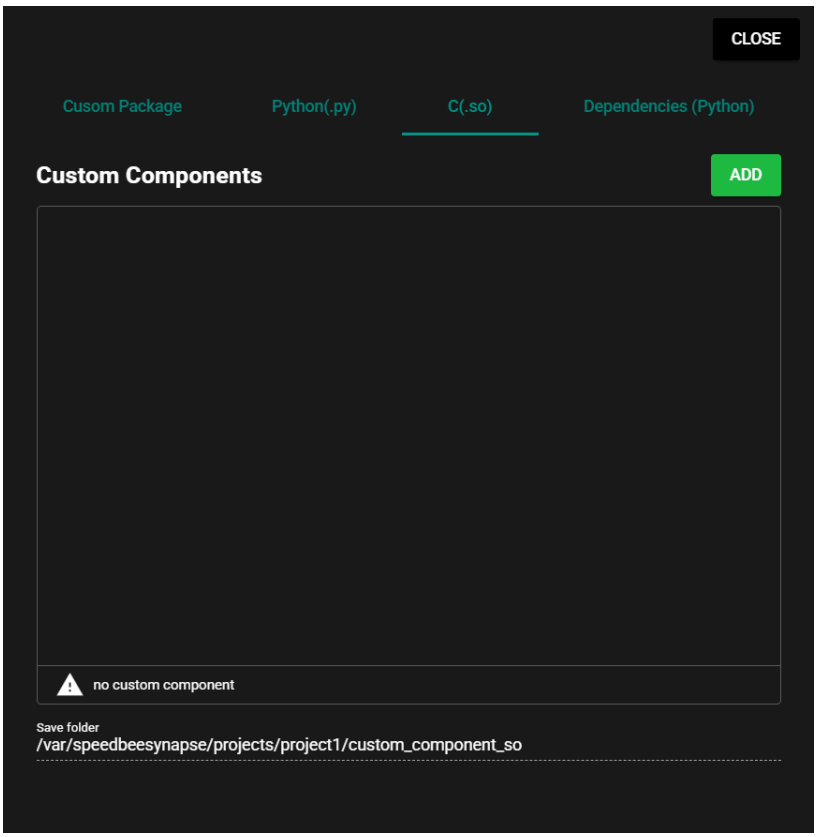
6.2.4.3.1 Download Development Files

You can download the development files as follows:

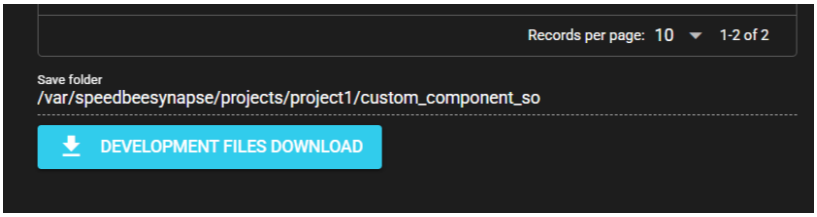
1. Press the Setting Menu icon and choose [Custom Components].



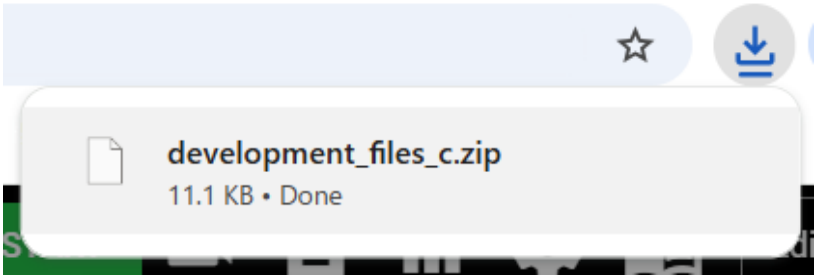
2. Select [C(.so)] in the tab at the top of the dialog.



3. Press the [DEVELOPMENT FILES DOWNLOAD] button.



4. The download will begin in your browser.



6.2.4.3.2 Development Files Description

The development files include the following files:

File name	Description
libhiveframework.so	This is a library file for Linux. It contains the implementation of API functions declared in `hiveframework.h`. Link with `-lhiveframework` at compile time. It will be downloaded only

File name	Description
	if the platform running the SpeeDBee Synapse is Linux.
hiveframework.lib	This is a library file for Windows. It contains the implementation of API functions declared in `hiveframework.h`. Link settings are required during compilation. It will be downloaded only if the platform running the SpeeDBee Synapse is Windows.
hiveframework.h	The primary header files required when developing SpeeDBee Synapse components. All C custom components require an `#include <hiveframework.h>`. For details, please refer to Custom Component Implementation Details.
hive_data_type.h	This is a header file dedicated to data type definitions. It is automatically included from the `hiveframework.h`. For details, please refer to Custom Component Implementation Details.

6.2.4.4 Custom Component Implementation Details

The following sections describe the implementation of custom components using as examples source codes that use the sample custom components `count_and_random` and `read_and_log`.

6.2.4.4.1 Include Header

When developing custom components in C, you must include the header file `hiveframework.h`.

```
#include <hiveframework.h>

 :  Followed by implementation of custom components
```

6.2.4.4.2 Custom Component Information Definition

In a C source code for a custom component, the name and identity of the component and the lifecycle callback function must be defined using the following global variables.

```
  :  Definition of functions used by custom components

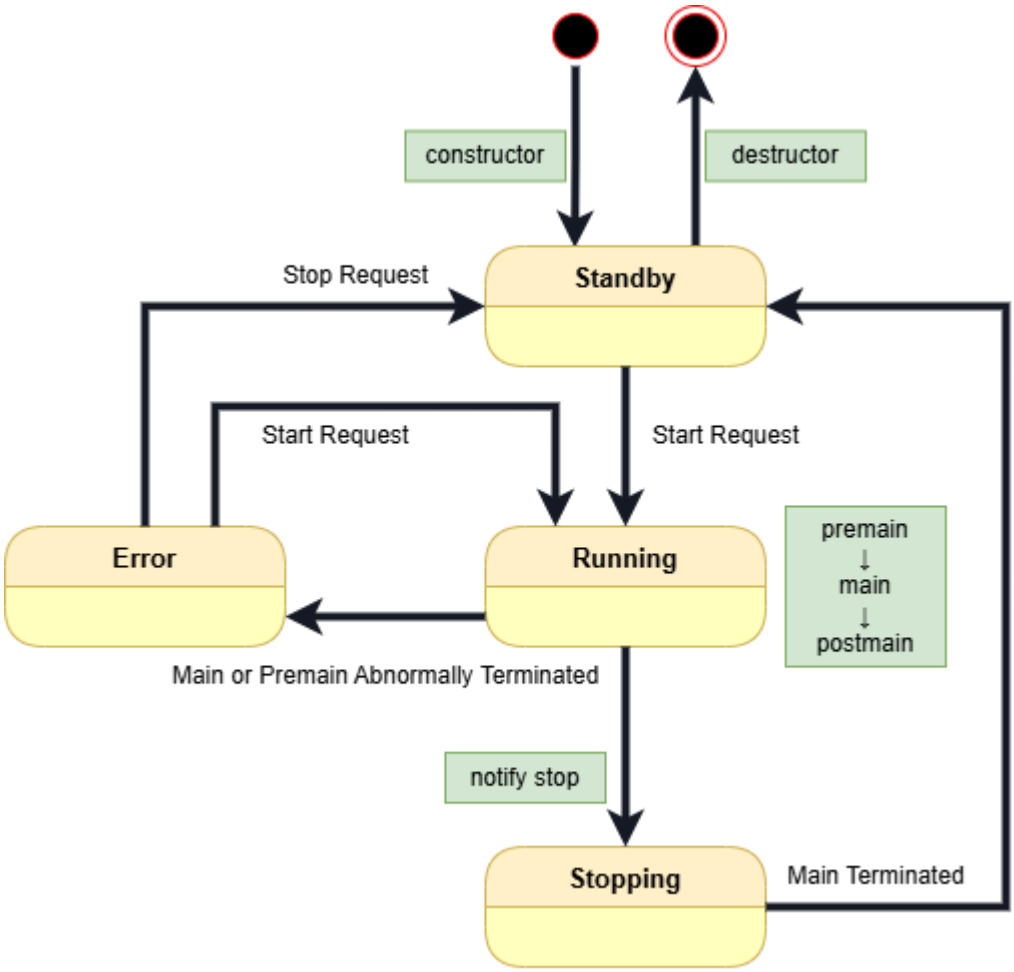
const static HIVE_COMPONENT_DEFINITION info = {
  .uuid            = "2e80ad02-1730-4bdc-b321-2fbc2d1b942e",
  .name            = "count and random",
  .parameter_type  = HIVE_COMPONENT_PARAMETER_NONE,
  .in_ports        = 0,
  .out_ports       = 1,
  .functions     = {
    .constructor         = NULL,      // optional
    .premain             = premain,   // optional
    .main                = mainloop,  // required
    .postmain            = postmain,  // optional
    .destructor          = NULL,      // optional
    .stop                = NULL,      // optional
  },
};
EXPORT_COMPONENT_DEFINITION(info);
```

Member name	Definition information	Description
`uuid`	UUID	The UUID is used to identify the component. Generate a random ID and set it here.
`name`	Component name	Set the name for the component. Since this name is displayed on the screen, it is better to use a name that distinguishes this component from other components.
`tag`	Tag	This adds a tag to the component. Use "collector," "emitter," "serializer," "action," or "logic" to specify under which component base to place the component. If this member is omitted, the component is placed under Custom.
`parameter_type`	Parameter type	While you can choose the type of parameter to be passed to this component from NONE, TEXT, and JSON, this item is currently not used.
`out_ports`	Number of output ports	Set the number of output ports for this component. To pass data to an input port of another component, set this to 1.
`in_ports`	Number of input ports	Set the number of input ports for this component. To receive data from an output port of another component, set this to 1.
`functions`	List of callback functions	Set the list of functions that implement the operations of this component. See the next section for details.

Always generate random IDs for UUIDs. If you use the same UUID as another component, you may not be able to use either component.

6.2.4.4.3 Lifecycle Callback Definition

The `functions` member explained in the previous section can specify the function pointers called based on the lifecycle of the component instance.



Callback function	Description
`constructor`	This function is called when the component instance is generated.
`destructor`	This function is called when the component instance is destroyed.
`premain`	This function is called before the "main" function when starting the component.

Callback function	Description
`main`	This function implements the main processing of the component. This function must continue to loop without exiting until the component is requested to terminate.
`postmain`	This function is called when a component exits, just after the "main" function ends.
`stop`	This function is executed when a component is requested to stop during the execution of the "main" function.

The `functions` definition as per the previous section is defined at the end of `count_and_random` of the sample component.

Because `constructor` and `destructor` are not required for this component, they are omitted (specified as NULL). The `premain` and `postmain` definitions can also be omitted if they are not necessary. (In this example, only log output is actually implemented.)

The `main` definition is required because it is the body of the component's processing. The actual function definition name must be something else, such as `mainloop`. Giving the name `main` would conflict with the `main` function of the C executable body.

The following sections explain the implementation of the `main` function in the sample program.

6.2.4.4.4 Creating Output Port Columns and Registering Data (`count_and_random`)

The following is the main processing of the custom component `count_and_random`. The procedure for outputting data is explained based on this operation.

```
static bool mainloop(HIVE_COMPONENT comp, const char *param, HIVE_STATUS status) {
  HIVE_LOG_INFO("create columns");
  HIVE_OUTCOLUMN clm_count = hive_outport_create_column(comp, OUTPORT1, "count",
HIVE_DATA_SCALAR(HIVE_TYPE_INT32), HIVE_COLUMN_OPTION_NONE);
  HIVE_OUTCOLUMN clm_random = hive_outport_create_column(comp, OUTPORT1, "random",
HIVE_DATA_SCALAR(HIVE_TYPE_DOUBLE), HIVE_COLUMN_OPTION_NONE);

  int count = 1;
  while (hive_component_runnable(comp)) {
    HIVE_LOG_INFO("insert count");
    if (!hive_outcolumn_insert(clm_count, &count)) {
      HIVE_API_ERROR err = hive_get_api_error();
      HIVE_LOG_ERROR("insert 'count' failed: errcode=0x%08x", err.code);
    }

    HIVE_LOG_INFO("insert random");
    double r = 100 * (double)rand() / (double)RAND_MAX;
    if (!hive_outcolumn_insert(clm_random, &r)) {
      HIVE_API_ERROR err = hive_get_api_error();
      HIVE_LOG_ERROR("insert 'count' failed: errcode=0x%08x", err.code);
    }

    usleep(3000000);
    count++;
  }

  return true;
}
```

1. Creating a column

   Start by creating a column.

   ```
   HIVE_OUTCOLUMN clm_count = hive_outport_create_column(comp, OUTPORT1, "count",
   HIVE_DATA_SCALAR(HIVE_TYPE_INT32), HIVE_COLUMN_OPTION_NONE);
   HIVE_OUTCOLUMN clm_random = hive_outport_create_column(comp, OUTPORT1, "random",
   HIVE_DATA_SCALAR(HIVE_TYPE_DOUBLE), HIVE_COLUMN_OPTION_NONE);
   ```

   This component uses `count`, a 32-bit integer column, and `random`, a double-precision floating point column. The `hive_outport_create_column()` function is used to create a column by specifying the column name, data type, etc. as arguments. Keep the return value in a variable, because you will use it later to populate the column.

   All data output by the component must be registered in this generated column.

About Column Names That Cannot Be Used

The following characters cannot be used in column names

- `\` - backslash
- `/` - slash
- `*` - asterisk
- `?` - question mark
- `"` - double quote
- `< >` - angle brackets
- `|` - pipe
- `'` - single quote
- ` ` - space

2. Repeating periodic operation

   To periodically register data, the "while" loop is used to implement repeated operation.

```
int count = 1;
while (hive_component_runnable(comp)) {
  :

  usleep(3000000);
  count++;
}
```

   The loop condition calls `hive_component_runnable()`, which returns true while the component is running. If you request to stop the component from the screen, this function will return false, so in that case you should promptly terminate the `mainloop` function.

   In addition to the while loop, you can use `hive_component_interval_call()` for iteration. Follow the link for instructions on how to use it.

3. Registering data in the column

   Data is registered in the column in an iterative process.

```
if (!hive_outcolumn_insert(clm_count, &count)) {
  HIVE_API_ERROR err = hive_get_api_error();
  HIVE_LOG_ERROR("insert 'count' failed: errcode=0x%08x", err.code);
}

double r = 100 * (double)rand() / (double)RAND_MAX;
if (!hive_outcolumn_insert(clm_random, &r)) {
  HIVE_API_ERROR err = hive_get_api_error();
  HIVE_LOG_ERROR("insert 'count' failed: errcode=0x%08x", err.code);
}
```

Data registration can be done using `hive_outcolumn_insert()`. Specify the previously created column variable as the first argument and the pointer to the data to be registered as the second argument. The pointer must match the data type previously specified when the column was created. Note that specifying the wrong type of data may cause the software to terminate abnormally.

> **Maximum size that can be registered with insert**
>
> The maximum size of data that can be registered with hive_outcolumn_insert() is 512 bytes (STRING is 511 bytes, not including null). When a value exceeding the maximum size is registered, the return value is false.

6.2.4.4.5 Reading Data from Input Port (`read_and_log`)

The following is the main processing of the custom component `read_and_log`. (Comments and log output are omitted.)

```
static bool mainloop(HIVE_COMPONENT comp, const char *param, HIVE_STATUS status) {
  HIVE_OUTCOLUMN clm_log = hive_outport_create_column(comp, OUTPORT1, "log",
HIVE_DATA_SCALAR(HIVE_TYPE_STRING), HIVE_COLUMN_OPTION_NONE);

  HIVE_CONTINUOUS_READER *creader
    = hive_inport_continuous_reader(comp, INPORT1, hive_timestamp()+5000000000UL);

  hive_time_t last_updated = 0;
  while (hive_component_runnable(comp)) {
    const HIVE_COLUMN_READ_RESULT *read_result = hive_continuous_reader_read(creader);
    if (!read_result) {
      continue;
    }

    if (last_updated < read_result->incolumn_list->updated_at) {
      last_updated = read_result->incolumn_list->updated_at;
    }

    HIVE_RECORD_ITERATOR iter = HIVE_GET_RECORD_ITERATOR(read_result);
    const HIVE_RECORD *record;
    while ((record = HIVE_RECORD_ITERATOR_GET_NEXT(&iter)) != NULL) {
      for (int i=0; i < record->data_count; i++) {
        const HIVE_RECORD_DATA *rd = HIVE_RECORD_GET_DATA(record, i);
        if (rd && rd->data_size != 0) {
          log_record(clm_log, record->timestamp, &read_result->incolumn_list->incolumns[i], rd);
        }
      }
    }
  }
  hive_continuous_reader_release(creader);

  return true;
}
```

1. Creating a column

In this component as well, start by creating a column.

```
HIVE_OUTCOLUMN clm_log = hive_outport_create_column(comp, OUTPORT1, "log",
HIVE_DATA_SCALAR(HIVE_TYPE_STRING), HIVE_COLUMN_OPTION_NONE);
```

This column is for outputting strings.

> **About Column Names That Cannot Be Used**
>
> The following characters cannot be used in column names
>
> - `\` - backslash
> - `/` - slash
> - `*` - asterisk
> - `?` - question mark
> - `"` - double quote
> - `< >` - angle brackets
> - `|` - pipe
> - `'` - single quote
> - ` ` - space

2. Data retrieval loop

The process of repeatedly receiving data from the input port is as follows.

```
HIVE_CONTINUOUS_READER *creader
  = hive_inport_continuous_reader(comp, INPORT1, hive_timestamp()+5000000000UL);

:
while (hive_component_runnable(comp)) {
  const HIVE_COLUMN_READ_RESULT *read_result = hive_continuous_reader_read(creader);
  if (!read_result) {
    continue;
  }

  :
}
hive_continuous_reader_release(creader);
```

The following four functions are used here:

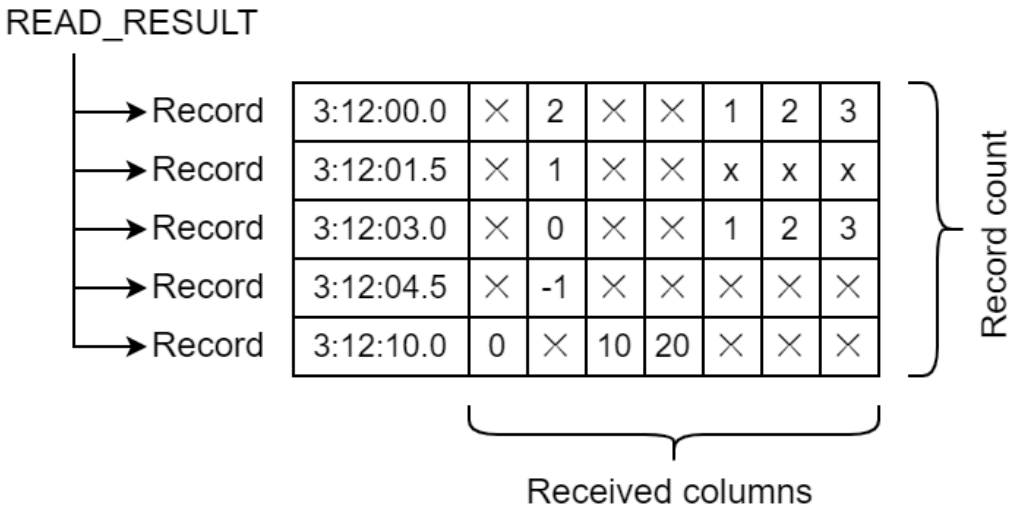
- `hive_inport_continuous_reader()`
  This function creates a handle to continuously retrieve data from an input port.
- `hive_continuous_reader_release()`
  This function is used to release the handle when stopping the retrieval of data.
- `hive_continuous_reader_read()`
  This function retrieves data from the created handle. The retrieved data is returned as a pointer to a

`HIVE_COLUMN_READ_RESULT` structure, but `NULL` may be returned if there is no data yet, in which case it should be ignored.

- `hive_component_runnable()`
  This function was also used in the previous section. It returns true while the component is in the Running state. If you request to stop the component from the screen, this function will return false, so in that case you should promptly terminate the `mainloop` function.

3. Reading the records of retrieved data

`hive_continuous_reader_read()` This function collectively retrieves data entering from the input port for a certain time width and stores it in `read_result`, a pointer to a `HIVE_COLUMN_READ_RESULT` structure. The `HIVE_COLUMN_READ_RESULT` structure consists of multiple records that store data from the same time as shown below. Each record stores multiple columns of data.



The following double loop allows for processing one record at a time from this structure. `HIVE_GET_RECORD_ITERATOR` () and `HIVE_RECORD_ITERATOR_GET_NEXT` are loops for records, and the for statement inside the loop, with variable `i`, is the repetition of data processing of each record.

```
HIVE_RECORD_ITERATOR iter = HIVE_GET_RECORD_ITERATOR(read_result);
const HIVE_RECORD *record;
while ((record = HIVE_RECORD_ITERATOR_GET_NEXT(&iter)) != NULL) {
  for (int i=0; i < record->data_count; i++) {
    const HIVE_RECORD_DATA *rd = HIVE_RECORD_GET_DATA(record, i);
    if (rd && rd->data_size != 0) {
      // rd holds 1 column worth of data here
      log_record(clm_log, record->timestamp, &read_result->incolumn_list->incolumns[i], rd);
    }
  }
}
```

4. Handling of each data object within a record

Information for a single data object can be obtained from the `incolumn_list` member of the `HIVE_RECORD` and `HIVE_COLUMN_READ_RESULT` structures.

Item	Type	Reference example in this sample program	Description
Data name	const char*	`read_result->incolumn_list->incolumns[i]->data_name`	Name of the column storing the data
Component name	const char*	`read_result->incolumn_list->incolumns[i]->source_name`	Name of component that generated the data
Data type	HIVE_DATA_TYPE	`read_result->incolumn_list->incolumns[i]->data_type`	Data type of the column storing the data
Data size	uint32_t	`HIVE_RECORD_GET_DATA(record, i)->data_size`	Data size of actual data
Data address	char *	`HIVE_RECORD_GET_DATA(record, i)->data`	Address storing the actual data

As shown above, the actual data is a `const char` pointer. Note that since it may accept different data types, you must identify the data with `HIVE_DATA_TYPE` and cast and reference it according to each type.

6.2.4.4.6 Log Output

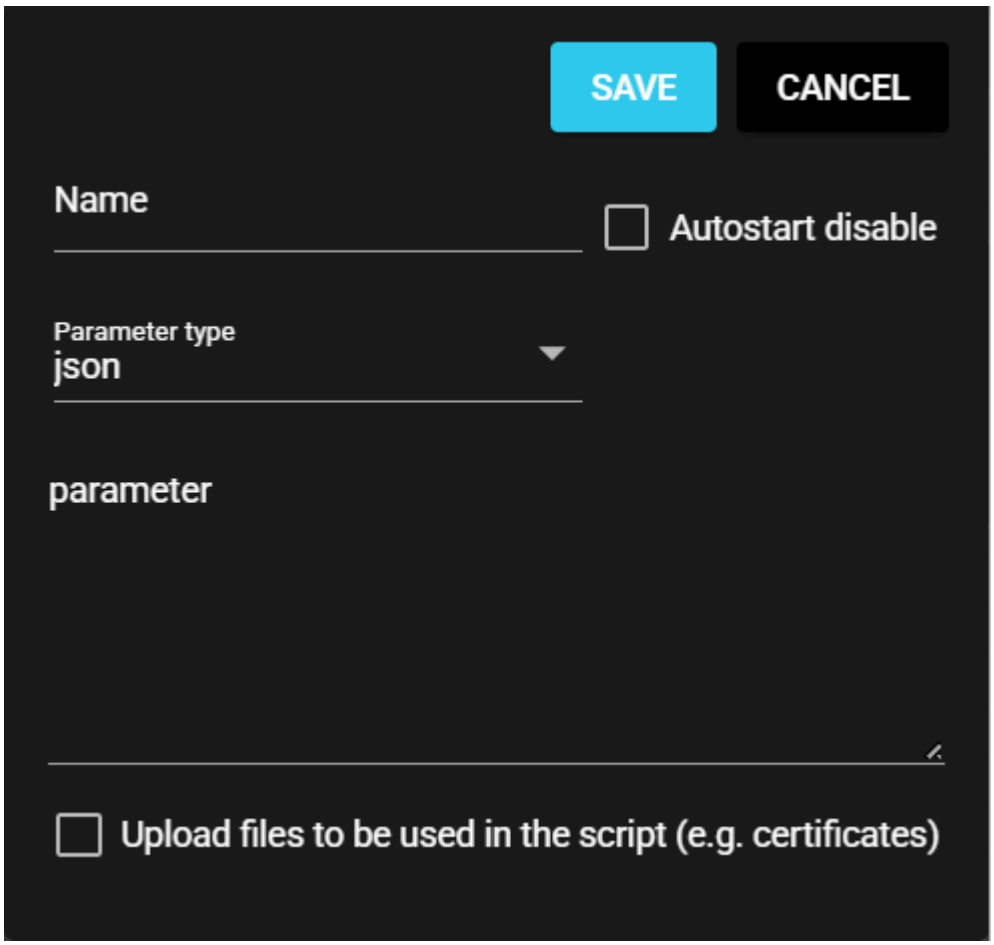
A component can output logs at any time.

```
HIVE_LOG_INFO("create columns");
HIVE_LOG_INFO("x=%d, y=%f, z=%s", x, 12.34, "aiueo");
HIVE_LOG_ERROR("insert 'count' failed: errcode=0x%08x", err.code);
```

You can output the log in the same format as `printf()` in the standard library, as shown above. The output log can be downloaded from the screen. See Various Logs.

6.2.4.4.7 Component Parameters

When you generate a component instance from the screen, you can set the component's runtime parameters as its settings.



Parameter type can be JSON or STRING. In either case, components implemented in C will be passed to the second argument of the main function as a `const char*` string, which can be parsed or handled directly as a string by the standard library sscanf.

```
static bool mainloop(HIVE_COMPONENT comp, const char *param, HIVE_STATUS status) {
  int val1, val2;
  sscanf(param, "%d,%d", &val1, &val2); // perform parameter as a number string

  // some process using 'val1' and 'val2'.
}
```

If you want to work with JSON format parameters, you also need to parse the JSON-format string stored in param. Since JSON is not handled by API in this system, consider using a general-purpose library.

Reference

- json-c
- jansson

6.2.4.5 API for Custom Components

Many API functions are provided in addition to the functions introduced in this section. For more information, see the Custom Component C-API Reference in the appendix.

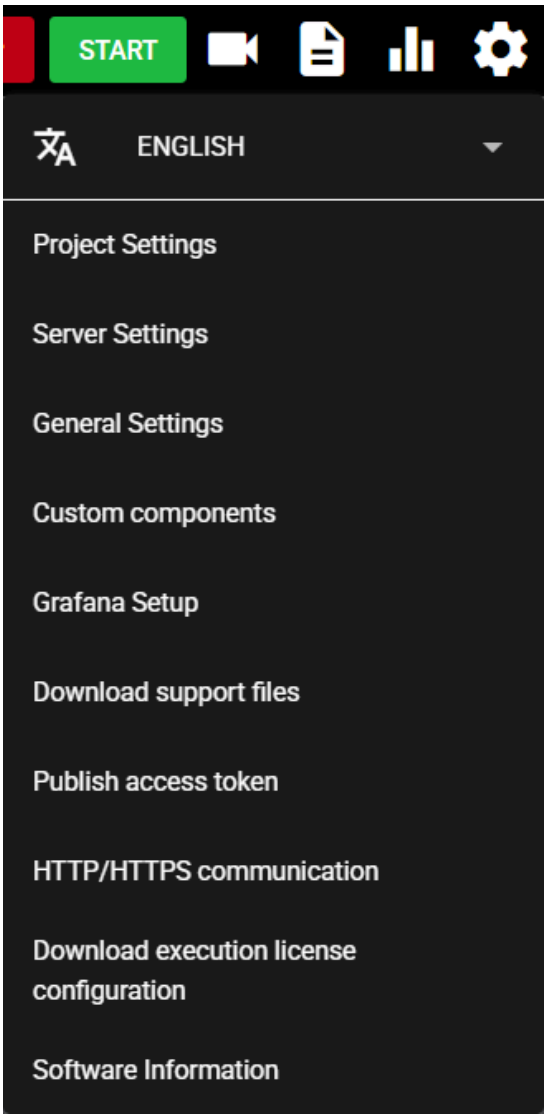
6.2.4.6 Using Custom Components

This section explains how to use custom components created in SpeeDBee Synapse.

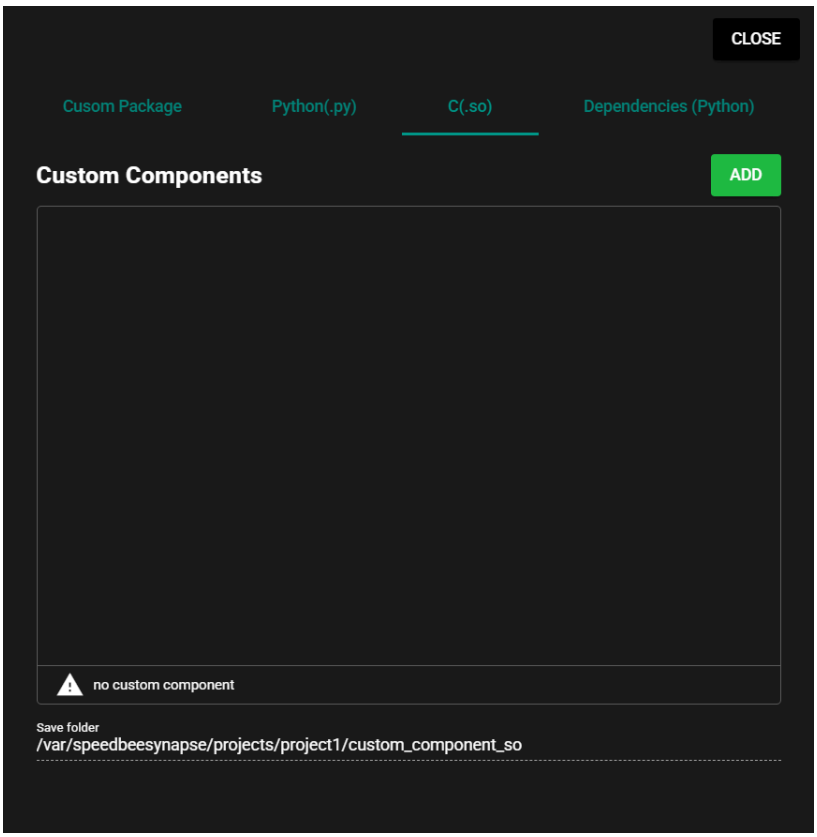
6.2.4.6.1 Add

You can register a built custom component in SpeeDBee Synapse using the following procedure.

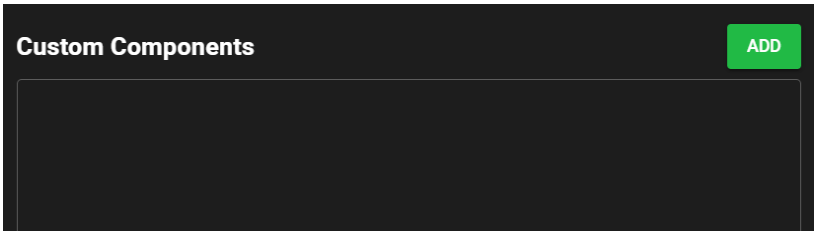
1. Press the Setting Menu icon and choose [Custom Components].



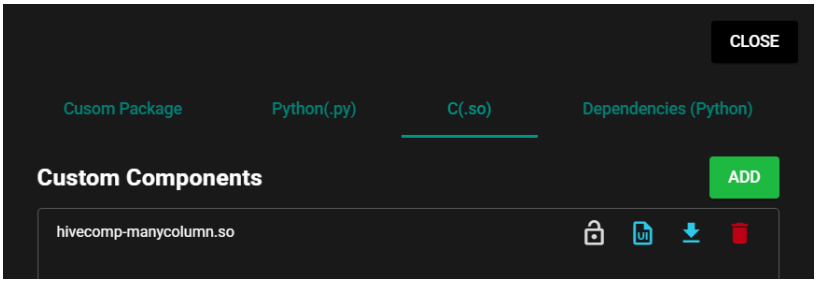
2. Select [C(.so)] in the tab at the top of the dialog.



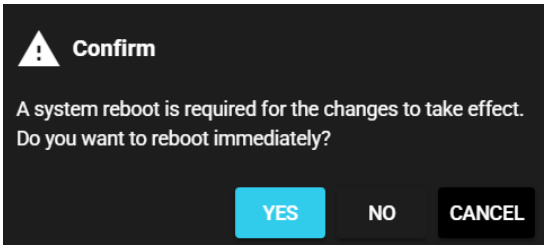
3. Press [Add] to register the so file for the custom component you have already built.



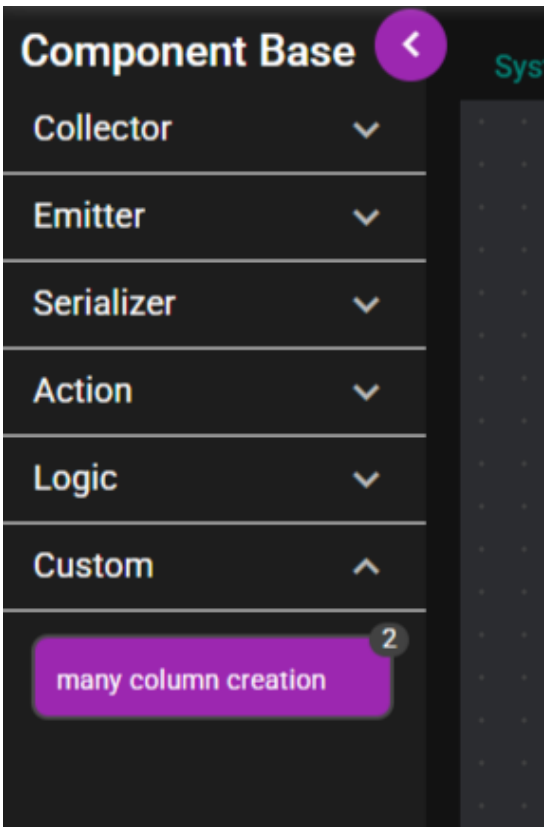
4. Press [Close].



5. Select [Yes] in the confirmation dialog to restart the system to apply the changes.



6. The left menu shows the registered custom component.



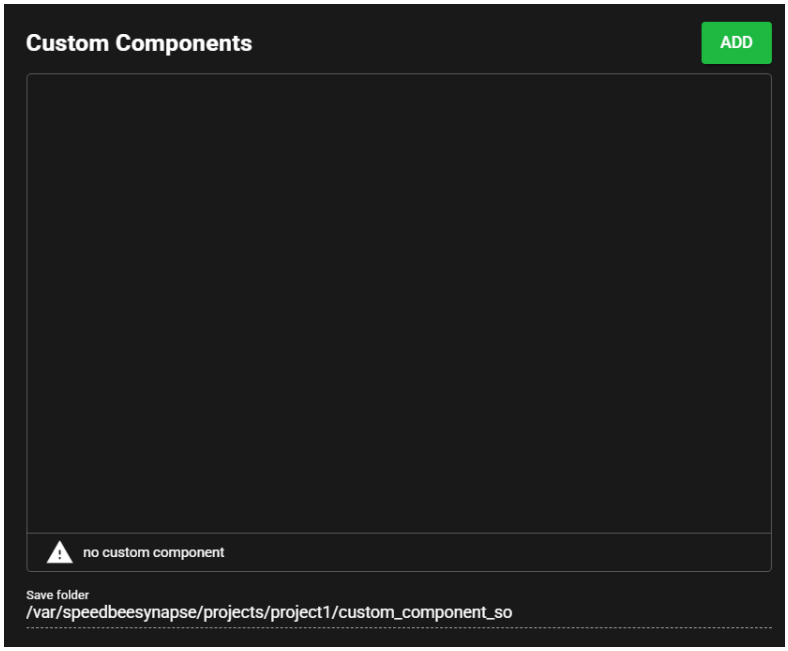
File Save Location

Registered custom component files are saved in the following directory:

Linux

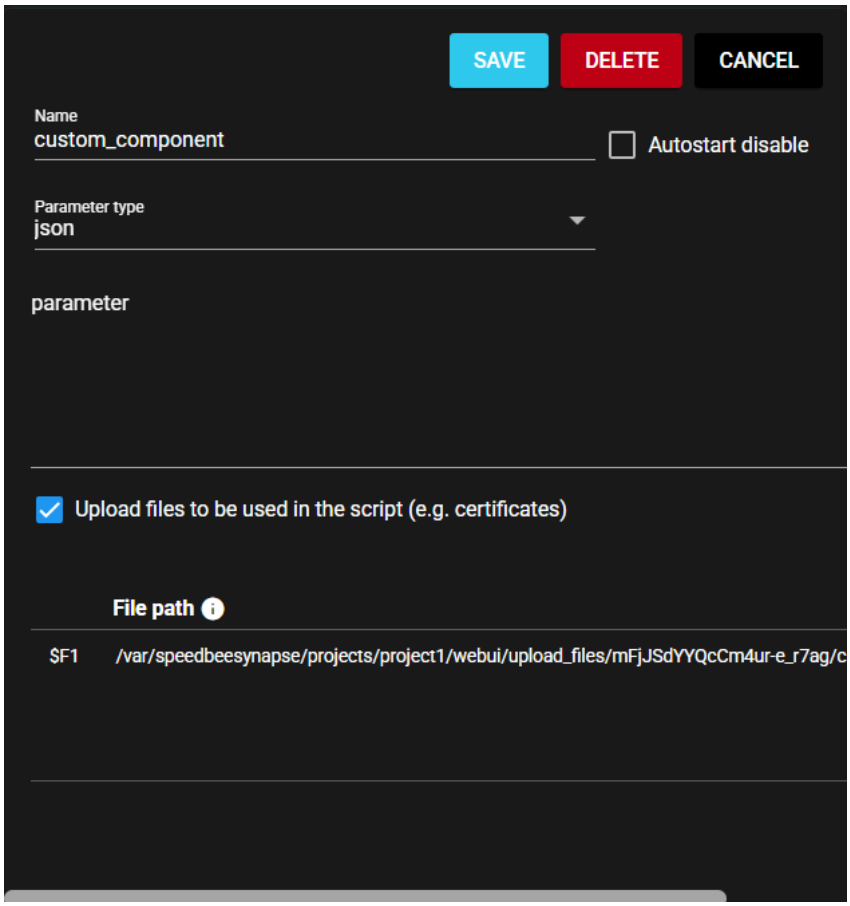
/var/speedbeesynapse/custom_component_so

The save directory is displayed in the [Save Location] screen item.



6.2.4.6.2 Settings Screen

The items on the custom component settings screen are as follows.



Item	Description
Name	Enter a name for the component. Note: Must be different from any other component name.
Autostart disable	Set to ON to disable the component's Autostart.
Parameter type	Select a parameter type from the following: • STRING: String • JSON: JSON string
Parameter	Enter component parameters.
Upload files to be used in the script (e.g. certificates)	Set to ON to upload a file, such as a certificate, for use in custom component processing.

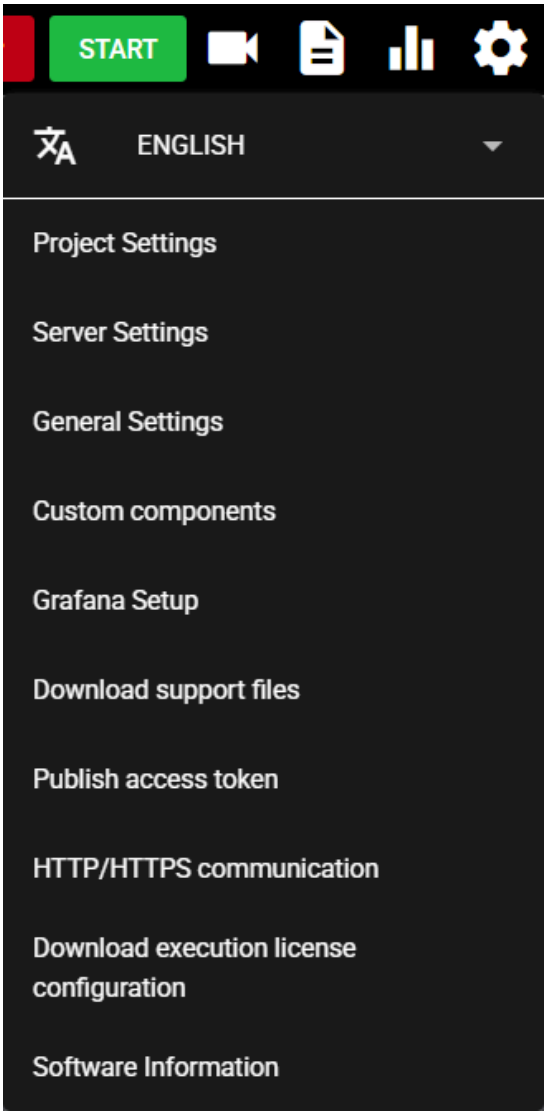
Item	Description
Add file	Press this button to upload the file to be used in the script. Note: Newly added files are uploaded when you save the component settings.
File path	Save location for uploaded files Note: When using a file path in a parameter, you can use a variable. For more information, click the icon to the right of [File Path] in the title.

You can change the parameter input area of the settings screen to display the screen items defined by the user. For more information, see Creating Custom UI.

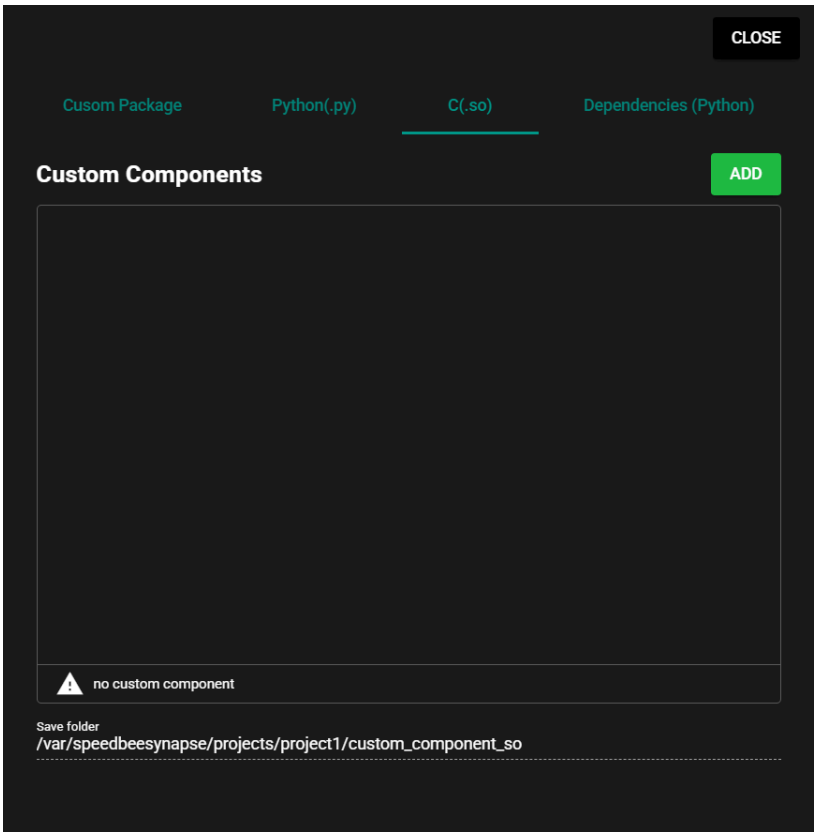
6.2.4.6.3 Delete

You can delete a registered custom component from SpeeDBee Synapse using the following procedure.

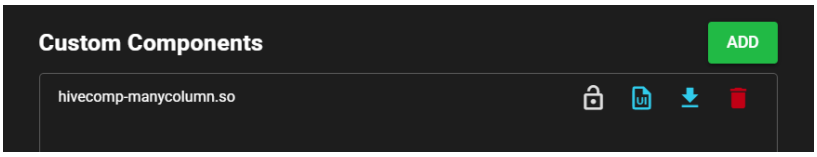
1. Press the Setting Menu icon and choose [Custom Components].



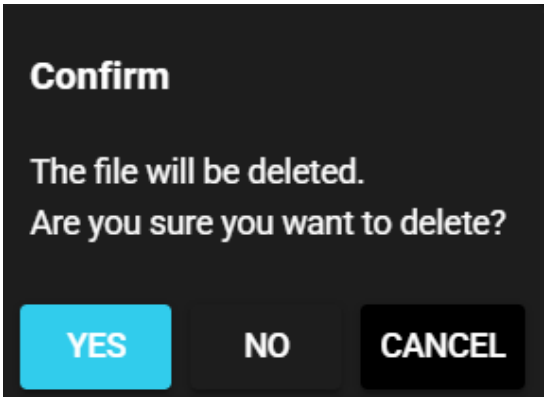
2. Select [C(.so)] in the tab at the top of the dialog.



3. Press the trash can icon (Delete) for the custom component.



4. Select [Yes] in the confirmation dialog.

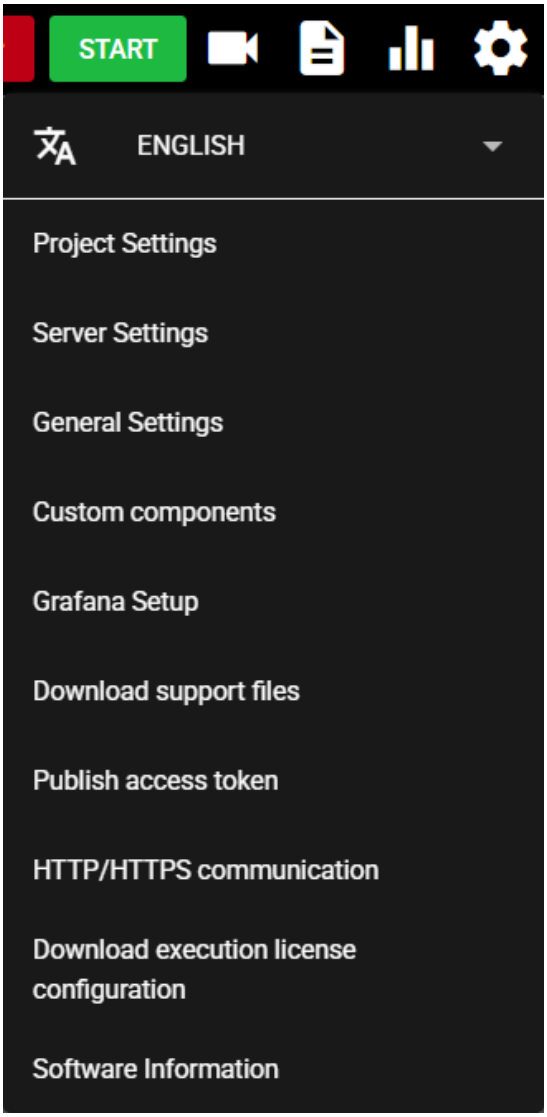


5. The custom component is deleted.

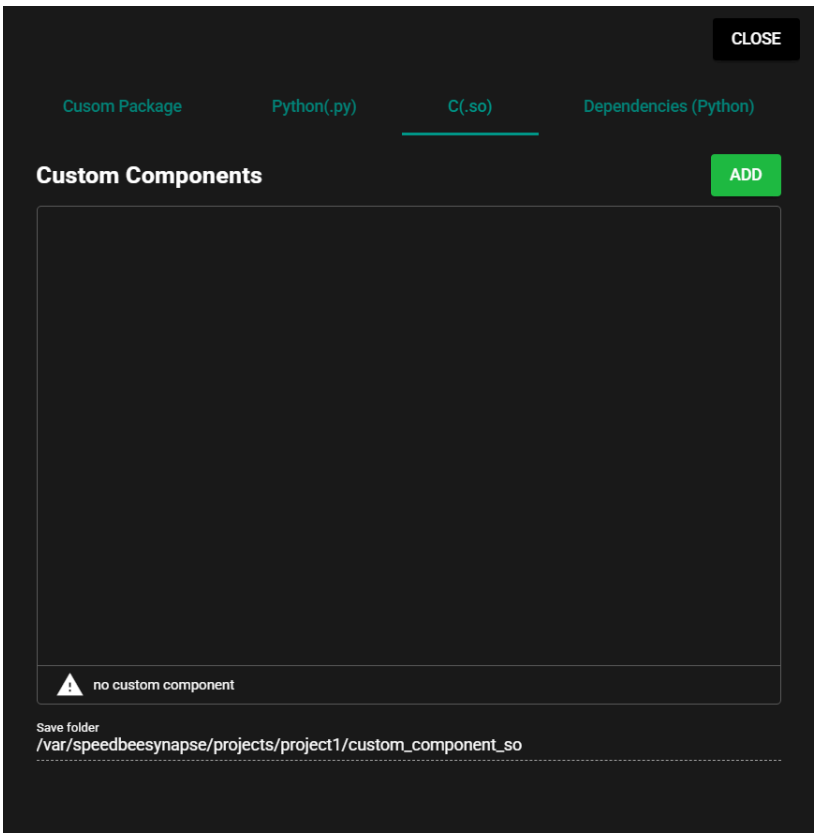
6.2.4.6.4 Download

You can download a registered custom component as follows:

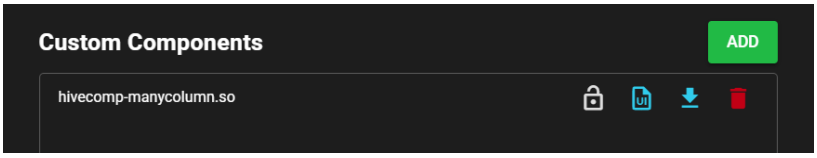
1. Press the Setting Menu icon and choose [Custom Components].



2. Select [C(.so)] in the tab at the top of the dialog.



3. Press the download icon (Get) for the custom component.



4. The custom component file is downloaded.

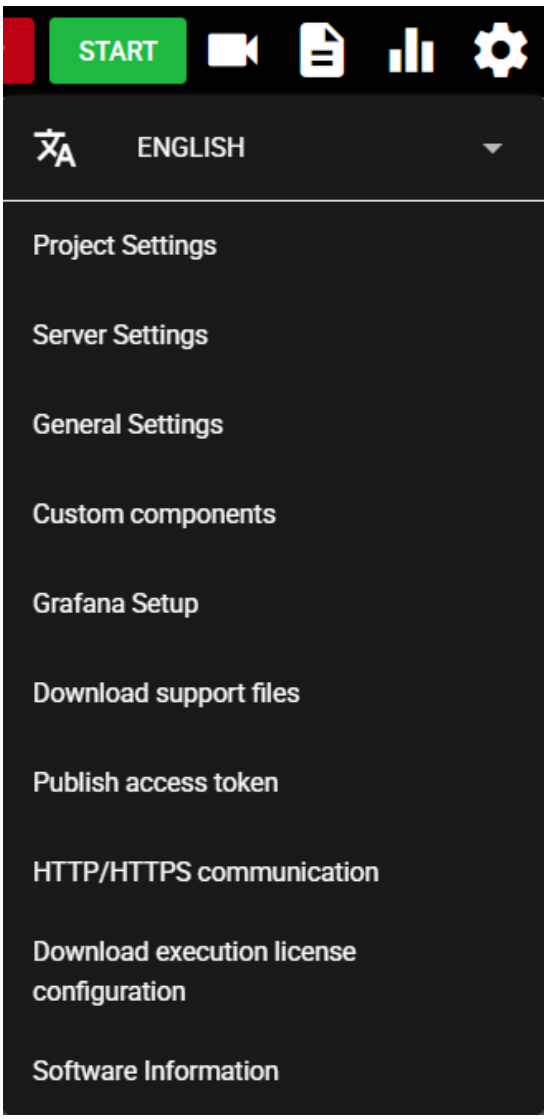
6.2.4.6.5 Protect

Registered components can be protected.
Protected components are not subject to the following operations and therefore are protected from unintended updates:

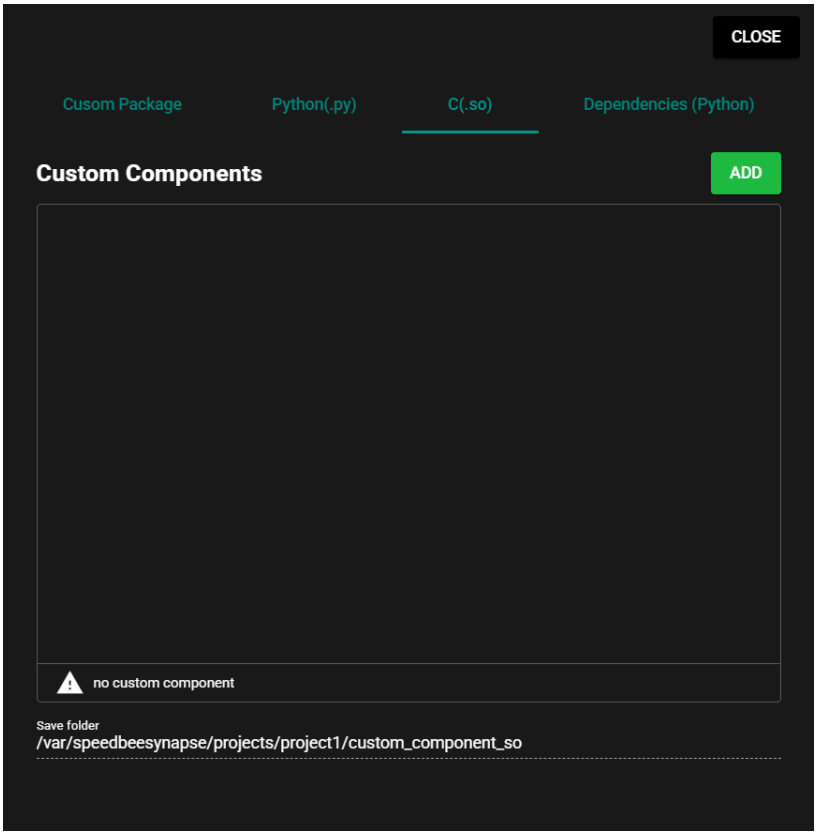
- Register
- Delete
- Download
- Set Custom UI

You can protect a custom component using the following procedure.

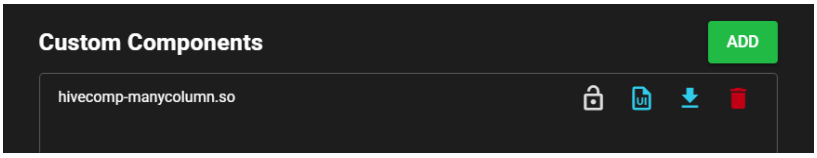
1. Press the Setting Menu icon and choose [Custom Components].



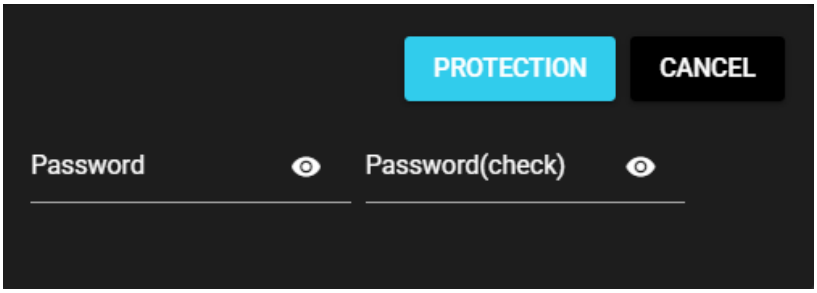
2. Select [C(.so)] in the tab at the top of the dialog.



3. Press the key icon (Protect) for the custom component.



4. Enter the password required to remove protection and press [Protection].

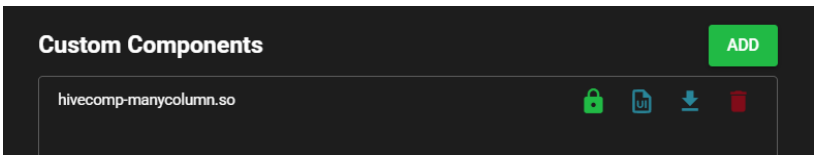


Warning

- The password is stored in a undecodable format and cannot be reset until protection is removed.
- If you forget the password for protection, the protected custom component becomes inoperable.

Be careful when handling passwords.

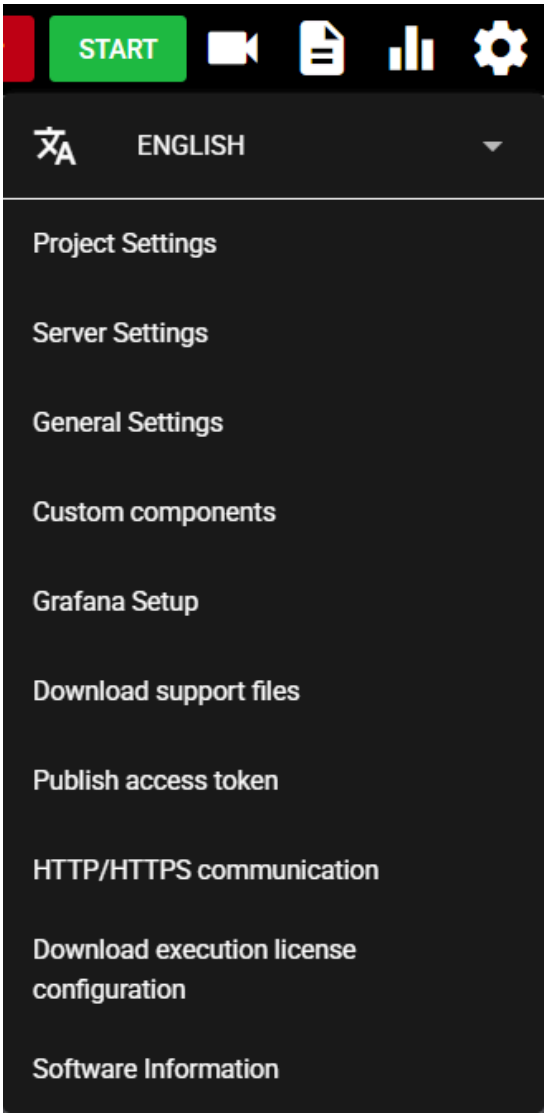
5. The custom component is now protected.



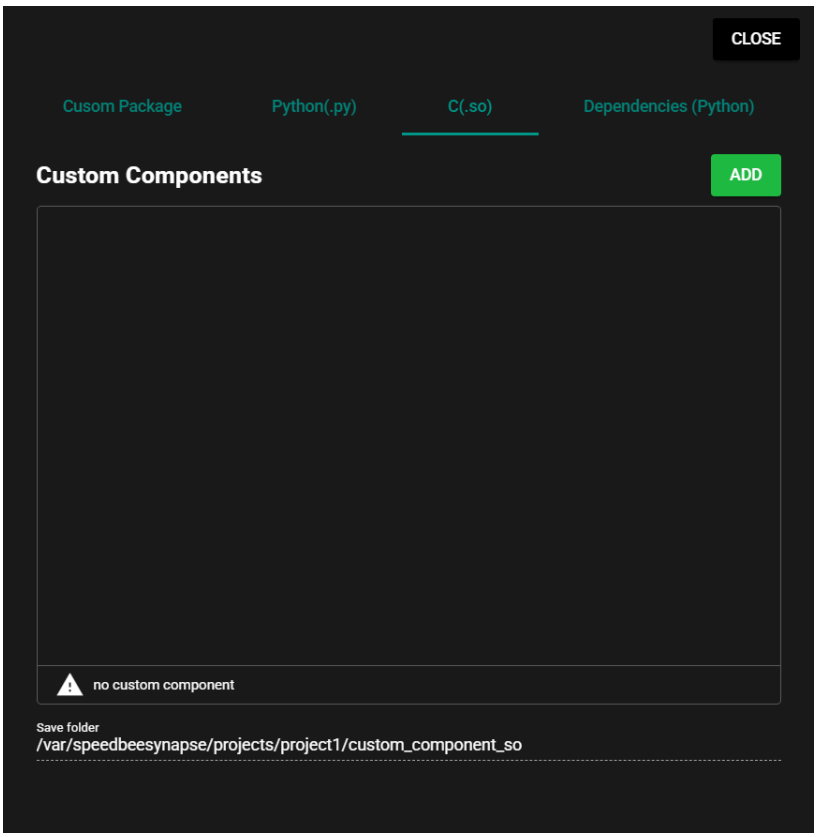
6.2.4.6.6 Unprotect

Follow the procedure below to remove the protection from a protected custom component:

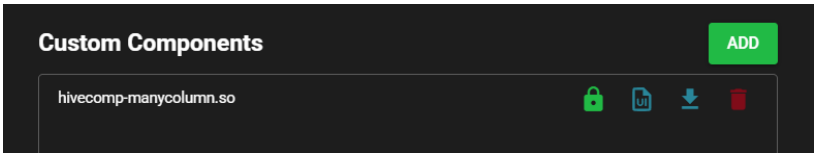
1. Press the Setting Menu icon and choose [Custom Components].



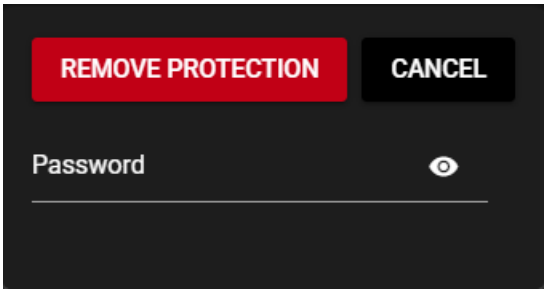
2. Select [C(.so)] in the tab at the top of the dialog.



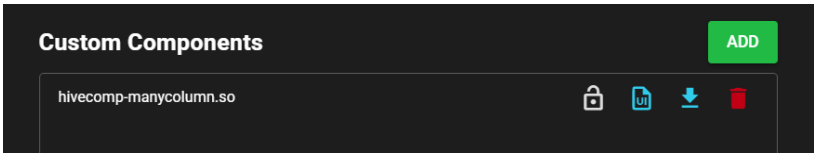
3. Press the key icon (Remove protection) for the custom component.



4. Enter the password set when applying protection and press [Remove protection].

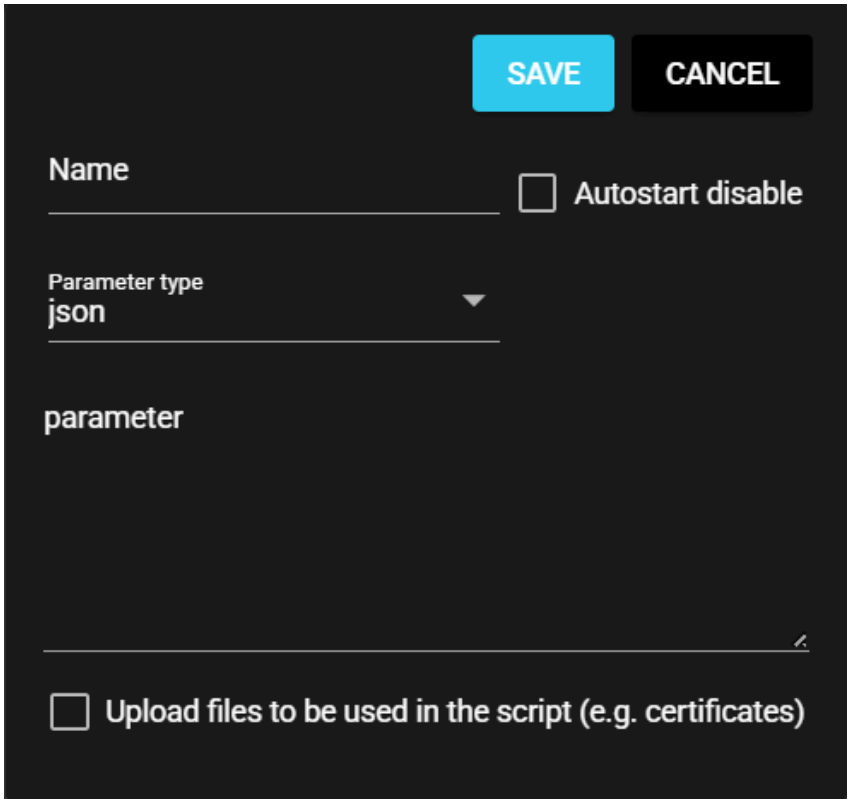


5. The protection is removed from the custom component.



6.2.5 Creating custom UI

The custom component settings screen is designed to allow users to freely enter JSON or string as parameter by default. However, this means that in cases where you are not the author of the program, you have to look at the source code of the custom component in order to find out what items are set as parameters, which requires extra time and effort.



To avoid this problem, with this product you can customize the parameter field on the custom component settings screen. You can define what items to place as parameter fields so as to make it easier for users to understand what parameters need to be entered.

6.2.5.1 Creating Custom UI

You can create a custom UI in the following ways. There are two forms of custom UI, each created in a different way:

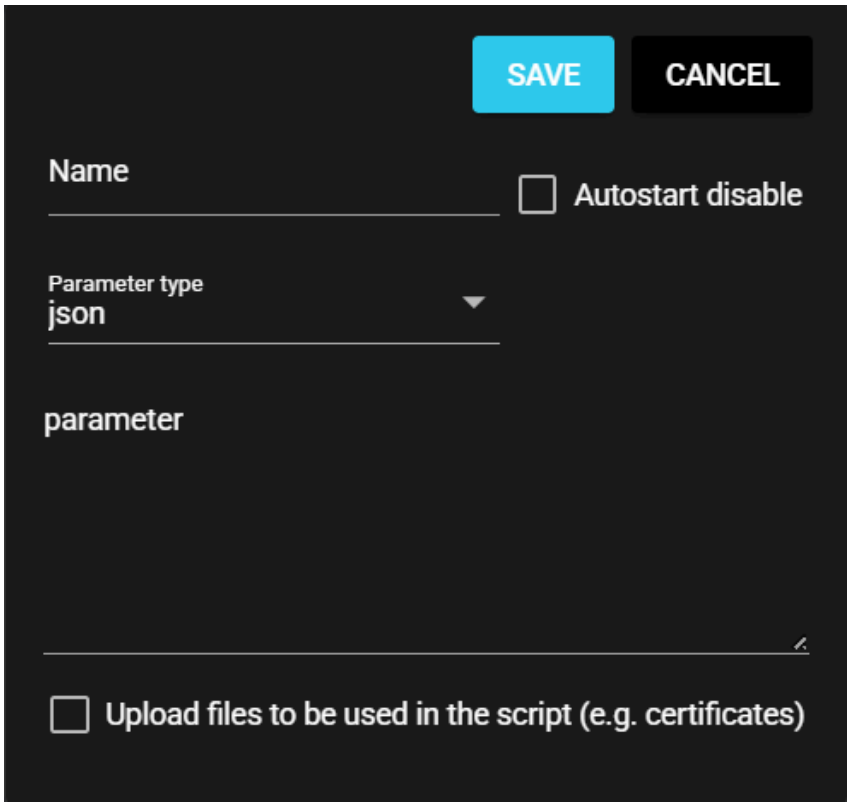
- JSON for Synapse custom UI

  In this method, the component settings are written in a JSON file. All you need to do is create a JSON file, and if you need to modify it, you can simply modify the JSON file. Instead, you can only define settings within the range of features provided by Synapse and cannot create complex UIs.

- HTML for Synapse custom UI

Use HTML and JavaScript to create component settings.
The HTML file can be created however the user wants, and complex UI can be implemented by utilizing JavaScript, CSS, images, etc. Furthermore, it supports a multilingual UI. However, basic knowledge of HTML and JavaScript is required to create HTML files.
Also, in a simple implementation, the look and feel will be different from the screens on Synapse.

Custom UI can only be created for the elements associated with the "parameters" within the red frame in the image; the elements outside cannot be edited.



- [Save] button

  Saves the component settings you entered.
- [Cancel] button

  Closes the Settings dialog.
- Name, Autostart disable

  This setting item is essential for running components.
- Upload files to be used in the script (e.g. certificates)

  Manages the files (such as certificates) required during component execution.

6.2.5.2 Creating JSON Files for Synapse Custom UI

A JSON file for custom UI can be created as follows:

6.2.5.2.1 Creating a JSON File

You can create a JSON file for custom UI by following the steps below:

1. Create an empty json file in a text editor.
2. Within the json file, define an array called "parameter_defines."
3. In "parameter_defines," define the input elements to be displayed on the custom component screen. Define other elements such as "window_size" and "layout" as needed.

Definition file (json)

```
{
  "window_size": "auto",
  "parameter_defines": [
    {
      // Input element 1
    },
    {
      // Input element 2
    },
    {
      // Input element 3
    }
    ...
  ]
}
```

The input elements displayed on the screen are defined using the following items:

Item	Value type	Required	Description
window_size	string	×	Window size { "window_size" : "small" } (example) ○ auto/No specification - min-width: "400px" ○ small - min-width: "33%" ○ middle - min-width: "50%" ○ large - min-width:"80%"
parameter_defines	object	○	Parent element of item (Nesting from the highest level to the next lower level can be specified. See sample)
name	string	○	Key to retain entered values in settings file
label	string	○	Item name displayed on the screen

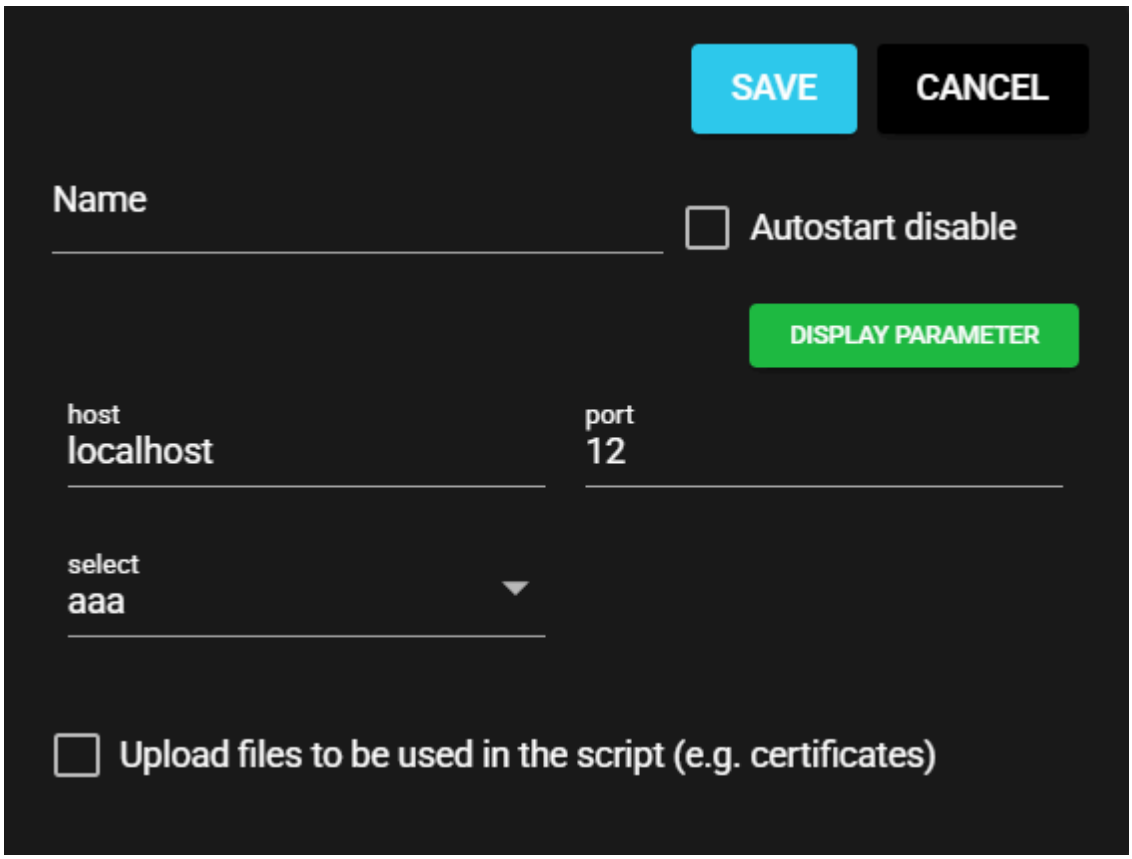
Item	Value type	Required	Description
type	string	○	Select the type of value you want to enter: string: String number: Integer real: Real number select: Option
max	number	○	Upper limit of input Note: Valid only if type is number or real.
min	number	×	Lower limit of input Note: Valid only if type is number or real.
select	array	×	Options Note 1: Valid only if type is select. Note 2: Options have "label" and "value" properties. Define by object array. Example: [{ "label":"option1", "value":1}, { "label":"option2", "value":2}]
default	any	×	Default value
require	bool	○	Whether input is required or not
array	bool	×	Use an array or not
layout	object	×	Specification of width of each item. See sample { "layout" : { "col" : 2 }} (This example specifies col as 1 to 12) When omitted, the default is 6. When array and object are omitted, the default is 12.

6.2.5.2.2 Sample

A sample custom UI definition file is shown below:

```
{
  "parameter_defines": [
    {
      "name": "host",
      "label": "Host",
      "type": "string",
      "default": "localhost",
      "require": true,
      "max":256
    },
    {
      "name": "port",
      "label": "Port",
      "type": "number",
      "max": 65535,
      "min": 1,
      "default": "12",
      "require": true
    },
    {
      "name": "select",
      "label": "Select",
      "type": "select",
      "select": [
        { "label":"aaa", "value":1},
        { "label":"bbb", "value":2}
      ],
      "default": 1,
      "require": false
    }
  ]
}
```

The resulting layout is as follows:

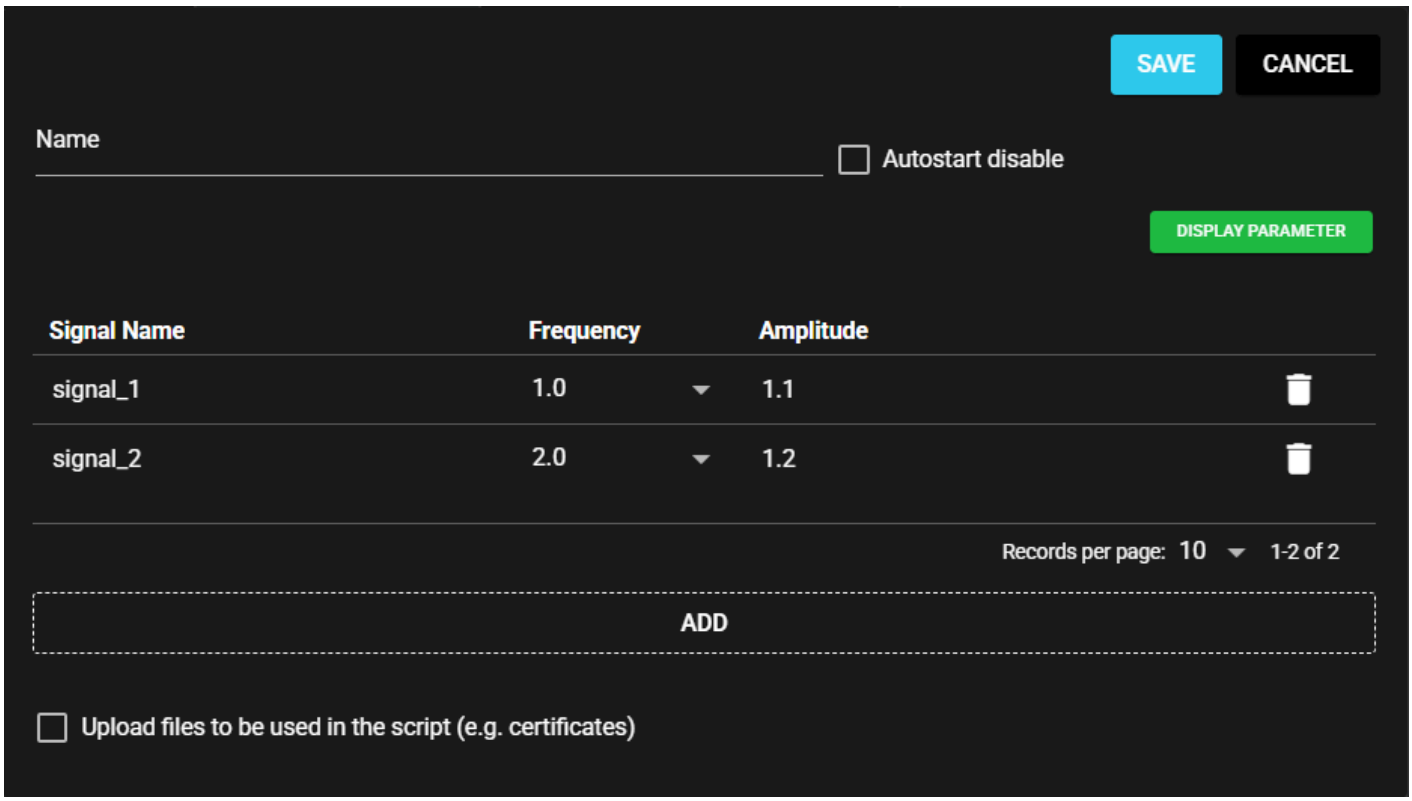


6.2.5.2.3 Sample with Nesting

A sample custom UI definition file using nesting is shown below:

```
{
  "window_size": "middle",
  "parameter_defines": [
    {
      "name": "nested",
      "label": "Nested",
      "type": "object",
      "array": true,
      "parameter_defines": [
        {
          "name": "name",
          "label": "Wave name",
          "type": "string",
          "require": true,
          "max": 32,
          "default": "signal_5"
        },
        {
          "name": "freq",
          "label": "Frequency",
          "type": "select",
          "require": true,
          "select": [
            { "label": "1.0", "value": 1.0 },
            { "label": "2.0", "value": 2.0 }
          ],
          "default": 1.0
        },
        {
          "name": "amp",
          "label": "Amplitude",
          "type": "real",
          "require": true,
          "max": 100.0,
          "min": 0.1,
          "default": 1.0
        }
      ],
      "require": false,
      "default": [
        {
          "name": "signal_1",
          "freq": 1,
          "amp": 1.1
        },
        {
          "name": "signal_2",
          "freq": 2,
          "amp": 1.2
        }
      ]
    }
  ]
}
```

The resulting layout is as follows:

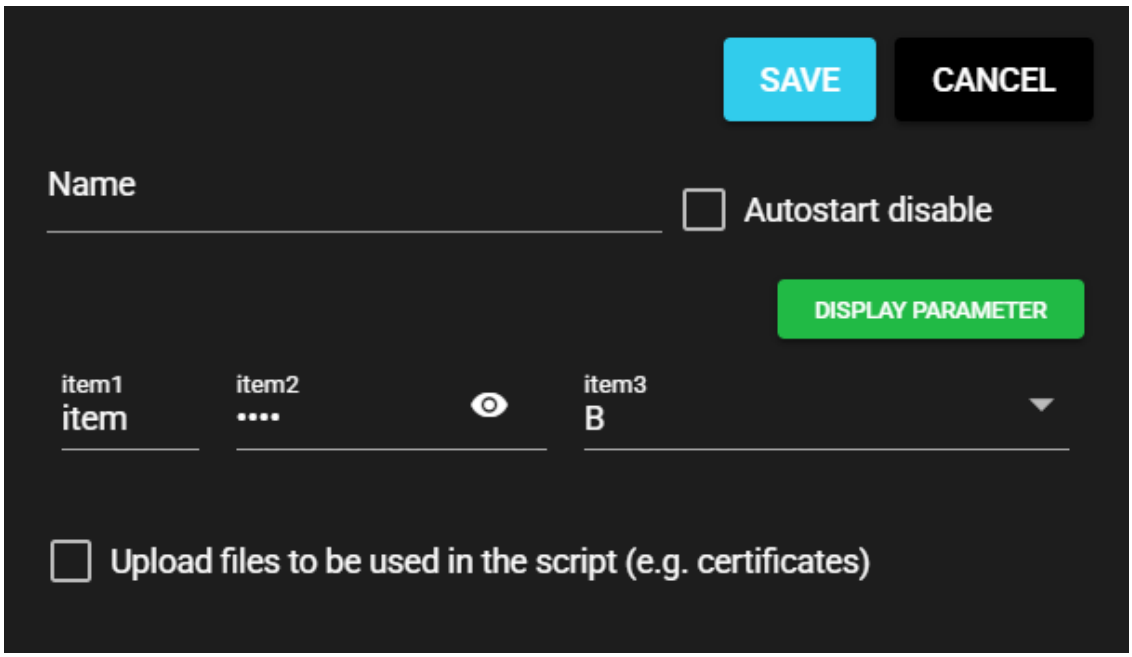


6.2.5.2.4 Sample Specifying Item Width

A sample custom UI definition file that specifies item width is shown below:

```
{
  "parameter_defines": [
    {
      "name": "item1",
      "label": "Item 1",
      "type": "string",
      "require": false,
      "max": 32,
      "default": "Komoku 1",
      "layout": {
        "col": 2
      }
    },
    {
      "name": "item2",
      "label": "Item 2",
      "type": "real",
      "require": false,
      "max": 32,
      "default": "12.3",
      "mask": true,
      "layout": {
        "col": 4
      }
    },
    {
      "name": "item3",
      "label": "Item 3",
      "type": "select",
      "require": false,
      "max": 32,
      "default": 2,
      "select": [
        { "label": "A", "value": 1},
        { "label": "B", "value": 2},
        { "label": "C", "value": 3}
      ],
      "layout": {
        "col": 6
      }
    }
  ]
}
```

The resulting layout is as follows:



6.2.5.3 Creating and Applying HTML for Synapse Custom UI

This section explains how to create a custom UI in HTML and JavaScript.

6.2.5.3.1 Creating Custom UI

The following environment is required to use custom UI in HTML format. Prepare this environment first.

- Custom Component Development Environment (SCCDE)

You can add a sample component including custom UI in HTML format by using one of the following commands:

```
$ sccde add -t collector -u html
$ sccde add -t emitter -u html
```

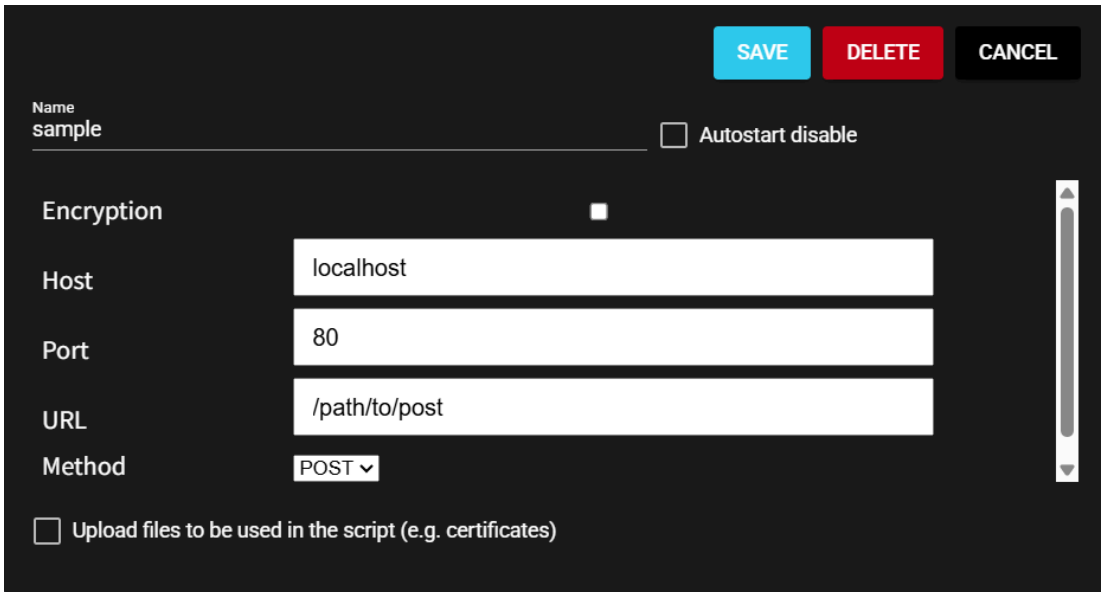
The actual HTML files are placed in the component-specific directories under `parameter_ui`. You can customize the html, js, and css files based on the sample you created.

6.2.5.3.2 HTML format details

- HTML file (index.html)

  As with regular HTML, you can write definitions using HTML and BODY tags. The definitions in this file are displayed in place of the text entry area in the parameter settings dialog of the custom component.

Sample display



- JavaScript file (index.js)

  Certain functions need to be called when displaying a dialog or when saving parameters, so those functions must be defined in JavaScript. The JavaScript file should be read by the `<script>` tag in the HTML file.

  Details will be described later.

Implementation Methods for Multilingual UI

When implementing a multilingual UI, it must be created according to the following format.

- Static implementation method(file: HTML)
  File management costs will increase, but it will result in a simpler operational method.
  Create files named `index.{language}.html` and prepare separate HTML files for each language. For example, `{language}` is written as `ja` for Japanese and `en` for English. The contents of the file must be written in the language matching the `{language}` of filename.
- Dynamic implementation method(file: JavaScript)
  File management is simplified, but processing within functions is required.
  Dynamic multilingual switching is possible by using the `locale` parameter in the function that loads the custom UI to identify the current display language. For details, see the usage of the `onLoad()` function.

- scc-info.json file

  To communicate to the system that your custom component uses custom UI (HTML), you need the following definition for the component:

Property	Description
parameter-ui-type	Specify 'html'
parameter-ui-size	Specify the Parameters dialog screen size. Selectable options are 'small', 'middle', 'large' and 'auto' (default is 'middle').
parameter-ui	Specify the path to the directory containing HTML and JavaScript files.

6.2.5.3.3 JavaScript File Coding

The following section explains the JavaScript processing when implementing custom UI in HTML, using the sample `index.js` source code as an example.

To enable JavaScript to work with Synapse information, a dedicated API object is provided for use with Synapse. This API object can be referenced as 'window.customUiLib.'

By registering a function with a specific property name in this object, you can exchange data with the Synapse screen.

```
async function onLoad(editable, param, isNewComponent, componentId, locale) {
  :
}
async function onSave() {
  :
}

// Registration of callback function
window.customUiLib.onSave = onSave;
window.customUiLib.onLoad = onLoad;
 :
```

There is also an API that can retrieve Synapse information.

For more information about these APIs, see Custom UI JS-API Reference.

6.2.5.3.4 Function Executed When the Parameters Dialog Appears

When the custom component's parameter settings dialog is displayed, the `onLoad()` function registered in the custom UI JavaScript is called. The following information is passed as arguments to this function, which should be reflected as elements in html as needed:

- Whether dialog can be edited

In Run Mode, `false` is passed to the `editable` argument to make the dialog input items not editable. See Controlling Edit Restrictions for more information.

- Set parameters

  When editing a component parameter that is already created, the previously set parameter is passed to the `param` argument.

- Create or update

  The value passed to the `isNewComponent` argument is `true` when creating a new component and is `false` if the component is being updated.

- Component ID

  The ID of the component is passed to the `componentId` argument. This information is passed as `undefined when you create a new component.

- Locale settings

  The code of the currently selected language, such as `ja` or `en`, is passed to the `locale` argument.

6.2.5.3.5 Function Executed When Saving Parameters Dialog

When the **Save** button is clicked in the custom component's parameter settings dialog, the `onSave()` function registered in the custom UI JavaScript is called. Return from this function the parameter to be saved.

6.2.5.3.6 Controlling Edit Restrictions

Synapse has two screens: **Run Mode** and **Edit Mode**. In Run Mode you can operate the data collection flow, and in Edit Mode you can make settings for the data collection flow.

Custom UI defined in standard components or JSON files are restricted so that they cannot be edited in Run Mode. However, custom UI implemented in HTML do not have this restriction. Instead, use the `editable` argument of the `onLoad()` function to receive information on whether the custom UI is editable or not to implement user restriction on editing.

Basically, deactivate the custom UI by setting the HTML element "disable" to true.

```
async function onLoad(editable, param, isNewComponent, componentId, locale) {
  // Deactivate 'host' item in form if "editable" is false
  document.forms['form']['host'].disabled = !editable;
}
window.customUiLib.onLoad = onLoad;
```

Reference: HTML attributes: disabled

6.2.5.3.7 Sample Using npm

Synapse uses `Vue` and `Quaser` to create screens.
While there may be some awkwardness in the screens made in plain HTML or CSS, by using `npm`, you can create UI using external packages such as `Vue` and `Quaser`. The samples used in the explanation so far only use plain HTML, Javascript, and CSS, but samples that use Vue and Quaser are also provided.

Samples using `Vue` and `Quaser`

Once the samples are downloaded, run the following command to generate html and js files under the `dist` directory: These files can be placed and used in the SCCDE environment, just like regular html.

```
$ ls
sample_custom_ui_vue.zip
$ unzip sample_custom_ui_vue.zip
$ cd sample_custom_ui_vue
$ npm install
$ npm run build
$ ls dist
assets  index.html
```

Using npm

`Node.js` must be installed in order to use `npm`. For more information about Node.js, including how to install it, visit the official Node.js site.
Official Node.js site: https://nodejs.org/en

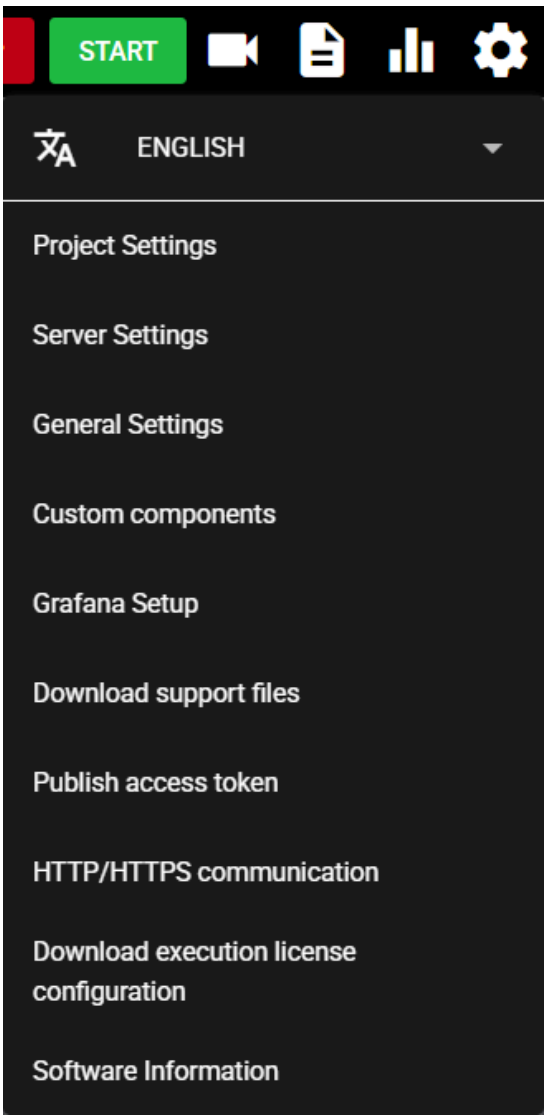
6.2.5.4 Applying Custom UI (JSON) to Python Custom Component

You can set custom UI in a Python custom component by registering a definition file created in Creating JSON File for Synapse Custom UI.

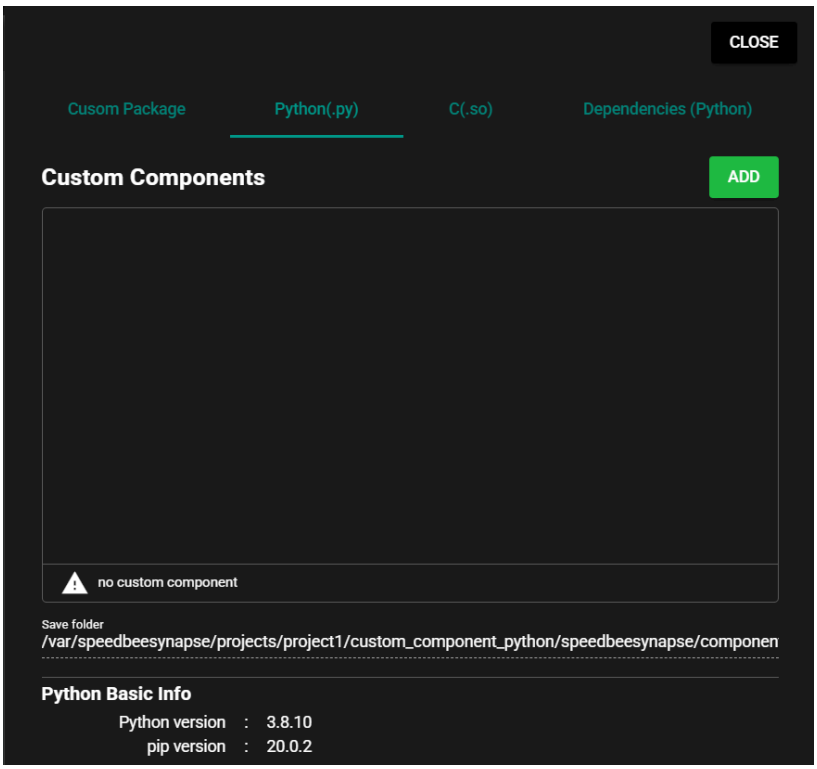
6.2.5.4.1 Register

You can register custom UI by following these steps:

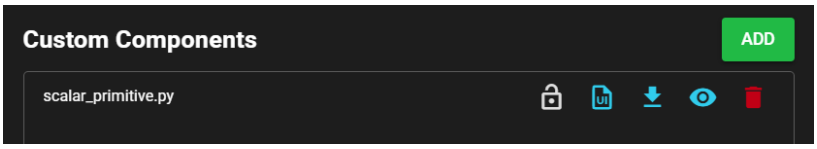
1. Press the Setting Menu icon and choose [Custom Components].



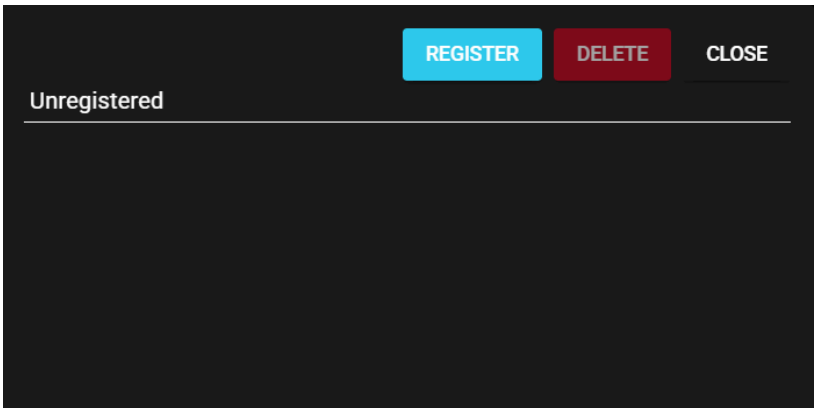
2. Select [Python(.py)] in the tab at the top of the dialog.



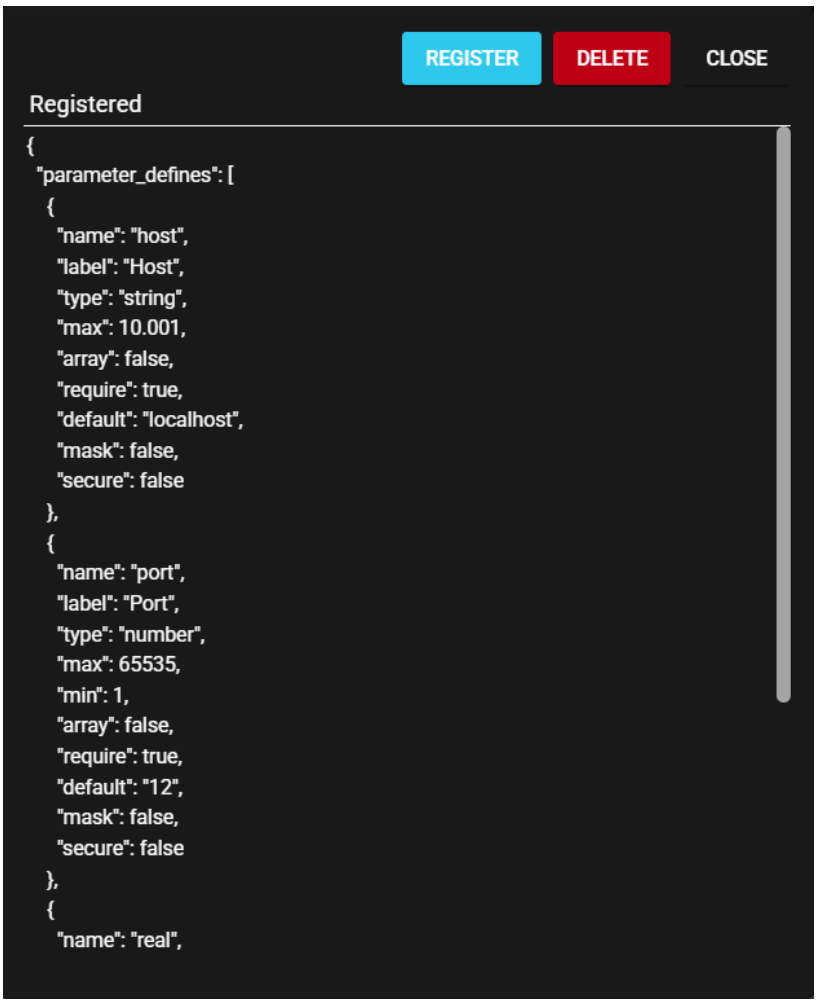
3. Press the UI icon (Set UI) for any custom component.



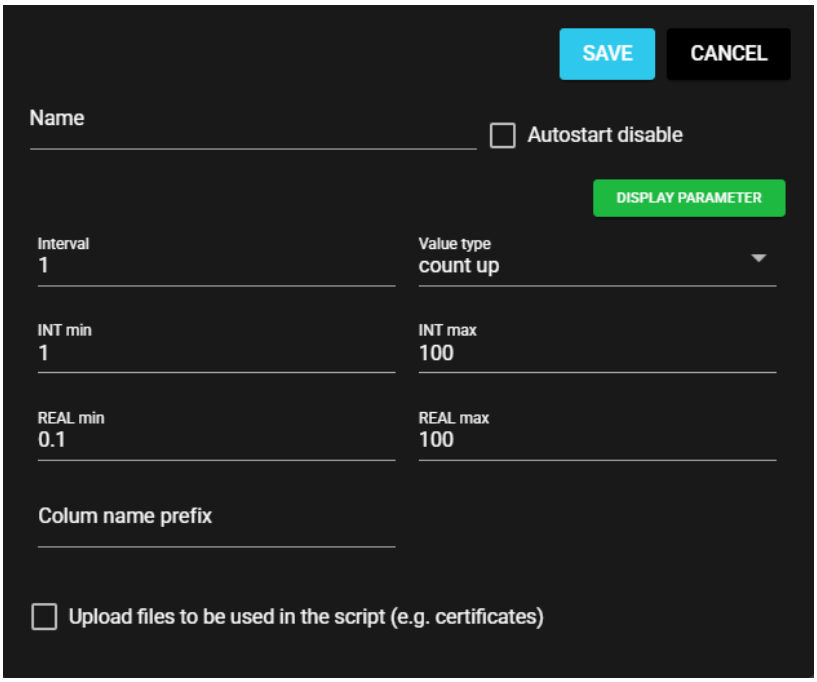
4. Press [Register] to register the definition file.



5. The wording "Registered" and the contents of the definition file are displayed.

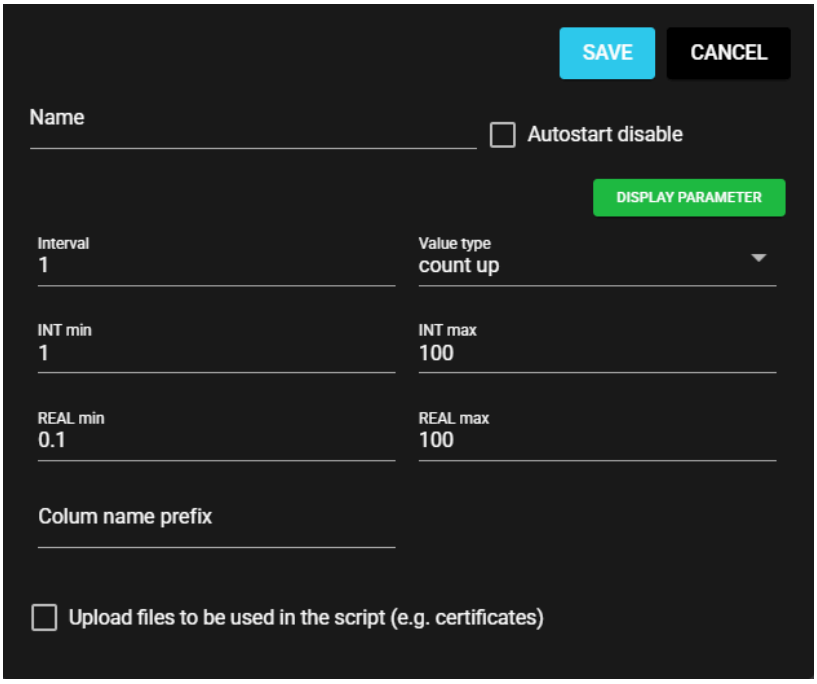


6. The items defined in the file are displayed in the settings screen of the custom component for which the definition file is registered.



Display parameter

1. Press [Display parameter].



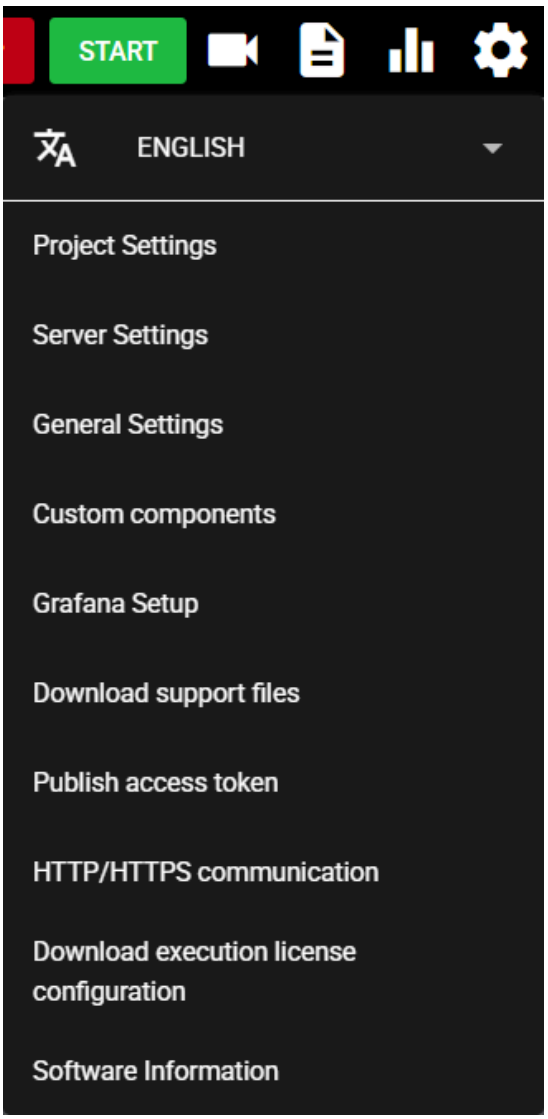
2. You can view the runtime parameters of the component that are generated from what you enter in the custom UI.



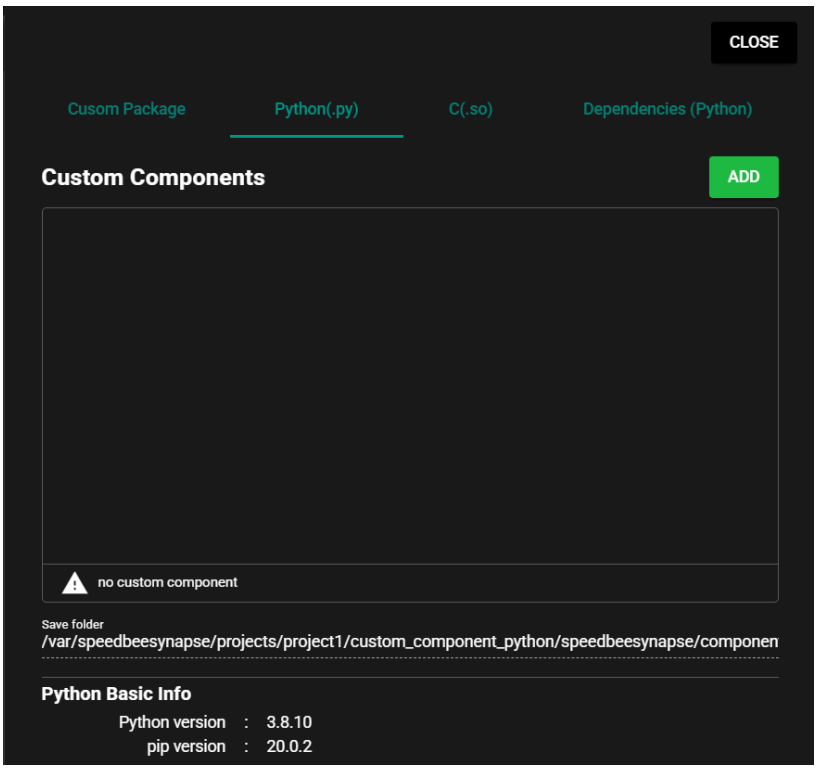
6.2.5.4.2 Delete

You can use the following procedure to delete the registered definition file and return the screen to its default state.

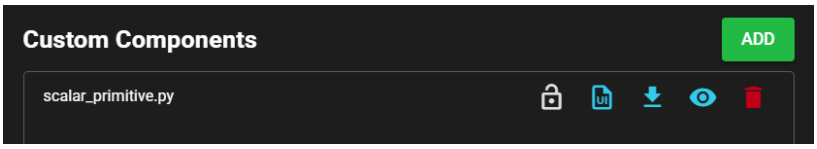
1. Press the Setting Menu icon and choose [Custom Components].



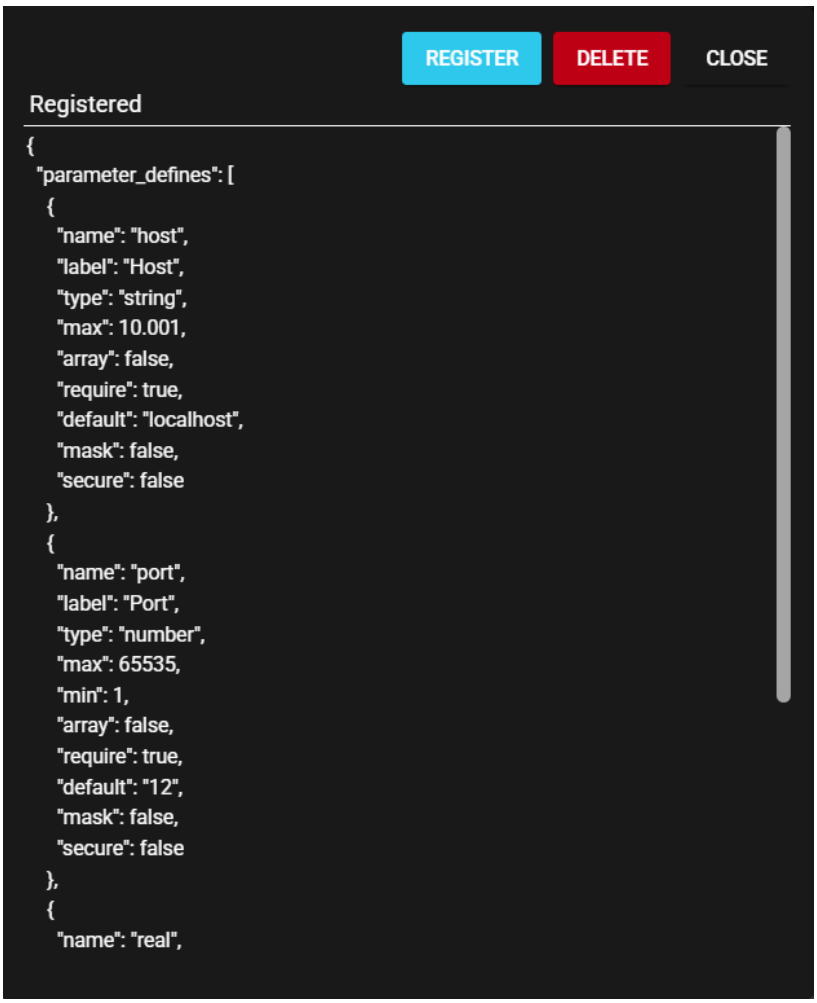
2. Select [Python(.py)] in the tab at the top of the dialog.



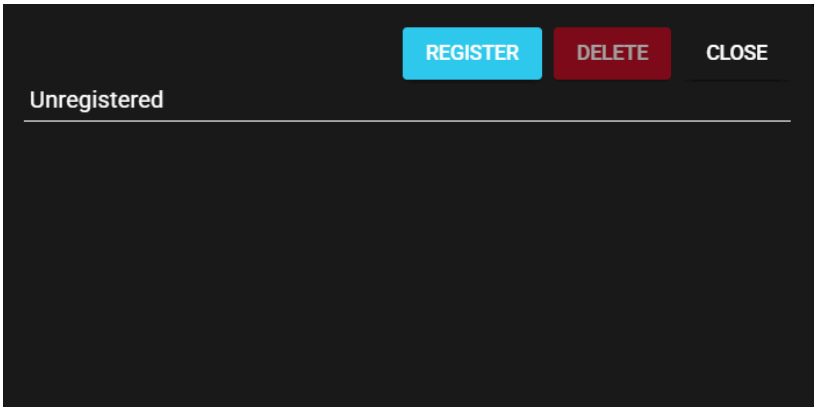
3. Press the UI icon (Set UI) for the custom component for which the definition file is registered.



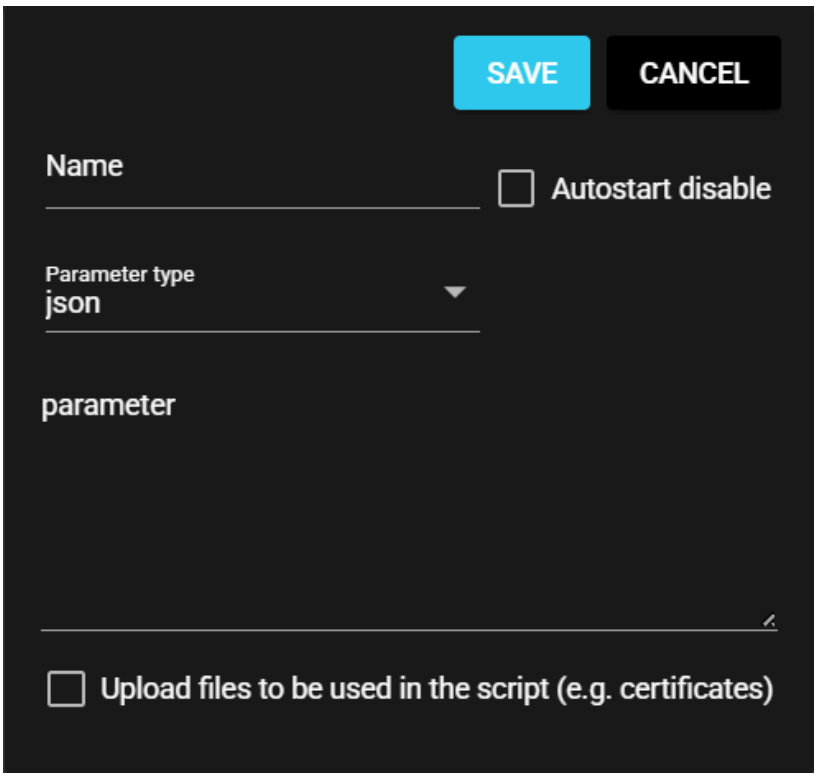
4. Press [Delete].



5. The registered definition file is deleted, and "Unregistered" is displayed.



6. The custom component settings screen returns to its default state.



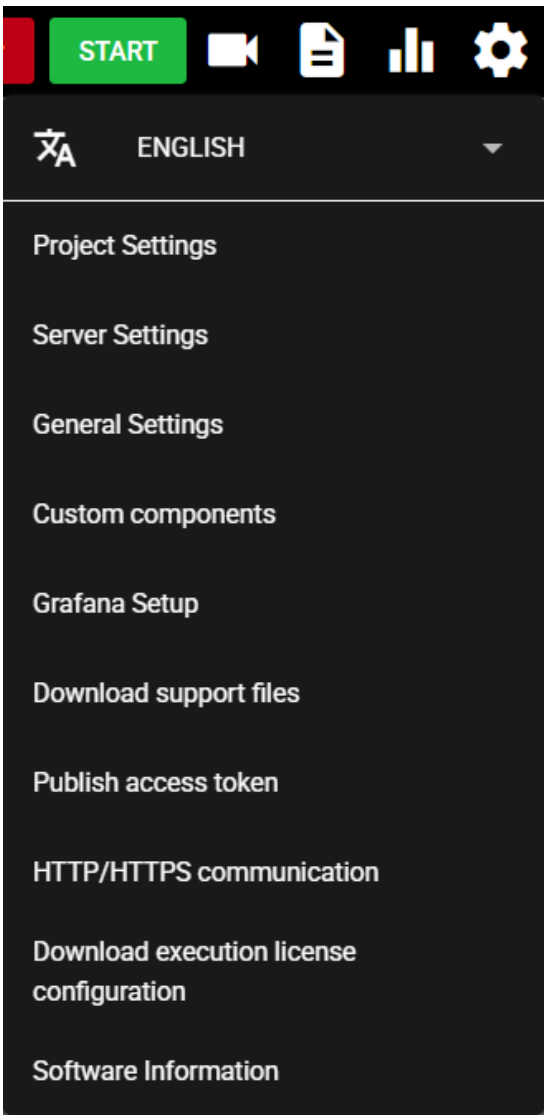
6.2.5.5 Applying Custom UI (JSON) to C Custom Component

You can set custom UI in a C custom component by registering a definition file created in Creating JSON File for Synapse Custom UI.

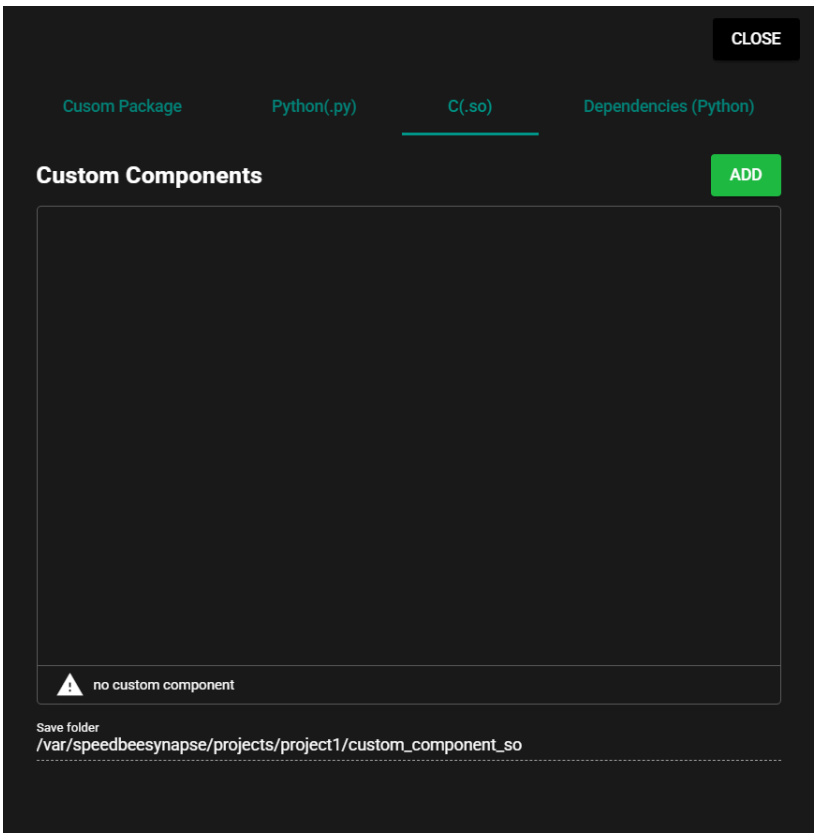
6.2.5.5.1 Register

You can register custom UI by following these steps:

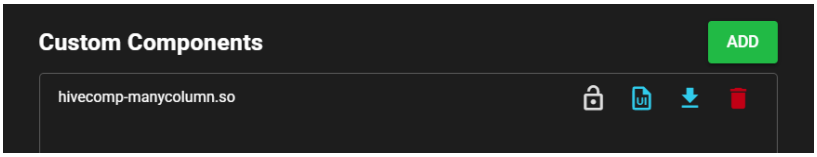
1. Press the Setting Menu icon and choose [Custom Components].



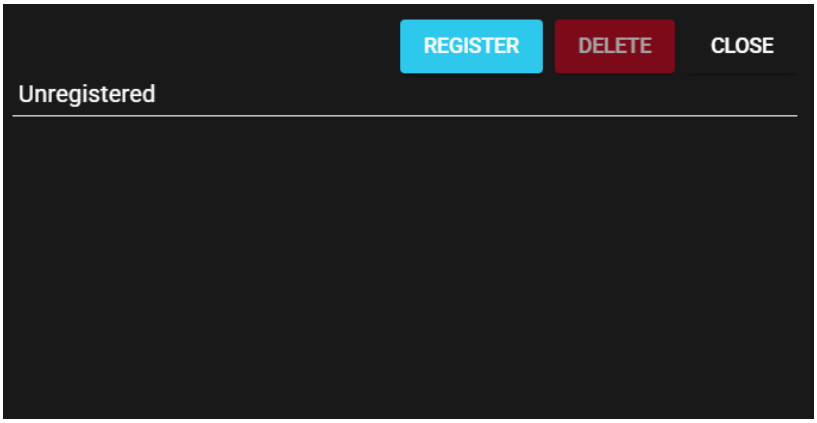
2. Select [C(.so)] in the tab at the top of the dialog.



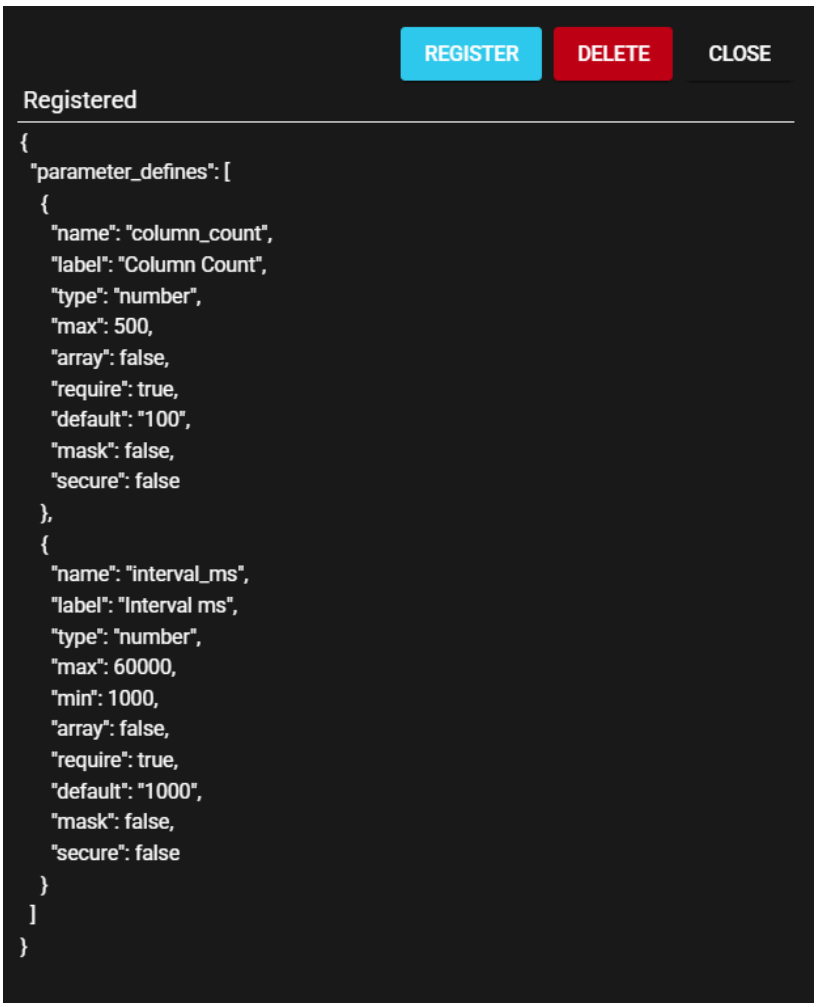
3. Press the UI icon (Set UI) for any custom component.



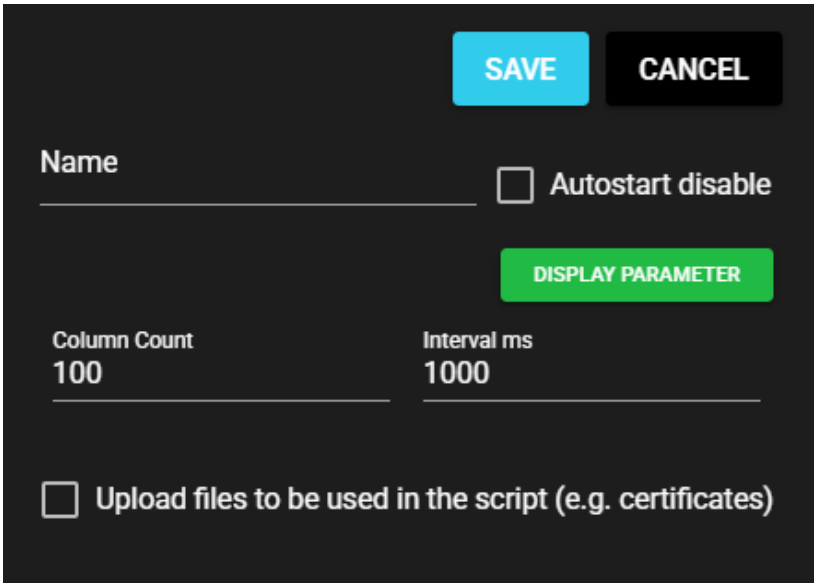
4. Press [Register] to register the definition file.



5. The wording "Registered" and the contents of the definition file are displayed.

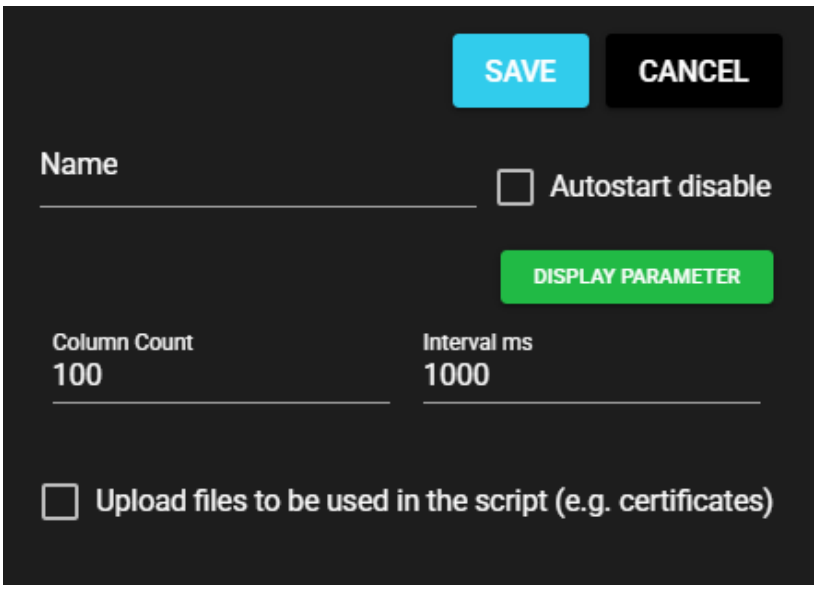


6. The items defined in the file are displayed in the settings screen of the custom component for which the definition file is registered.

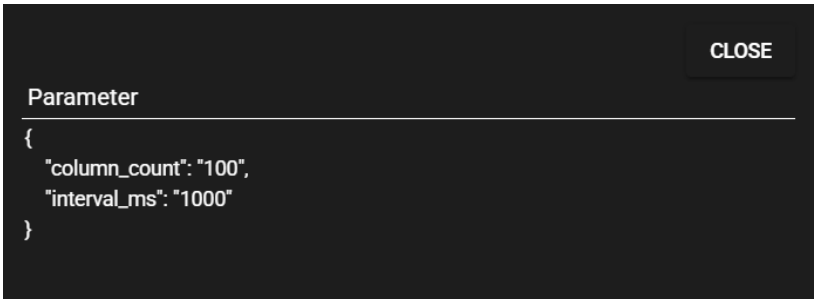


Display parameter

1. Press [Display parameter].



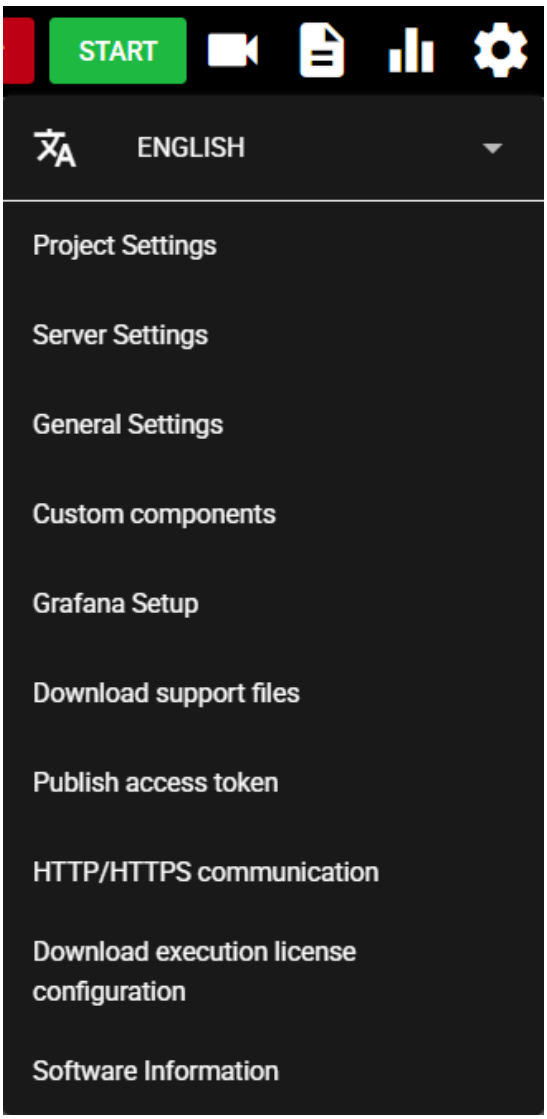
2. You can view the runtime parameters of the component that are generated from what you enter in the custom UI.



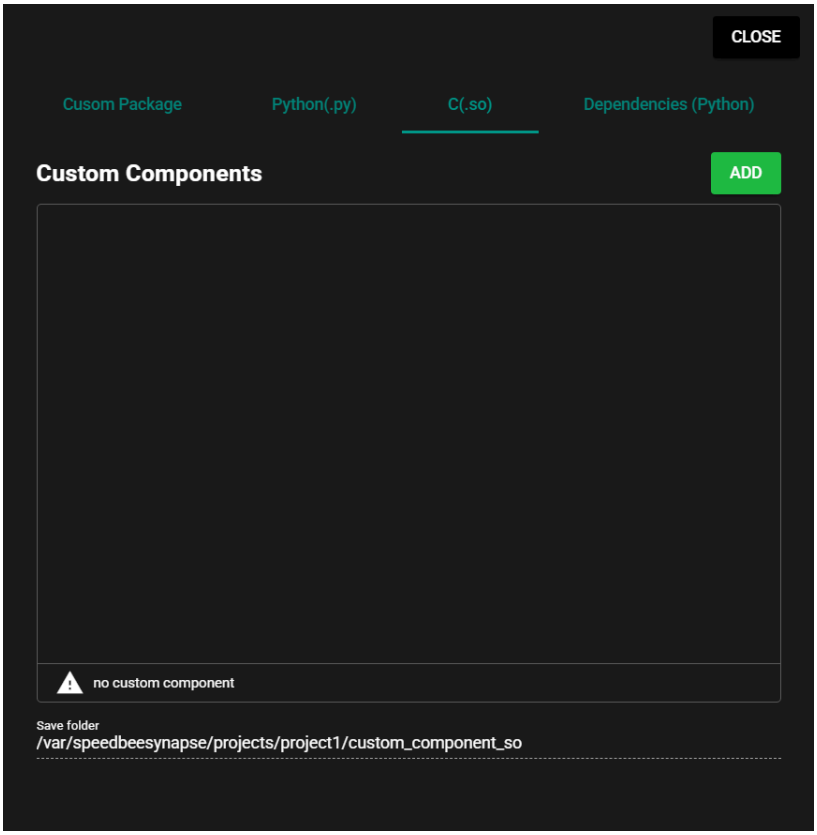
6.2.5.5.2 Delete

You can use the following procedure to delete the registered definition file and return the screen to its default state.

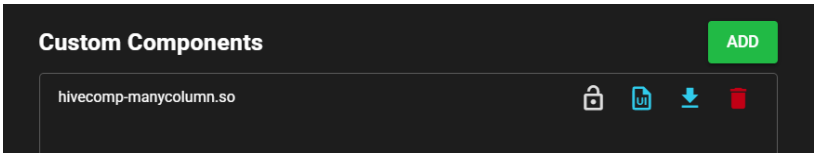
1. Press the Setting Menu icon and choose [Custom Components].



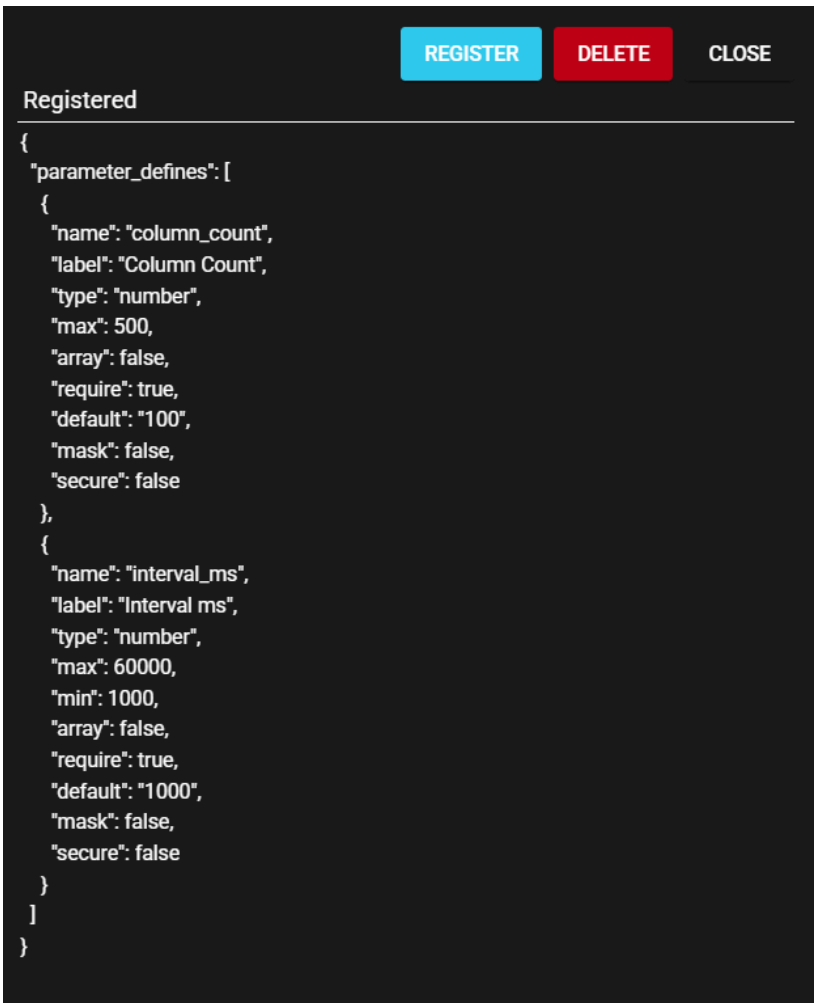
2. Select [C(.so)] in the tab at the top of the dialog.



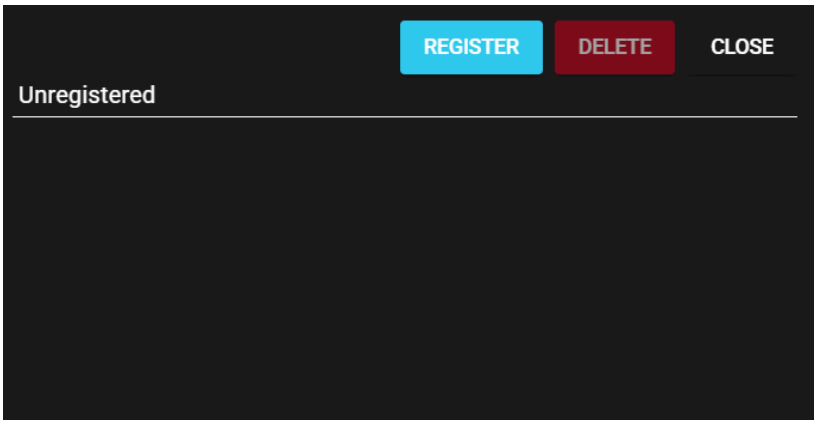
3. Press the UI icon (Set UI) for the custom component for which the definition file is registered.



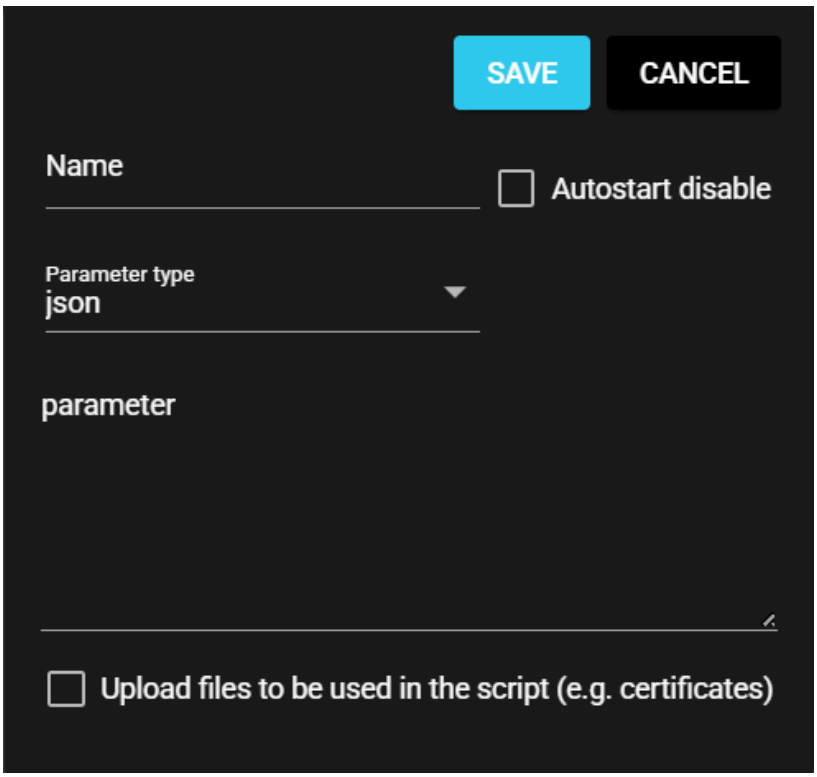
4. Press [Delete].



5. The registered definition file is deleted, and "Unregistered" is displayed.



6. The custom component settings screen returns to its default state.


SAVE
CANCEL
Name
Autostart disable
Parameter type
json
parameter
Upload files to be used in the script (e.g. certificates)

6.2.6 Custom component development environment (SCCDE)

This section explains the Synapse custom component development environment (SCCDE).

6.2.6.1 Overview of SCCDE

The Synapse custom component development environment (SCCDE) is a collection of tools and packages for developing custom components for Synapse. It currently has the following features:

- Initialization of custom component administration directory (init)

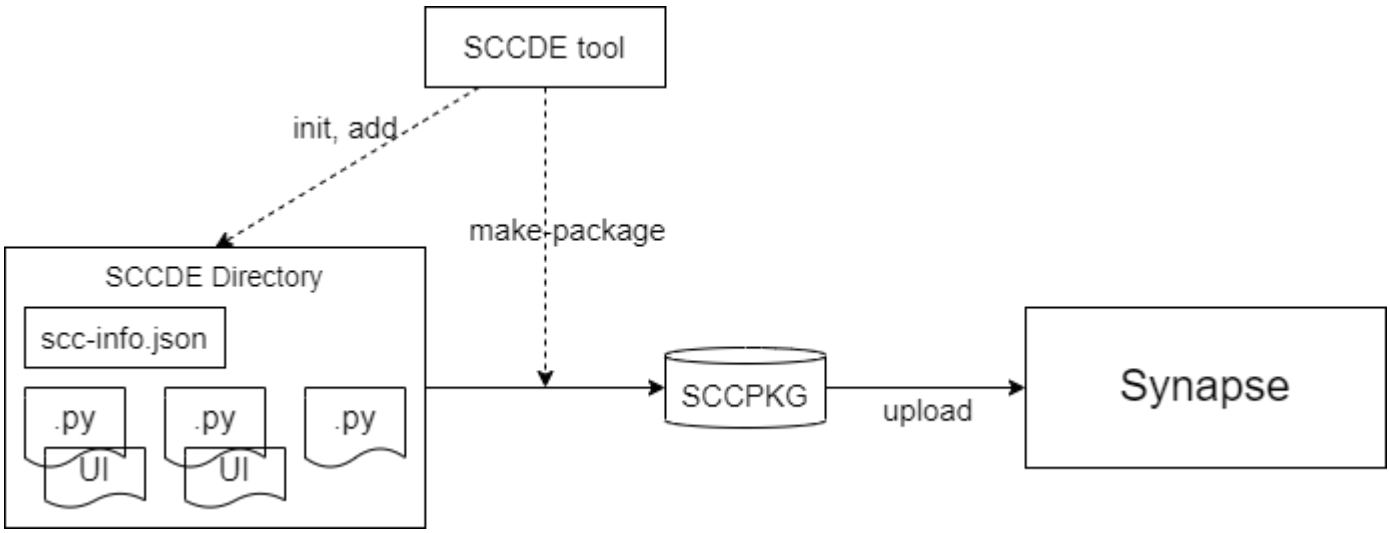
  The administration directory is easy to use with version control systems such as Git and Subversion.
- Addition of sample programs for custom components (add)

  You can add simple sample programs such as collectors, serializers, and emitters.
- Packaging of custom components (make-package)

  Component packaging allows you to combine multiple custom components and custom UIs into a single file (SCCPKG).

  By uploading the SCCPKG file from the Synapse administration screen, you can collectively manage multiple custom components.
- Execution of test environment for custom components (serve)

  You can launch a test environment to quickly run your custom components.



Currently, only custom components implemented in Python are supported. Support for components implemented in C is not planned.

6.2.6.2 Uninstalling SCCDE

SCCDE can be installed in the following ways:

- Installation using the WHL package file distributed with Synapse
- Installatrion from PyPI (pip standard index)

You can install SCCDE on Windows or Linux. However, Synapse must be installed on the same device to perform the test environment execution (serve) function.

Linux

```
$ python3 -m venv ~/sccde
$ . ~/sccde/bin/activate
(sccde) $ pip3 install speedbeesynapse-sccde
```

Reference: venv

Windows(pwsh)

```
PS> python -m venv "$HOME\sccde"
PS> . "$HOME\sccde\Scripts\Activate.ps1"
(sccde) PS> pip3 install speedbeesynapse-sccde
```

Reference: venv

Since this package is a standard Python package, various package management tools, including Poetry and uv, can be used in addition to the installation methods using pip and venv as described above.

Use the tool of your choice.

Note: The following procedure assumes that you are using the `(sccde)` environment that you activated as above.

6.2.6.3 Using SCCDE and Packaging

6.2.6.3.1 Initializing the Custom Component Administration Directory

You must provide and initialize an empty directory to manage your custom components. Follow the procedure below to initialize the directory.

Linux

```
(sccde) $ mkdir sccpack-test
(sccde) $ cd sccpack-test
(sccde) $ sccde init
(sccde) $ ls
parameter_ui  source  scc-info.json
```

Windows(pwsh)

```
(sccde) PS> mkdir sccpack-test
(sccde) PS> cd sccpack-test
(sccde) PS> sccde init
(sccde) PS> ls
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          2025/xx/yy    hh:mm                parameter_ui
d----          2025/xx/yy    hh:mm                source
-a---          2025/xx/yy    hh:mm            644 scc-info.json
```

Initialization of the current directory involves the following operations:

- Generation of `scc-info.json`
- Addition of sample program for custom component (collector)

The `scc-info.json` file contains information about the custom components to be managed in this directory. It is in JSON format, so edit it in a text editor as necessary.
The file supports multilingual properties that enable display of multiple languages in the custom component dialog and component base tooltips.

The major properties are listed below.

Property	Description
package-name	Name of the package. You can give it any name. Since this is a required property, make sure to include it in the scc-info.json file. If no valid value is entered, nothing will be displayed.
package-name.{language code}	Multilingual property for `package-name`. When adding this property, please be careful not to remove the required property `package-name`. Enter supported language codes by Synapse (e.g., `ja` or `en`) in `{language code}`, and the display will switch according to the language settings. Therefore, the property value must be written in the language corresponding to `{language code}`. If no valid value is entered, the value of `package-name` will be displayed; if `package-name` also has no valid value, nothing will be shown.

Property	Description
package-version	A string that indicates the version of the package. As a basic rule, specify three numbers delimited by "."(period). The version string should conform to Semantic Versioning 2.0.0.
package-uuid	ID used to identify the package. A random UUID is set by default. Since this is a required property, make sure to include it in the scc-info.json file. As a basic rule, you do not need to change this value.
package-description	A description of the package. Enter any character string. If no valid value is entered, nothing will be displayed.
package-description.{language code}	Multilingual property for `package-description`. Add this when you want to display package descriptions in multiple languages. Enter supported language codes by Synapse (e.g., `ja` or `en`) in `{language code}`, and the display will switch according to the language settings. Therefore, the property value must be written in the language corresponding to `{language code}`. If no valid value is entered, the value of `package-description` will be displayed; if `package-description` also has no valid value, nothing will be shown.
author	You can include the name of the package creator.
license	You can enter the license type for using this package. (BSD, MIT, UNLICENSED, etc.). If it is not necessary, leave this as an empty string.
license-file	You can specify a relative path to the file that contains the license details for this package. If it is not necessary, leave this as an empty string.
python-components-source	Deployment path for Python custom components to be managed in this directory. This generally does not need to be changed.
components	An object for defining multiple custom components. The key must be the UUID of the custom component.
components.*.name	Name of custom components. You can give it any name.
components.*.description	Description of custom components. Displayed in component base tooltips. If no valid value is entered, the default text will be displayed.
components.*.description.{language code}	Multilingual property for `components.\*.description`. Displayed in component base tooltips.

Property	Description
	Add this when you want to display custom component descriptions in multiple languages. Enter supported language codes by Synapse (e.g., `ja` or `en`) in `{language code}`, and the display will switch according to the language settings. Therefore, the property value must be written in the language corresponding to `{language code}`. If no valid value is entered, the value of `components.\*.description` will be displayed; if `components.\*.description` also has no valid value, "This component was created and registered by the user." will be displayed.
components.*.component-type	Implementation type of custom components. It must be either 'python' or 'c'.
components.*.modulename	Module name of Python custom components. Enter the name of module for importing custom components relative to the path specified by `python-components-source`.
components.*.parameter-ui-type	Specify the type in the custom UI definition file. It can be either 'json' or 'html'.
components.*.parameter-ui	Specify a relative path to the custom UI. Specify the path to the file in the case of 'json' format, or the path to the directory where the html file is located in the case of 'html' format.
python-requirements	An array for describing Python dependency module definitions. Examples "python-requirements": ["sqlalchemy", "psycopg[binary]", "requests ==2.32.4", "aiohttp >=3,<4;sys_platform == 'win32'", "httpx ==0.28.1;sys_platform == 'linux' and python_version>='3.10'"] For detailed description format, see Dependency specifiers. *Note: URLs are not supported.

Multilingual Property Configuration Rules (scc-info.json)

- Required Properties
  The `package-name` property is required in the `scc-info.json` file. When adding the multilingual property `package-name.{language code}`, be careful not to accidentally delete `package-name`.

  Definition file example (scc-info.json)

  ```
  {
      "package-name": "example package name",       // Existing property (required)
      "package-name.ja": "カスタムパッケージ名",     // Multilingual property (Japanese)
      "package-name.en": "Custom package name",     // Multilingual property (English)
      ..
      .
  }
  ```

- Download File Names
  File names when downloading packages are always generated from `package-name`, regardless of language settings.
- Priority between multilingual and existing properties
  Multilingual properties are referenced with priority, and existing properties are referenced when multilingual support is not available or when no valid values are entered.

  1 Multilingual properties
  Examples: `package-name.ja`, `package-description.en` etc.
  2 Existing properties
  Examples: `package-name`, `package-description` etc.
- Invalid values
  When the corresponding values are entered, those properties become invalid. The behavior when invalid varies by property, so please check the description of each property.

  a. Empty strings
  Example: `"package-description": ""`

  b. Strings with only whitespace
  Example: `"package-description.ja": "  "`

  c. Only tabs and line breaks
  Example: `"package-description.en": "\t\n"`

6.2.6.3.2 Adding Custom Components

After you initialize the administration directory, you can add sample programs for custom components with the `add` subcommand.

Linux

```
(sccde) $ sccde add -t collector
```

Windows(pwsh)

```
(sccde) PS> sccde add -t collector
```

The string after `-t` can be replaced with `collector`, `serializer`, `emitter`, etc.

The added sample programs are initially placed under `source/python.` These samples are meant to be run out of the box, but you can use them as a basis to create your own custom components.

For more information on how to implement a custom component, see below.

- Creating Custom Components in Python
- Custom Components Python-API Reference

The added sample components also come with custom UI sample. You can also modify this as needed.

- Creating Custom UI

Custom UI is in json format as default. You can also create a custom HTML-format UI by specifying the `-u html` option when adding.

6.2.6.3.3 Testing Custom Components

If Synapse is also installed in the same environment where you installed the SCCDE commands, you can test run the custom components with the `serve` command.

Linux

```
(sccde) $ sccde serve --port 8123
Preparing Synapse data directories..
Synapse server ready: http://127.0.0.1:8123
:
```

Windows(pwsh)

```
(sccde) PS> sccde serve --port 8123
Preparing Synapse data directories..
Synapse server ready: http://127.0.0.1:8123
:
```

The first time you run it, it will take some time to initialize the data directory and install dependent modules. `ready: http://..` message indicates that the URL is ready to be opened in a browser.

When you open the URL, a screen similar to the Edit Mode in the Synapse flow editor appears. You can now select the custom component you created from the component base list on the left.

You can stop the test Synapse with `Ctrl-C`.

Options

Options	Description
`--host`	You can specify an IP address for use by the test Synapse server. If you want to access it from outside the running device, specify `0.0.0.0`. The default is `127.0.0.1`.
`--port`	You can specify a TCP port number for use by the test Synapse server. Specify a port number so that it does not conflict with another service. The default is `8000`.
`--verbose`	This displays the test Synapse execution log.

Data Directory and Logs

Data running on the test Synapse is placed in the `.synapse/var-speedbeesynapse` directory in the sccde working directory.

Linux

```
(sccde) $ ls .synapse/var-speedbeesynapse/
certs  config  dblog  hivecore_exec_status.json  logs  projects  sdts.db  sdts.file  webui_tmp
```

Windows(pwsh)

```
(sccde) PS> ls .\.synapse\var-speedbeesynapse\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          2025/xx/yy    hh:mm                config
d----          2025/xx/yy    hh:mm                dblog
d----          2025/xx/yy    hh:mm                logs
d----          2025/xx/yy    hh:mm                projects
d----          2025/xx/yy    hh:mm                sdts.db
d----          2025/xx/yy    hh:mm                sdts.file
d----          2025/xx/yy    hh:mm                webui_tmp
-a---          2025/xx/yy    hh:mm            105 hivecore_exec_status.json
```

If you want to clear the data and restore the default state, delete the entire `.synapse` folder and then start the test Synapse.

You can also look at `speedbeesynase.log` in `.synapse/var-speedbeesynapse/logs` if you want to view the log for debugging custom components.

Development Components

Dedicated development components are available while running the test Synapse. For more information about development components, see Development Components.

6.2.6.3.4 Packaging

After you add the custom component, you can create a package using the `make-package` subcommand.

Linux

```
(sccde) $ sccde make-package
(sccde) $ ls
custom_component_package_example.sccpkg  parameter_ui  scc-info.json  source
```

Windows(pwsh)

```
(sccde) PS> sccde make-package
(sccde) PS> ls
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          2025/xx/xx    hh:mm                parameter_ui
d----          2025/xx/xx    hh:mm                source
-a---          2025/xx/xx    hh:mm           2481 custom_component_package_example.sccpkg
-a---          2025/xx/xx    hh:mm            644 scc-info.json
```

The file with the `.sccpkg` extension is the SCCPKG. You can register this file in Synapse.

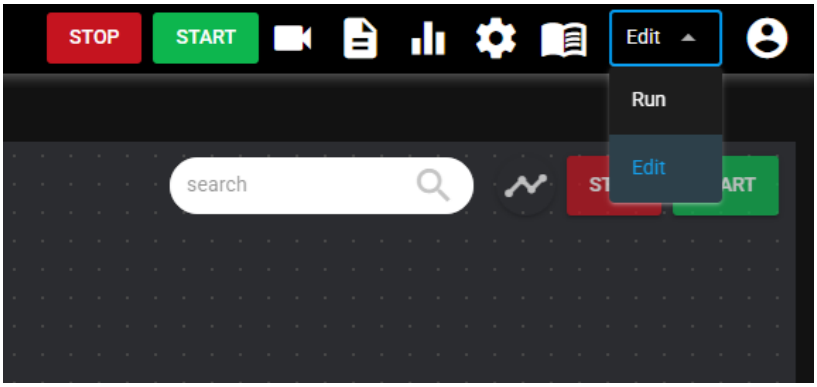
6.2.6.4 Registering SCCPKG File in and Deleting SCCPKG File from Synapse

You can use the custom components contained in the package by registering the package file (SCCPKG) created by packaging explained in the previous section from the administration screen of Synapse.

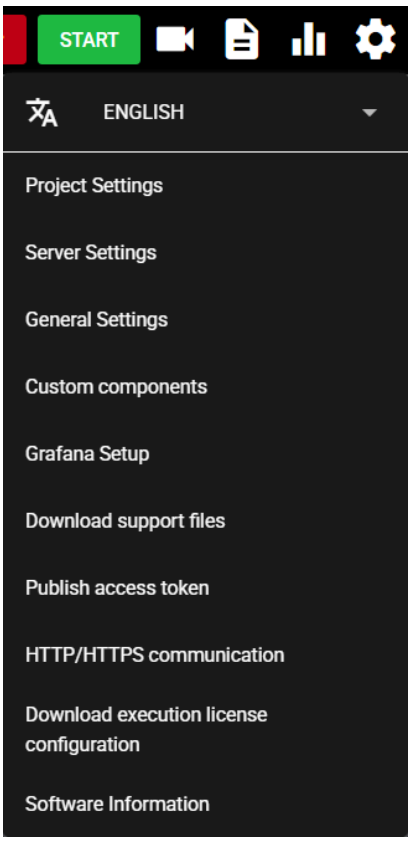
6.2.6.4.1 Register/Update

You can register a created SCCPKG file in SpeeDBee Synapse using the following procedure.

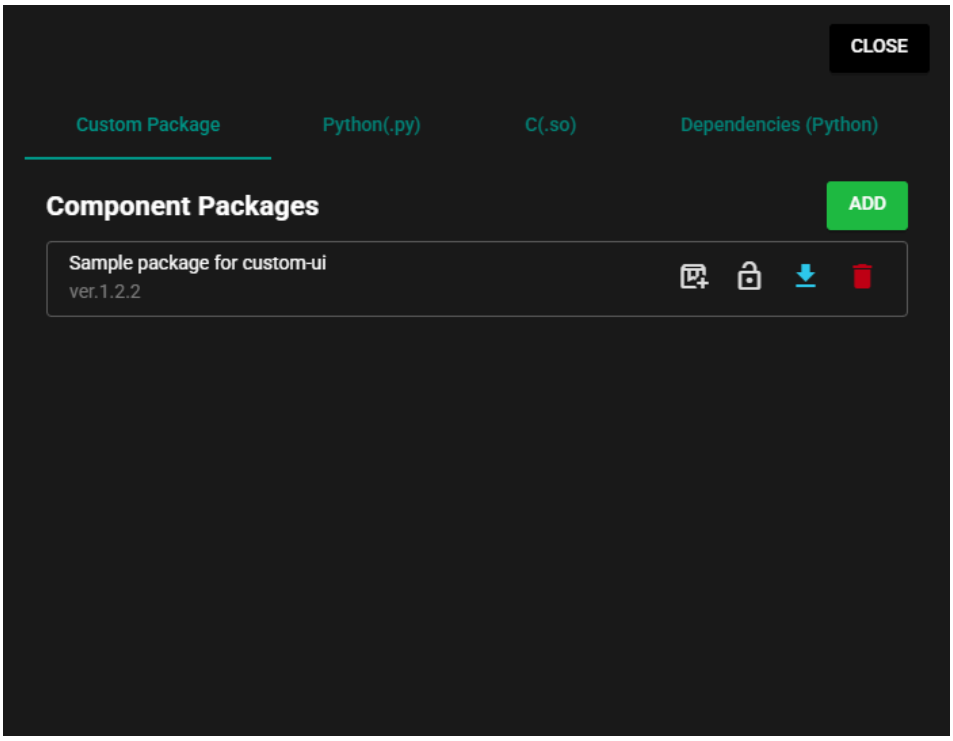
1. Select "Edit mode" in the header mode selection to open the Edit mode window.



2. Press the Setting Menu icon and choose [Custom Components].



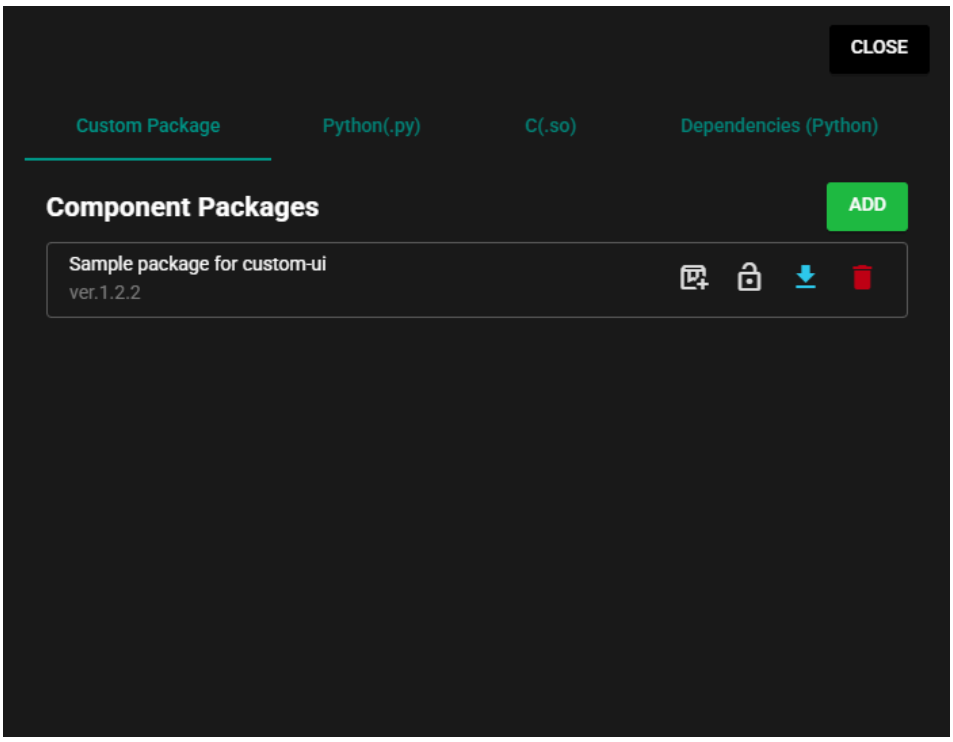
3. Press [Add] to register the custom component file.



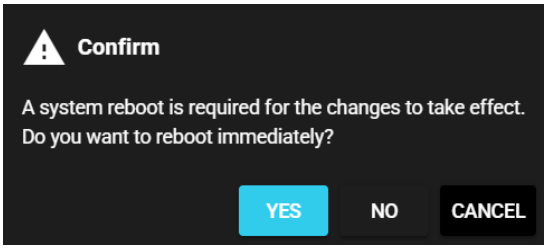
If the same package is already registered, a dialog appears asking you to confirm overwriting. Select [Yes] to overwrite.

If there are dependent modules, a dialog box will appear asking you to confirm installation. If you want to install them, select [Install].

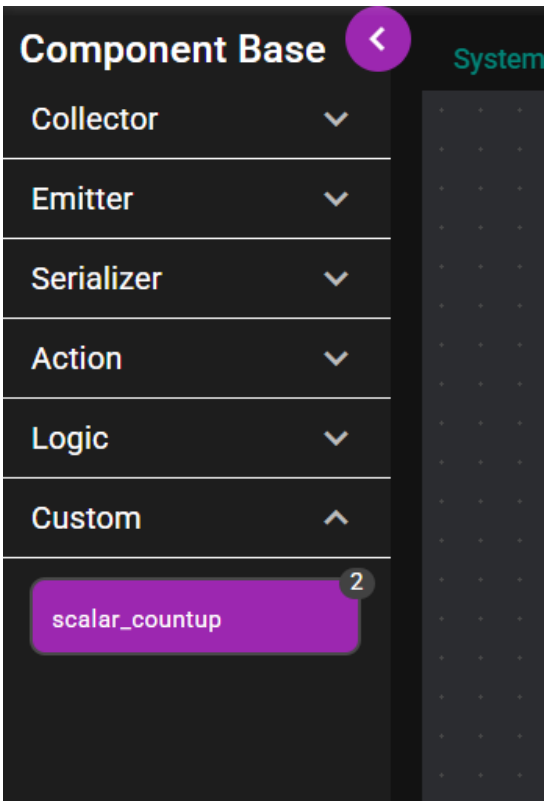
4. Press [Close].



5. Select [Yes] in the confirmation dialog to restart the system to apply the changes.



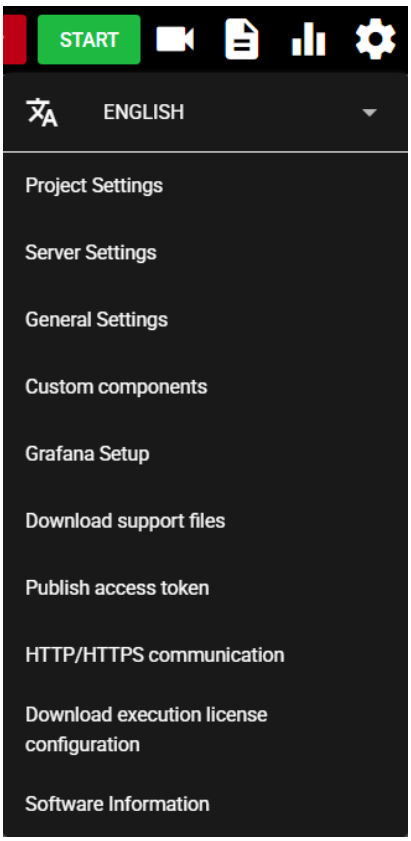
6. The left menu shows the registered custom component.



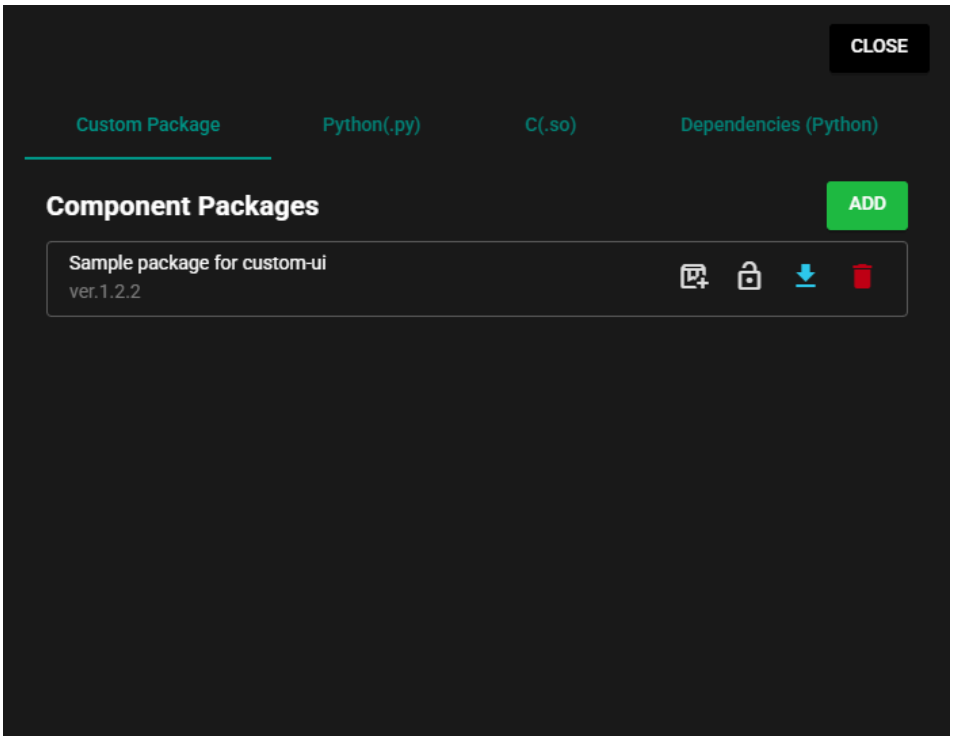
6.2.6.4.2 Dependent modules

You can install dependent modules as follows:

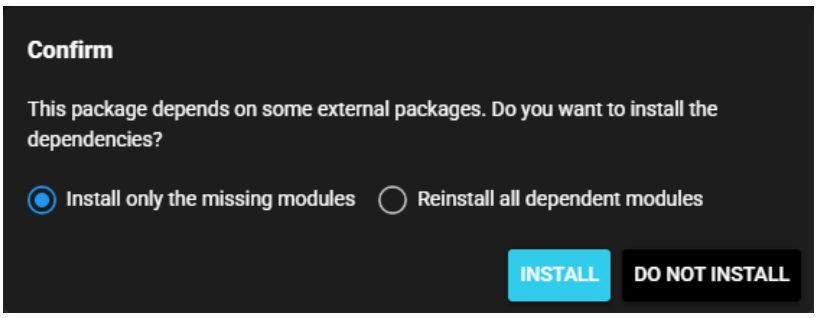
1. Press the Setting Menu icon and choose [Custom Components].



2. Press the download icon (Dependent modules) for the component package.



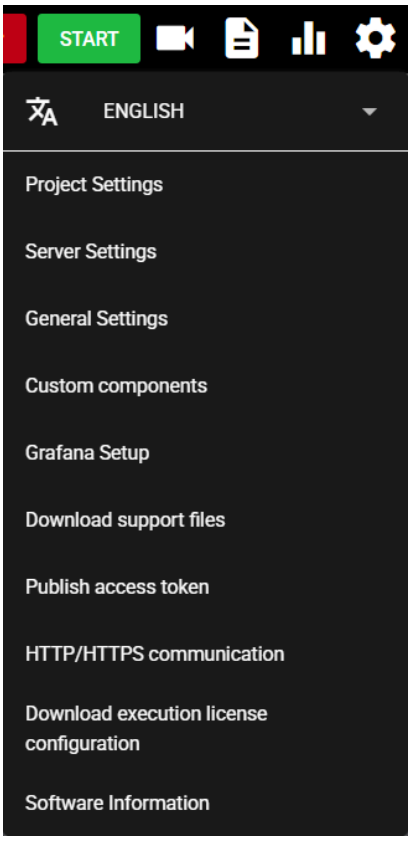
3. In the confirmation dialog, select [Install only the missing modules] or [Reinstall all dependent modules] then select [Install].



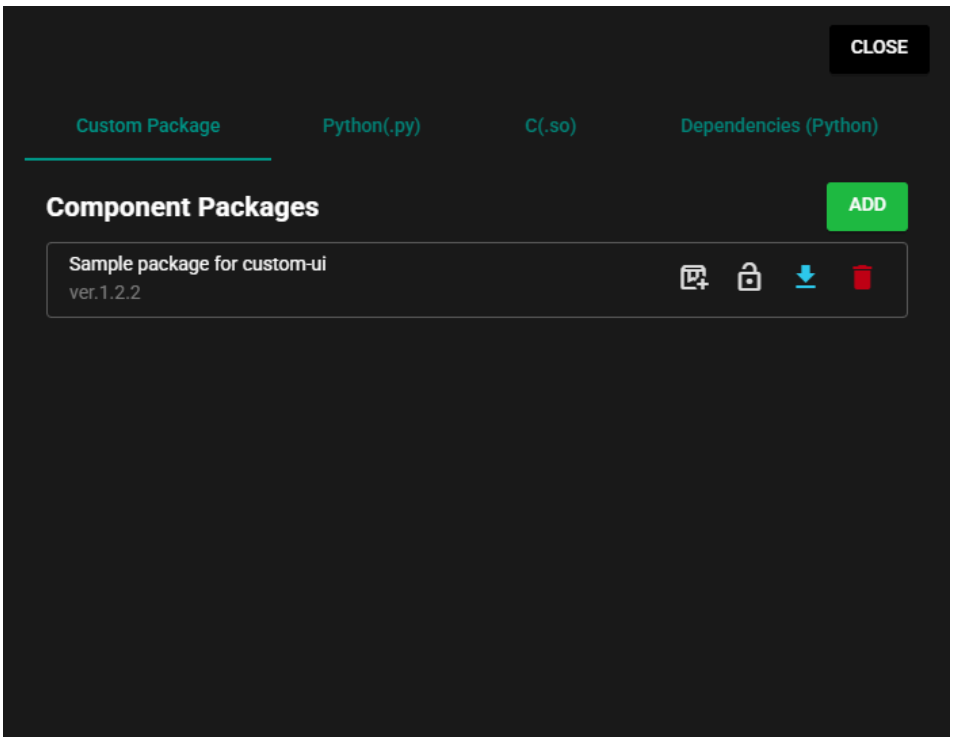
6.2.6.4.3 Download

You can download a registered component package as follows:

1. Press the Setting Menu icon and choose [Custom Components].



2. Press the download icon (Get) for the component package.

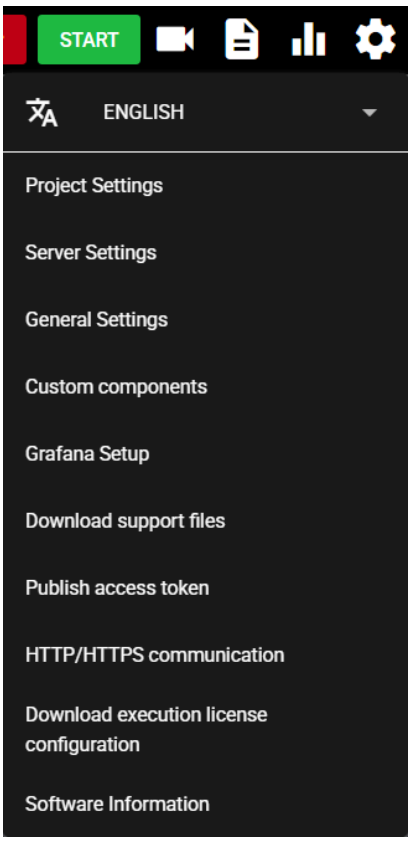


3. The component package file is downloaded.

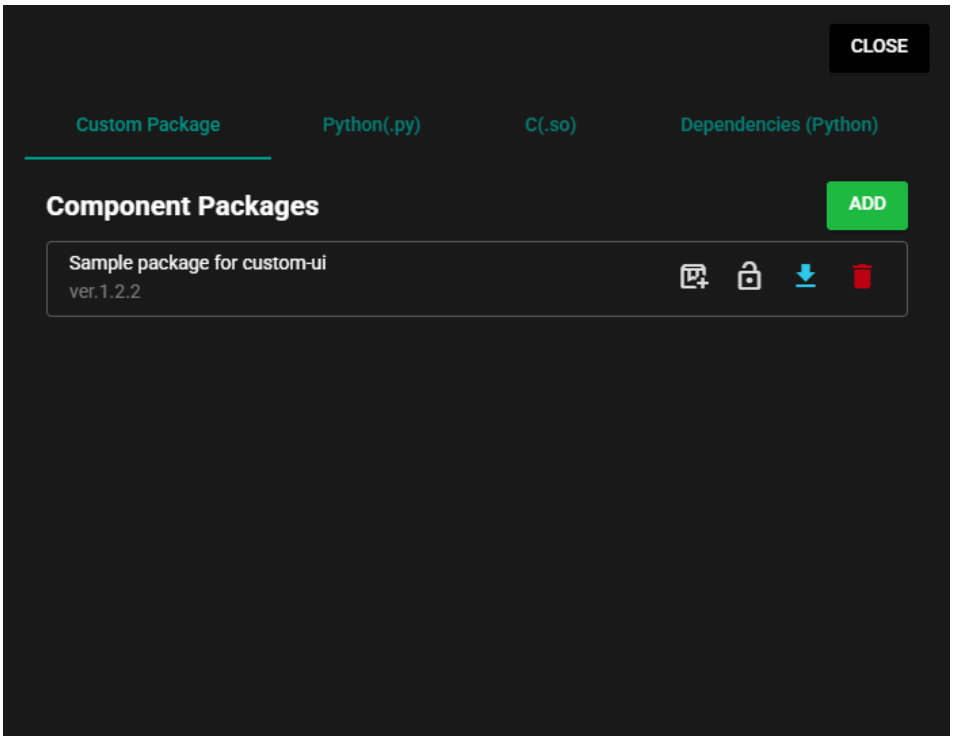
6.2.6.4.4 Delete

You can delete a registered SCCPKG file from SpeeDBee Synapse using the following procedure.

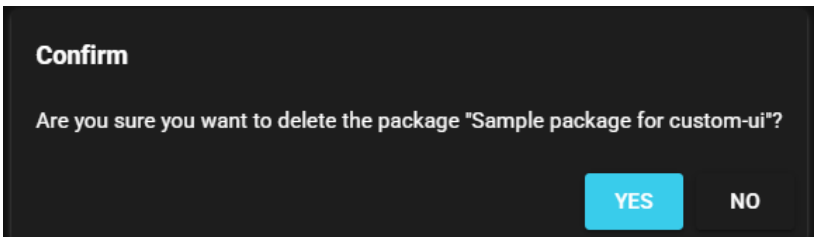
1. Press the Setting Menu icon and choose [Custom Components].



2. Press the trash can icon (Delete) to the right of the target package.



3. Select [Yes] in the confirmation dialog.



4. The SCCPKG file is deleted.

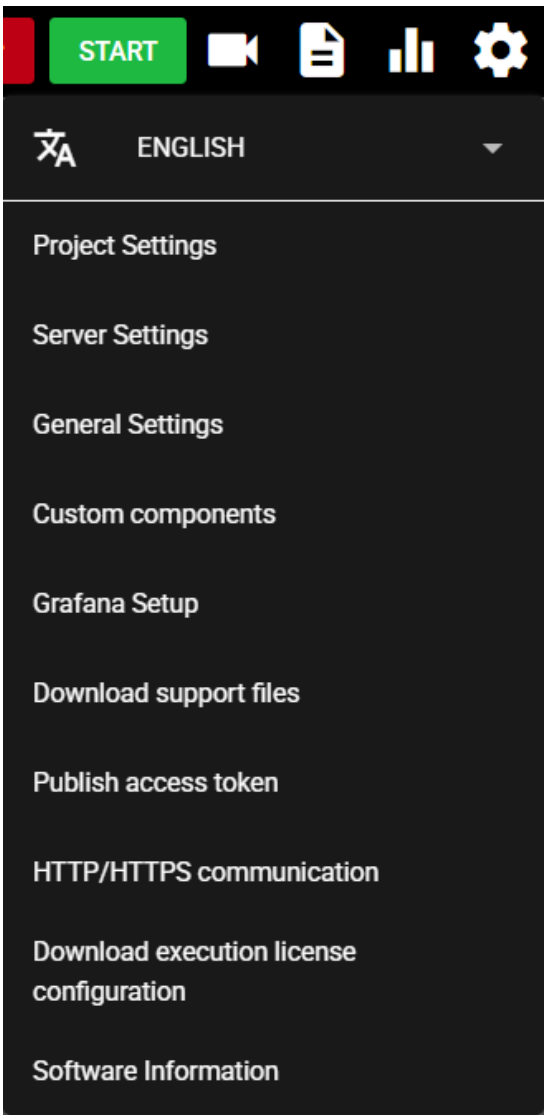
6.2.6.4.5 Protect

Registered packages can be protected.
Protected packages are not subject to the following operations and therefore are protected from unintended updates:

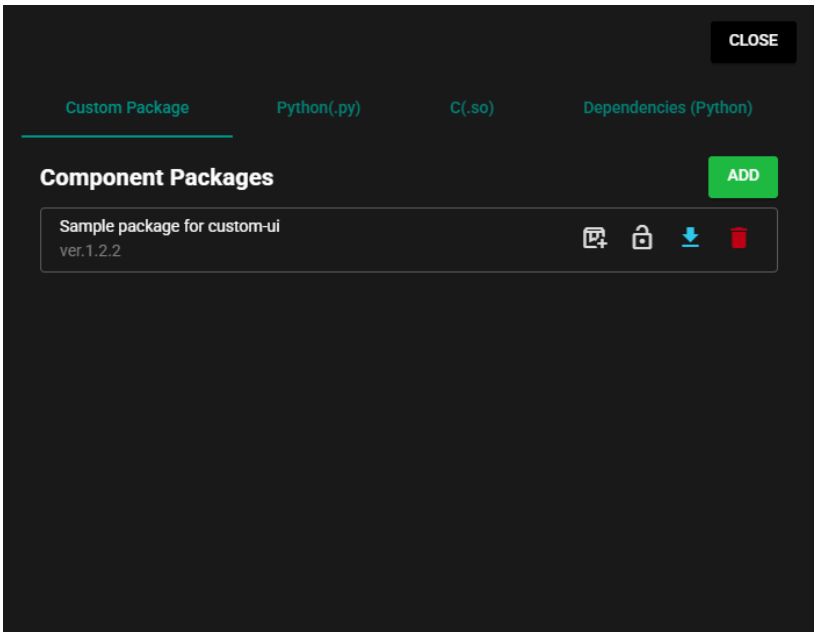
- Register/Update
- Delete
- Download

Packages can be protected as follows:

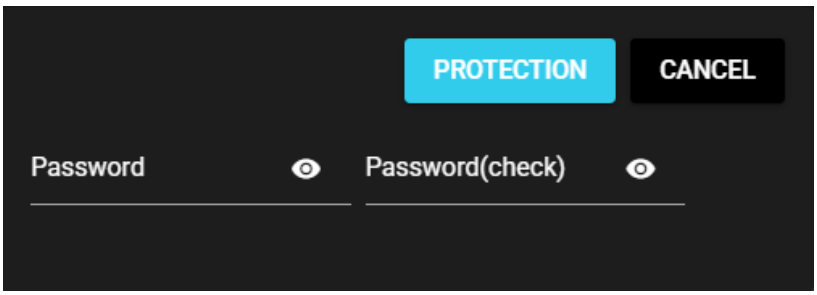
1. Press the Setting Menu icon and choose [Custom Components].



2. Press the key icon (Protection) to the right of the target package.

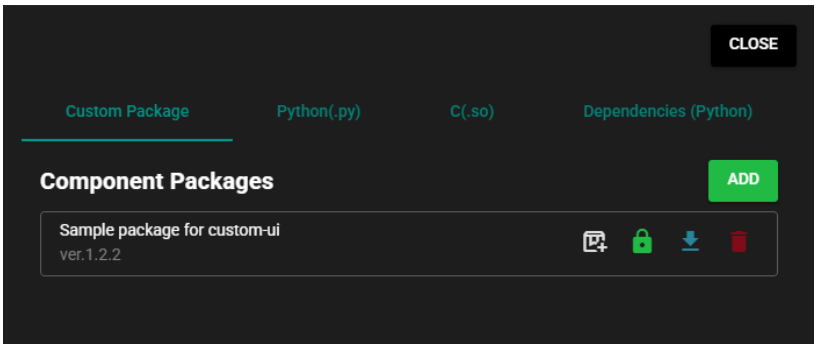


3. Enter the password required to remove protection and press [Protection].



> **Warning**
>
> - The password is stored in a undecodable format and cannot be reset until protection is removed.
> - If you forget the password for protection, the protected package becomes inoperable.
>
> Be careful when handling passwords.

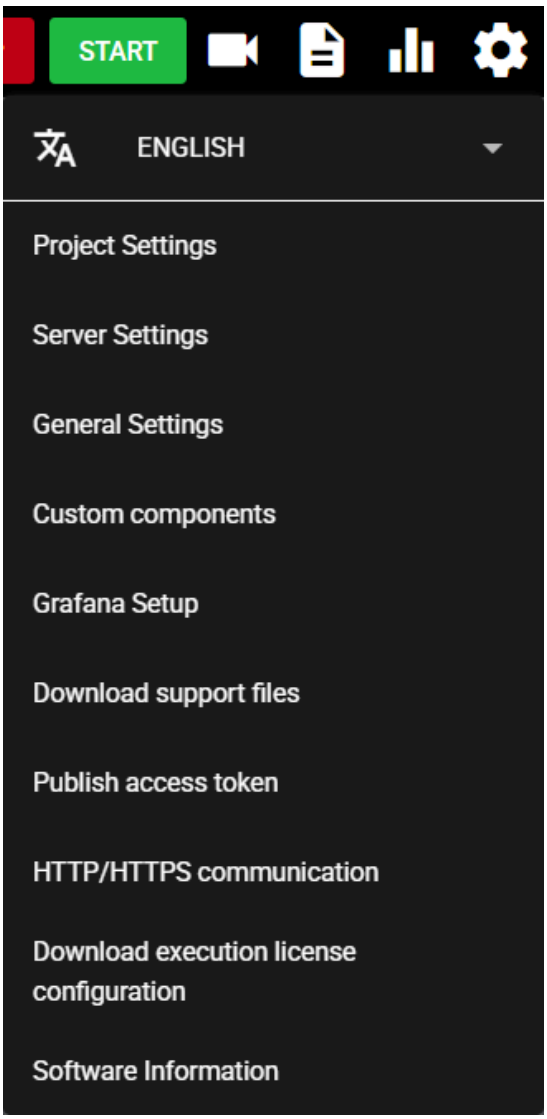
4. The component package is now protected.



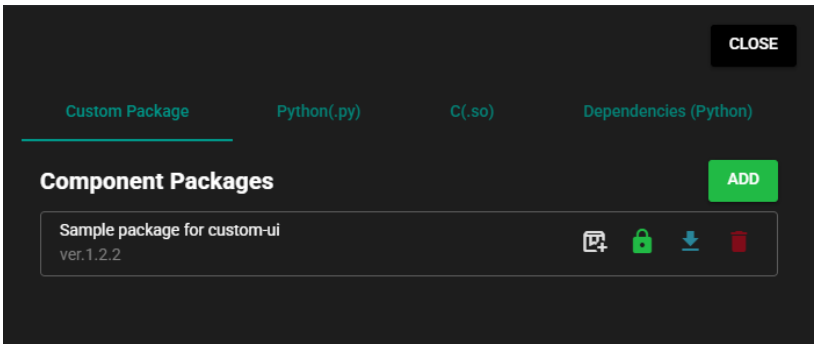
6.2.6.4.6 Unprotect

To remove protection from a package, follow the procedure below:

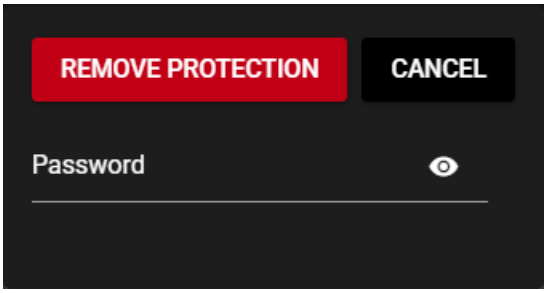
1. Press the Setting Menu icon and choose [Custom Components].



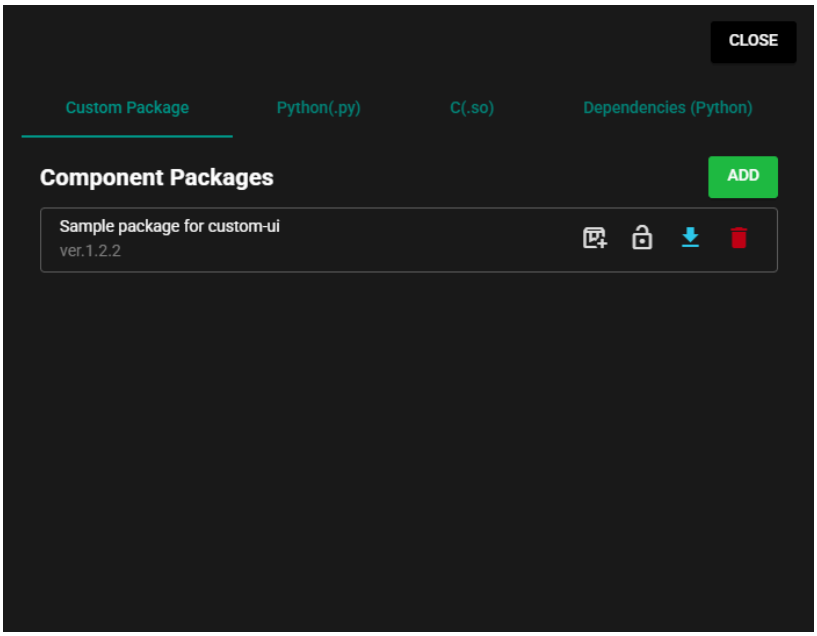
2. Press the key icon (Remove protection) to the right of the target package.



3. Enter the password set when applying protection and press [Remove protection].



4. The protection is removed from the component package.



6.3 Linking with Control Programs (WebAPI)

6.3.1 Introduction to WebAPI

Some operations in SpeeDBee Synapse can be performed the outside through the API.

6.3.2 How to run the WebAPI

As an example, the method to execute the API using the curl command is shown here.
If you don't have curl installed, download it from the official curl site and install it on your machine.

1. Open the environment in which you want to run the curl command. Open Terminal in the case of Linux and open Command Prompt in the case of Windows.
2. Copy the “Request with curl Command” for each API in the API Specification and paste it into the environment where you want to run the curl command.
   Note: Some of the “requests by curl command” need be replaced. For more information, see “Common Issues”.
3. Press Enter to run the API and the response is displayed. For details about the response, see "Response" of each API in "API specification."

The curl command requests for each API in the “API Specification” specify the following: If you use a method other than the curl command to execute the API, you must specify the same items.

Item	Description
The specification is shown in the form of	HTTP method
This is the URL starting from	Request URL
The specification is shown in the form of	HTTP header
The specification is shown in the form of	Data to send with POST method

6.3.3 WebAPI Specification

Regarding the operations that can be performed by the API, the outline, request method and response are listed here. By understanding the API specification and sending the same request from various programs, it is possible to use them from various programming languages and to collaborate with control programs.

For detailed API specification documentation, see External API Interface. If the link is not accessible, please refer to this site.

6.3.3.1 Common Items

The following list is common to the contents shown in the sample requests thereafter.

- Replace with the host port where SpeeDBee Synapse is running.
- Replace with the access token issued in “Publish access token”.
- specifies the ID of the panel you want to retrieve. This acquires the execution status of all components registered on that panel. You can obtain the panel ID by "Acquire execution status of all components."
- specifies the ID of the component whose execution status you want to obtain. You can obtain the component ID by "Acquire execution status of all components."

6.3.3.2 Acquire the execution status of all components

- Overview

  This acquires the execution status of all components registered on SpeeDBee Synapse.
- Request URL

```
http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status
```

- Requests with the curl command

Linux

```
$ curl -X "GET" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>"
```

Windows

```
$ curl.exe -X "GET" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>"
```

- Response

Code	Description
200	Normal
403	Authentication error

Code	Description
503	SpeeDBee Synapse is unable to process requests.

A successful response has the following format:

```
{
  "status": "Status of SpeeDBee Synapse",
  "components": [
    {
      "id": "Component ID",
      "name": "Component name”,
      "panel_id": "ID of the panel in which the component is registered ",
      "panel_name": "Name of the panel in which the component is registered ",
      "status": "Component Status ",
      "errors": [
        {
          "id": "Error ID",
          "parameter": ["Error Message"]
        }
      ]
    }
  ]
}
```

Note 1: "components" stores the execution state of the components. If there are no components, this is an empty array.
Note 2: "errors" stores error information for the component. If there are no errors, this is an empty array.

6.3.3.3 Acquire the execution status of individual component

- Overview

  This acquires the execution status of an individual component registered on SpeeDBee Synapse.

- Request URL

```
http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status
```

- Requests with the curl command
  - Per Panel

Linux

```
$ curl -X "POST" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>" -H 'Content-Type:
application/json' -d "{\"panel_ids\": [<Panel ID>]}"
```

Windows

```
$ curl.exe -X "POST" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"panel_ids\": [<Panel ID>]}"
```

- Per Component

Linux

```
$ curl -X "POST" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>" -H 'Content-Type:
application/json' -d "{\"component_ids\": [\"<Component ID>\"]}"
```

Windows

```
$ curl.exe -X "POST" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/status" -H
"accept: application/json" -H "X-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"component_ids\": [\"<Component ID>\"]}"
```

- Response

Code	Description
200	Normal
403	Authentication error
422	Validation error
503	SpeeDBee Synapse is unable to process requests.

A successful response has the following format:

```
{
  "status": "Status of SpeeDBee Synapse",
  "components": [
    {
      "id": "Component ID",
      "name": "Component name”,
      "panel_id": "ID of the panel in which the component is registered ",
      "panel_name": "Name of the panel in which the component is registered ",
      "status": "Component Status ",
      "errors": [
        {
          "id": "Error ID",
          "parameter": ["Error Message"]
        }
      ]
    }
  ]
}
```

Note 1: "components" stores the execution state of the specified component. If there are no components, this is an empty array.
Note 2: "errors" stores error information for the component. If there are no errors, this is an empty array.

6.3.3.4 Start batch execution of components

- Overview

  This starts component registered on SpeeDBee Synapse.

- Request URL

```
http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/start
```

- Requests with the curl command
  - Per Panel

    Linux

```
$ curl -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/start" -H
"accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"panel_ids\": [<Panel ID>]}"
```

    Windows

```
$ curl.exe -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/start" -H
"accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"panel_ids\": [<Panel ID>]}"
```

  - Per Component

Linux

```
$ curl -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/start" -H
"accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"component_ids\": [\"<Component ID>\"]}"
```

Windows

```
$ curl.exe -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/start" -H
"accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
application/json" -d "{\"component_ids\": [\"<Component ID>\"]}"
```

- Response

Code	Description
200	Normal
400	Component cannot start executing due to insufficient points.
403	Authentication error
422	Validation error
503	SpeeDBee Synapse is unable to process requests.

A successful response has the following format:

```
[
  {
    "id": "Component ID",
    "panel_id": "Panel ID",
    "success": "Component start success or failure ",
    "error": "Error Information"
  }
]
```

Note: The successful response array contains the list of components specified in the request. If there are no components, this is an empty array.

6.3.3.5 Stop batch execution of components

- Overview

  This stops components registered on SpeeDBee Synapse.

- Request URL

```
http://<SpeeDBee Synapse Host Ports>/ext_api/v1/components/stop
```

- Requests with the curl command
  - Per Panel

    Linux

    ```
    $ curl -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/stop" -H
    "accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
    application/json" -d "{\"panel_ids\": [<Panel ID>]}"
    ```

    Windows

    ```
    $ curl.exe -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/stop" -H
    "accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
    application/json" -d "{\"panel_ids\": [<Panel ID>]}"
    ```

  - Per Component

    Linux

    ```
    $ curl -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/stop" -H
    "accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
    application/json" -d "{\"component_ids\": [\"<Component ID>\"]}"
    ```

    Windows

    ```
    $ curl.exe -X "PUT" "http://<SpeeDBee Synapse Host: Ports>/ext_api/v1/components/stop" -H
    "accept: application/json" -H "x-hive-api-key: <Access token>" -H "Content-Type:
    application/json" -d "{\"component_ids\": [\"<Component ID>\"]}"
    ```

- Response

Code	Description
200	Normal
403	Authentication error
422	Validation error
503	SpeeDBee Synapse is unable to process requests.

A successful response has the following format:

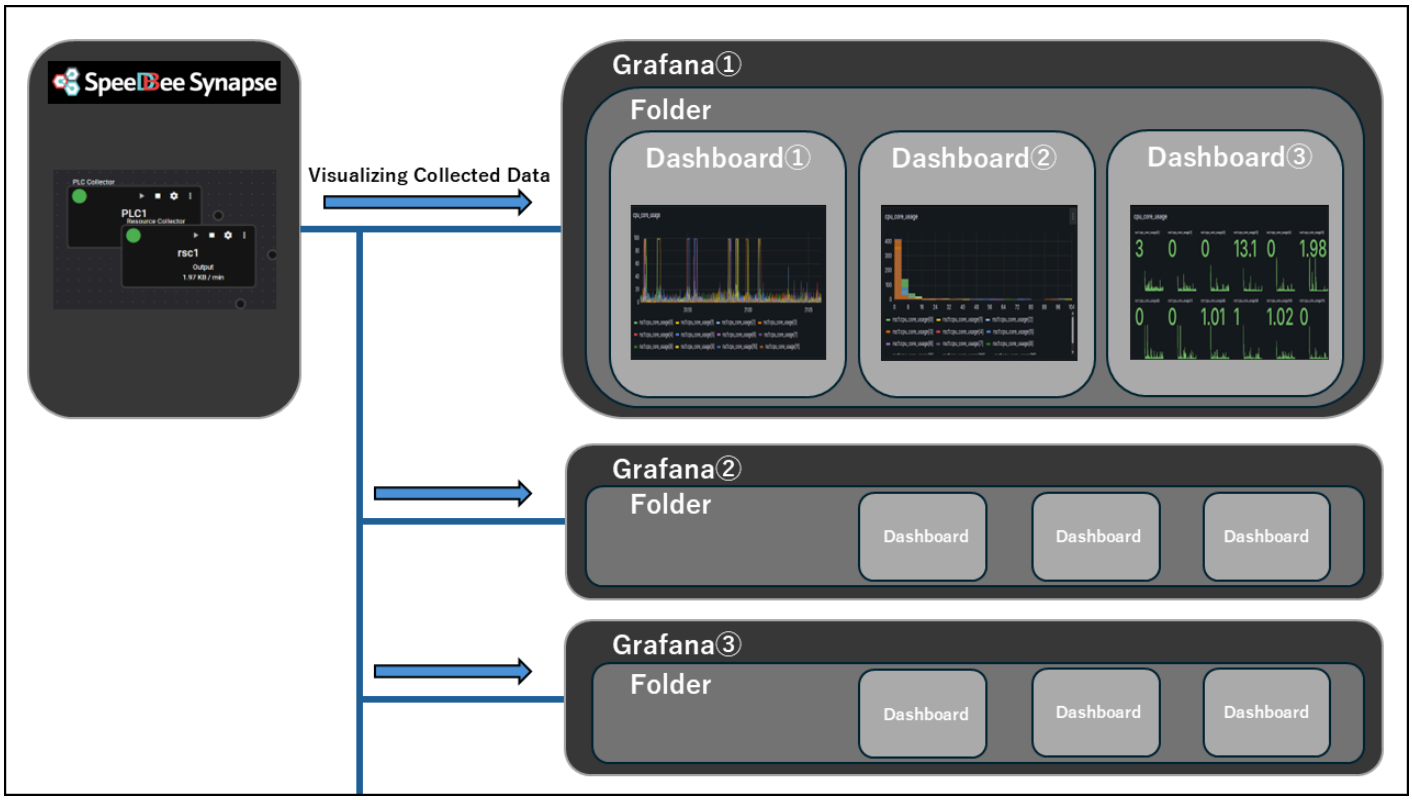
```
[
  {
    "id": "Component ID",
    "panel_id": "Panel ID",
    "success": "Component stop success or failure ",
    "error": "Error Information"
  }
]
```

Note: The successful response array contains the list of components specified in the request. If there are no components, this is an empty array.

6.4 Linking with Grafana

6.4.1 Grafana Integration Overview

Grafana is a tool for visualizing system data.
By working with Grafana, you can create graphs using data collected by SpeeDBee Synapse.
You can connect a single SpeeDBee Synapse to multiple Grafana to create dashboards, and you can also create multiple dashboards within a single Grafana.



For more information about Grafana, visit the official website.

Notes

Version 11.5.2 of Grafana is used in the explanations in this chapter. There may be differences in operations depending on the version.

In the SpeeDBee Synapse Grafana integration feature, the data source, folder, and dashboard in Grafana are created automatically. Since these resources are managed and updated by Synapse, please do not modify them manually from the Grafana interface. If you modify and save them, any user-made changes will be overwritten when Synapse performs an update operation in Grafana. If you prefer to manage these resources manually, please create the data source, folder, and dashboard yourself.

6.4.2 Installing Grafana

- Perform this procedure in an environment in which Grafana is installed.
- The following Linux procedures use sudo for all operations that require system administration privileges.
- In Linux, to operate as a user with root privileges, enter the command without sudo.
- On Windows, start Windows PowerShell with “Run as administrator.”
- For the administrator privileges, consult the system administrator of the OS.

In the Data Flow Controller version, Grafana is pre-installed by default, so no installation is required.

1. Install Grafana. For the installation procedure, refer to the official website.
2. Execute the following command to create the directory to which the plug-in is to be installed:

Linux

```
$ sudo mkdir -p /var/lib/grafana/plugins/SpeeDBSynapse/
```

Windows

```
PS >New-Item -ItemType Directory -Path "C:\Program
Files\GrafanaLabs\grafana\data\plugins\SpeeDBeeSynapse" -Force
```

Standalone Linux Binaries

```
$ sudo mkdir -p <Destination directory>/grafana-9.1.3/data/plugins/SpeeDBeeSynapse/
```

3. Place "synapse_gplugin.tar.gz" in a directory of your choice.

"synapse_gplugin.tar.gz" is a plug-in for linking Grafana and SpeeDBee Synapse. It is distributed together with the installer of SpeeDBee Synapse. If you have any questions, feel free to contact us.

Contact Form: https://www.saltyster.com/contact/

4. Execute the following commands to move to the directory where you located "synapse_gplugin.tar.gz":

Linux

```
$ cd <The directory where synapse_gplugin.tar.gz is located>
```

Windows

```
PS >cd <The directory where synapse_gplugin.tar.gz is located>
```

Standalone Linux Binaries

```
$ cd <The directory where synapse_gplugin.tar.gz is located>
```

5. Execute the following command to install the plug-in:

Linux

```
$ sudo tar xzf synapse_gplugin.tar.gz -C /var/lib/grafana/plugins/SpeeDBSynapse/
```

Windows

```
PS >tar xzf synapse_gplugin.tar.gz -C "C:\Program
Files\GrafanaLabs\grafana\data\plugins\SpeeDBeeSynapse"
```

Standalone Linux Binaries

```
$ sudo tar xzf synapse_gplugin.tar.gz -C <Destination directory>/grafana-
9.1.3/data/plugins/SpeeDBeeSynapse/
```

6. Restart Grafana by executing the following commands:

Linux

```
$ sudo systemctl restart grafana-server
```

Windows

```
PS >Restart-Service grafana
```

Standalone Linux Binaries

```
$ cd < Destination directory >/grafana-9.1.3
$ sudo ./bin/grafana-server
```

Service registration to systemd

If you want to start a service with systemctl, follow the procedure below to register the service with systemd.

a. Creating Service Unit File

/lib/systemd/system/grafana-server.service

```
[Unit]
Description=Grafana server

[Service]
Type=simple
WorkingDirectory=<Destination directory>/grafana-9.1.3
ExecStart=<Destination directory>/grafana-9.1.3/bin/grafana-server
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

b. Load the service to systemd and start the service.

```
$ sudo systemctl daemon-reload
$ sudo systemctl start grafana-server
```

6.4.3 Basic Information for Grafana Integration

To integrate with Grafana, you need to configure the “destination” settings and the “dashboard.”

- **Destination**

  When integrating SpeeDBee Synapse with Grafana, this is the setting that specifies which Grafana to communicate with.

- **Dashboard**

This is the setting for visualizing the data collected by SpeeDBee Synapse on a single screen.

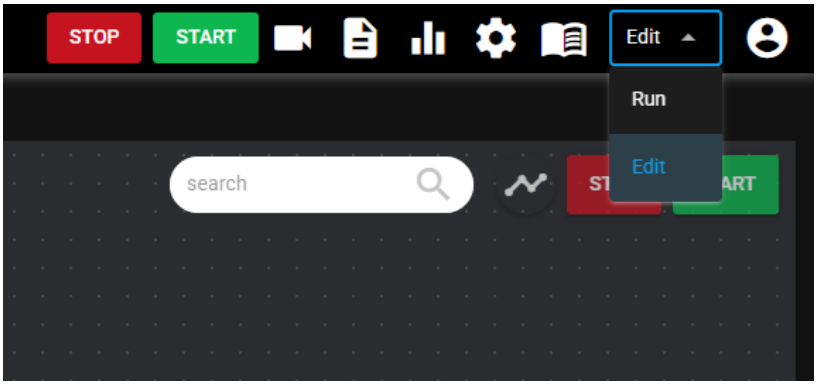
A dashboard is created for the specified destination.

6.4.4 Destination

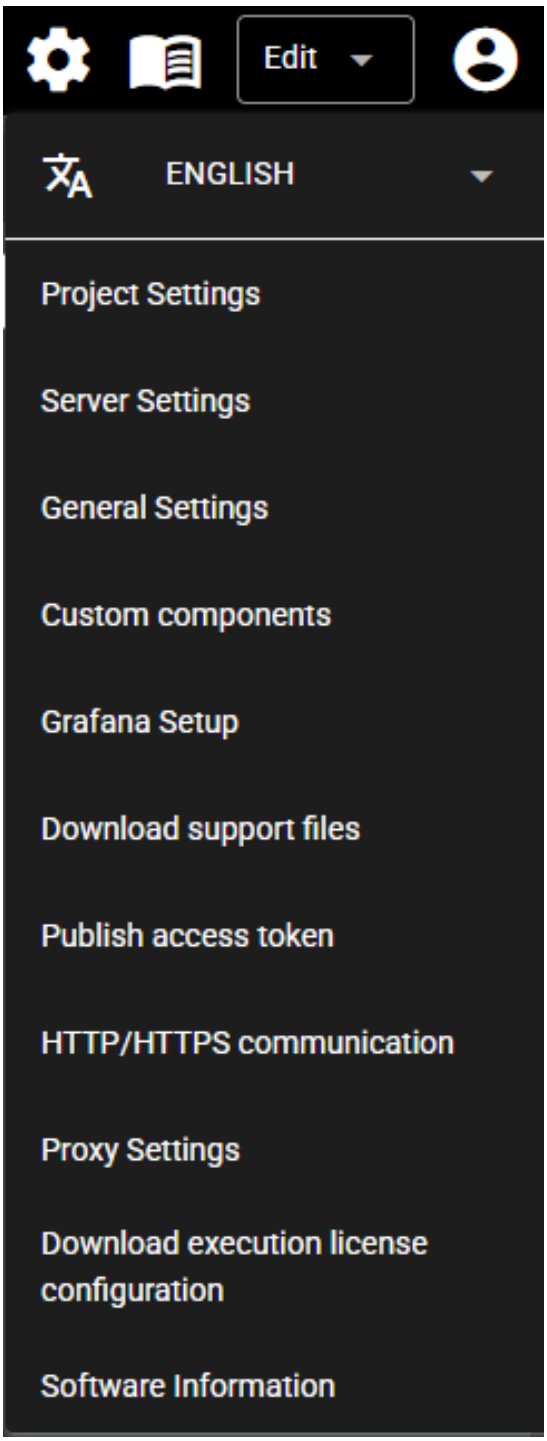
> Before proceeding with this procedure, you must have completed "Installing Grafana."

6.4.4.1 Adding Destination

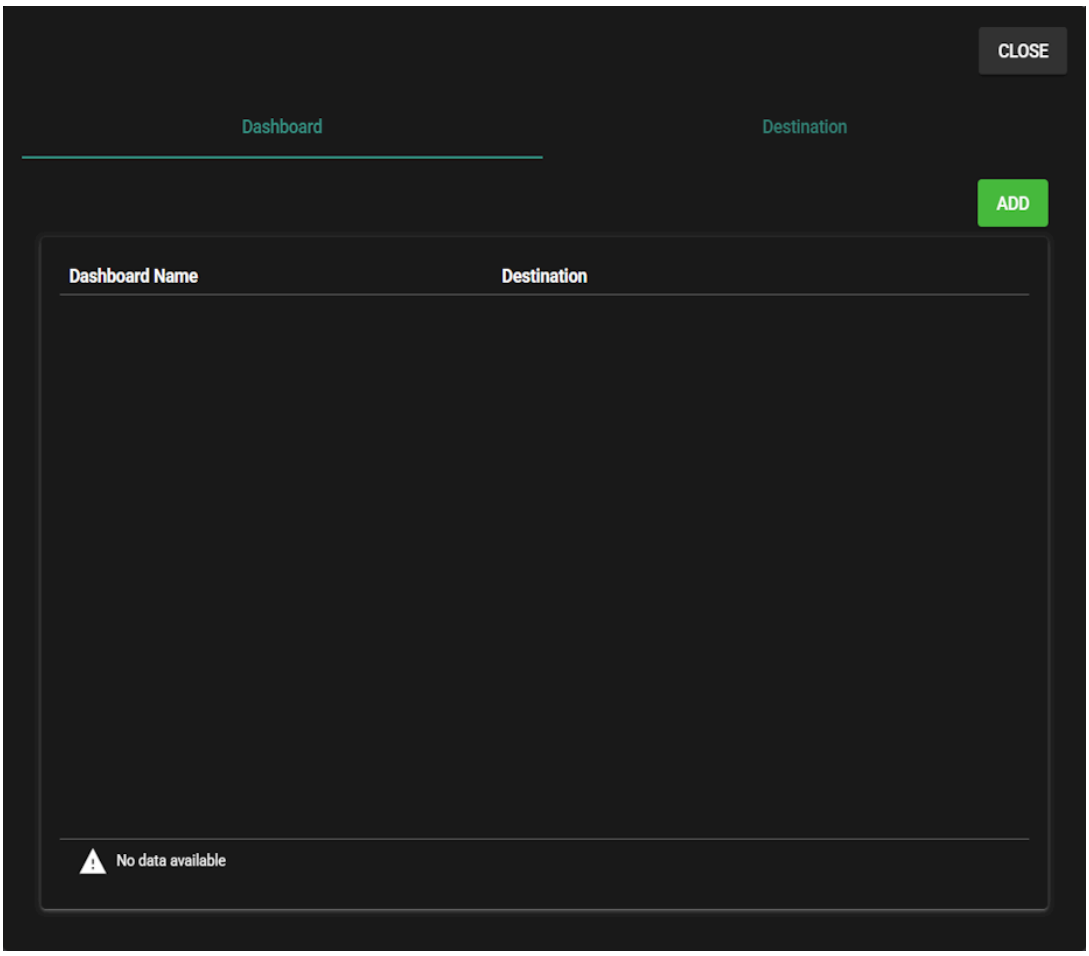
1. Open the Edit mode screen of SpeeDBee Synapse.



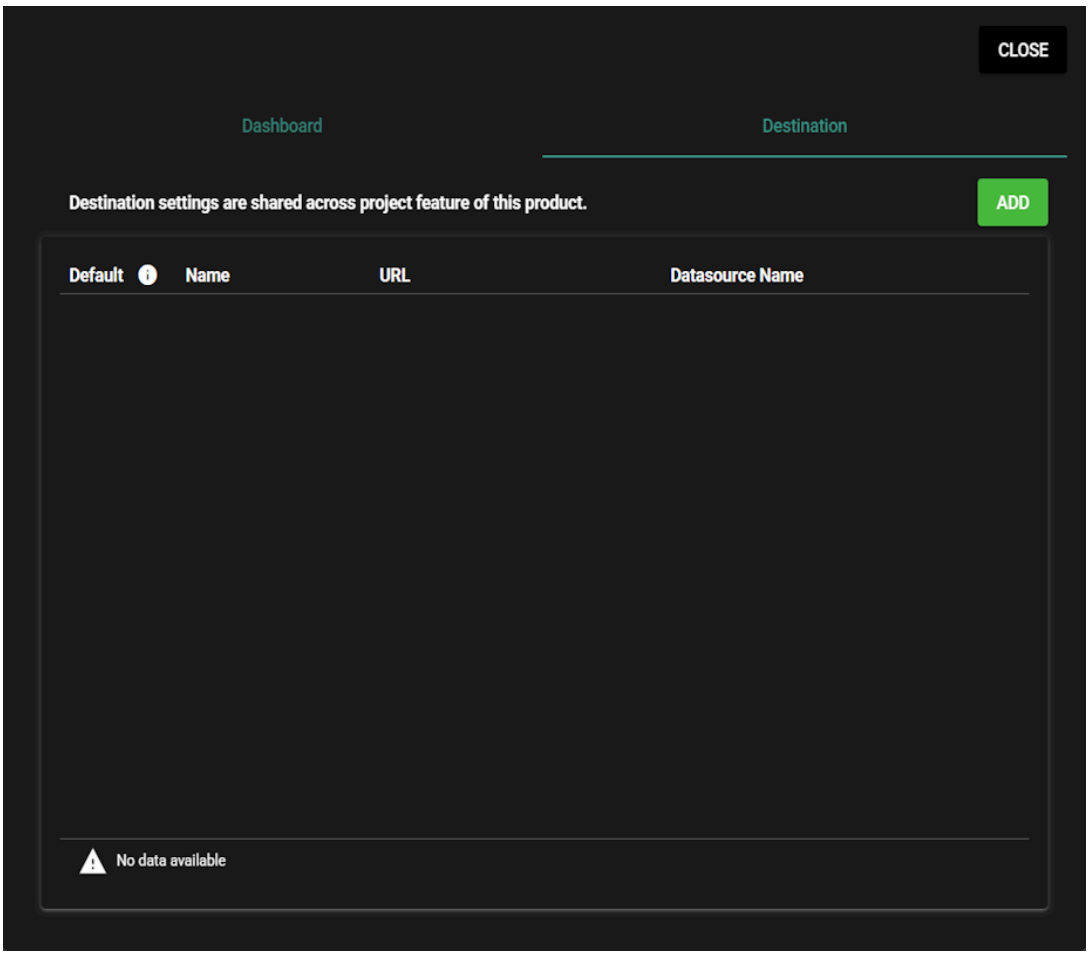
2. Press the Setting Menu icon and choose “Grafana Setup.”



3. The dashboard settings list will open. For details about dashboard settings, see Dashboard.

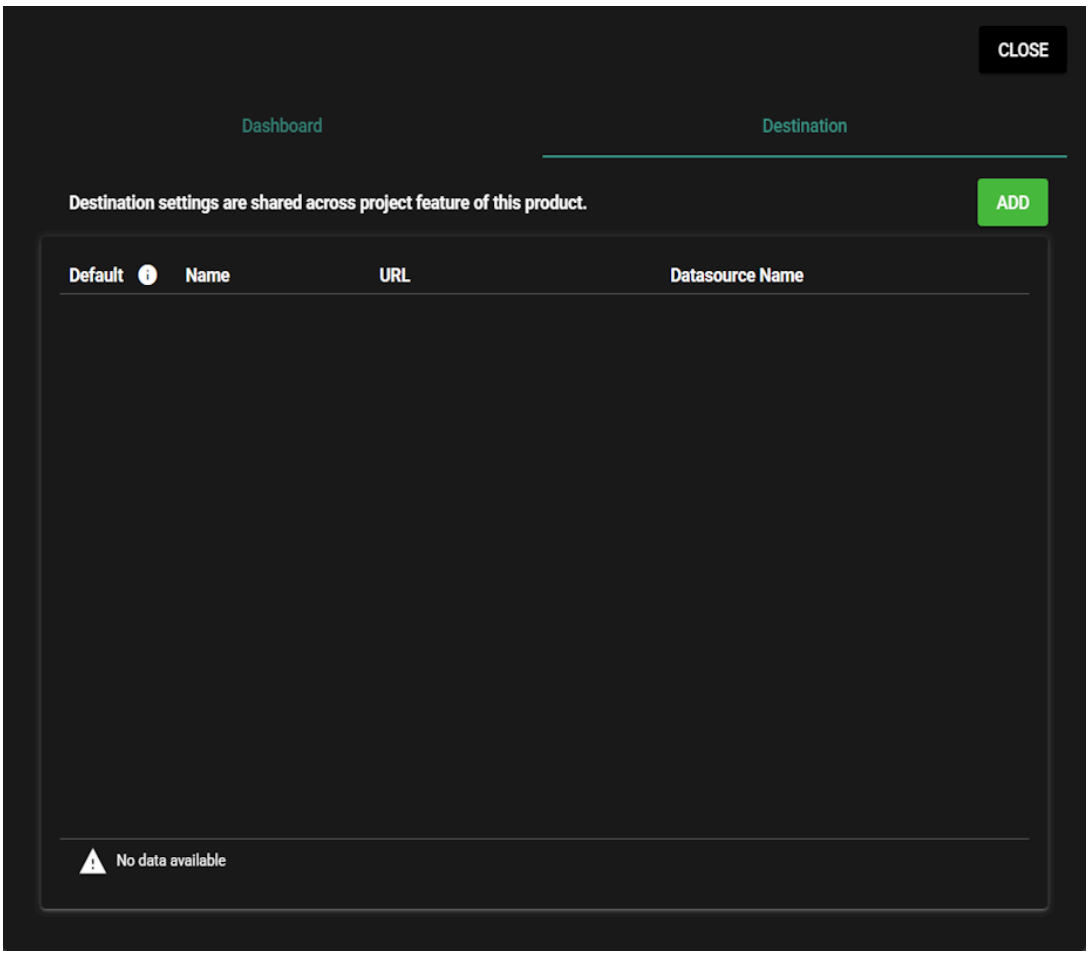


4. Press the “Destination” tab to display the list of destination settings.

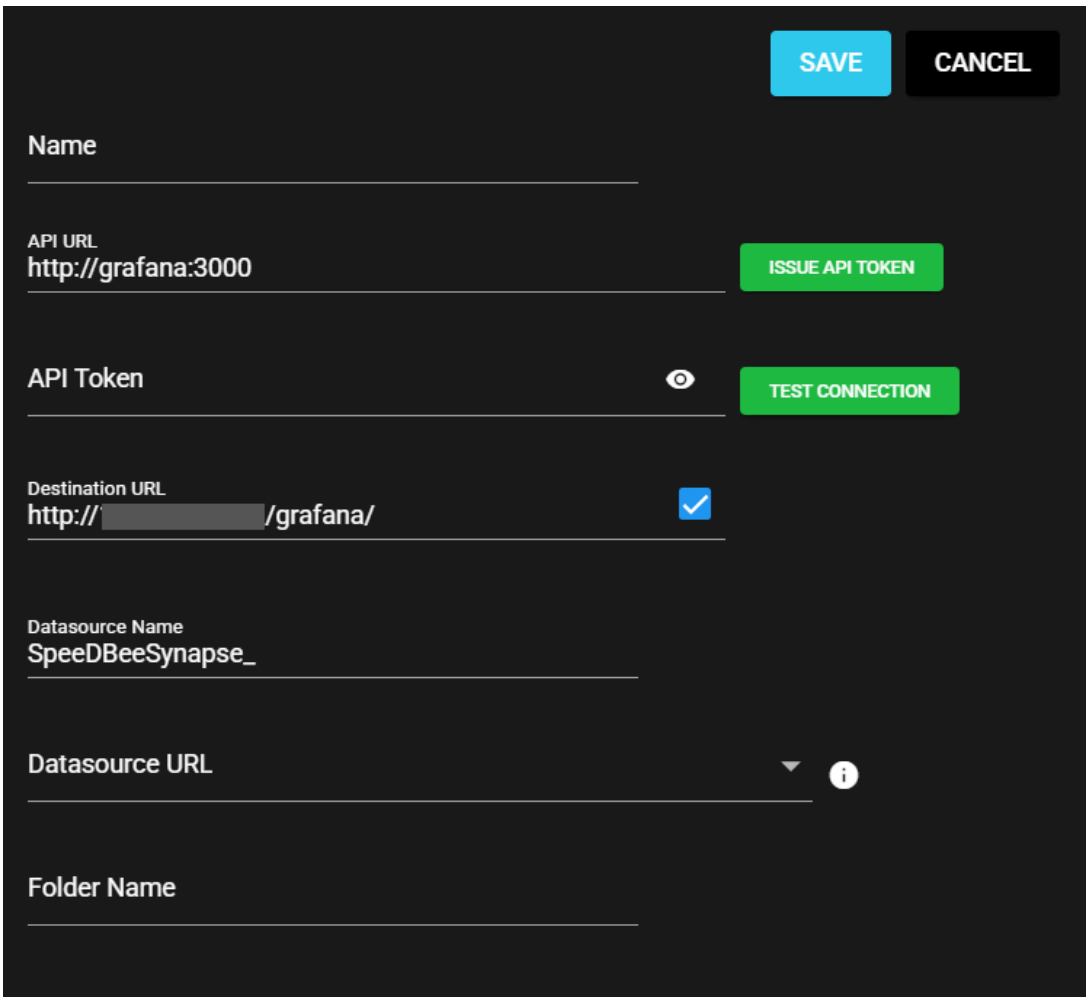


Item	Description
Default	Default destination (used as the target when navigating from the graph icon in the header)
Name	Destination name used in the destination list
URL	URL of the Grafana to connect to
Datasource Name	Data source name used in Grafana

5. Press “Add.”



6. Detailed destination settings will open, and you can enter the information.



Item	Description
Name	Displayed in the destination settings list. Enter any desired text.
API URL	Enter the URL of the Grafana instance to connect to. In the Data Flow Controller edition, "http://grafana:3000" is set by default and can be used without modification. If using an external Grafana, enter the URL that can be accessed from a web browser.
API Token	This is the token used when connecting to Grafana. If you already have one, enter it directly. If not, you can generate one using the “ISSUE API TOKEN” button. For details, see “API Token Issuance Process.”
Destination URL	Enter the URL for the graph icon in the header at the top of the screen. This is normally the same as the API URL, so no changes are required. In the Data Flow Controller edition, "http://<device-IP-address>/grafana/" is set by default. No changes are required if using the built-in Grafana.

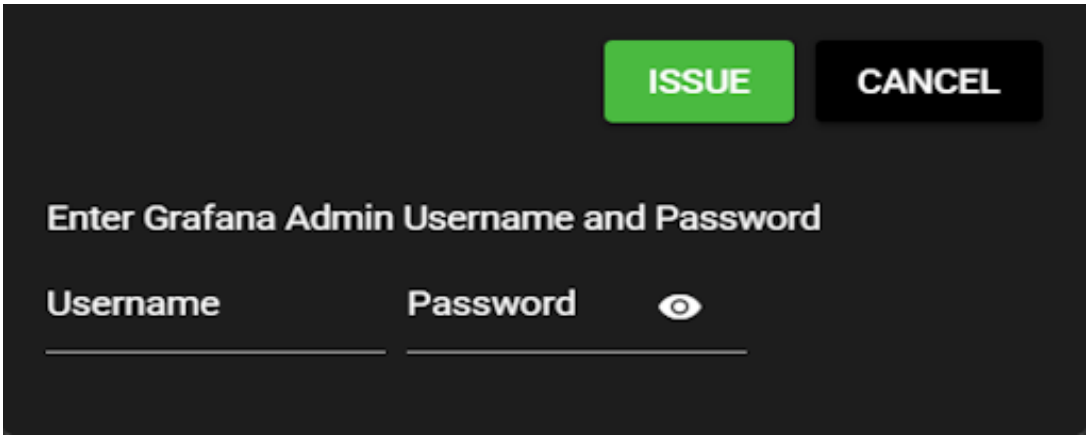
Item	Description
	If using an external Grafana, enter the URL that can be accessed from a web browser. To modify this setting, select the checkbox to enable editing.
Datasource Name	This is the data source name used in Grafana. Normally, it reflects the value entered in the Name field, so no changes are required. If you want to modify it, enter any desired text.
Datasource URL	This is the URL that Grafana uses to access this software's DB query endpoint. Select a URL from the list, or enter one manually if none of the options apply. Example: http://127.0.0.1:8081/dbquery The IP address must be one that Grafana can use to reach this software. For details, see "DB Query Service" and "DB Query Service Web API Specification".
Folder Name	This is the folder name used in Grafana dashboards. Enter any desired text.

API Token Issuance Process

Enter the URL of the Grafana you want to connect to, then press "Issue API Token."



The following dialog appears. After entering the required fields, press "Issue."

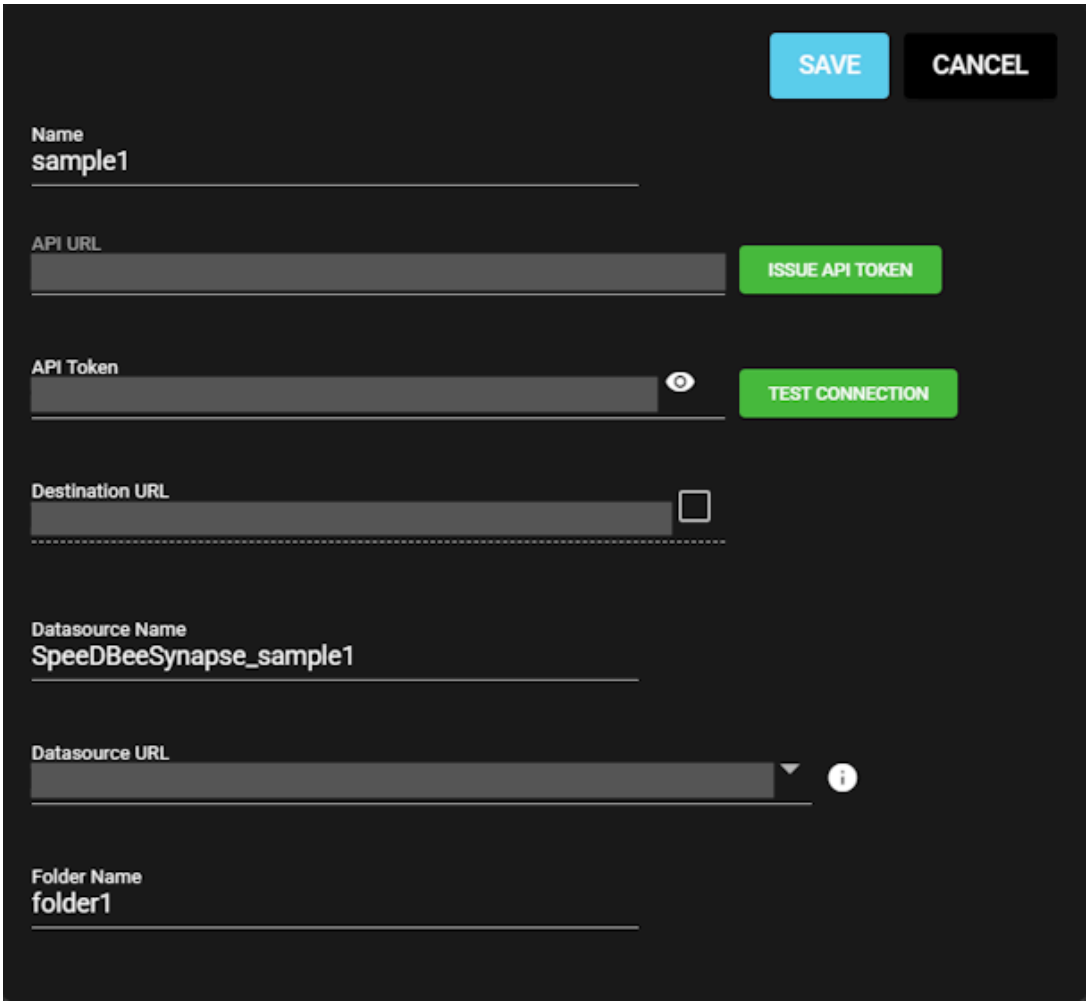


Item	Description
Username	Username required to log in to Grafana Note: The default value is "admin."
Password	Password required to log in to Grafana Note: The default value is "admin."

When the setup is completed, the API token will be filled in automatically.
You can also press “Test Connection” to display the validation result on the right side.



7. Press “Save.”

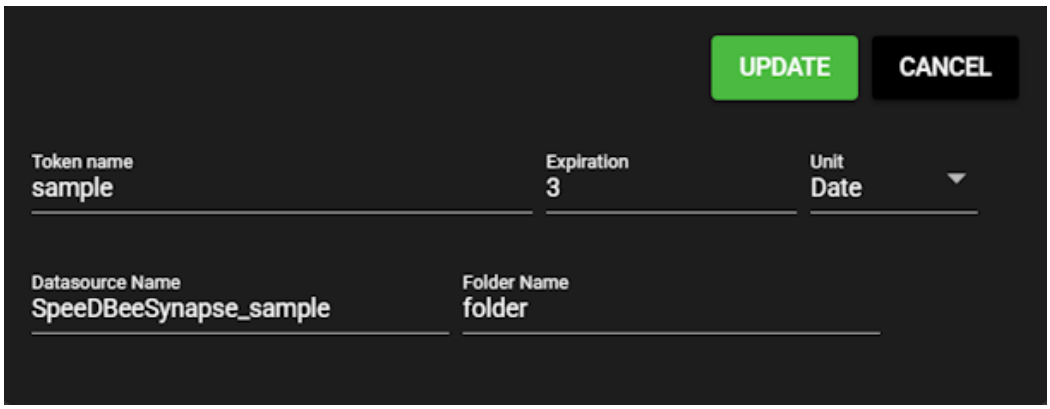


a. After pressing “Save,” the following validations are performed for the access token, data source, and folder.

- The access token registered in the data source
- If no data source exists, whether the configured data source name is already in use
- If no folder exists, whether the configured folder name is already in use

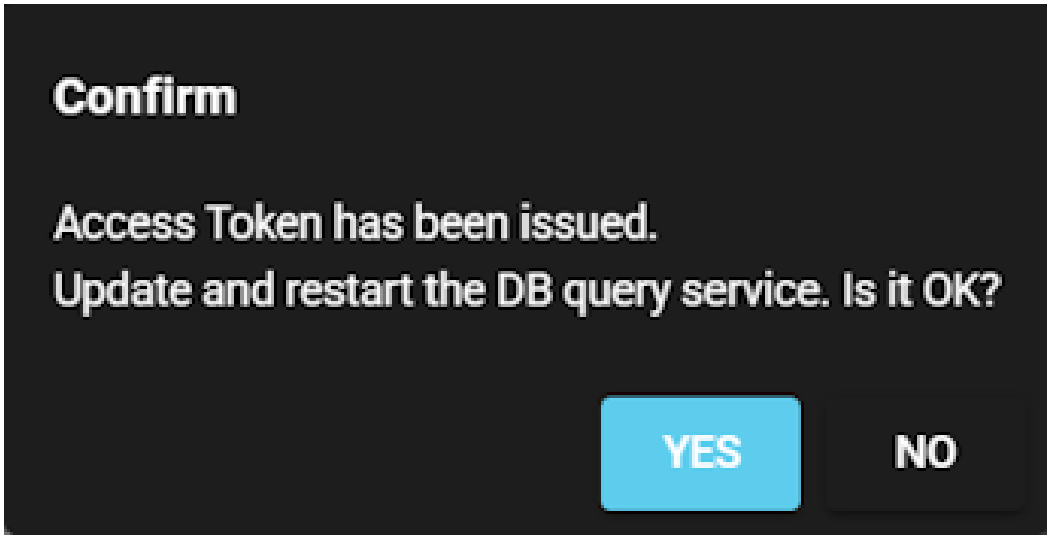
If the validation fails, a dialog like the one shown in the image will appear.
When the dialog appears, adjust your input values as needed and press “Update.”

Note: If the validation succeeds, you can skip this step.

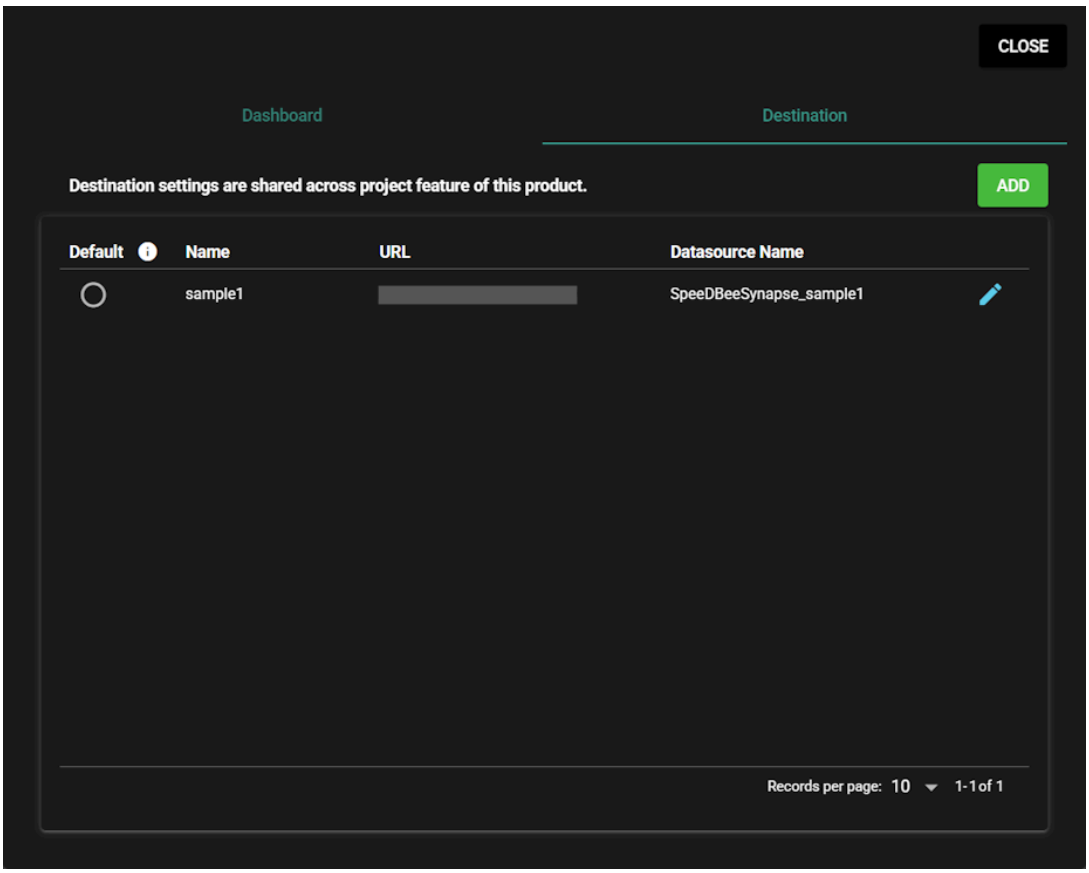


Item	Description
Token Name	The token name you want to assign when issuing a new token
Expiration	Token validity period
Unit	The unit for the value entered in “Expiration”
Datasource Name	The data source name to use when creating a new data source
Folder Name	The folder name to use when creating a new folder

b. If you generate a new access token, a dialog will appear. Press “Yes.”
Note: If you generate a new access token and update it, you will need to update the settings of the DB query service and restart it. If you need to update and start DB query service manually, see “Starting DB Query Service.”
If you did not generate a new access token, you can skip this step.



8. You can confirm that the destination you just configured has been added to the destination list.

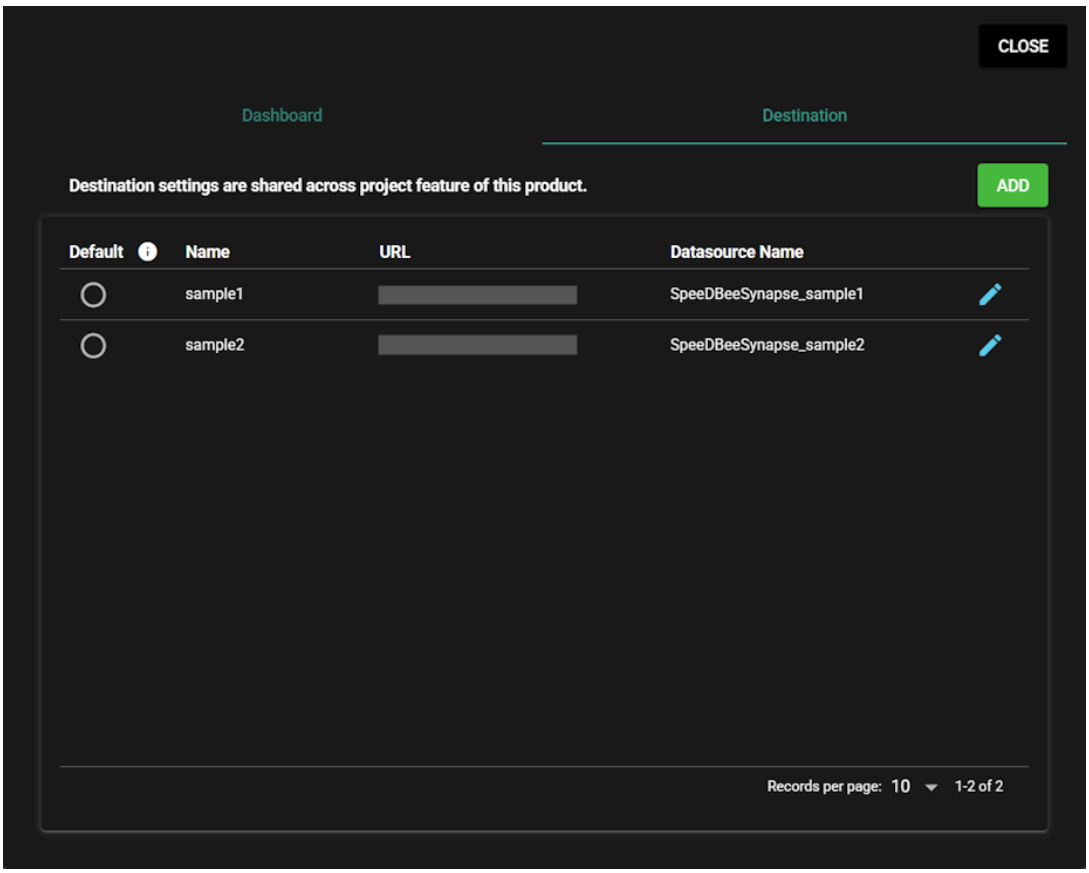


6.4.4.2 Modifying or Deleting Destination

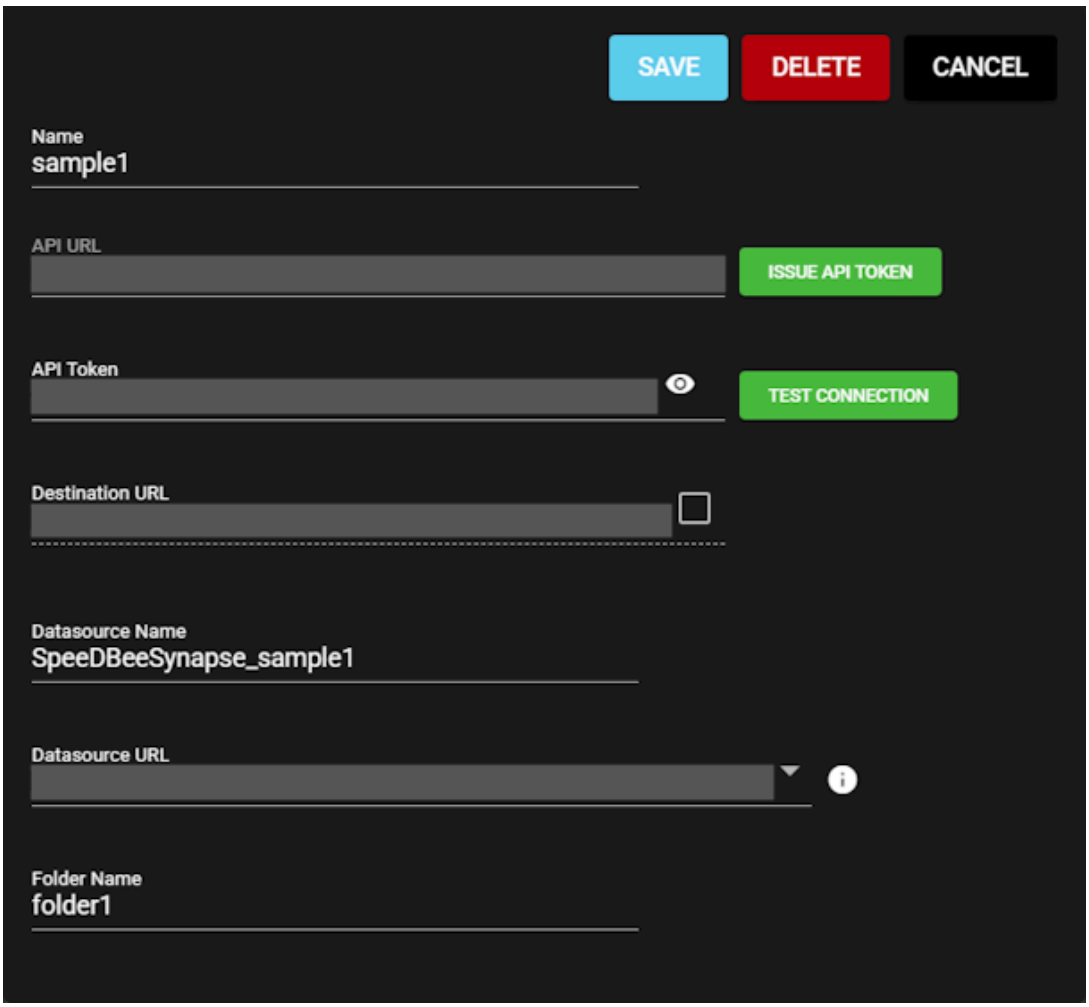
Edit

You can edit a destination settings by following the steps below.

1. Open the destination list and press the edit icon.



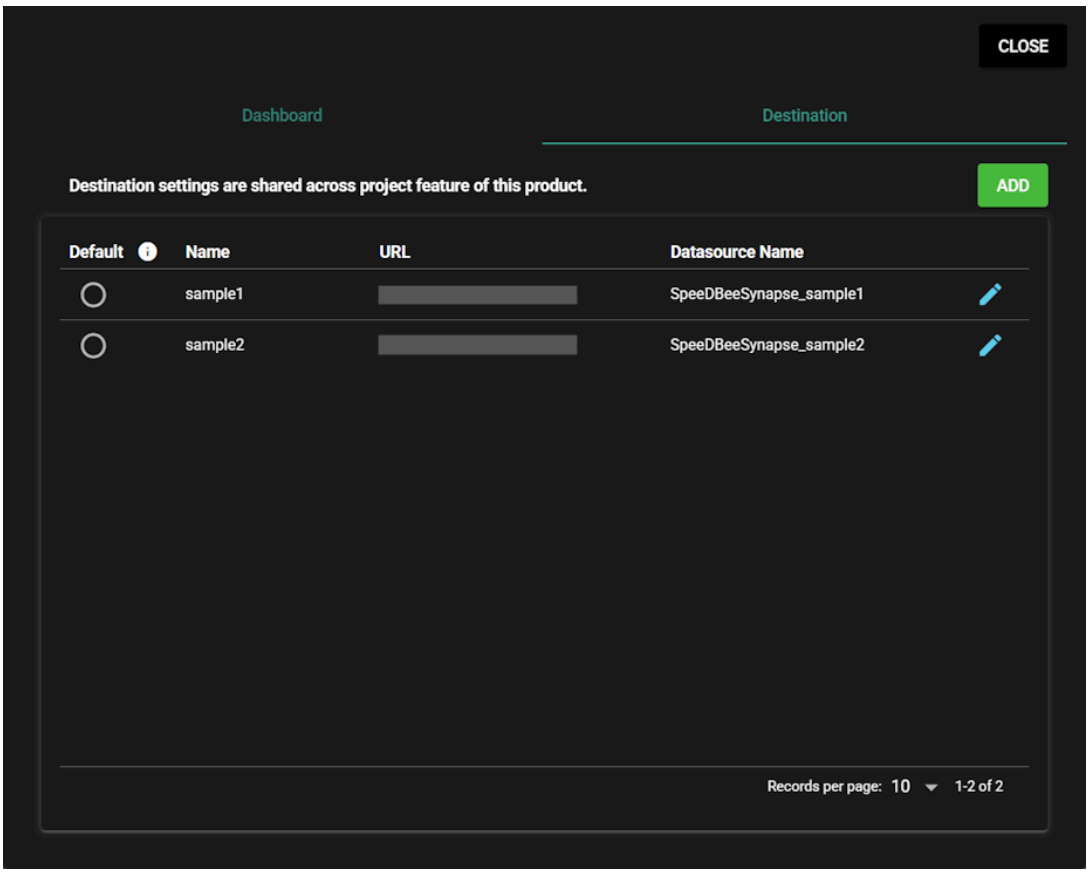
2. Enter the following items and press “Save.”
   For details on each item, see “Adding Destination.”



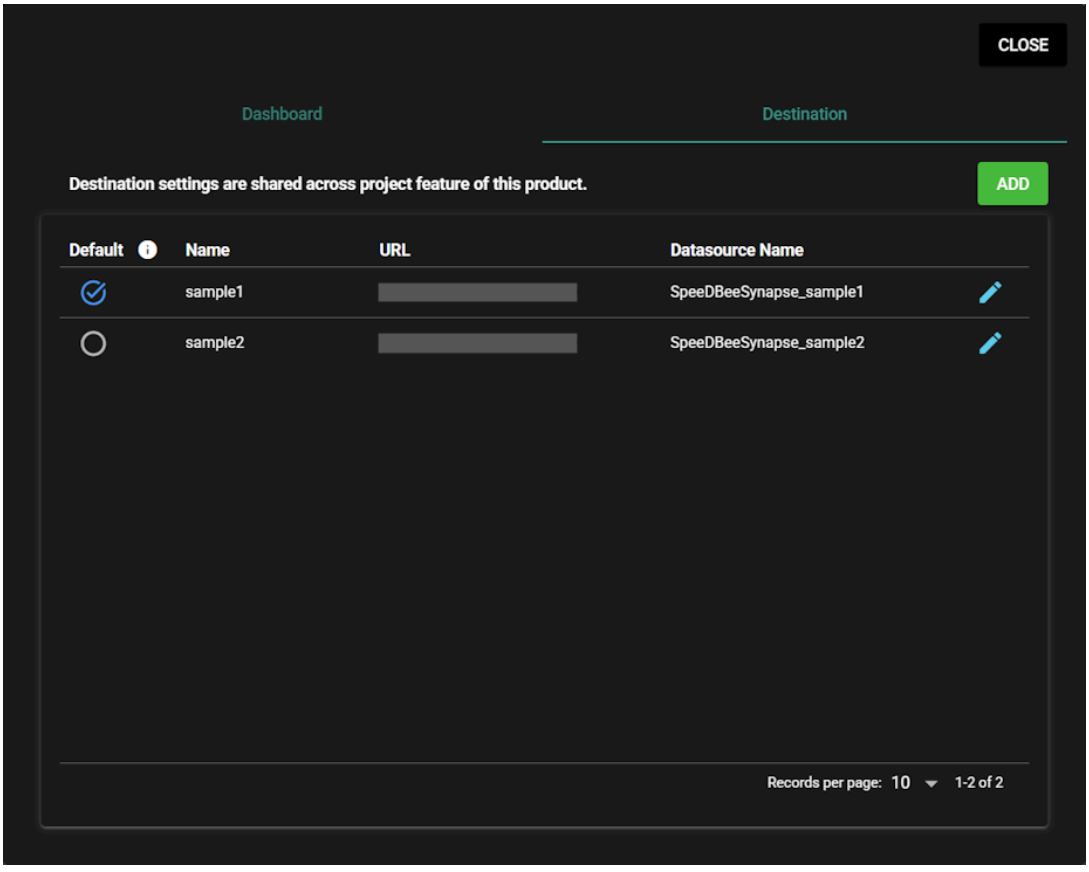
Default Destination

You can set the default destination by following the steps below.
The URL set as the default destination will be used as the target when navigating from the graph icon in the header.

1. Open the destination settings list and check the destination you want to set as the default.



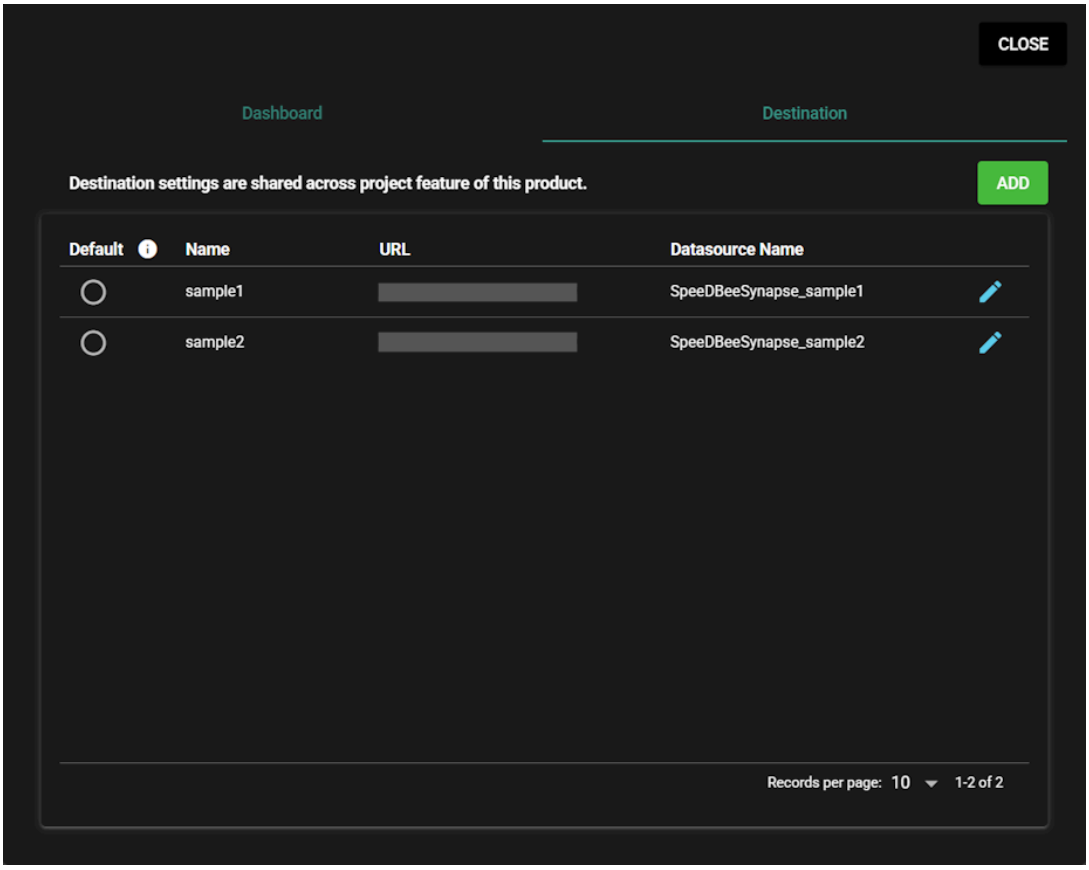
2. Note: Pressing a destination that is already set as the default will unset it as the default destination.



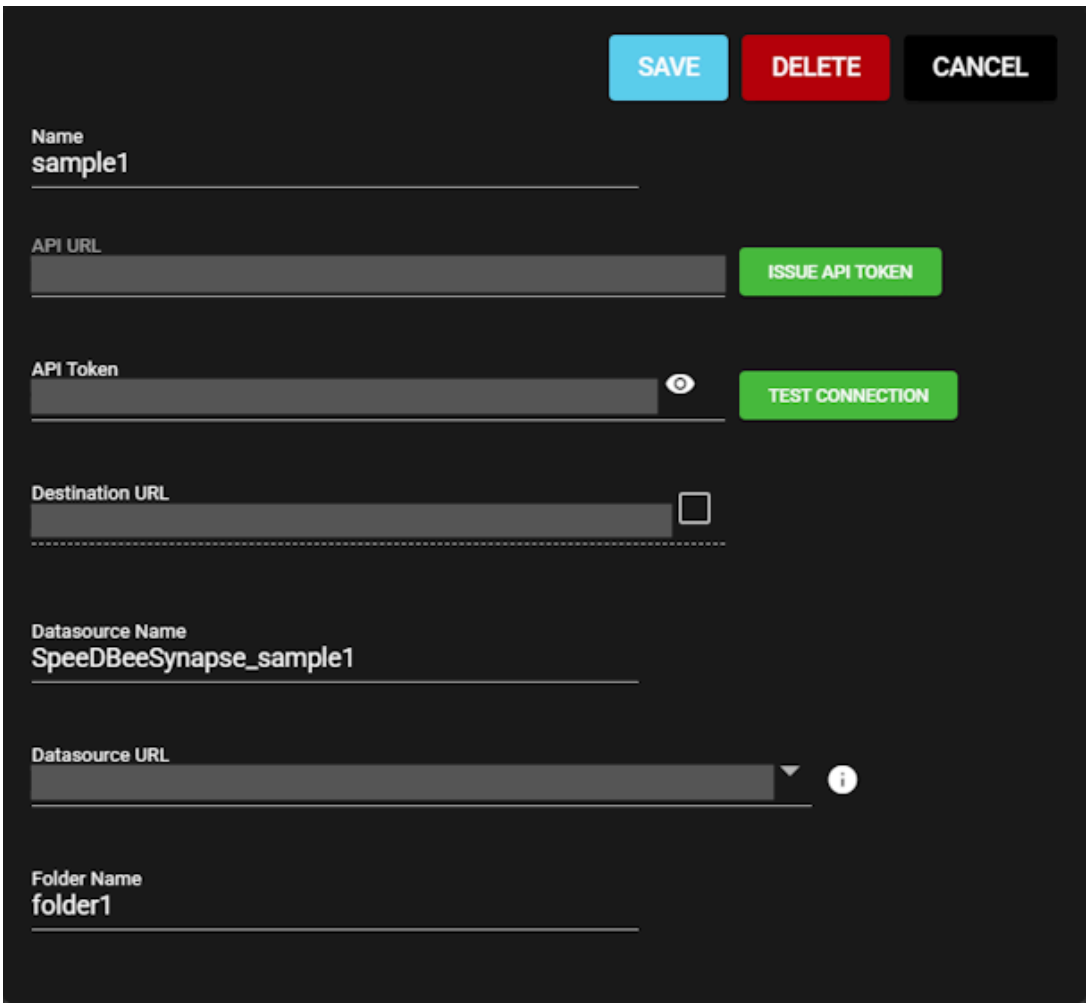
Delete

You can delete destination settings by following the steps below.

1. Open the destination list and press the edit icon.



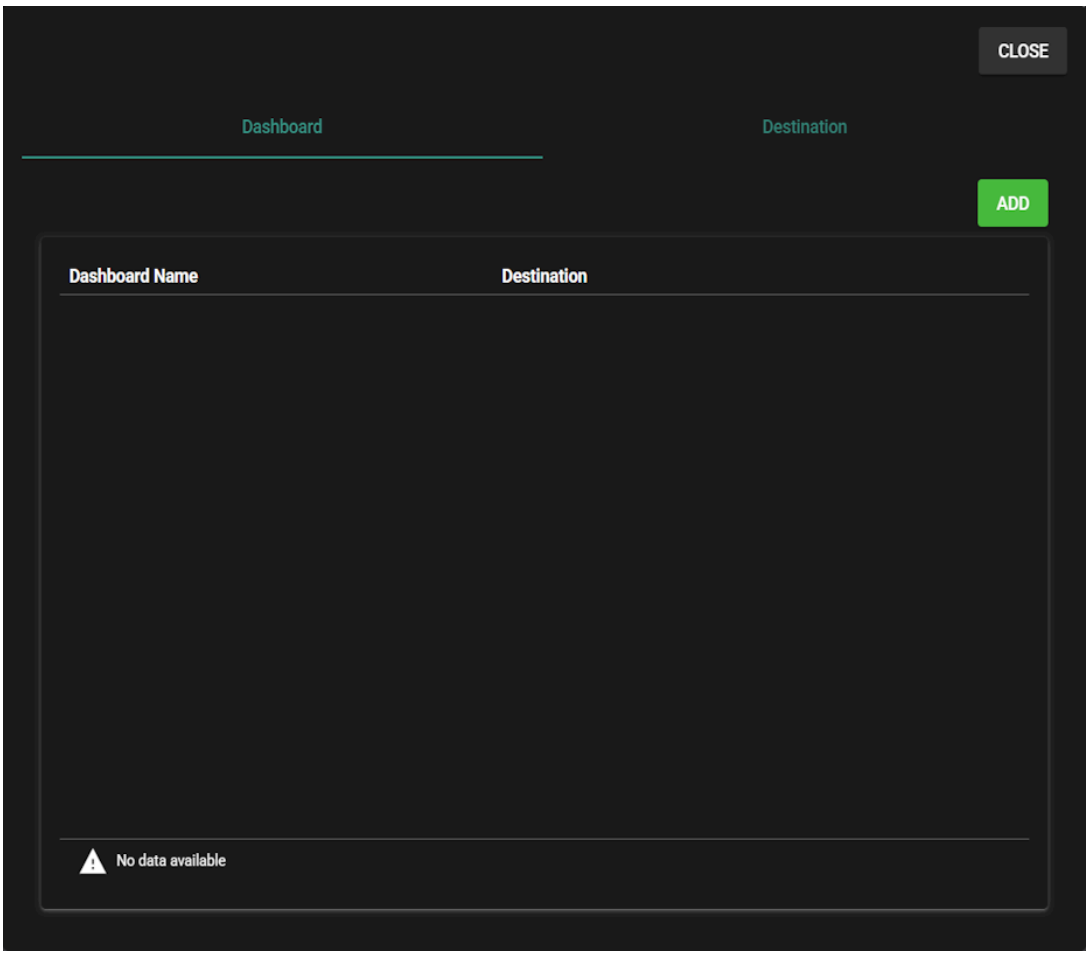
2. Press “Delete.”
   Note: If the destination settings you are deleting is associated with any resources on the Grafana side, those resources will also be removed when you delete the destination settings.



6.4.5 Dashboard

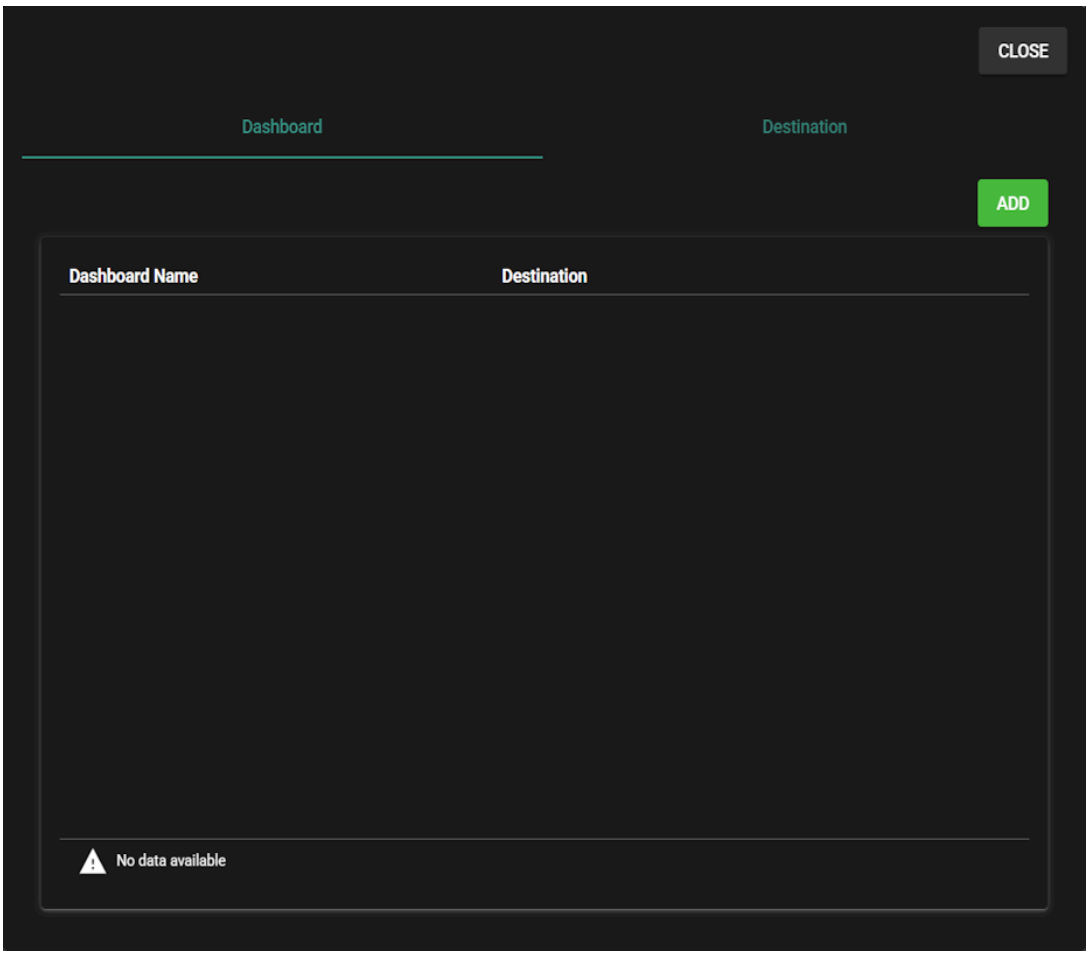
6.4.5.1 Adding Dashboard

1. Press the “Dashboard” tab to display the list of Dashboard settings.

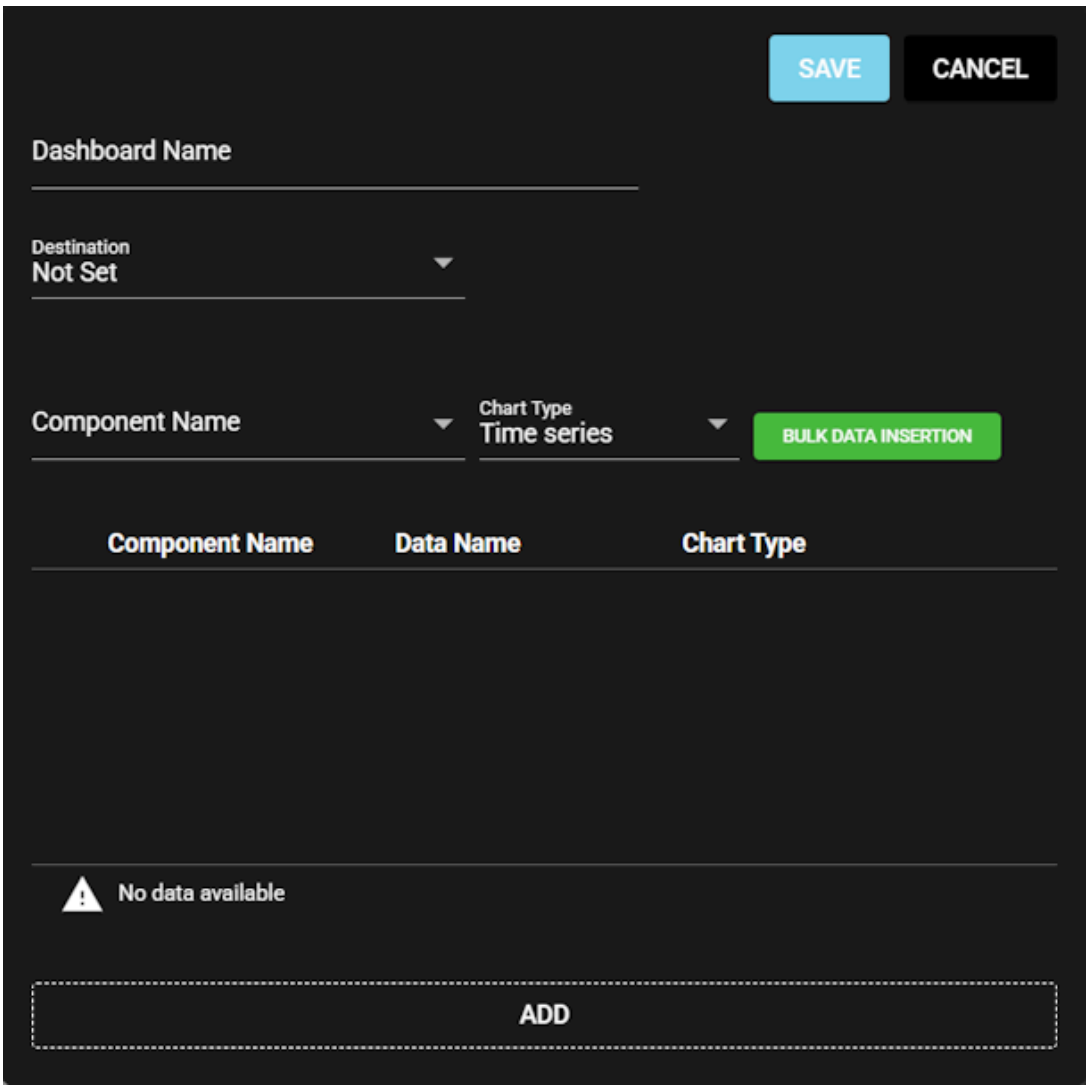


Item	Description
Dashboard Name	Dashboard name used in the dashboard list
Destination	The destination used by the dashboard

2. Press “Add.”

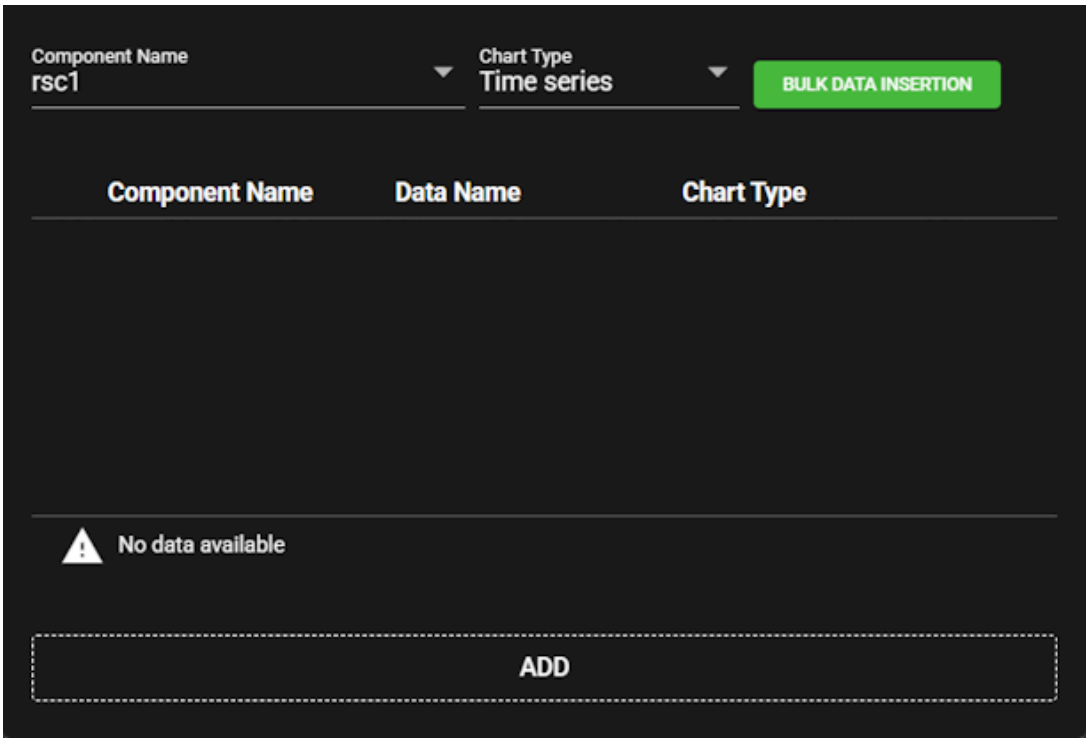


3. Detailed dashboard settings will open, and you can enter the information.



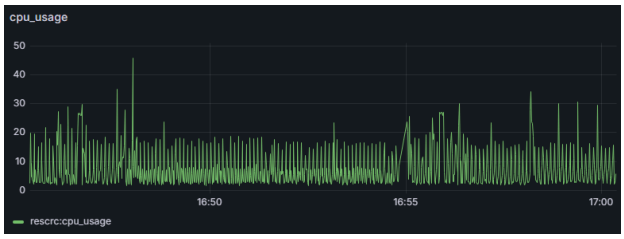
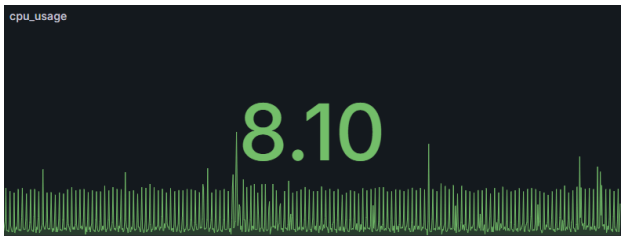
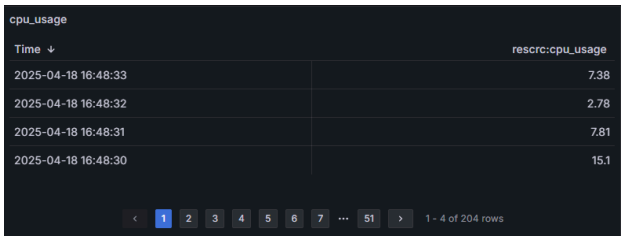
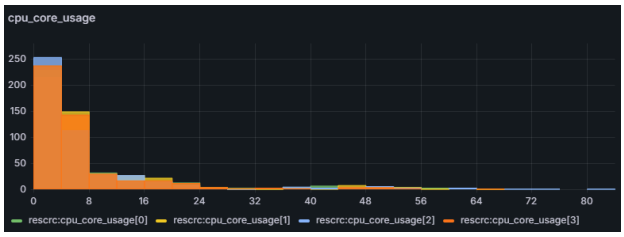
Item	Description
Dashboard Name	This will be displayed in the dashboard settings list and as the dashboard name in Grafana. Enter any desired text.
Destination	This is the destination used by the dashboard. Select the destination configured in “Destination.” If the destination is set to “Not Set”, it will not integrate with Grafana and will only be added to the dashboard settings list.
Component Name	This is the component name used by the “Bulk Data Insertion” feature. For details, see “Bulk Data Insertion.”
Chart Type	This is the chart type used by the “Bulk Data Insertion” feature. For details, see “Bulk Data Insertion.”

The dashboard displayed in Grafana is configured using the items in the table shown in the lower half of the image.
Pressing “Add” will add a new row.
Note: when multiple rows exist, you can reorder them by dragging the icon on the left.



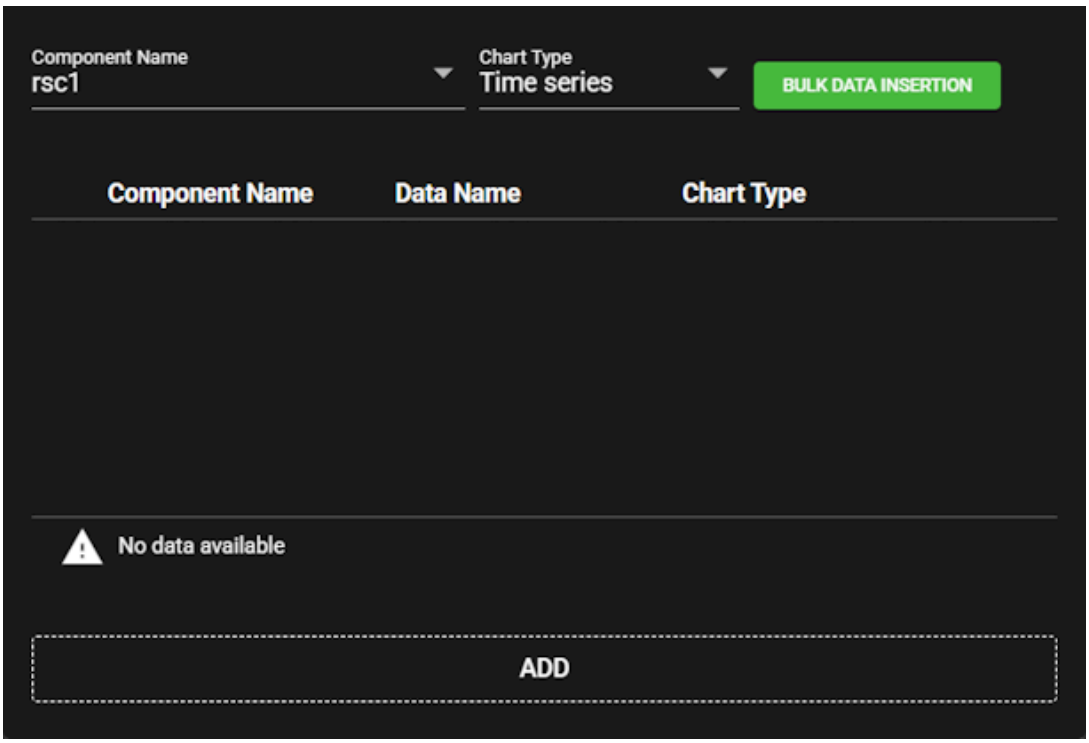
Item	Description
Component Name	This is the component name displayed in the Grafana dashboard. You can select it with a left-click, and manual input is also available.
Data Name	This is the name of the data output by the component. You can select it with a left-click, and manual input is also available. You can select the data name output by the component in the same row.
Chart Type	This is the chart type displayed in the Grafana dashboard. Select the graph type you want to configure.

The available chart types are as follows.

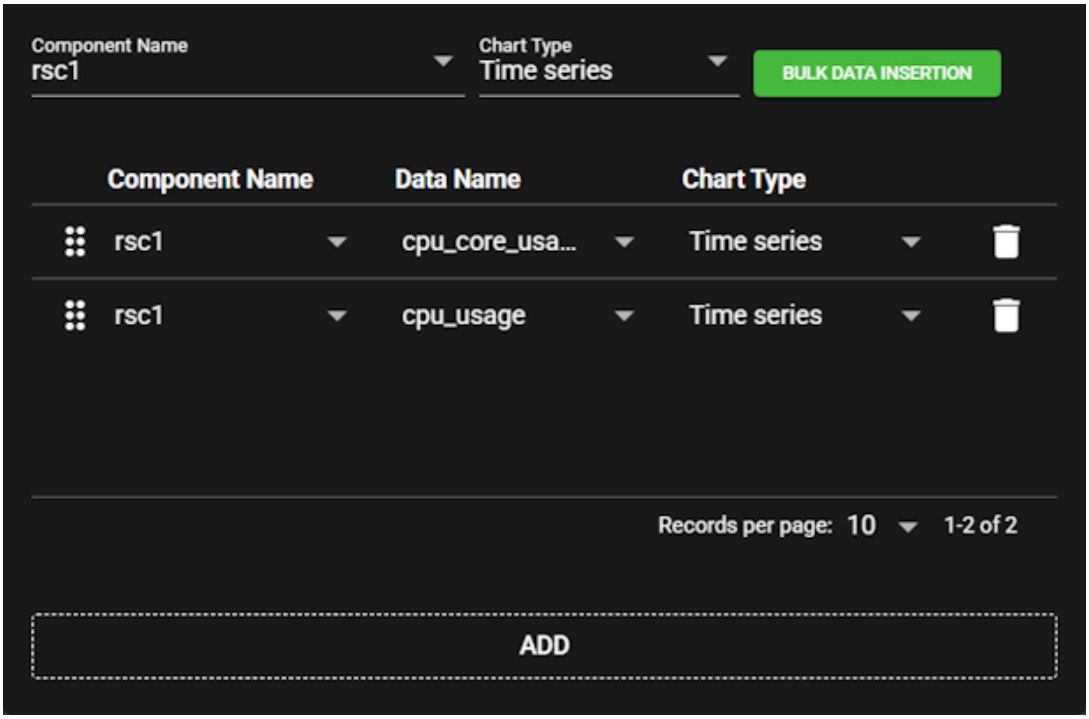
Chart Type	Description	Example
Time series	Displays values over time.	
Stat	Displays the current value over a background line chart.	
Table	Displays values in a table format.	
Histogram	Calculates the distribution of values and displays it as a bar chart.	
Unselected	The default value when selecting a chart type. It cannot be saved while unselected.	

Bulk Data Insertion

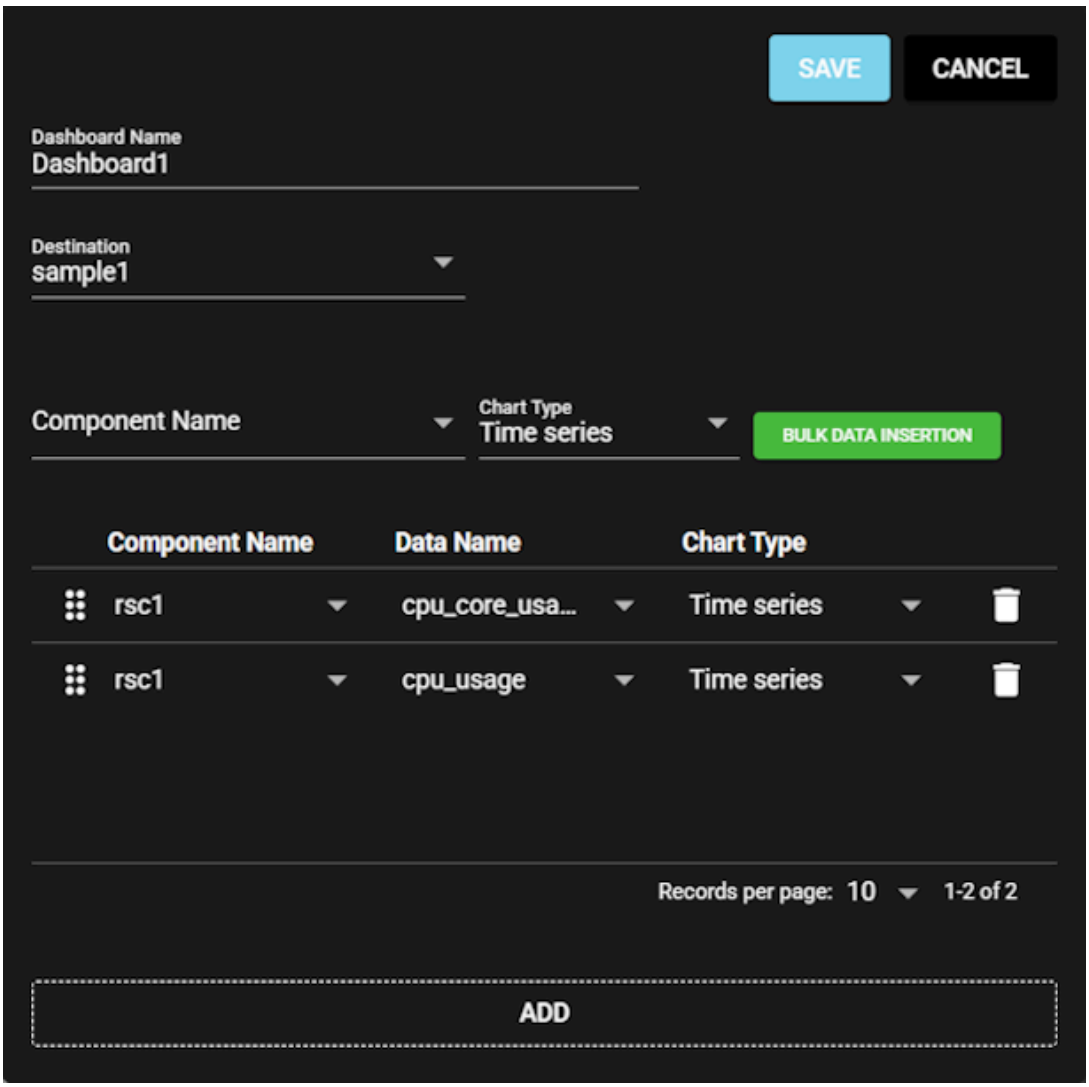
Select the component and chart type, then press “Bulk Data Insertion.”



All data output by the selected component will be added at once using the selected chart type.



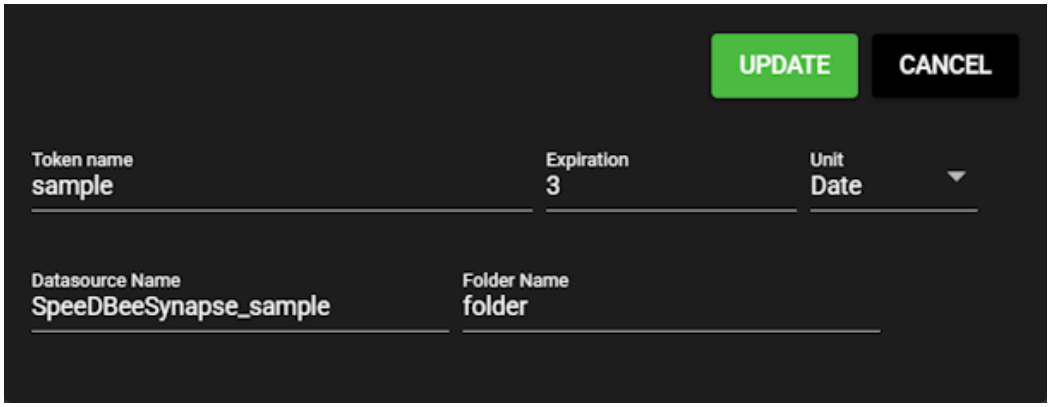
4. Press “Save.”



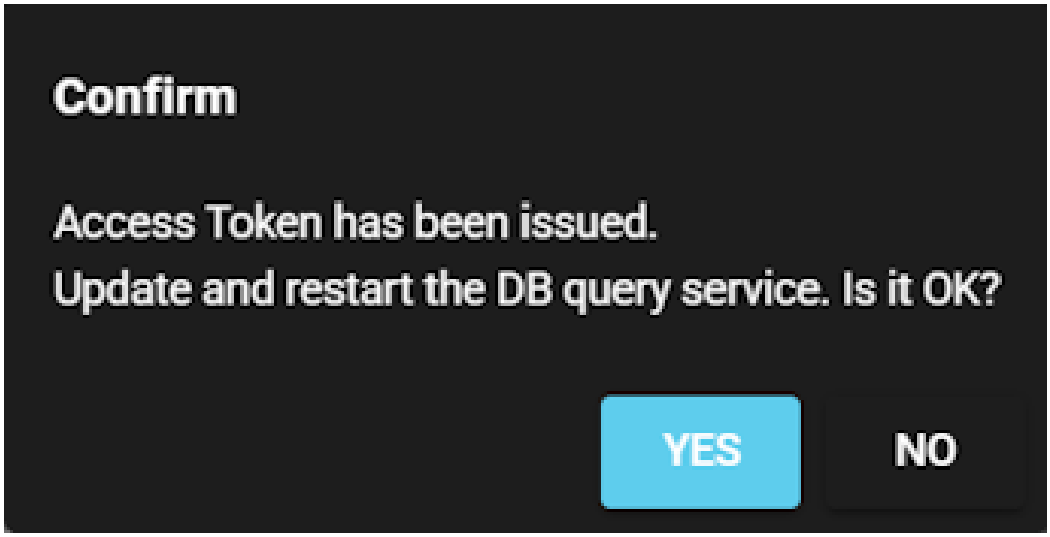
a. As with the destination settings, pressing “Save” validates the access token, data source, and folder. (For details, see “Destination.”)

If this dialog appears, adjust your input values as needed and press “Update.”

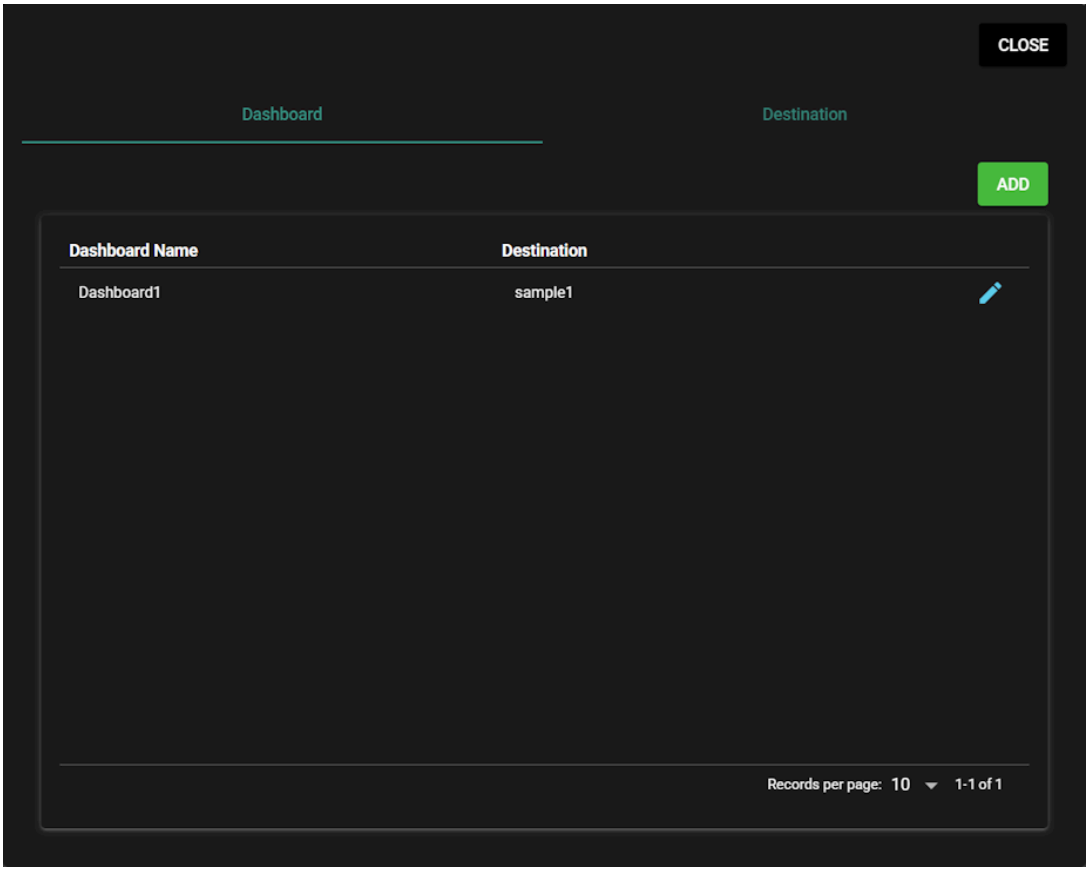
Note: If the validation succeeds, you can skip this step.



b. If you generate a new access token, a dialog will appear. Press “Yes.”
Note: If you generate a new access token and update it, you will need to update the settings of the DB query service and restart it. If you need to update and start DB query service manually, see “Starting DB Query Service.”
If you did not generate a new access token, you can skip this step.



5. You can confirm that the dashboard you just configured has been added to the dashboard settings list.

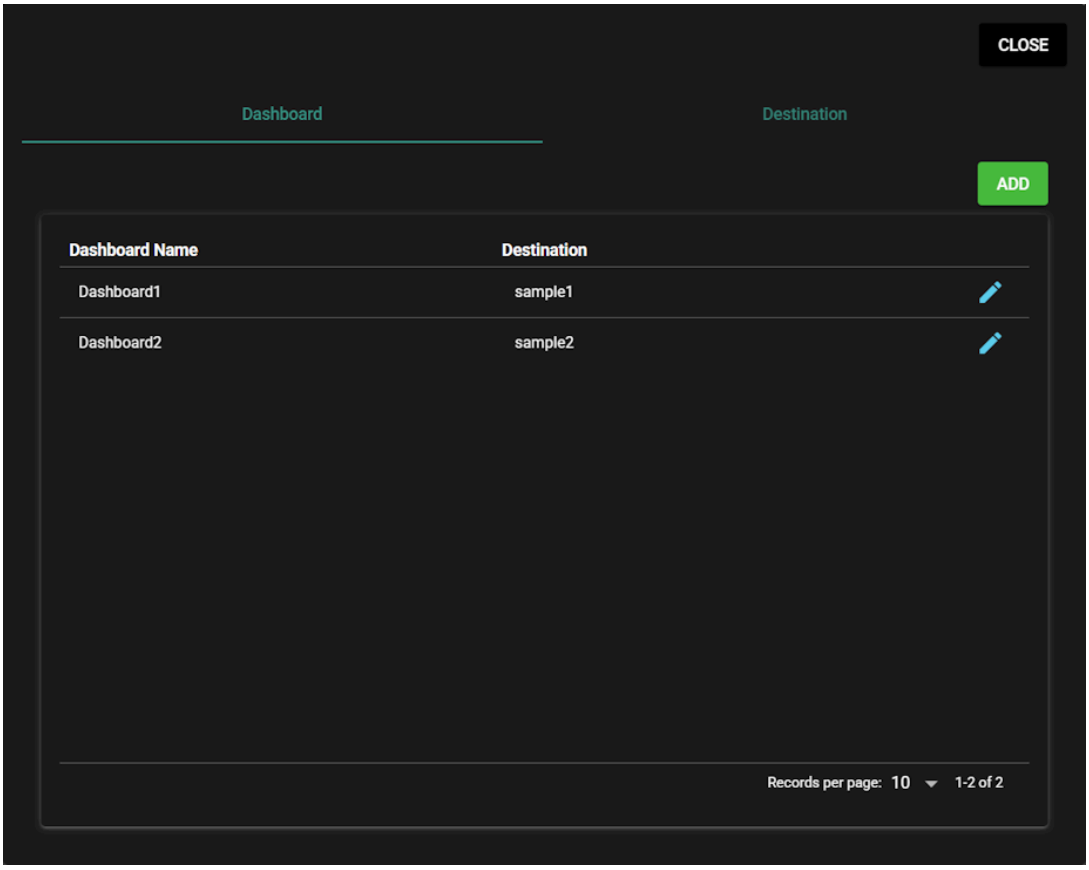


6.4.5.2 Modifying or Deleting Dashboard

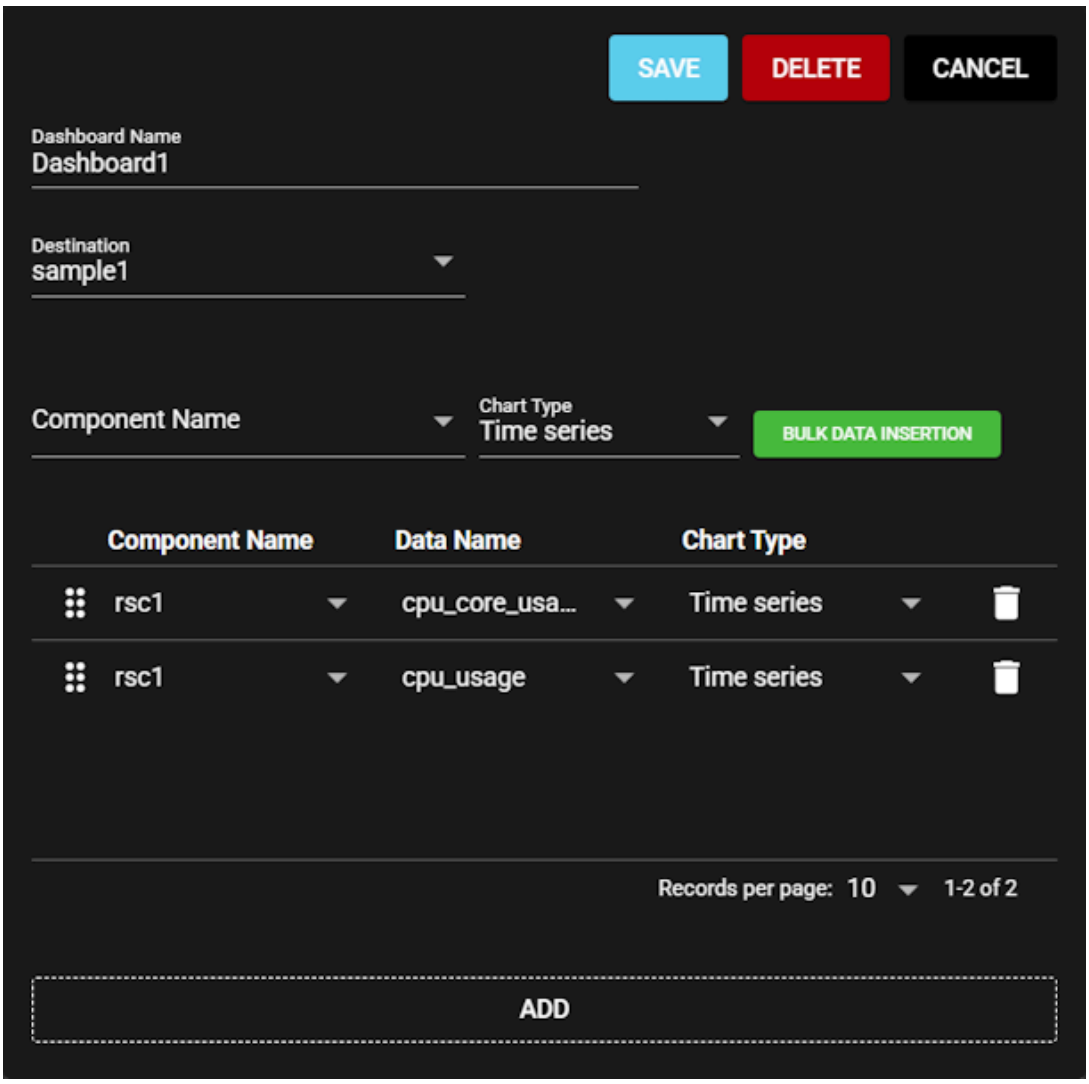
Edit

You can edit a dashboard settings by following the steps below.

1. Open the dashboard settings list and press the edit icon.



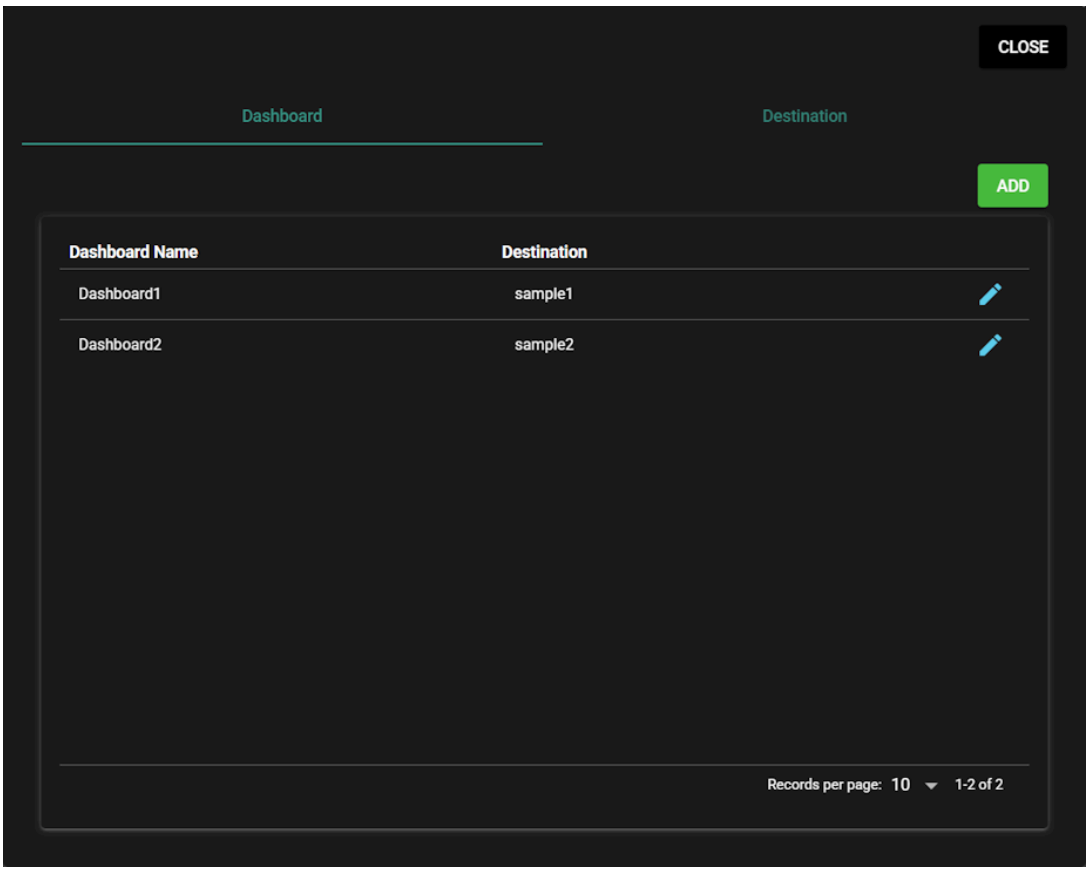
2. Enter the following items and press “Save.”
   For details on each item, see “Adding Dashboard.”



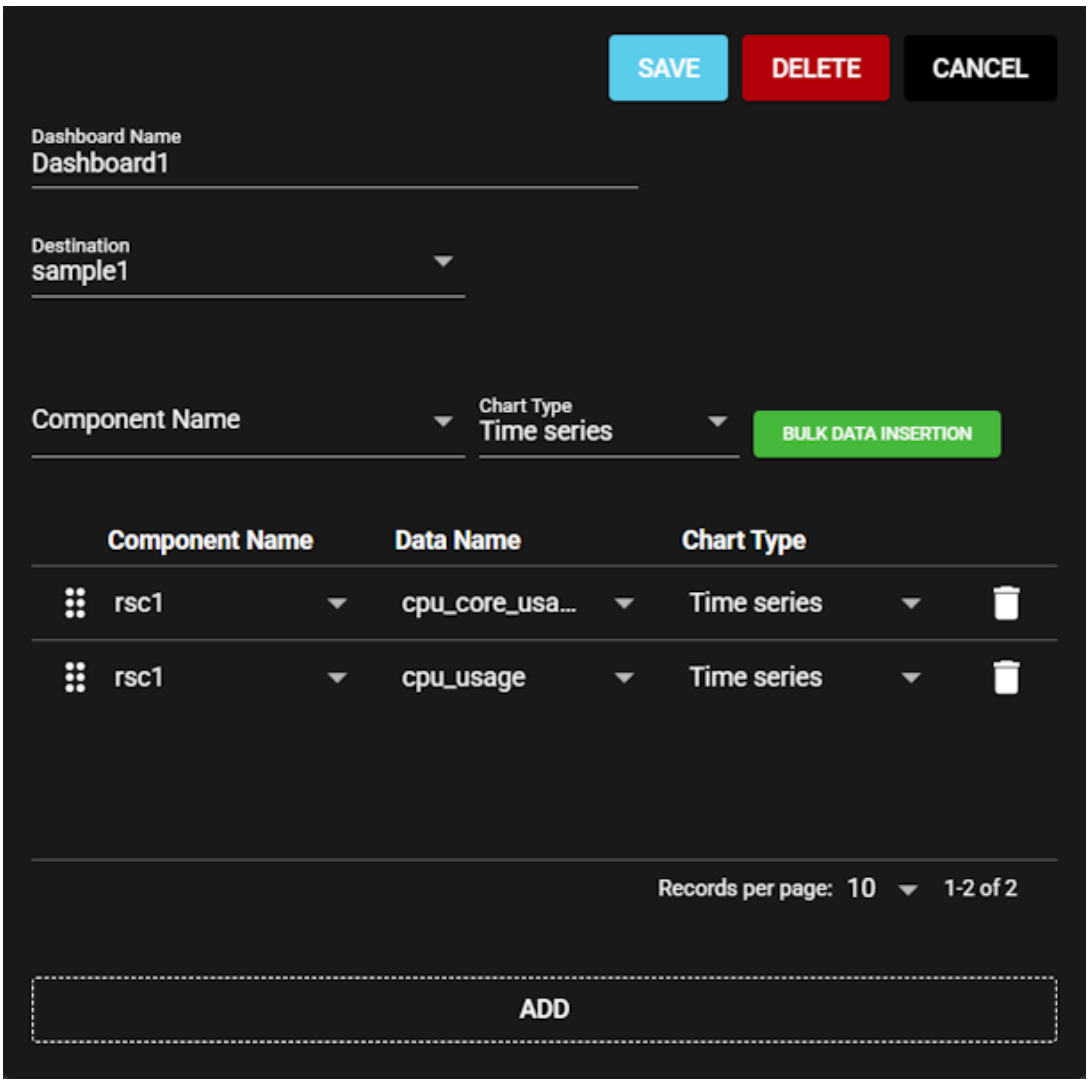
Delete

You can delete dashboard settings by following the steps below.

1. Open the dashboard settings list and press the edit icon.



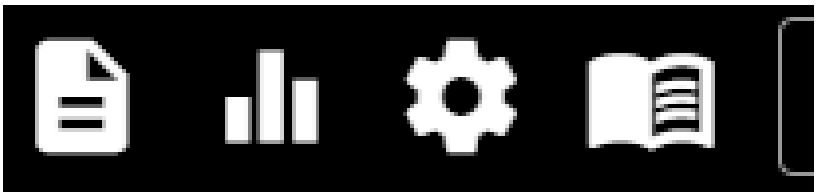
2. Press “Delete.”
   Note: If the destination settings you are deleting is associated with any resources on the Grafana side, those resources will also be removed when you delete the destination settings.



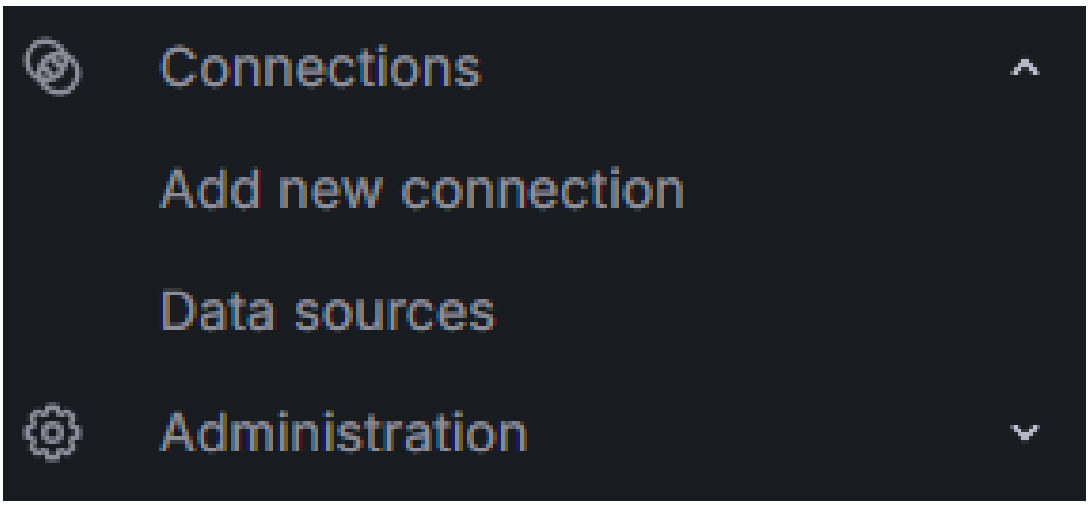
6.4.6 Checking in Grafana

Check the configured destination and dashboard in Grafana.

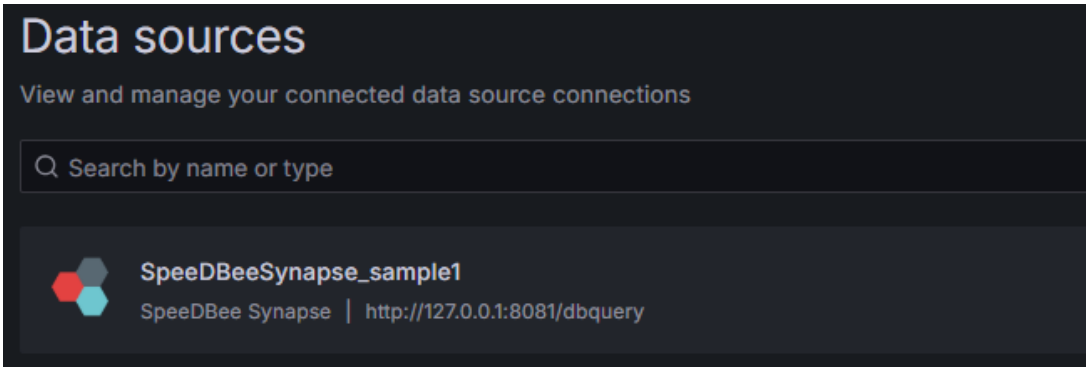
1. Press the graph icon in the header to open Grafana.



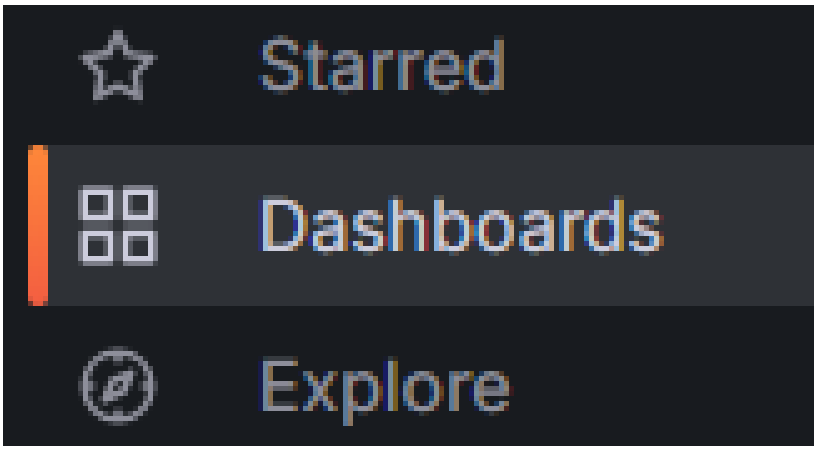
2. Press “Data sources” in the left menu.



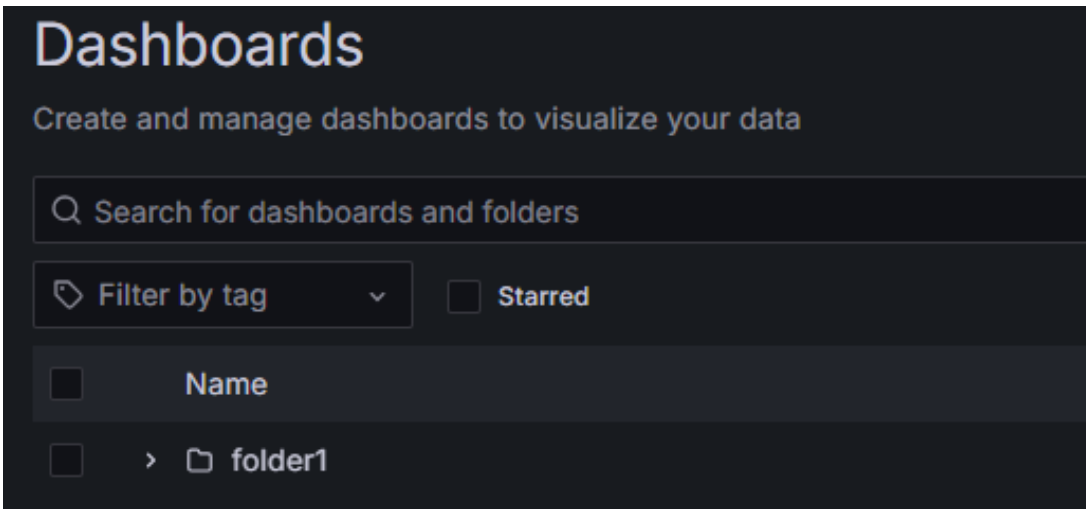
3. You can confirm that the destination configured in “Destination” has been registered as a data source.



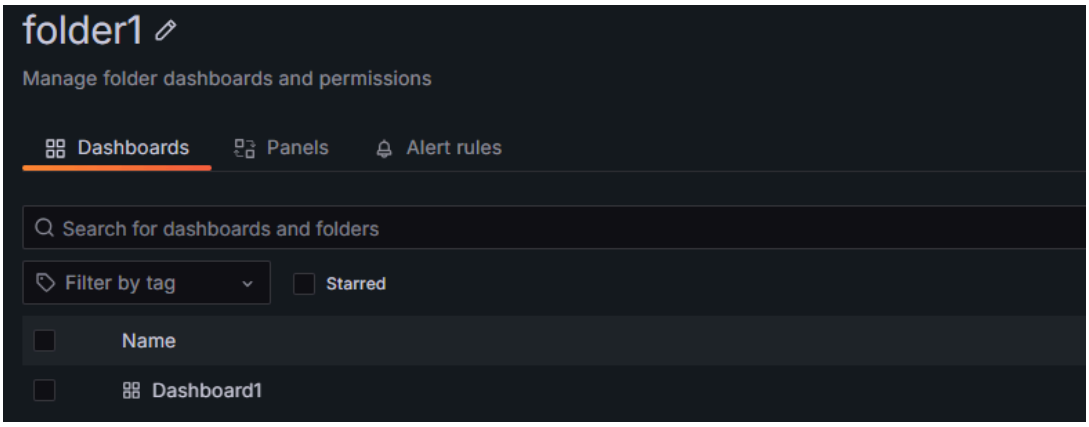
4. Press “Dashboards” in the left menu.



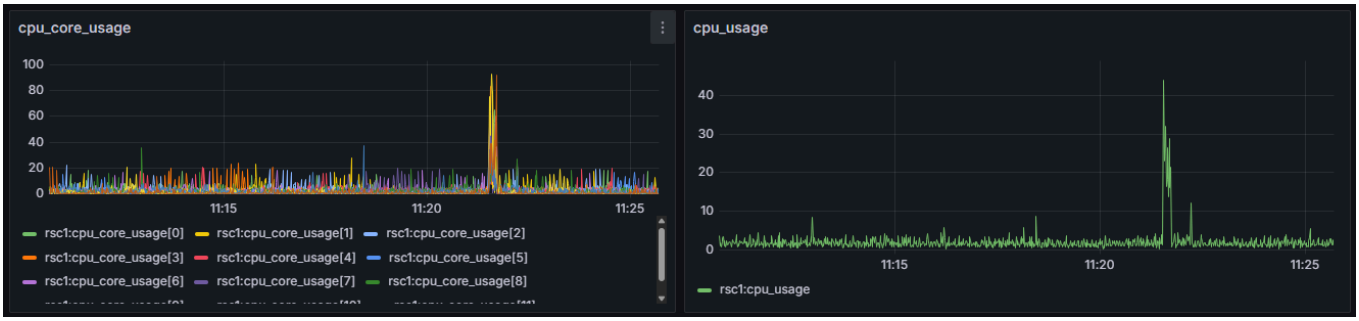
5. The folder configured in “Destination” exists, so select it.



6. Select the dashboard configured in “Dashboard”.



7. You can confirm the graph you configured.



6.4.7 Grafana and DB Query Service

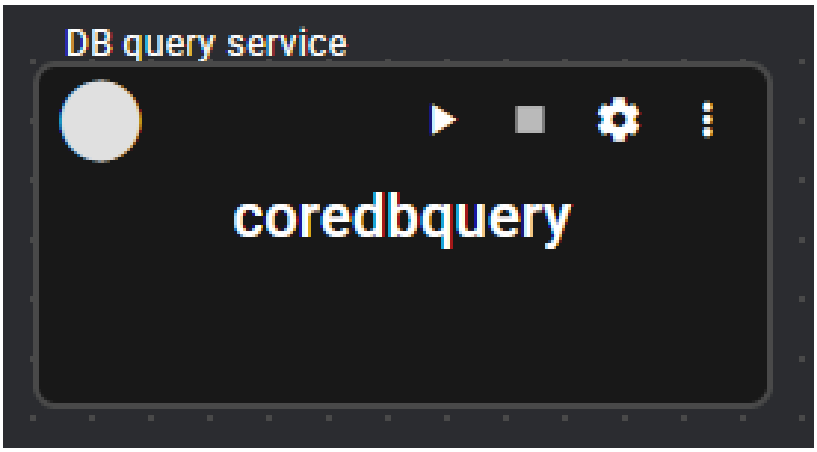
Grafana retrieves the data required to display graphs by using the DB query service provided by SpeeDBee Synapse. DB query service is one of the components and enables external access to data. For details, see “DB Query Service.”

6.4.7.1 Starting DB Query Service

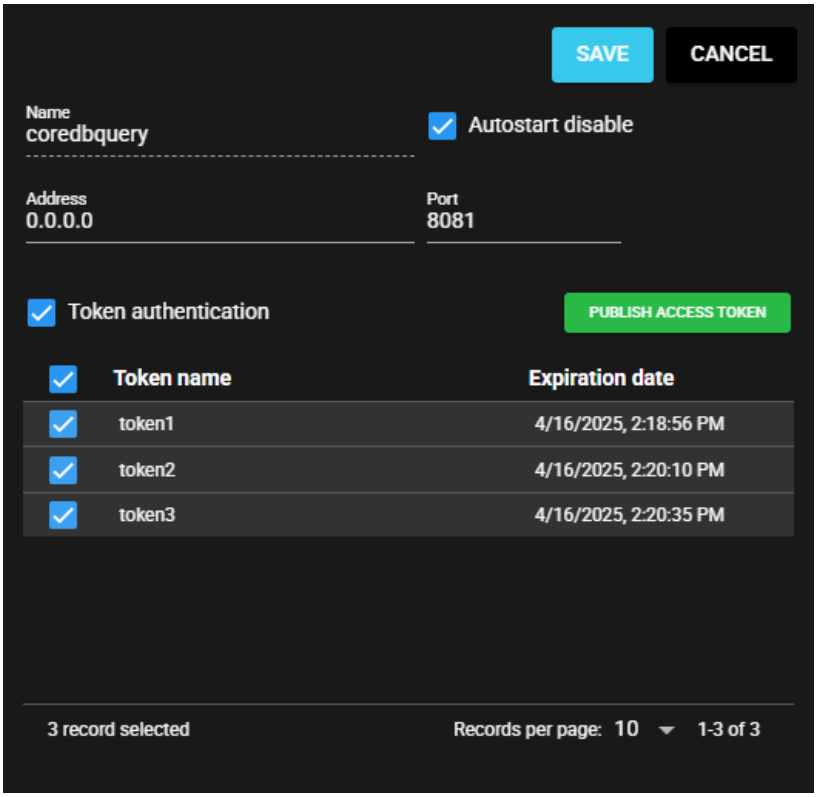
1. Press the panel tab of “System” to open the System panel.



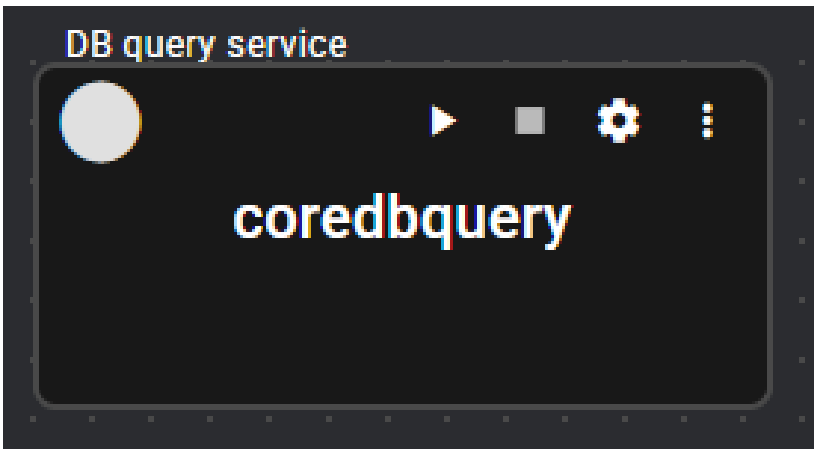
2. Press the settings icon of DB Query Service.



3. Turn on "Token authentication," select all tokens and press "Save."



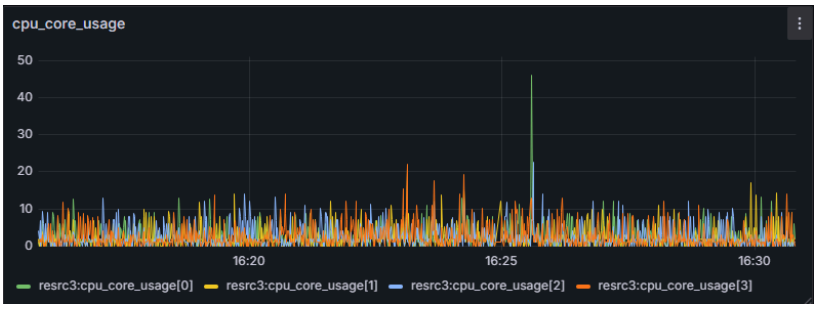
4. Press the Start Core DB Query icon. If it is already running, press the Stop icon and then the Start icon.



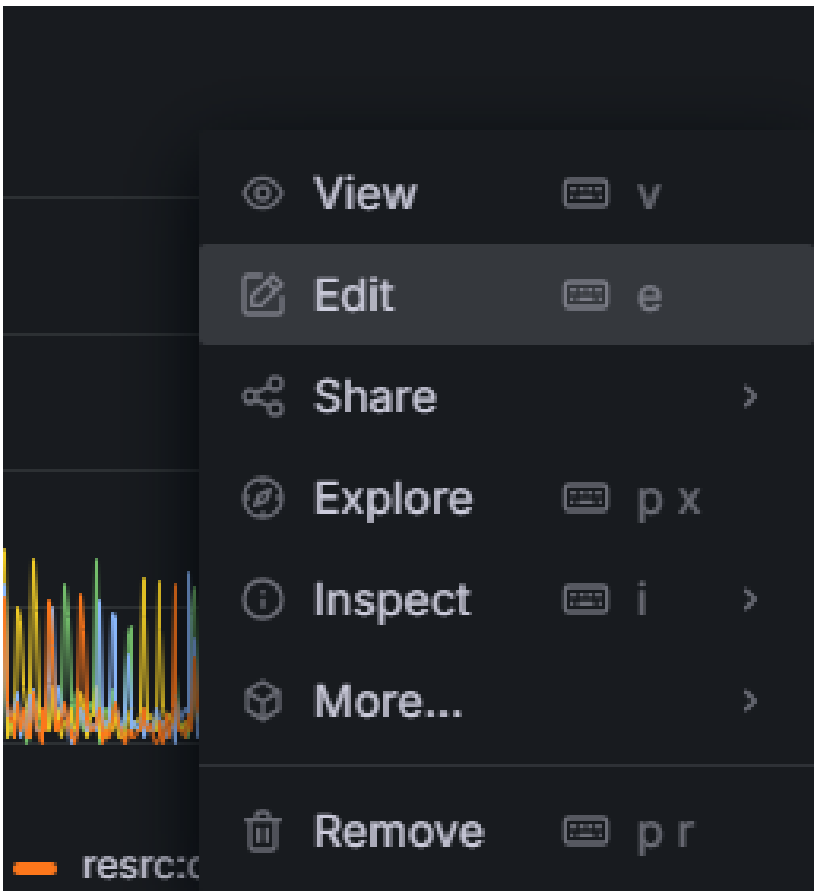
6.4.8 Changing the data count limit for a graph

If only a portion of the data is displayed, the number of data items may have exceeded the maximum. You can use the procedure below to change the maximum number of data items.

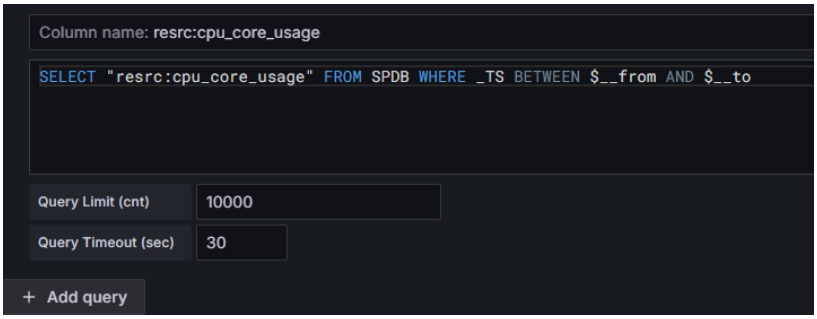
1. Execute the procedure shown in “Dashboard” to display a list of graphs.
2. Press the menu icon at the top right of the graph.



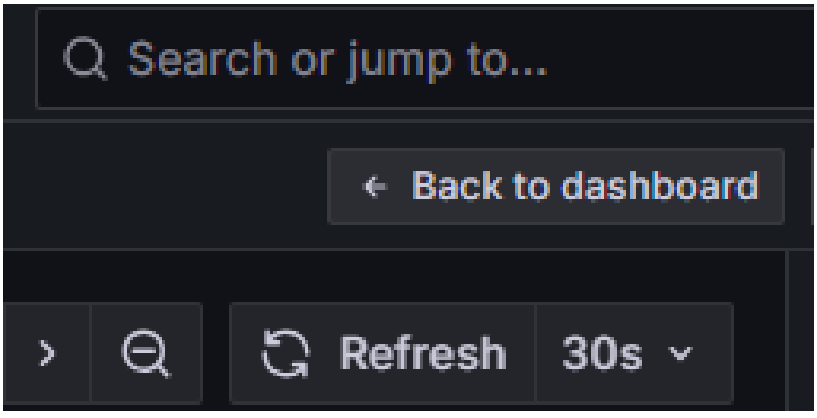
3. A menu appears. Select “Edit.”



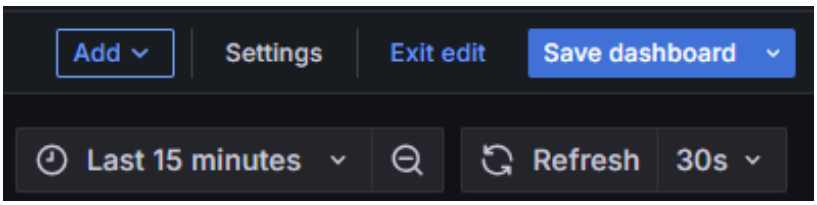
4. This displays the graph setting screen. The upper limit on the number of data items is set in the setting item "Query Limit (cnt)" on this screen. Changes to the values are reflected in the graph.
(The default value for “Query Limit (cnt)” is 10,000.)



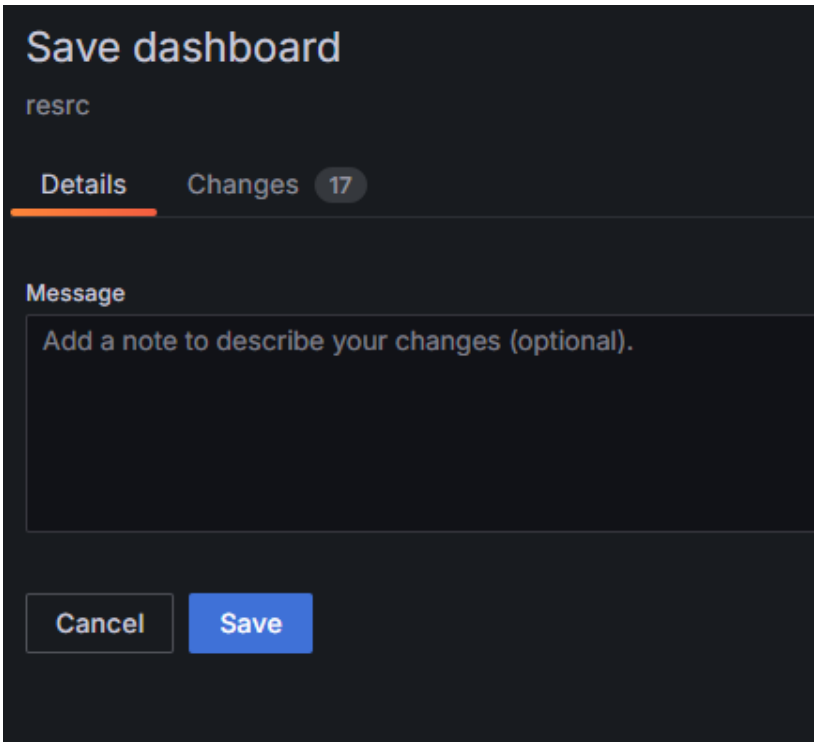
5. After changing the value, press the ESC key or the “Back to dashboard” icon in the upper right of the screen.



6. This returns the screen to the graph list screen. Press the “Save dashboard” icon at the top right of the screen.



7. Press “Save” to save the settings.

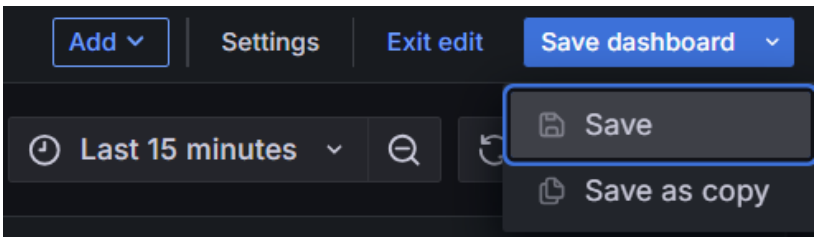


> **When Settings are Overwritten**
>
> After applying dashboard settings from SpeeDBee Synapse to Grafana, any changes made to the dashboard on Grafana will be overwritten when you run the update procedure again. For details, see Dashboard Overwrite.

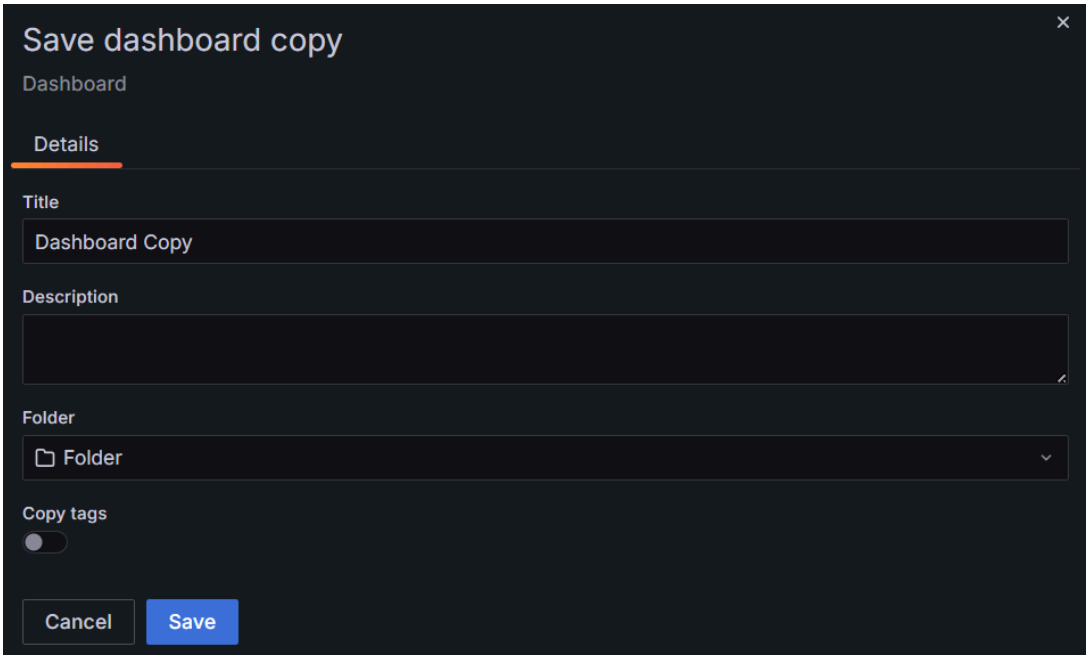
6.4.9 Dashboard Overwrite

When applying Dashboard settings from SpeeDBee Synapse to Grafana, any manually edited content on Grafana (such as changing data count limits) will be overwritten. If you need to preserve your manual customizations on Grafana, use one of the following methods.

1. **Copy the dashboard on Grafana**
   Open the target dashboard on Grafana, then click the "Save dashboard" dropdown menu and select "Save as copy".



Enter any information as the copy destination and click "Save". The copied dashboard will be outside the management of SpeeDBee Synapse.



By following the above steps, when SpeeDBee Synapse applies the settings, only the original dashboard will be updated, and you can preserve the existing dashboard.

2. **Add a separate dashboard on SpeeDBee Synapse**
   Follow the procedure in "Adding Dashboard" to add a new dashboard configuration to the same destination.

If you accidentally overwrite the dashboard

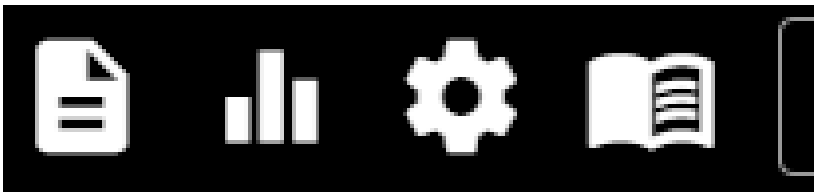
You can revert to a previous version using the version history in [Settings] > [Versions] on the Grafana dashboard.

6.4.10 Manually Setting a Grafana Data Source

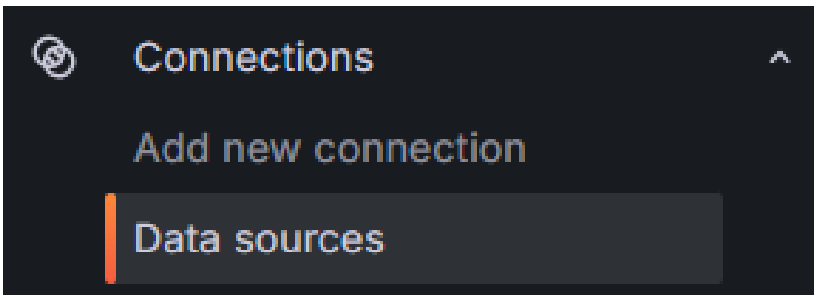
Learn how to manually interact with Grafana to view the data.
There is no need to configure this setting if it has already been configured in the “Destination”.

1. Issue an access token as described in “Publish access token”.
   Note: Keep the issued access token because it will be used later.
2. Restart the DB Query Service by following the procedure shown in “Start the DB Query Service”.

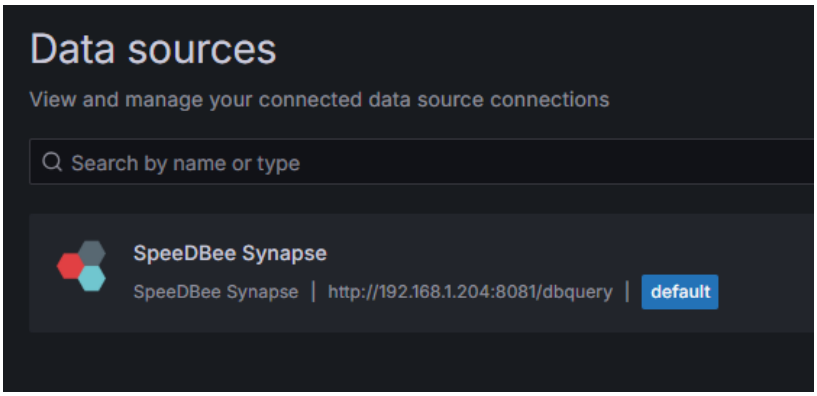
3. Press the Grafana icon in the header to open the Grafana screen.



4. Hover over the “Connections” icon in the left menu and select “Data sources” from the menu that appears.

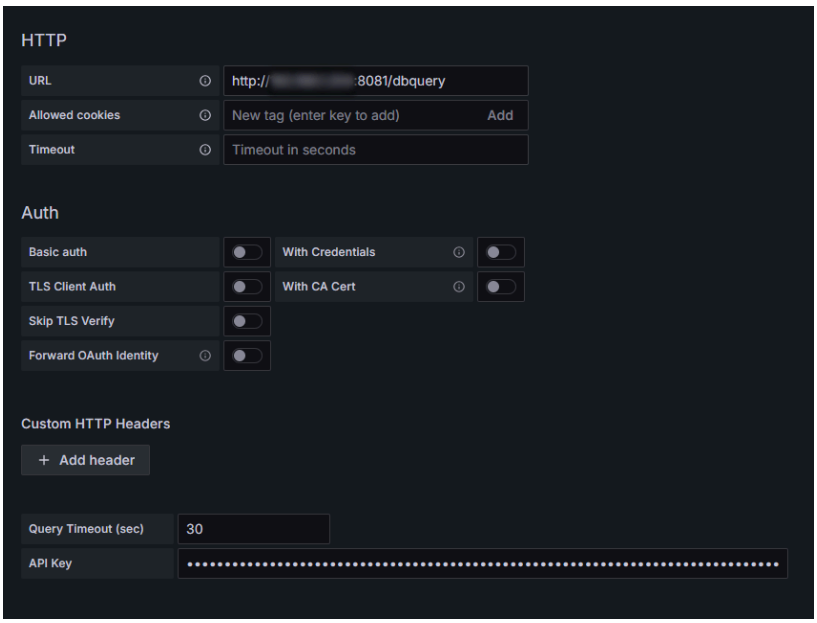


5. Select “SpeeDBee Synapse.”

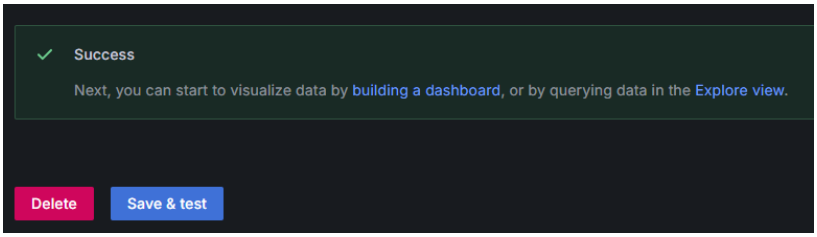


6. Enter the following items.

Item	Description
URL	Enter "http://{IP address of the environment where SpeeDBee Synapse is running}: {port where the DB query service is running}/dbquery." Note: You can check or change the port on which the DB query service by accessing the “Port” setting item of the DB query service component. For details, see “DB Query Service”.
API Key	Enter the access token you issued in the previous step.



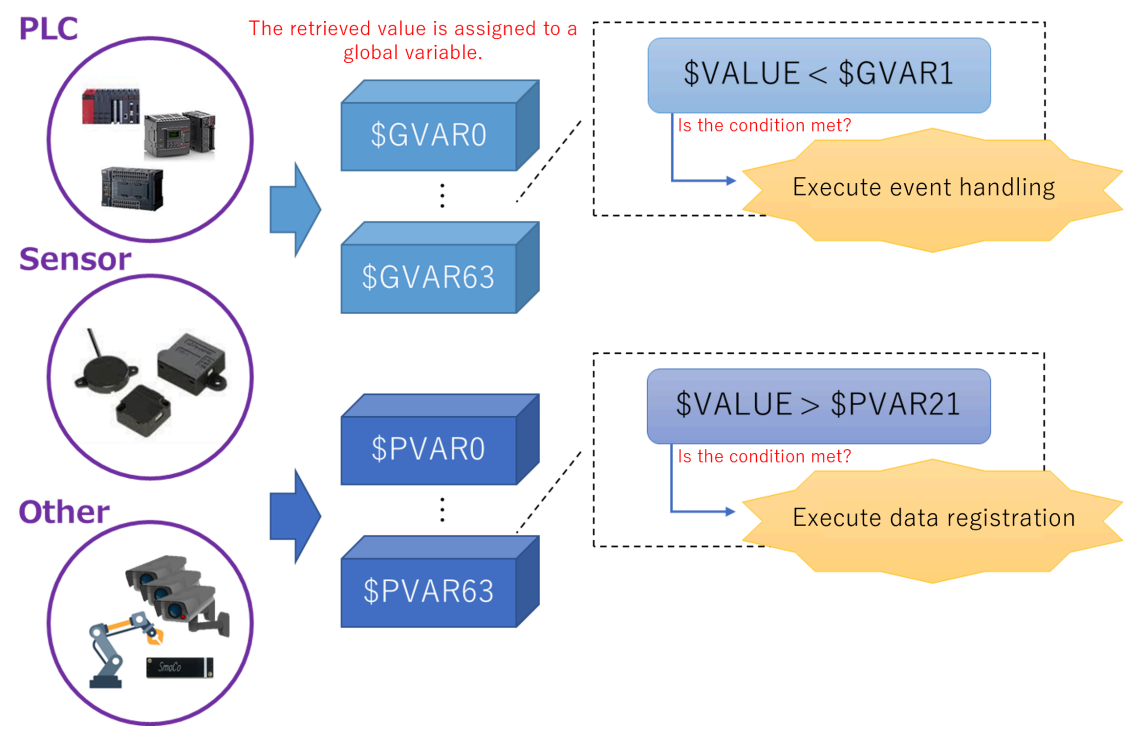
7. When you press “Save & test” and the message "Success" appears, the data source configuration is complete.



6.5 Global Variables

6.5.1 Overview of Global Variables

Global variables store frequently changing data in the variables on the memory. These variables can be used in various functions of SpeeDBee Synapse, such as conversion, conditions, and arithmetic and conditional expressions of components, and SQL.
This prevents, for example, updates of conditional expression thresholds during system operation and registration of other data by specific data values.



Global variables can be used in arithmetic expressions, conditional expressions, and SQL. See here for information on arithmetic and conditional expressions.
See here for information on SQL.

6.5.2 Syntax of Global Variable

There are two ways to specify a global variable: one is using the standard global variable and the other is using the persistence global variable. Both can use up to 64 variables.
With standard global variable, once the core of SpeeDBee Synapse is stopped (such as when an operating machine is restarted), the value is lost. The persistence global variable allows for reading of the final value at next startup even when the SpeeDBee Synapse core has been stopped.

- $GVAR0 to $GVAR63 (Global variable)

  GVAR (global variable) is initialized to 0 each time you restart the SpeeDBee Synapse core. This remains 0 until a value is assigned.

- $PVAR0 to $PVAR63 (Persistence global variable)

  PVAR (persistence global variable) is initialized to 0 when the SpeeDBee Synapse core is started for the first time. When a value is assigned, the value is retained even after core restart.

To use the value assigned at previous startup when the SpeeDBee Synapse is restarted, use a persistence global variable.

1. Assign method

```
GVAR0 := $VALUE
```

```
PVAR15 := ($VALUE * 20)
```

Note: The "$" is not necessary at the beginning of a global variable to be assigned.

- Notes on assignment in numeric conversions

```
PVAR0 := $VALUE * 2
```

When an assignment is specified as shown above, the value registered is $VALUE multiplied by 2. Note that the registered value changes.

- Notes on assignment in conditional expressions

```
GVAR0 := $VALUE
```

You cannot specify only assignment in a conditional expression. An assignment within a conditional expression must be specified as a conditional expression as shown below.

```
0 < (GVAR0 := $VALUE)
```

In this case, the condition is satisfied if $VALUE is greater than 0. Assignment to GVAR0 is constantly done.

2. Reference

```
$GVAR0 < $VALUE
```

```
$PVAR15 > ($VALUE + 20)
```

Note: The "$" is added to the beginning of a global variable to be referenced.

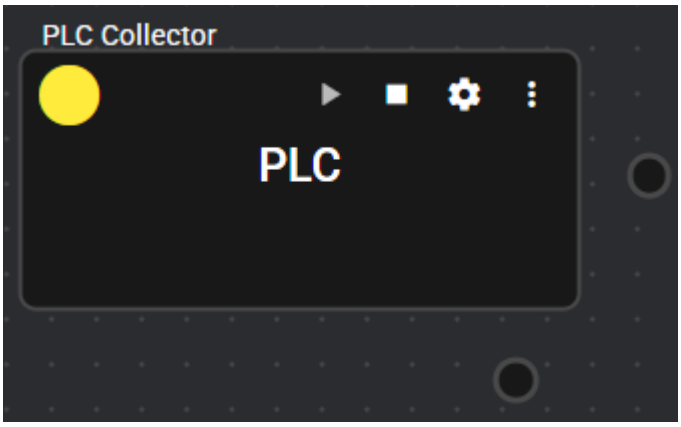
7 Troubleshooting

This chapter explains the countermeasures when troubles occur.

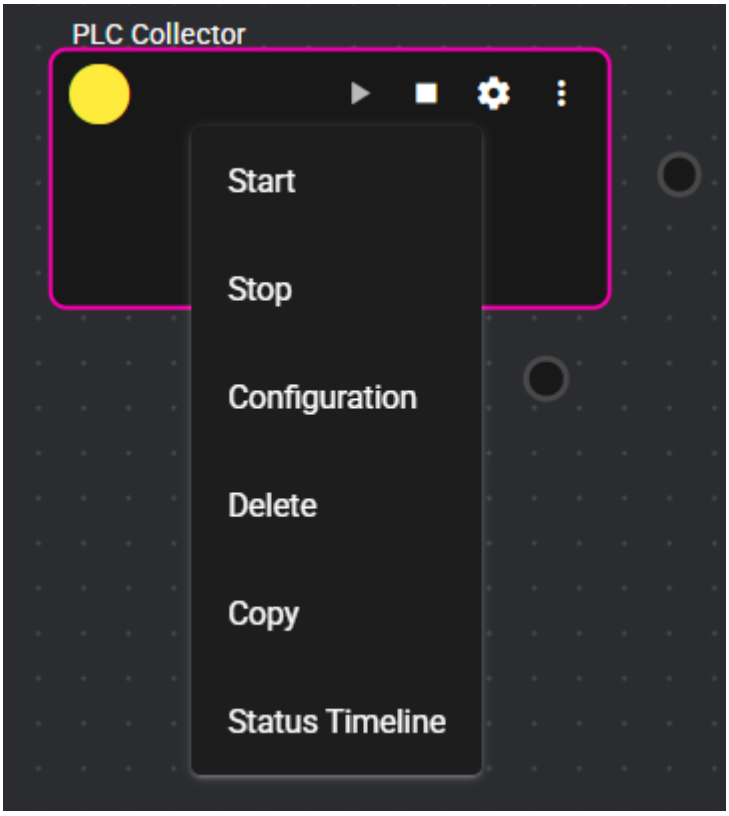
7.1 Common procedure when a trouble occurs

This section describes basic actions to take if you encounter problems with SpeeDBee Synapse.

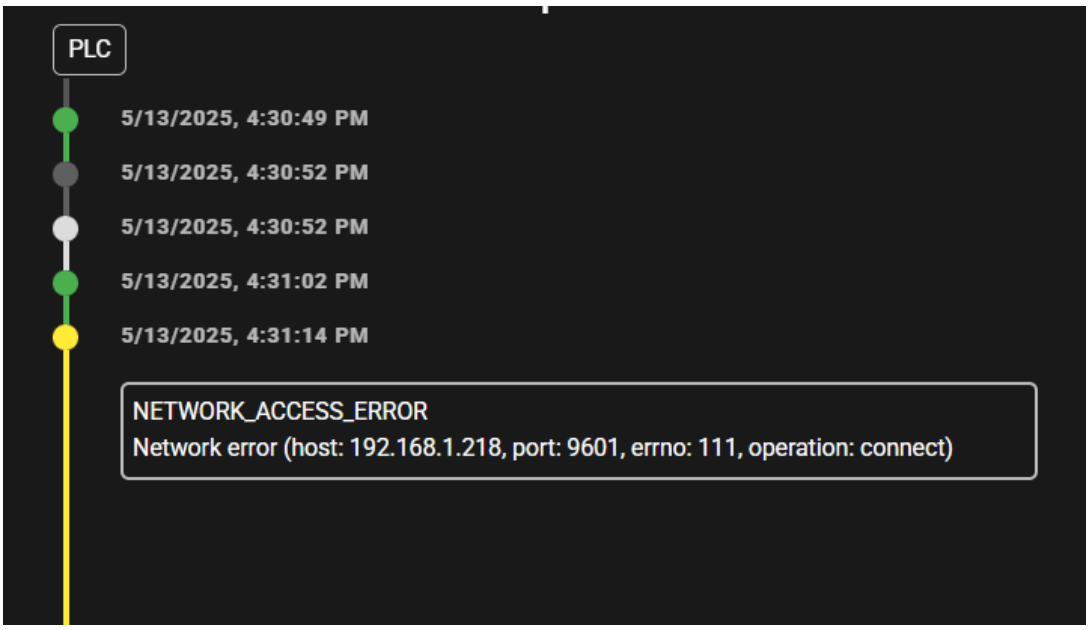
1. Right-click a component with a red-yellow status light.



2. Select “Status Timeline”.



3. Check the error code displayed in the timeline.



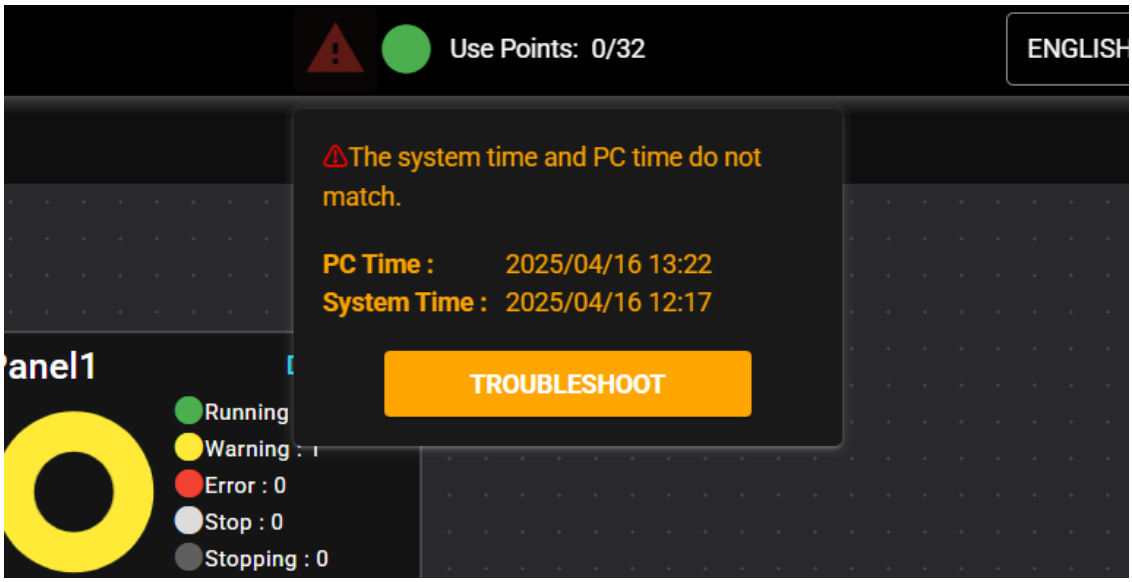
4. Use the search box in the upper right of this manual to search for the error code displayed in the timeline.
5. Understand the error code overview and take corrective action.
6. If the error persists after taking the corrective action, contact the manufacturer referring to "Contact manufacturer."

7.2 Common troubles and their solutions

This section explains the common troubles and the countermeasures for them.

7.2.1 Time Setting Problems and Workarounds

If the time on your browser differs significantly from the machine on which SpeeDBee Synapse is running, the following warning appears at the top of the screen:

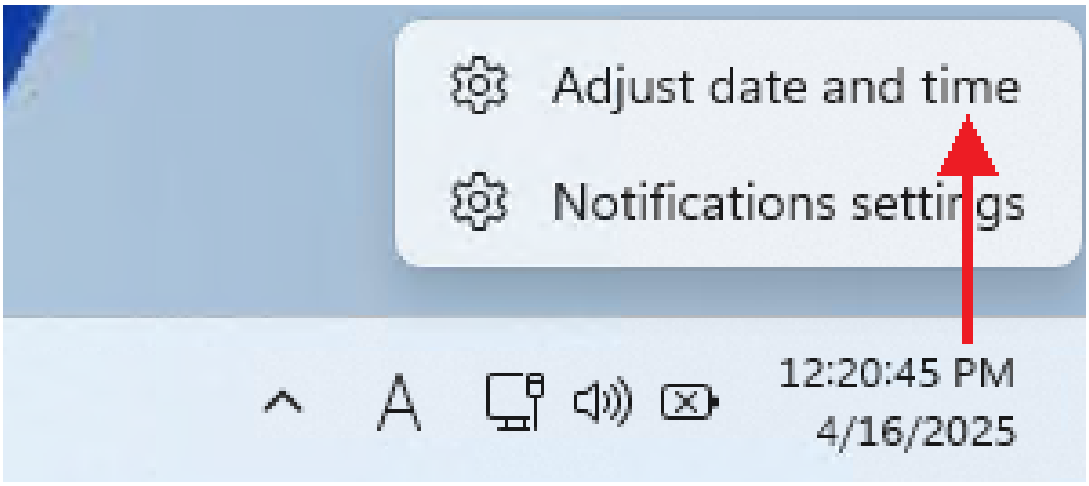


When this warning is displayed, SpeeDBee Synapse will fail during operation.
To avoid this, if there is a way to set the system time in the relevant environment, please align the settings with the PC time settings. For details, please refer to the manual for the relevant environment.

7.2.1.1 If SpeeDBee Synapse is running on Windows,

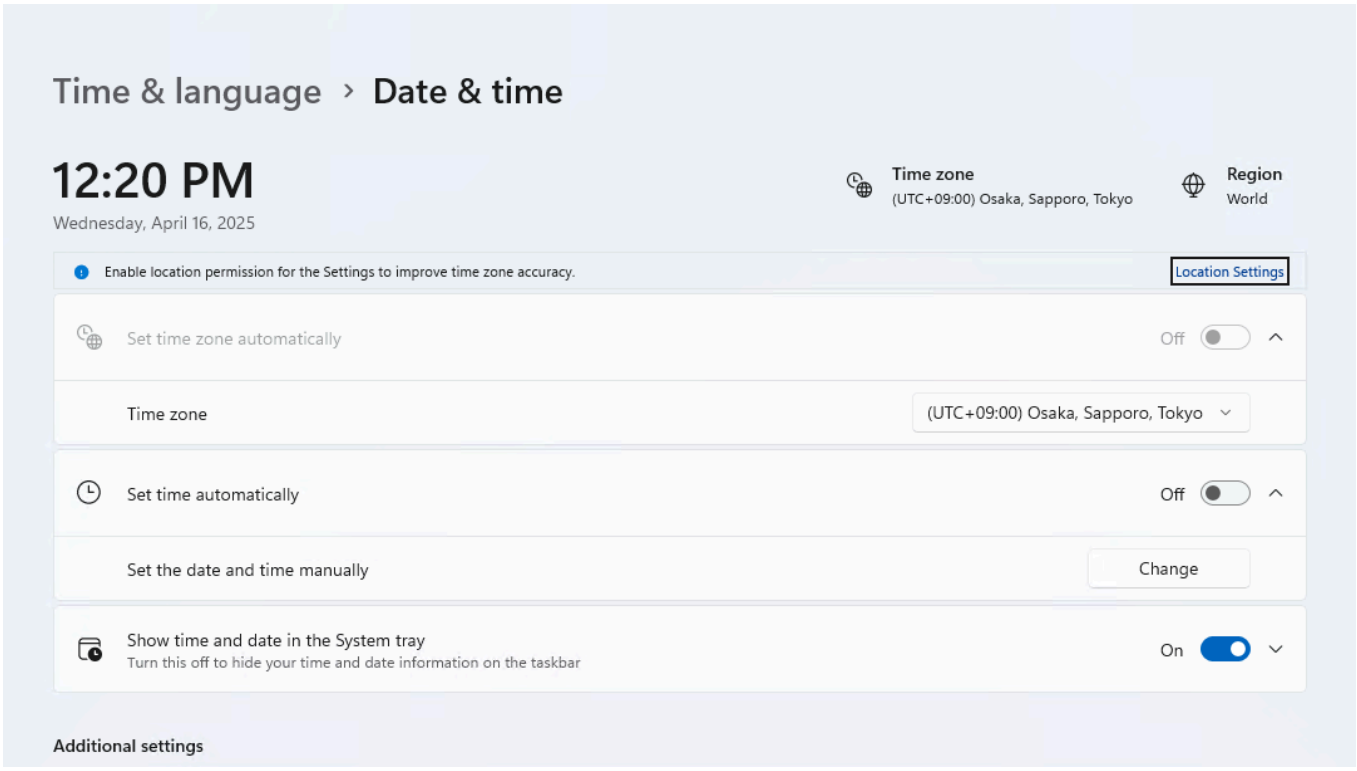
follow the steps below to configure Windows time settings.

1. On the machine where SpeeDBee Synapse is running, stop Synapse. See here for how to stop running.
2. Right-click the date and time at the far right of the taskbar, and then click “Adjust Date and Time”.



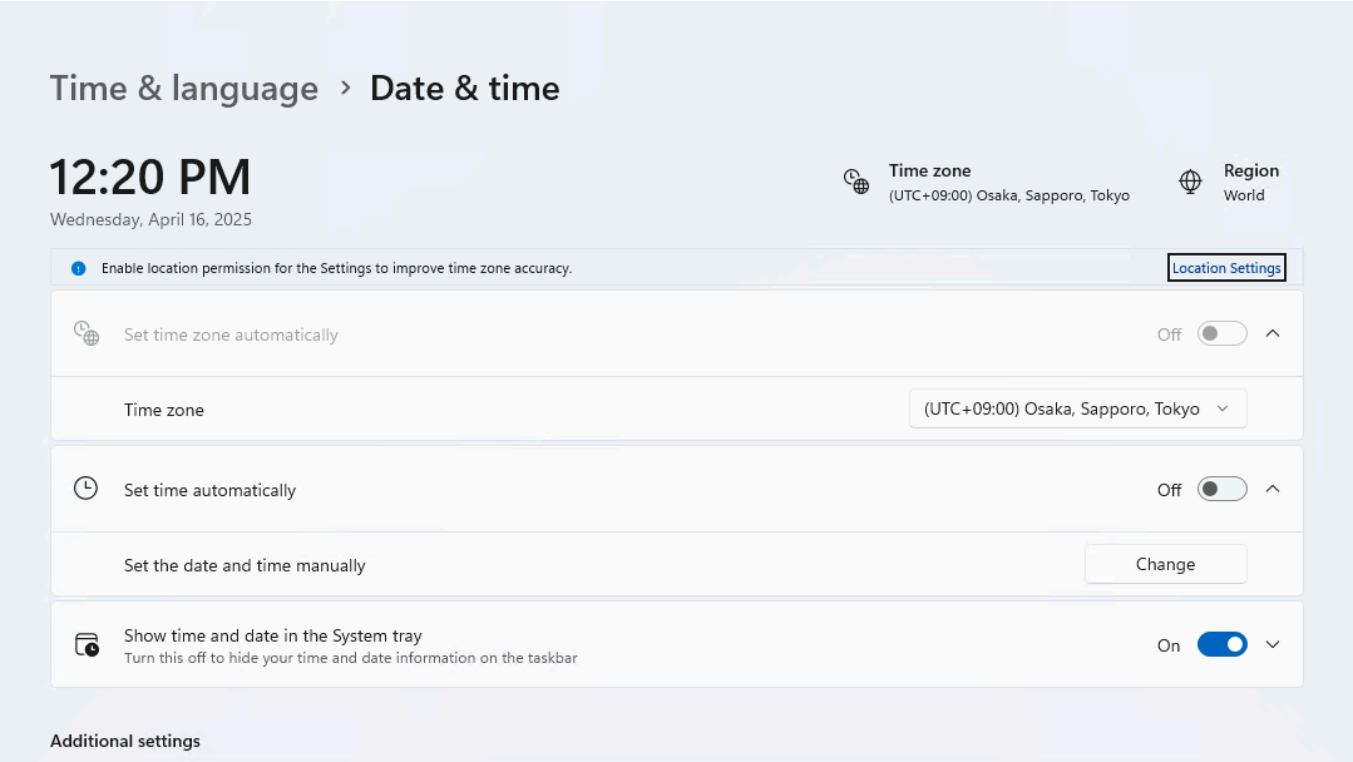
If the PC is connected to the Internet, proceed to the next step. If the PC is not connected to the Internet, proceed from step 4.

3. The “Date and Time” screen appears. If your PC is connected to the Internet, turn on “Set Time Automatically” and wait one minute. Then go to step 6.

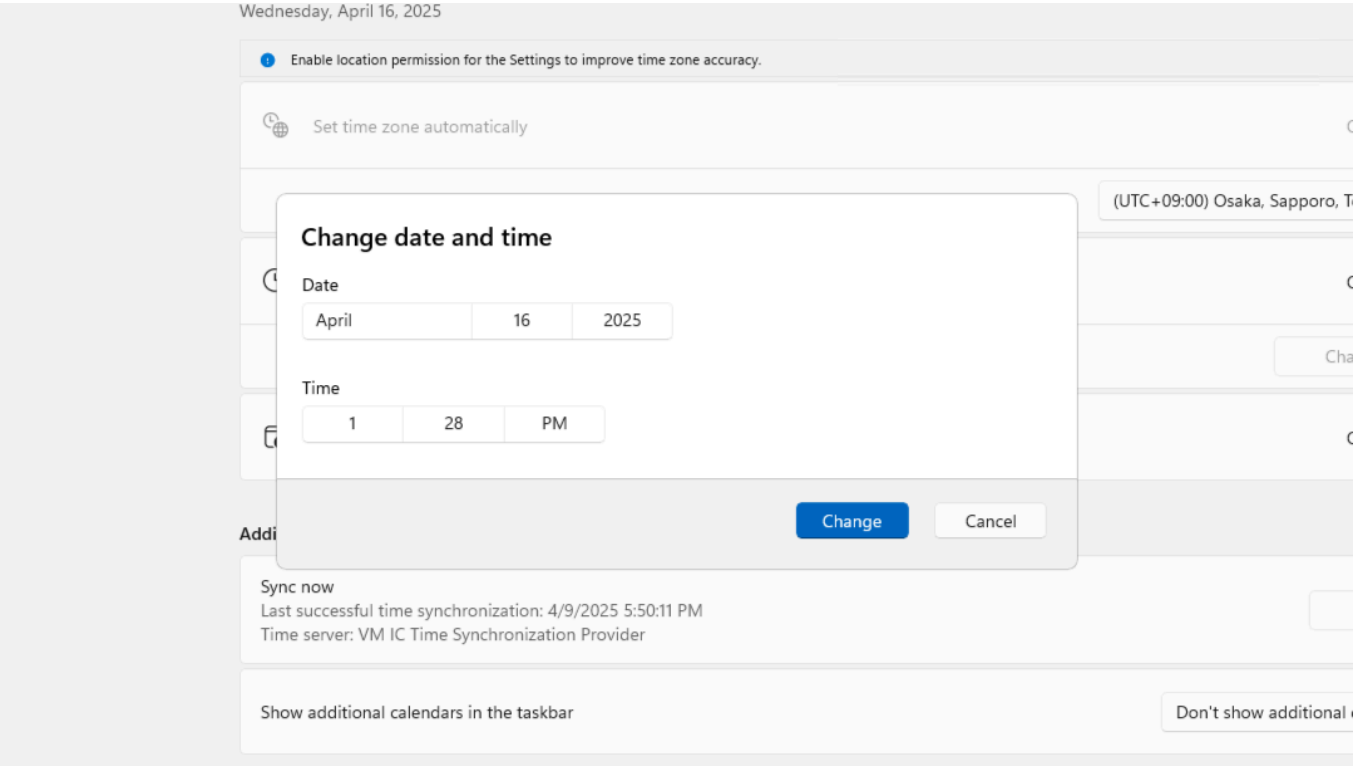


If automatic time synchronization does not work or does not solve the problem, or if your PC is not connected to the Internet, you may need to set the time manually. Then go to the next step.

4. When "Set time automatically" is turned off and the "Set date and time manually" button is selected, the time setting menu opens.

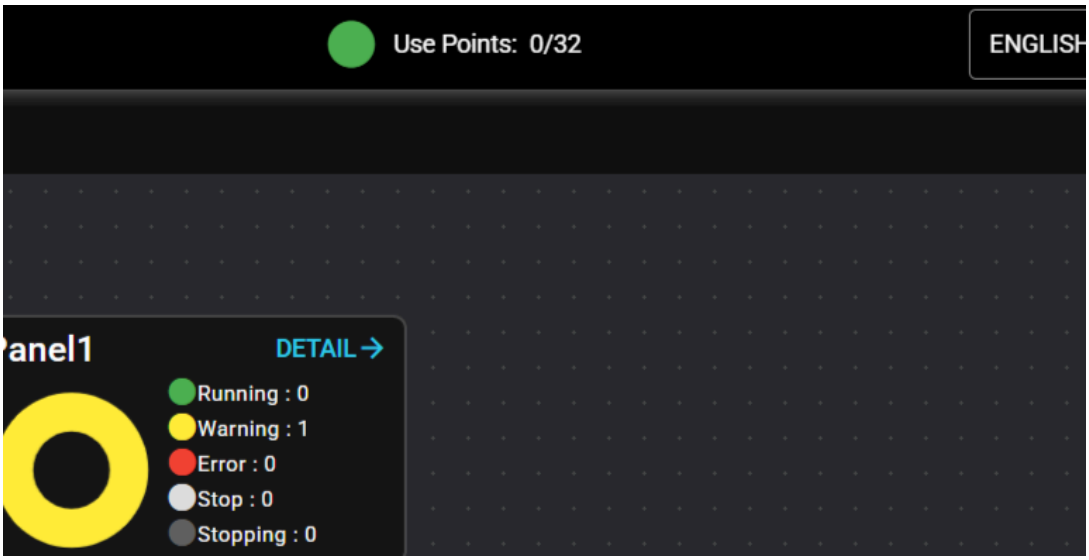


5. Set the current time in the menu, choose the “Change” button, and wait one minute.



6. On the machine where SpeeDBee Synapse is running, start Synapse. See here how to start running.

7. If the time is set correctly, the warning will no longer appear at the top of the SpeeDBee Synapse screen.



7.2.1.2 If SpeeDBee Synapse is Running on Linux,

follow the steps below to configure Linux time settings.

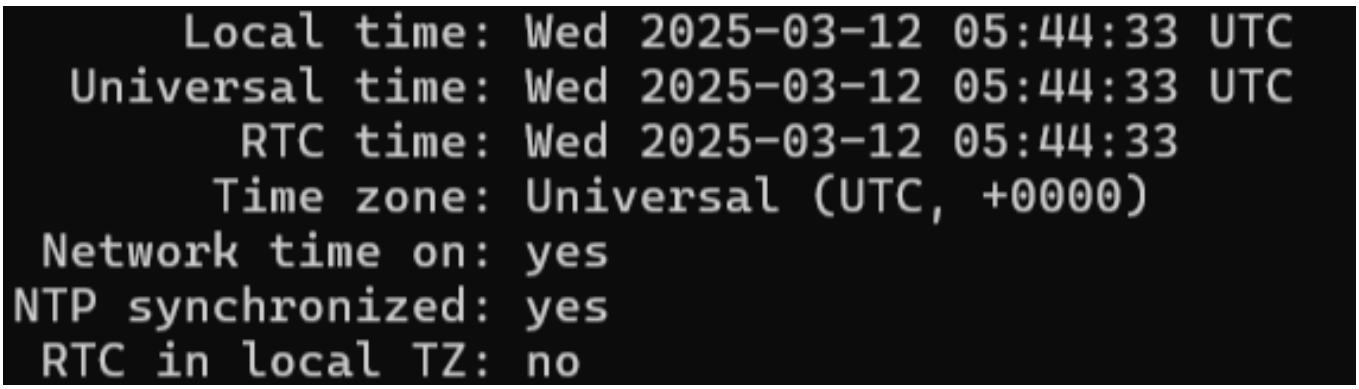
> In the Data Flow Controller edition, you can configure this in the WebUI of Data Flow Controller.

1. On the machine where SpeeDBee Synapse is running, stop Synapse. See here for how to stop running.
2. Check the current time on Linux. The following command is used to check the time currently set in the system.

   command

   ```
   $ timedatectl
   ```

   Here is an example of the result with the time displayed: “Local time” is the currently set system time. If the time is wrong, you need to change the setting.

3. If necessary, change the time zone to the correct region. Since this setting is optional, skip this step if the time zone is correct or if you do not need to configure the setting.
   For example, to change the machine running SpeeDBee Synapse to JST, you should use the following command:

command

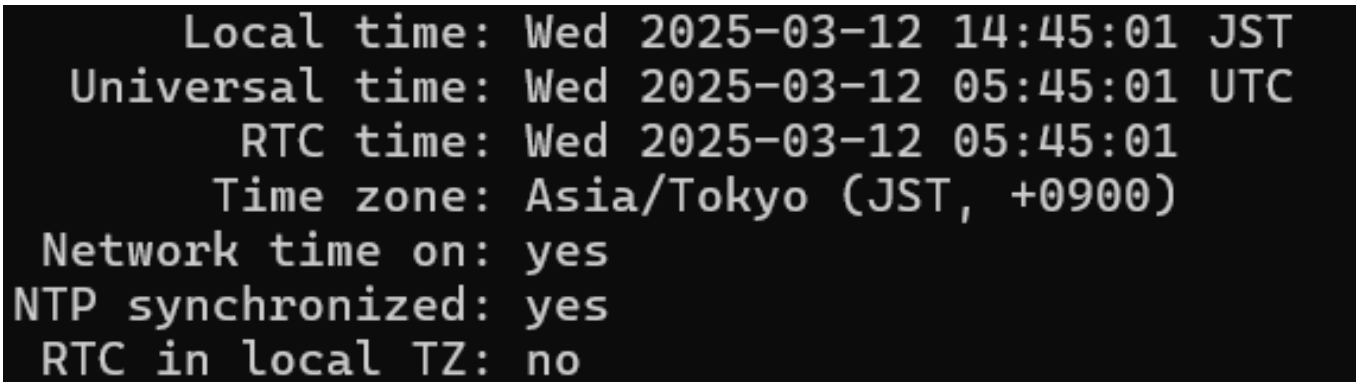
```
$ sudo timedatectl set-timezone Asia/Tokyo
```

If the time zone you want to set is not JST, you can use the following command to check the available time zones.

command

```
$ timedatectl list-timezones
```

The following examples show the results of changing the time zone. You can check the set value by using the command in step 1.



If the PC is connected to the Internet, proceed to the next step. If the PC is not connected to the Internet, go to step 5.

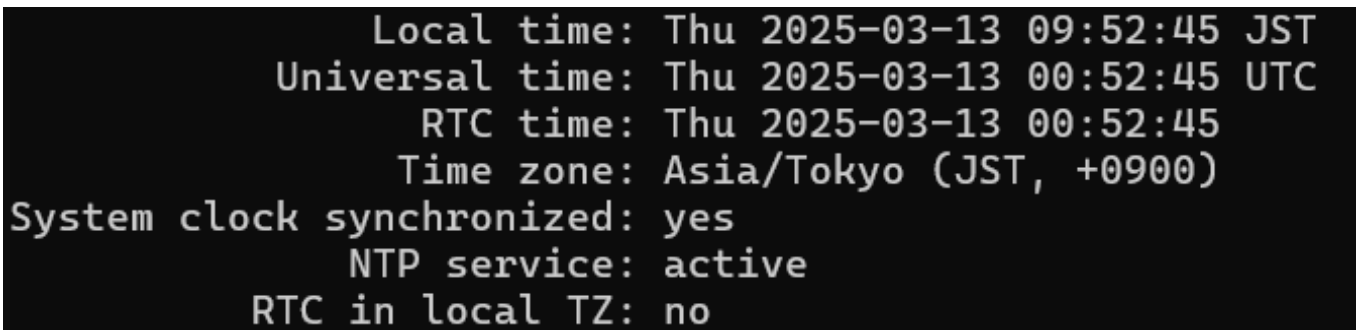
4. If the PC is connected to the Internet, enable NTP Server. After setting by using the command below, wait one minute and go to step 7.

command

```
$  sudo timedatectl set-ntp yes
```

The following example shows the result of enabling the NTP Server: You can check the set value by using the command in step 1.

If the OS is other than BlackBear



If the OS is BlackBear

```
      Local time: Wed 2025-03-12 15:48:42 JST
  Universal time: Wed 2025-03-12 06:48:42 UTC
        RTC time: Wed 2025-03-12 06:48:42
       Time zone: Asia/Tokyo (JST, +0900)
 Network time on: yes
NTP synchronized: yes
 RTC in local TZ: no
```

If synchronizing the NTP server does not solve the problem, or if your PC is not connected to the Internet, you may need to set the time manually. Then go to the next step.

5. To set the time manually, disable NTP server. It can be set using the following command:

command

```
$ sudo timedatectl set-ntp no
```

The following example shows the result of disabling the NTP Server: You can check the set value by using the command in step 1.

If the OS is other than BlackBear

```
               Local time: Wed 2025-03-12 16:03:22 JST
           Universal time: Wed 2025-03-12 07:03:22 UTC
                 RTC time: Thu 2025-03-13 00:57:15
                Time zone: Asia/Tokyo (JST, +0900)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
```

If the OS is BlackBear

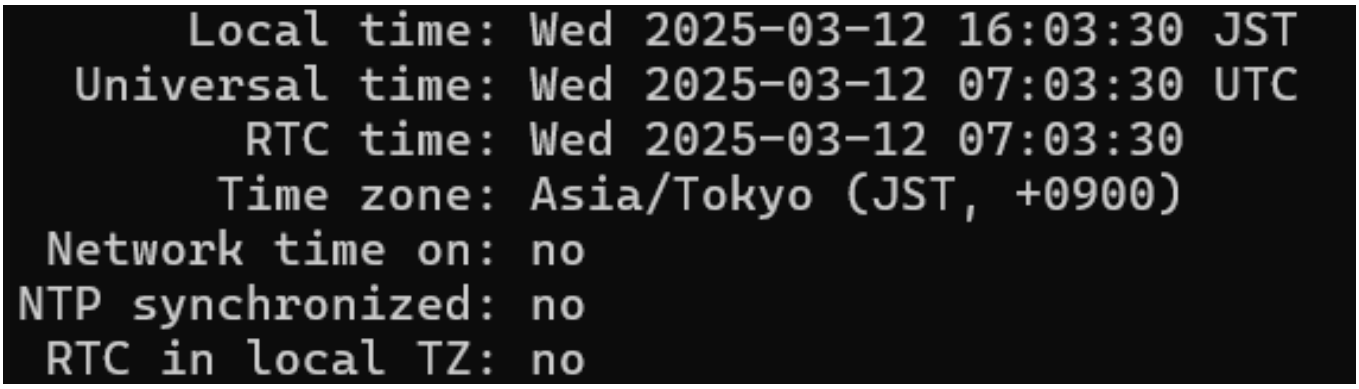
```
      Local time: Wed 2025-03-12 14:42:04 JST
  Universal time: Wed 2025-03-12 05:42:04 UTC
        RTC time: Wed 2025-03-12 06:48:09
       Time zone: Asia/Tokyo (JST, +0900)
 Network time on: no
NTP synchronized: no
 RTC in local TZ: no
```

6. Then change the date and time. The following example shows how to set the date to 16:03:20 on March 12, 2025. In actual use, set the date and time to the desired values. Wait 1 minute after setting.

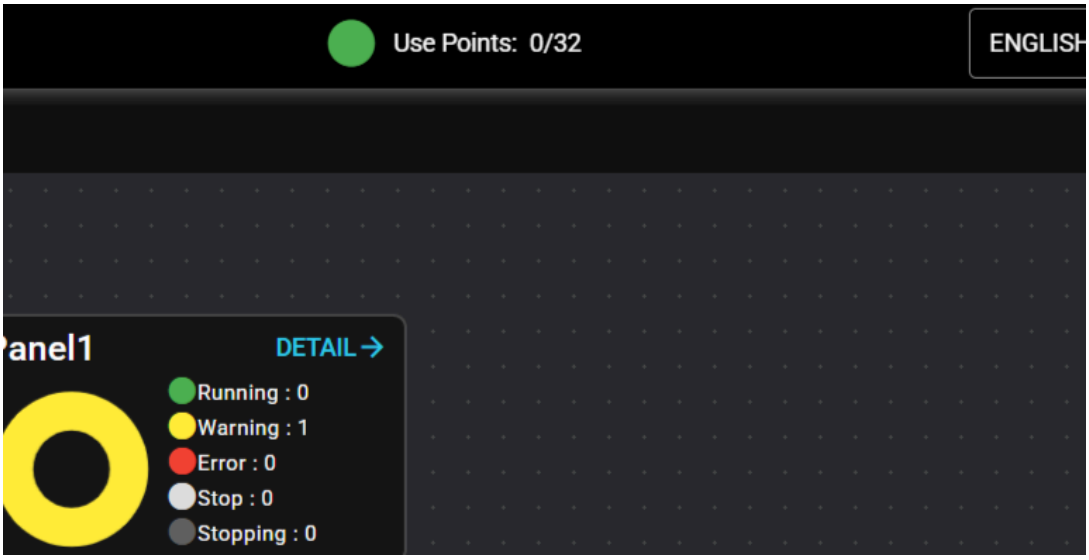
command

```
$ sudo date -s "2025-3-12 16:03:20"
```

Here is an example of the result of changing the date and time: You can check the set value by using the command in step 1.



7. On the machine where SpeeDBee Synapse is running, start Synapse. See here how to start running.
8. If the time is set correctly, the warning will no longer appear at the top of the SpeeDBee Synapse screen.



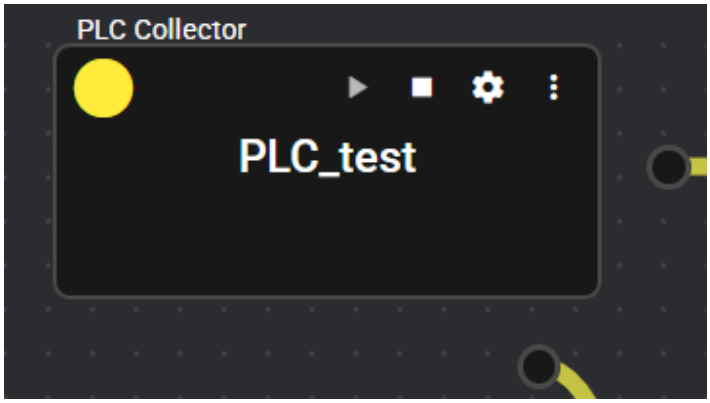
7.2.1.3 Notes on Time Settings

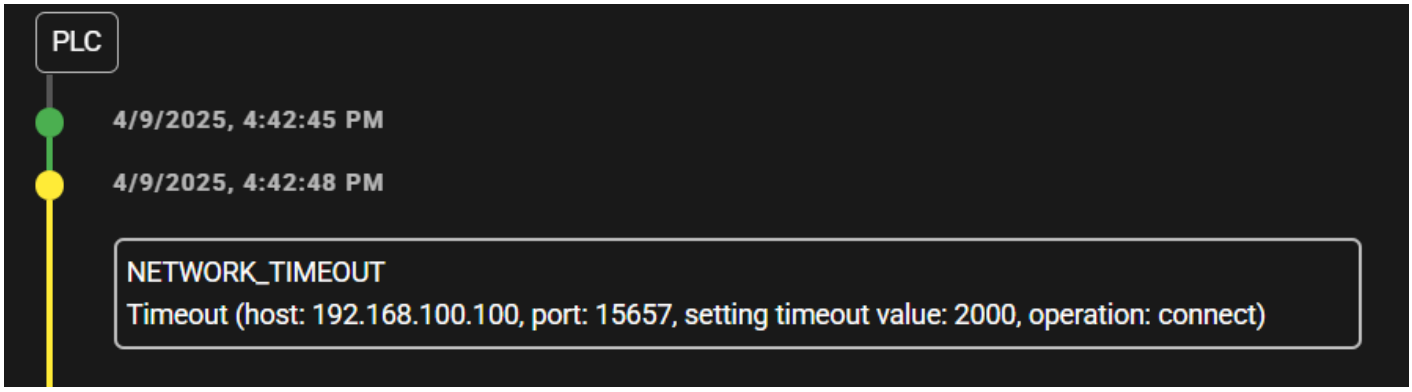
You can avoid problems with time settings by following the procedures described in this paragraph. However, machines that are not connected to the Internet or that do not have a Real Time Clock (RTC) will need the same setting after rebooting. Therefore, we recommend that you consider one of the following:

- Establishing proper time synchronization over an Internet connection (only when there are no security concerns)
- Set up an NTP Server within the LAN where the operating machine is working, and automatically synchronize.
- Prevent clock slippage by using RTC module.

7.2.2 Workaround when you cannot connect to the PLC collector

If the PLC collector cannot be connected, the component displays a yellow status light and the status timeline displays NETWORK_TIMEOUT.
See here how to check the status.





7.2.2.1 Checking IP Address

Confirm the IP address of the PLC being used.

> **About Devices That Can Be Used for Confirmation**
>
> The procedures in this section do not need to be performed on the device running SpeeDBee Synapse. You can also perform them on the PC you are using to view the WebUI of SpeeDBee Synapse.
> However, it is more reliable to perform these steps from a PC on the same network as the device running SpeeDBee Synapse.
> Note that in the Data Flow Controller edition, you can configure this in the WebUI of Data Flow Controller.

1. On a networked PC, open PowerShell (or a command prompt) on Windows, while on Linux, run the following command from the terminal.

For Windows

```
PS >ping IP address of target PLC
```

- Success screen

```
Pinging 192.168.100.106 with 32 bytes of data:
Reply from 192.168.100.106: bytes=32 time=106ms TTL=61
Reply from 192.168.100.106: bytes=32 time=111ms TTL=61
Reply from 192.168.100.106: bytes=32 time=132ms TTL=61
Reply from 192.168.100.106: bytes=32 time=151ms TTL=61

Ping statistics for 192.168.100.106:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 106ms, Maximum = 151ms, Average = 125ms
```

- Failure screen

```
Pinging 192.168.100.100 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.100.100:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

On Linux

```
$ping IP address of target PLC
```

Note: Linux continues to send ping until you type [ctrl] +[C].

- Success screen

```
PING 192.168.100.106 (192.168.100.106) 56(84) bytes of data.
64 bytes from 192.168.100.106: icmp_seq=1 ttl=62 time=38.8 ms
64 bytes from 192.168.100.106: icmp_seq=2 ttl=62 time=38.2 ms
64 bytes from 192.168.100.106: icmp_seq=3 ttl=62 time=38.7 ms
64 bytes from 192.168.100.106: icmp_seq=4 ttl=62 time=38.3 ms
64 bytes from 192.168.100.106: icmp_seq=5 ttl=62 time=38.6 ms
64 bytes from 192.168.100.106: icmp_seq=6 ttl=62 time=38.9 ms
64 bytes from 192.168.100.106: icmp_seq=7 ttl=62 time=38.4 ms
^C
--- 192.168.100.106 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6009ms
rtt min/avg/max/mdev = 38.224/38.577/38.942/0.239 ms
```

- Failure screen

```
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.

--- 192.168.100.100 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3006ms
```

If ping communication cannot be confirmed, check the settings of your PLC side.

7.2.2.2 Checking Port

Confirm the port number of the PLC being used.

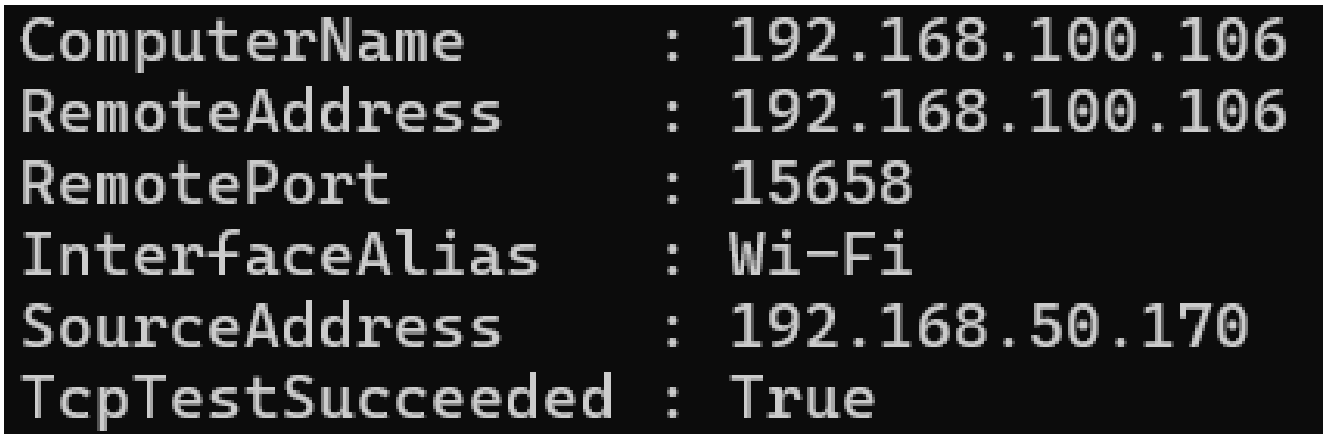
> **About Devices That Can Be Used for Confirmation**
>
> The procedures in this section do not need to be performed on the device running SpeeDBee Synapse. You can also perform them on the PC you are using to view the WebUI of SpeeDBee Synapse.
> However, it is more reliable to perform these steps from a PC on the same network as the device running SpeeDBee Synapse.

1. On a networked PC, open PowerShell (or a command prompt) on Windows, while on Linux, run the following command from the terminal.

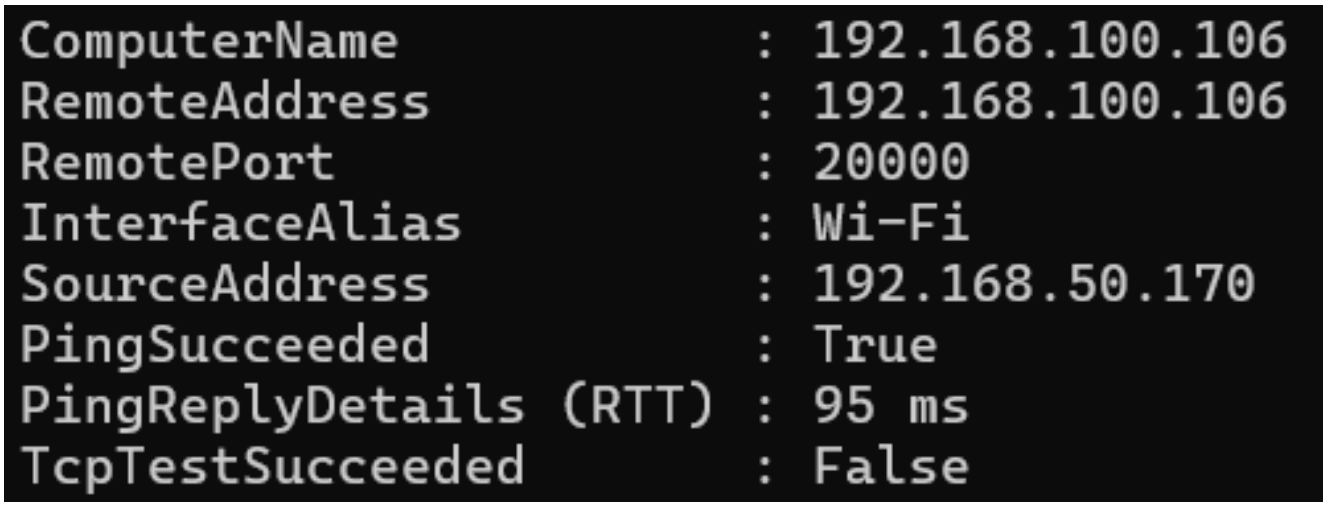
For Windows

```
PS >IP address of Test-NetConnection target PLC - Port number of Port target PLC
```

- Screen when the relevant port is open



- Screen when the relevant port is not open



On Linux

```
Port number of IP address target PLC of $nc-zv target PLC
```

- Screen when the relevant port is open



- Screen when the relevant port is not open



If the port cannot be confirmed to be open, check the settings of PLC side that you are using.

7.2.2.3 PLC Side Settings

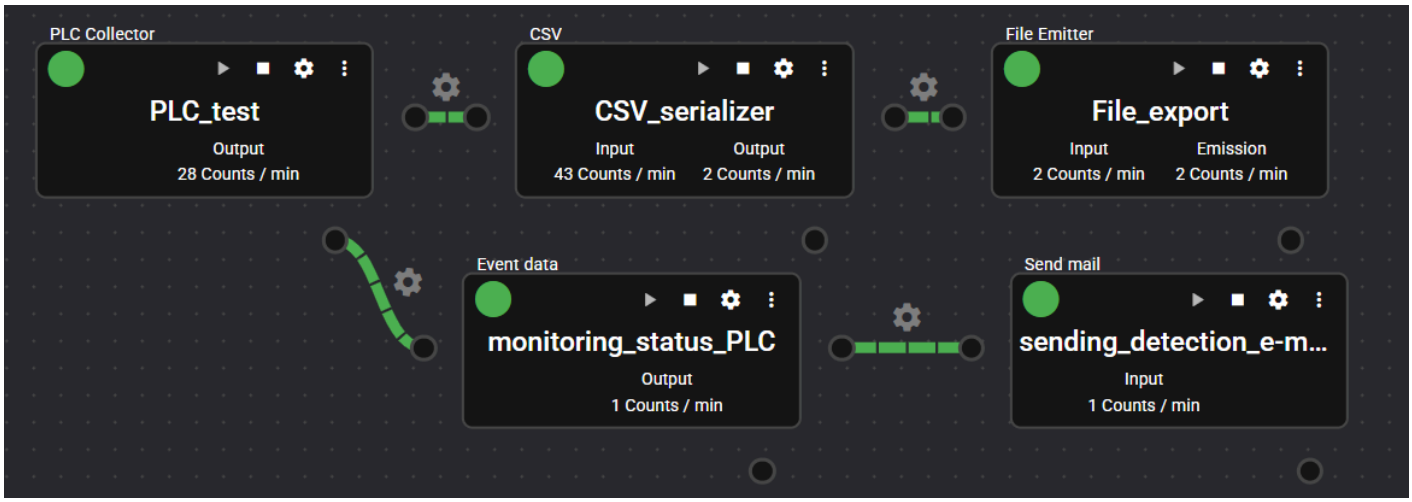
For how to set the IP address and port of the PLC, refer to the instruction manual of the PLC you are using.

7.3 How to Detect When a Problem Occurs

This section describes how to detect when a component error occurs.

7.3.1 How to Detect When a Problem Occurs at a Specific Component

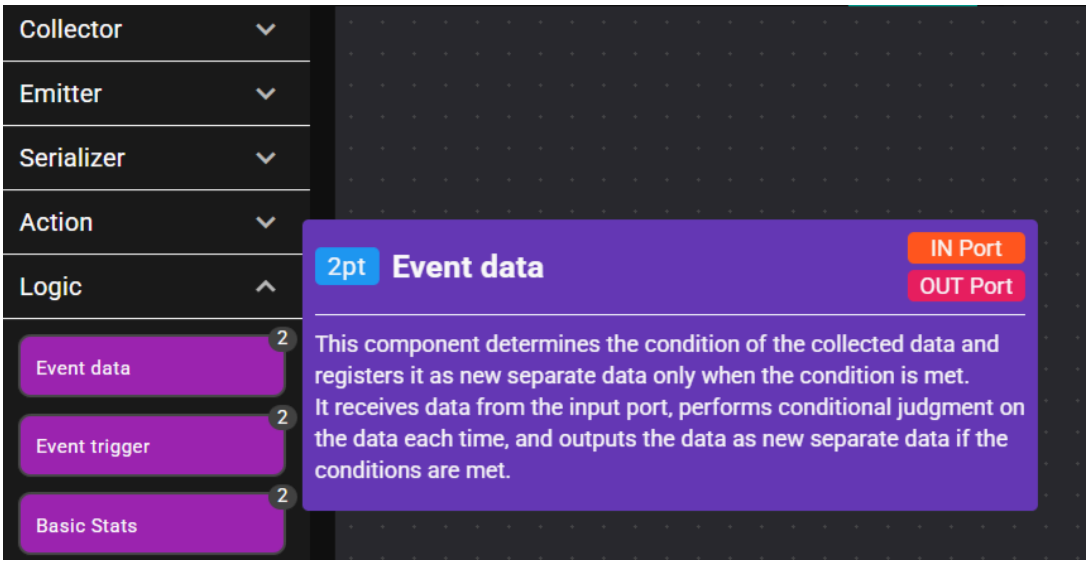
This section describes how to send an email to notify you when an error occurs in a specific component.



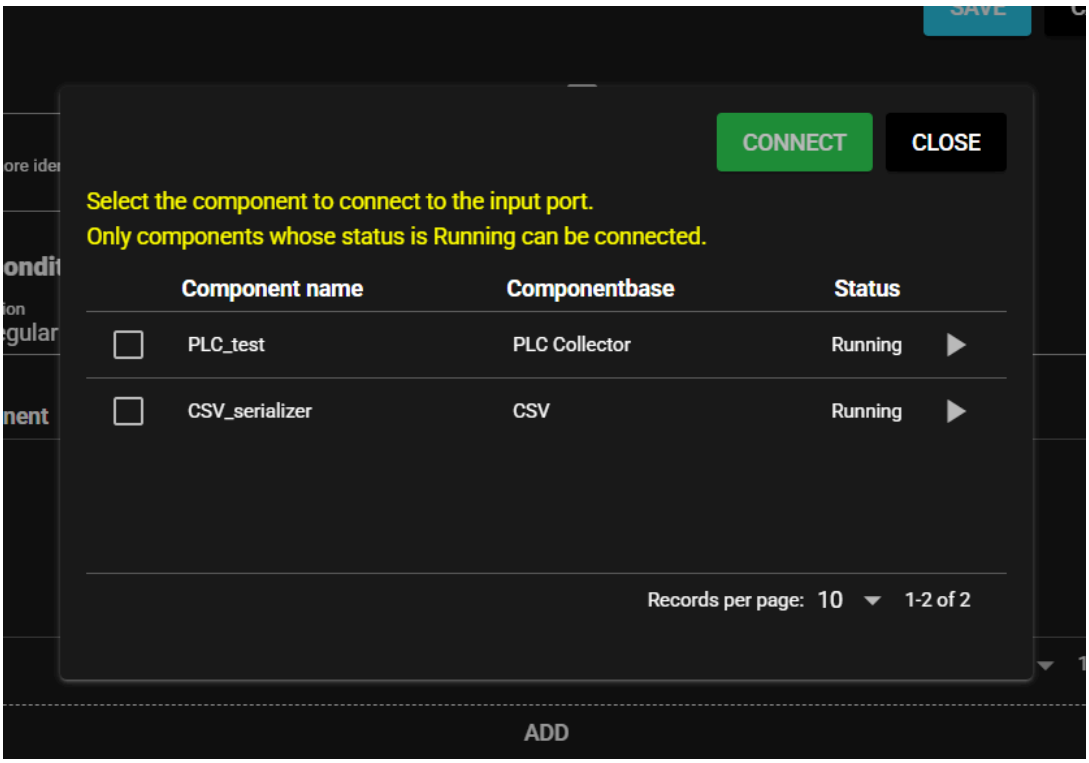
7.3.1.1 Configuring Event Data

Specify the settings so that event data is registered when the value of errors_run_status output from the detection target component is a value other than Normal or Stopped. Provides email notification when the component status is Warning or Error.

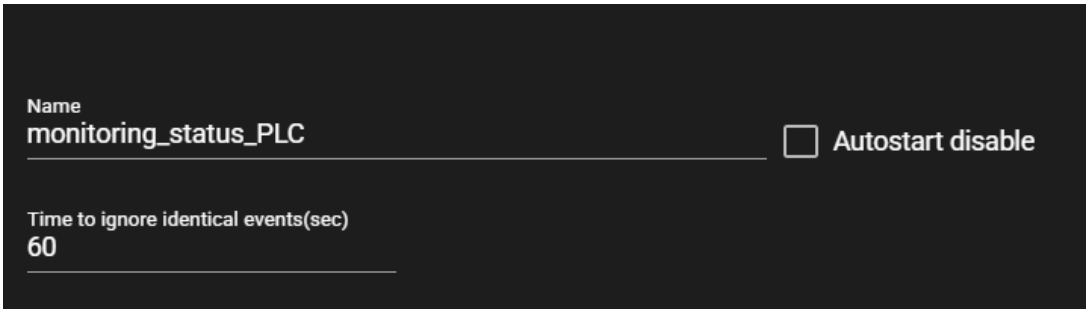
1. Open the panel on which the component you want to detect an error is running, and in Edit Mode, drag and drop ComponentBase- > Logic- > Event Data onto the panel.



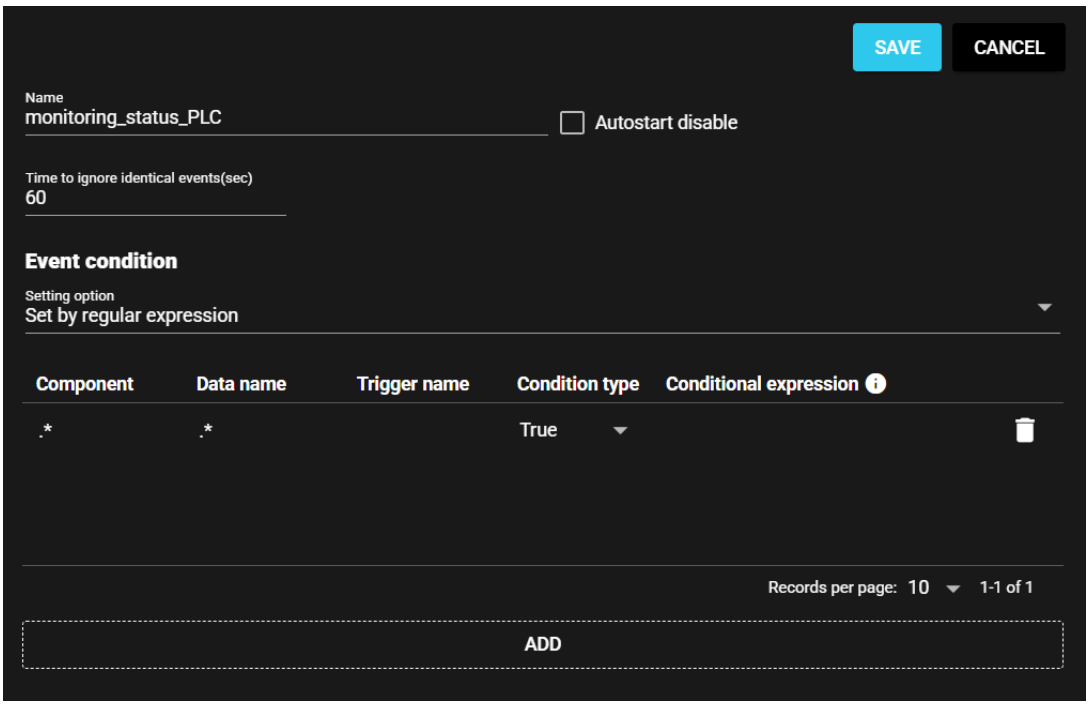
2. The destination of connection menu for the input port opens. Select “Close” this time to connect to the system port.



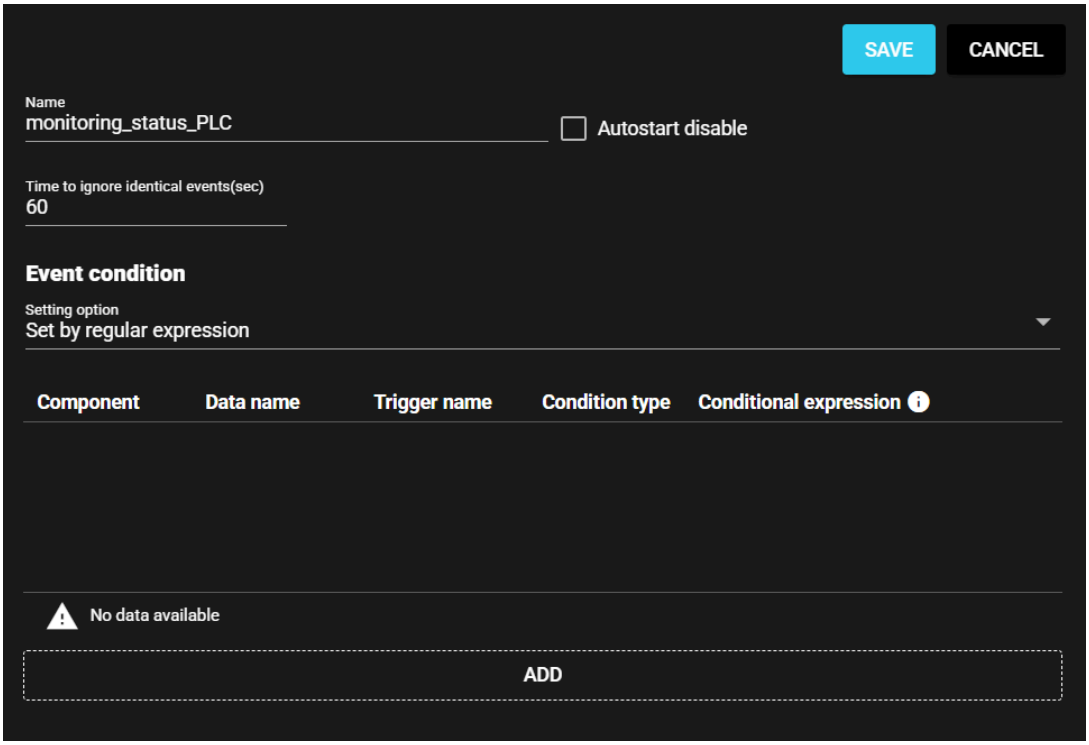
3. Enter any name. In this example, let us assume that the name is “monitoring_status_PLC”.



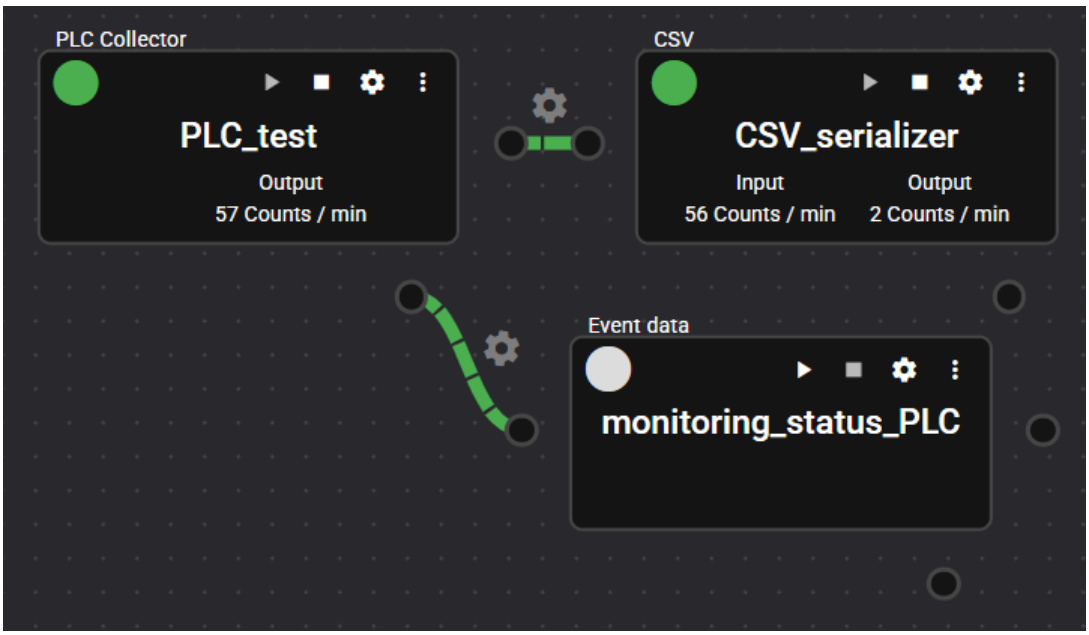
4. Select the trash can mark and delete the data.



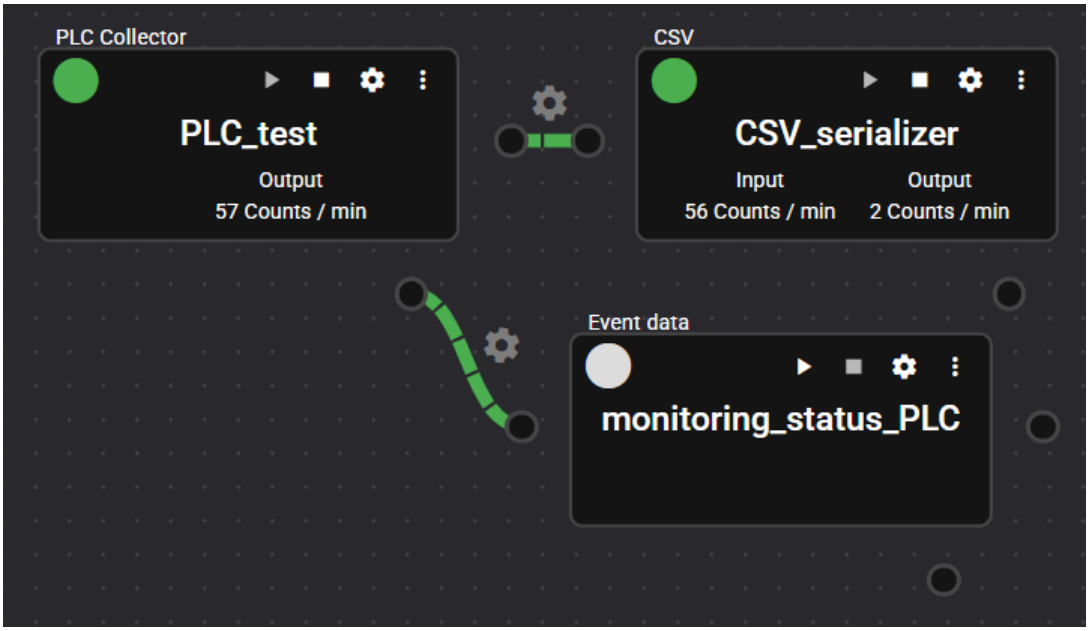
5. Select “Save.”



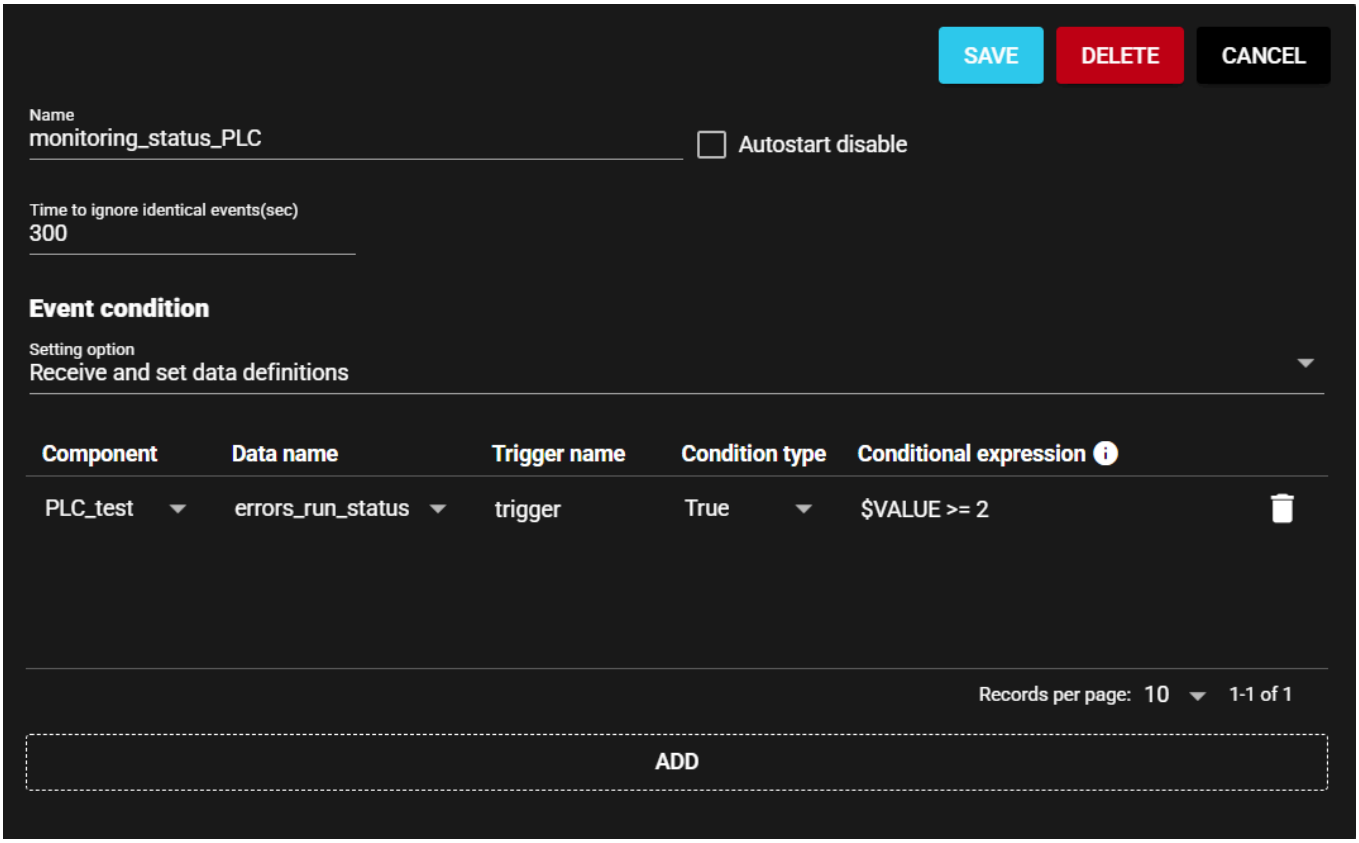
6. Connect to the system port of the component whose event data you want to detect. In this example, the event data is assumed to be connected to the PLC collector.



7. Open the Setting Menu of event data.



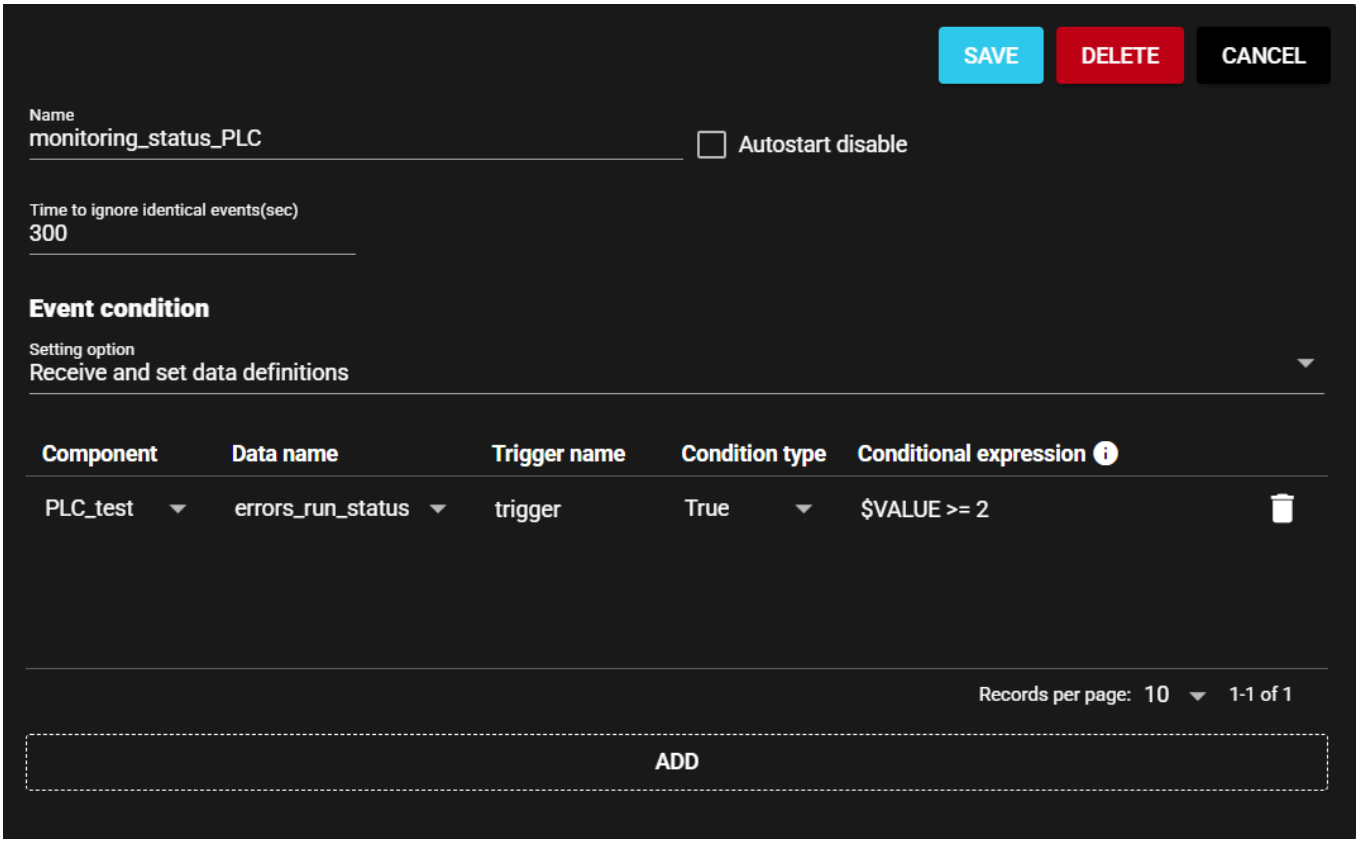
8. Set the event condition. Select "Receive and set data definitions" as the setting method. Set the data name as “errors_run_status”, the trigger name as “trigger”, the condition type as “True”, and the conditional expression as “$VALUE >= 2”.
   Then, set the Time to ignore identical events(sec) to 300.



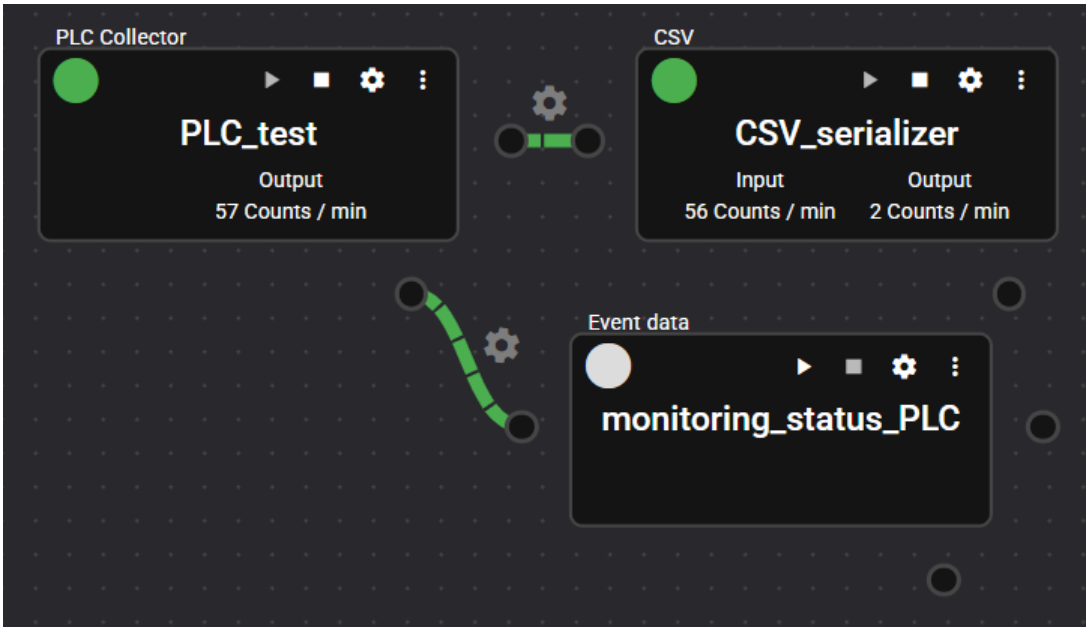
Notes on sending email

With this setting, an email is always sent as long as the problem persists, even after 5 minutes (300 seconds) have elapsed. Please note that the amount of notifications can be huge. To send an email only once when an error occurs, set the condition type to “Raise”.

9. Select “Save.”



10. Start the event data.

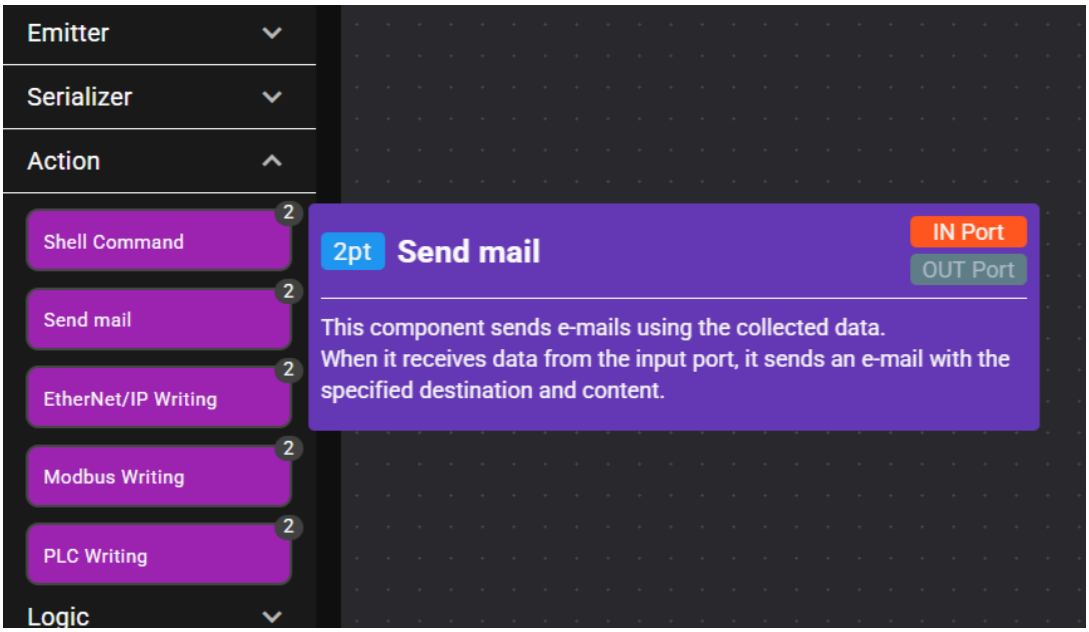


The event data settings are now complete.

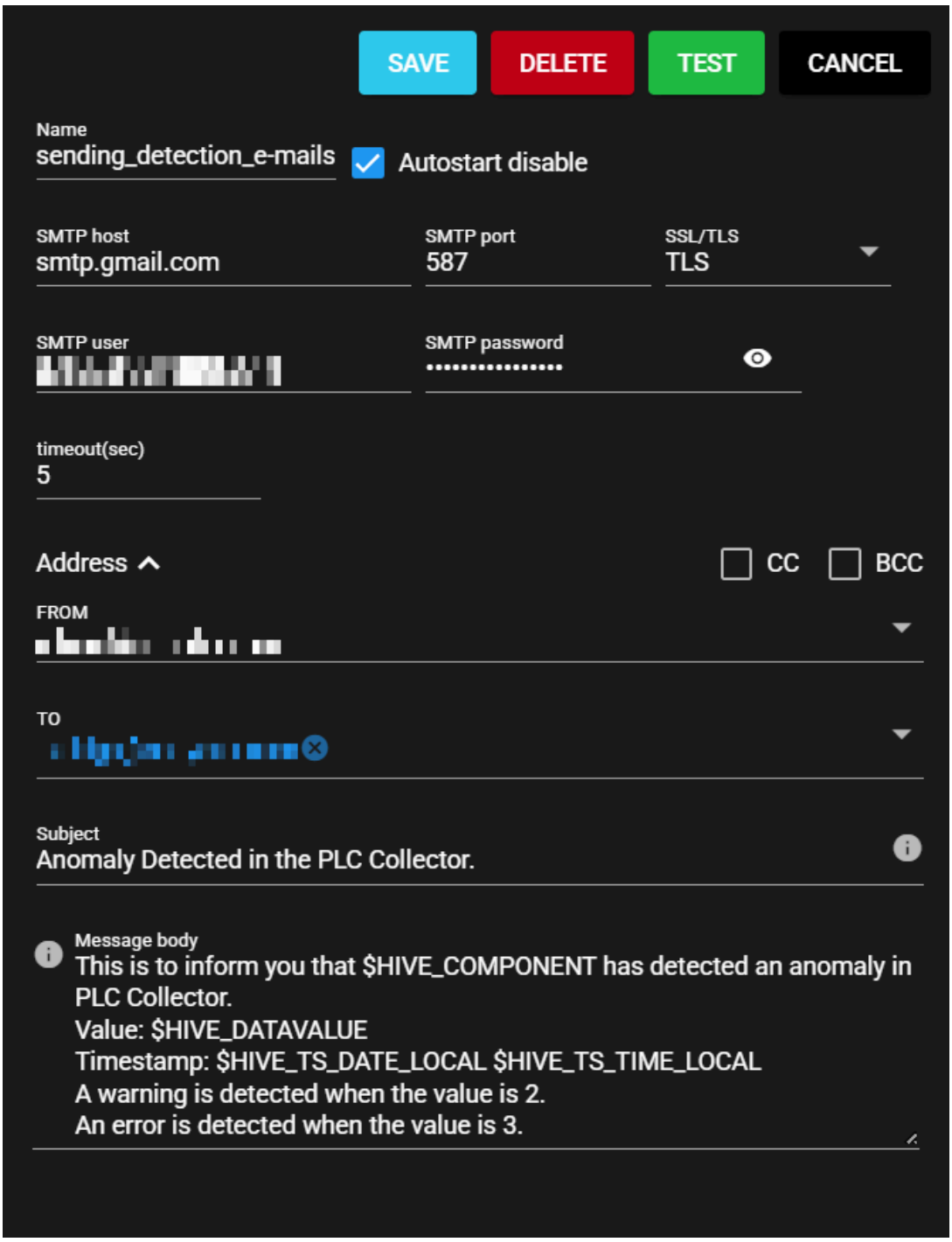
7.3.1.2 Configuring Email Sending

Set the value received from the event data to be sent by email.
In this example, the setting for sending by gmail is explained. Make sure you have a usable gmail account, set up two-factor authentication, and create an app password.

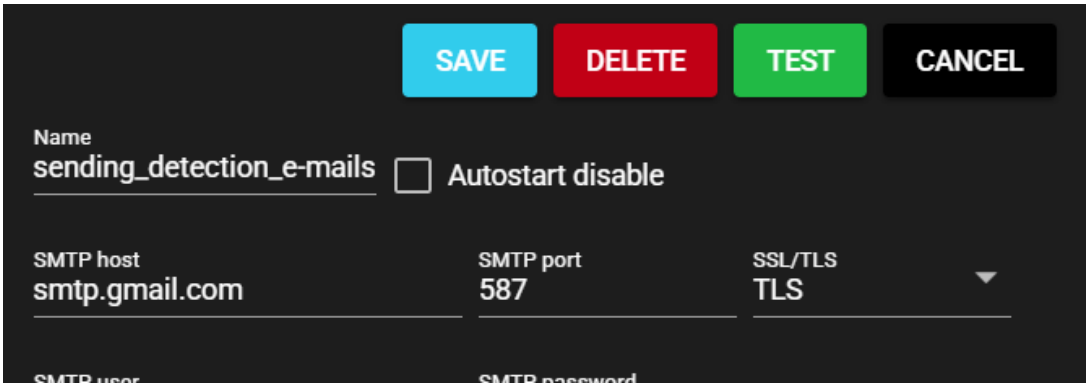
1. Open the panel on which the component you want to detect an error is running, and in Edit Mode, drag and drop ComponentBase - > Action - > Send email onto the panel.



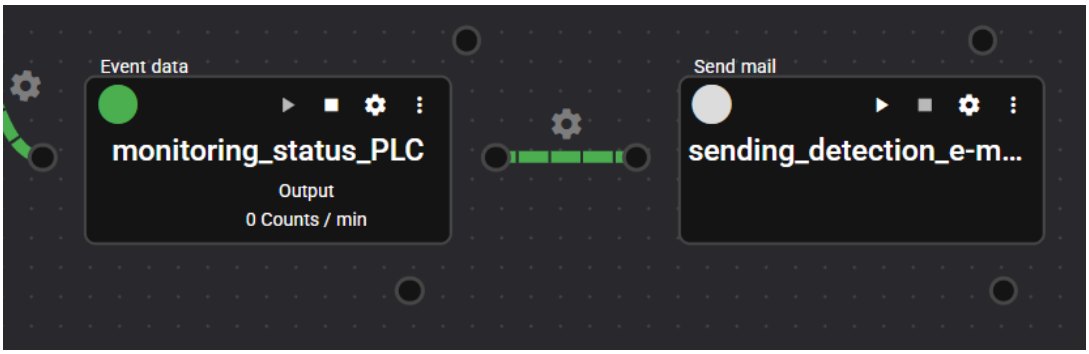
2. Make the necessary settings. For gmail, the SMTP server is "smtp.gmail.com" and the port (TLS) is "587." For the SMTP password, enter the app password that you created.



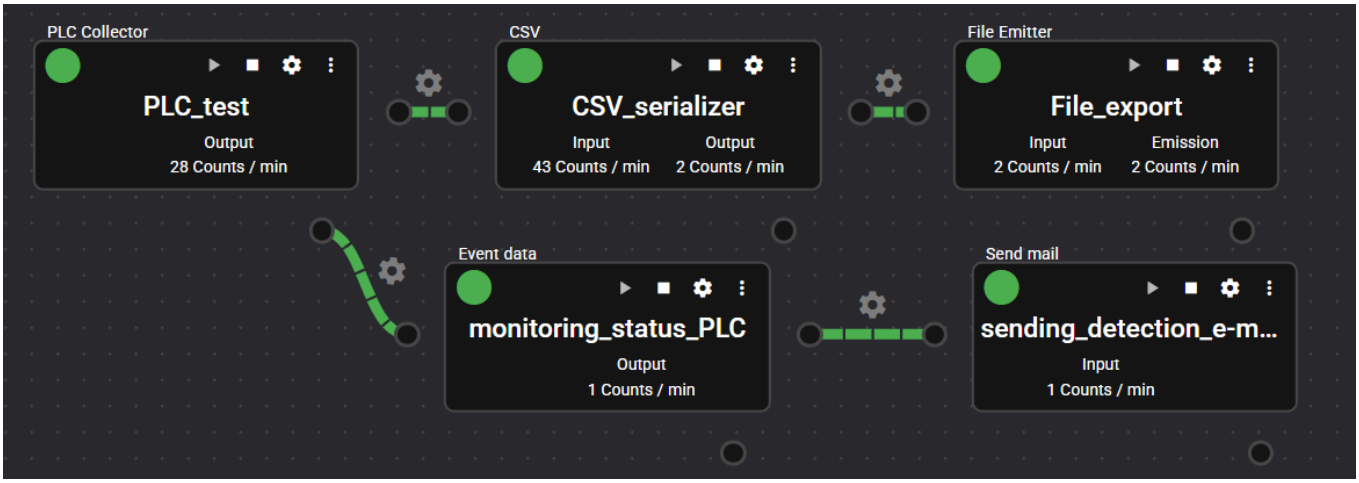
3. Select “Save.”



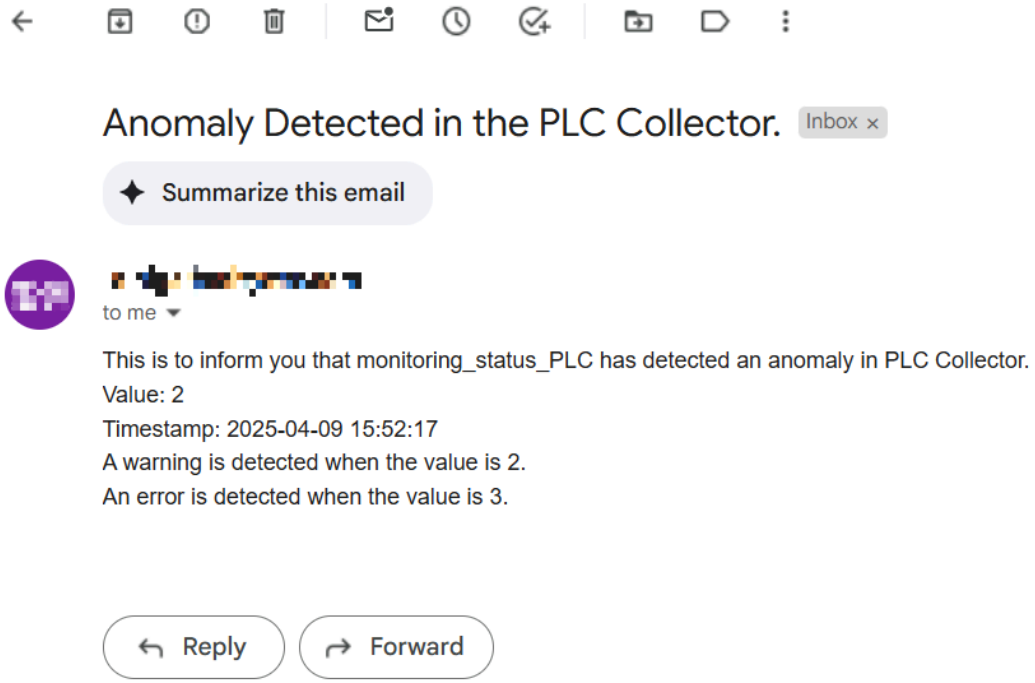
4. Connect email sending to event data and start the component.



The settings for sending email are now complete.

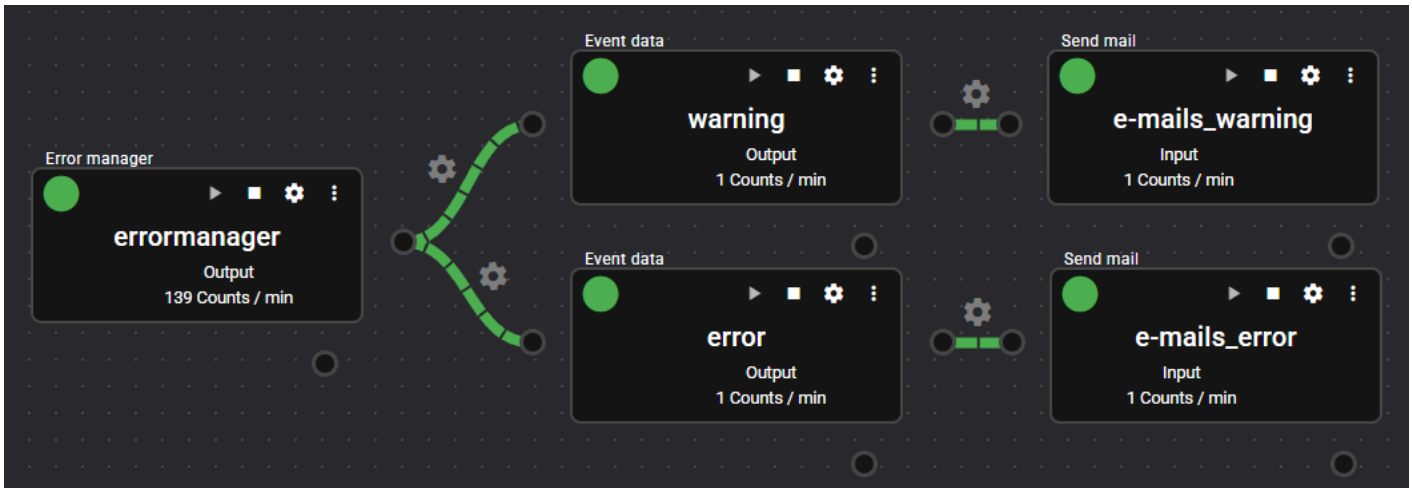


When a component actually fails, the data is sent to the configured email account. Below is an example of an email received.



7.3.2 How to detect when any one of components in Synapse fails

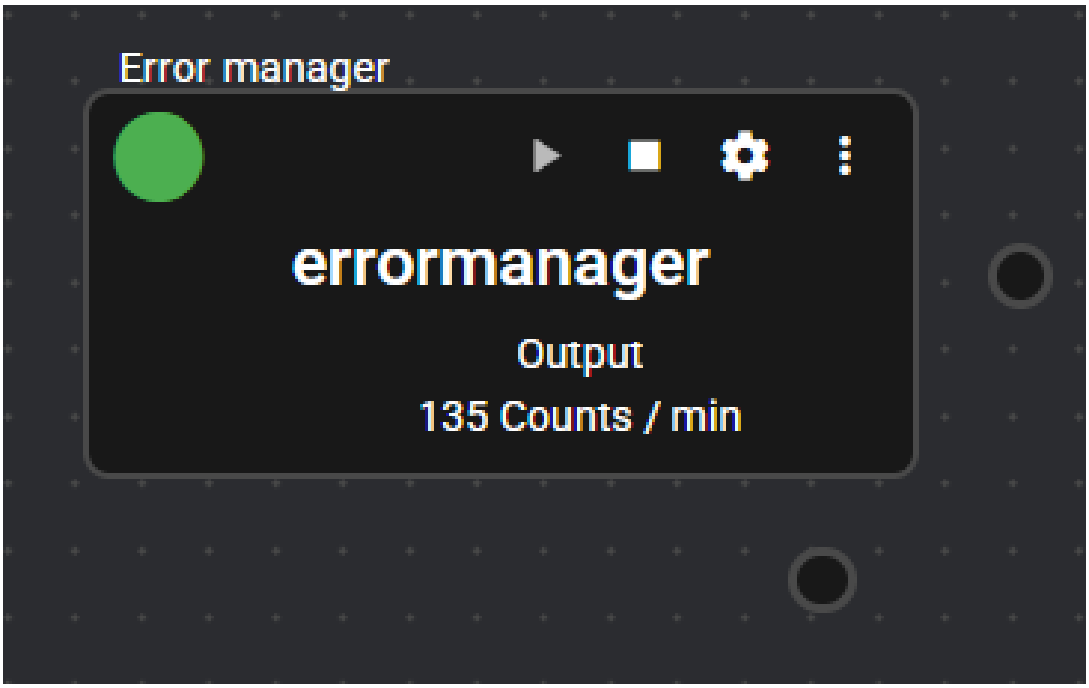
This section describes how to send an email to notify you when an error occurs in any one of components.



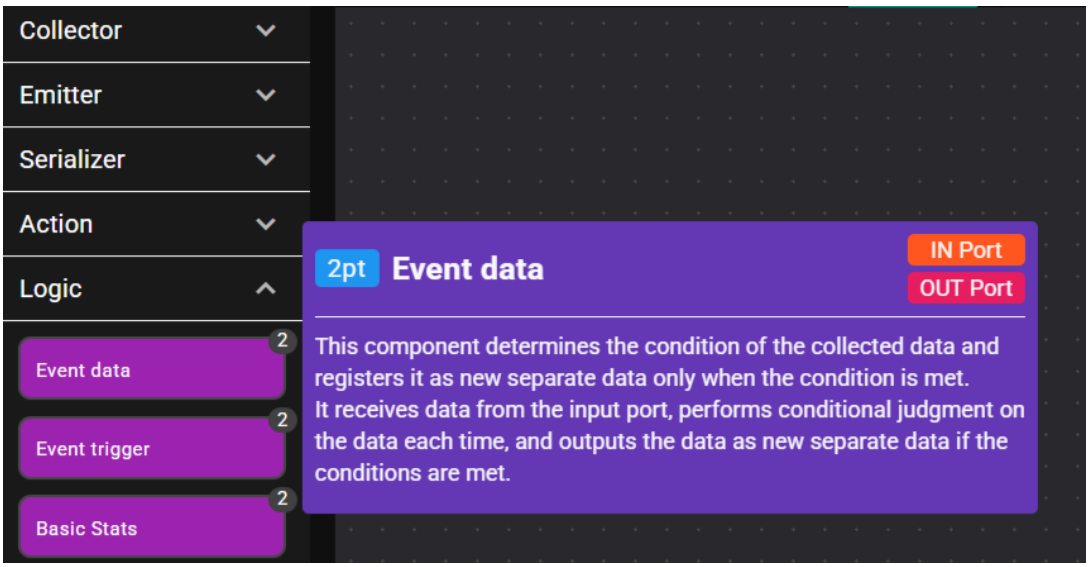
7.3.2.1 Configuring Event Data

Set event data to be registered when the values of "latest *error_component* count" and "latest *warning_component* count" outputted from the error manager increase. Provides email notification when any component status within Synapse is Warning or Error.

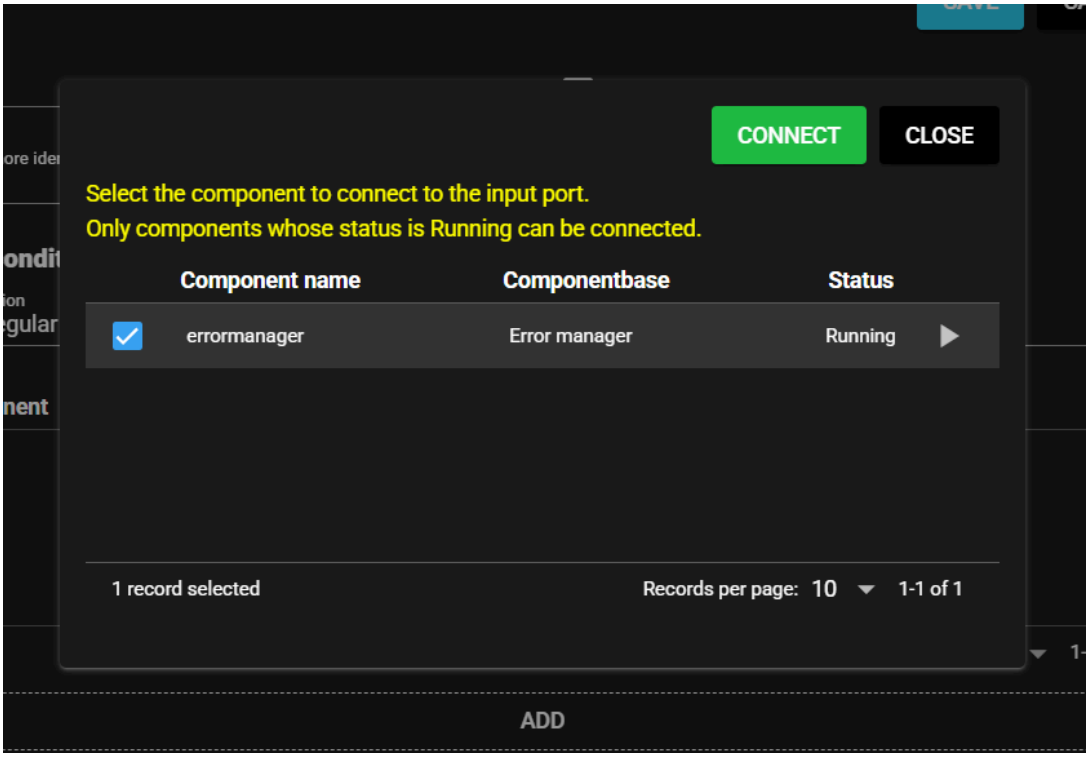
1. Open the system panel in Edit Mode and start the Error Manager.



2. Drag and drop ComponentBase- > Logic- > Event data onto the panel.



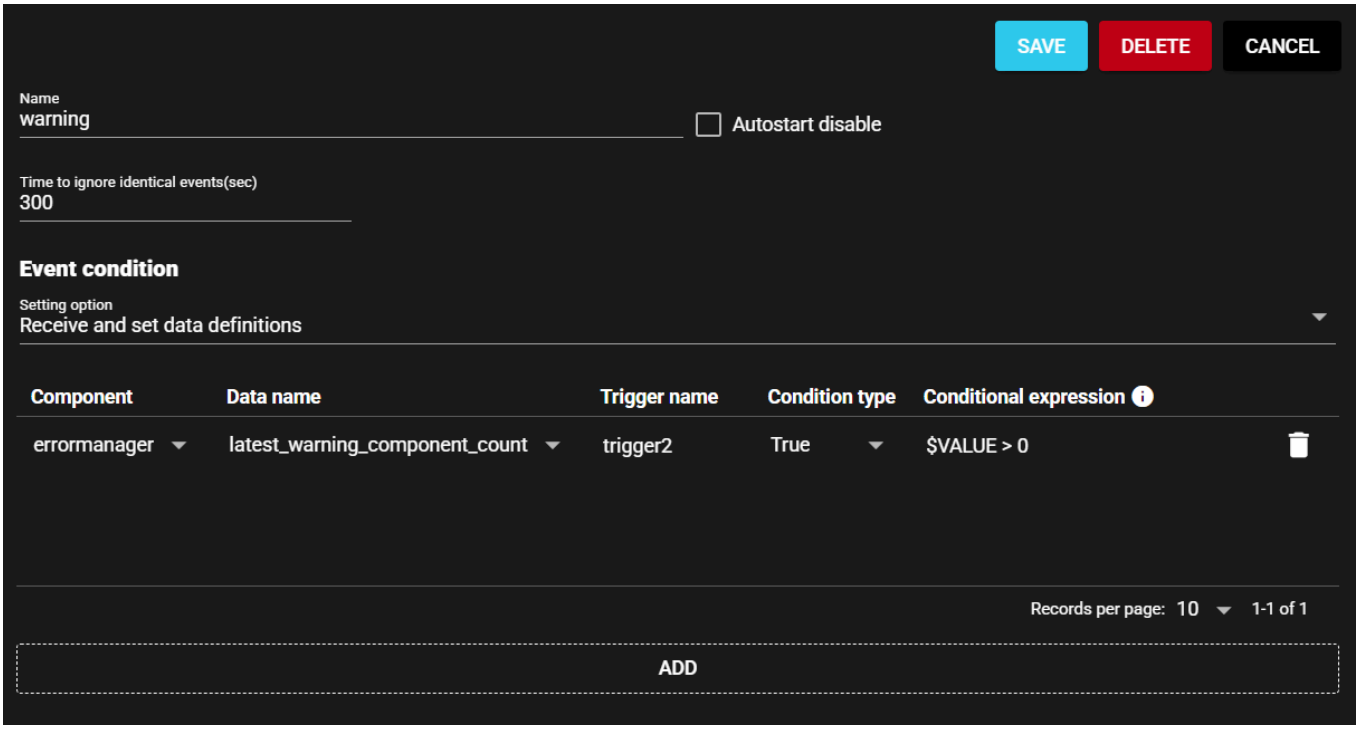
3. The destination of connection menu for the input port opens. Check errormanager and choose “Connect”.



4. Enter any name, set the event conditions, and select “Save.” Event conditions are created individually for components for which the following conditions (1) and (2) are set.

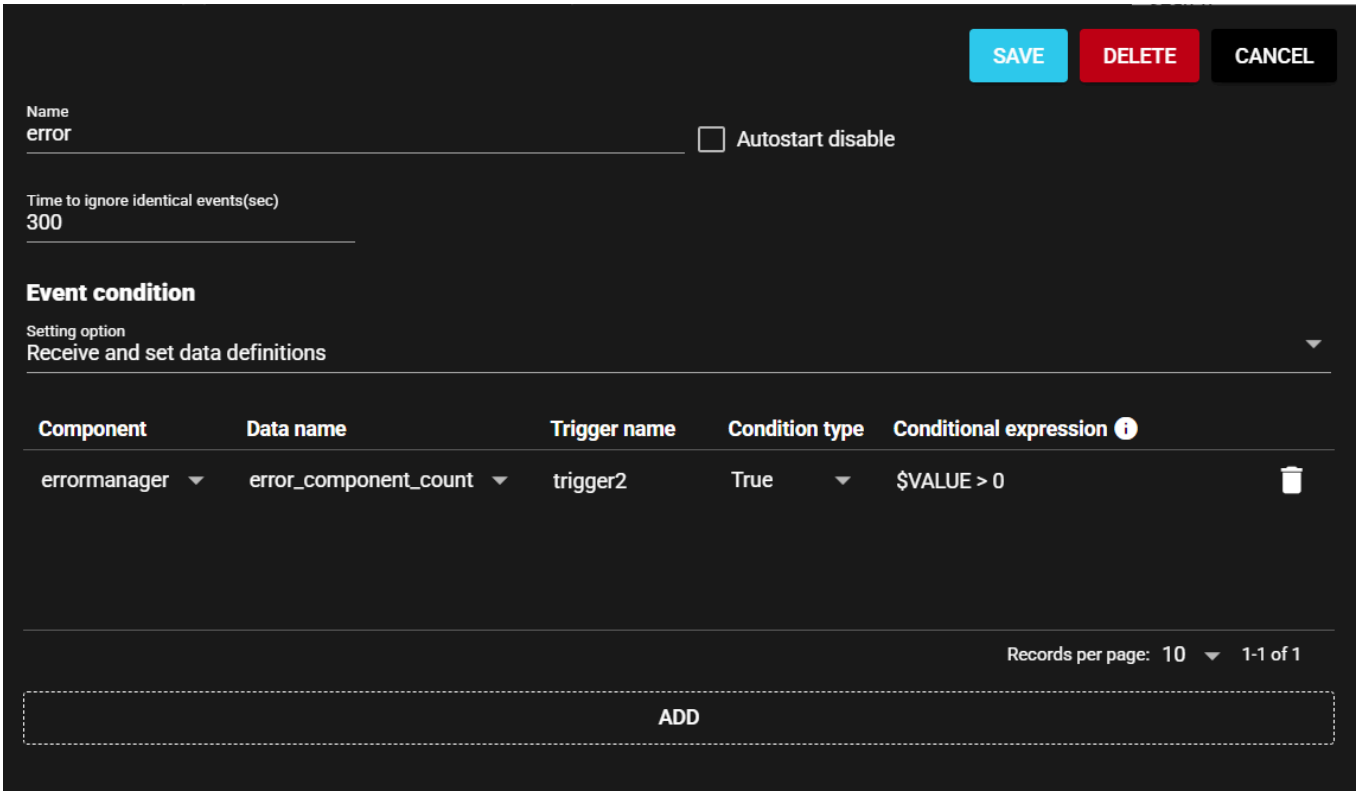
Condition (1)

Time to ignore identical events(sec) “300”, data name “latest_error_component_count”, trigger name “trigger1”, condition type “True”, conditional expression “$VALUE > 0”



Condition (2)

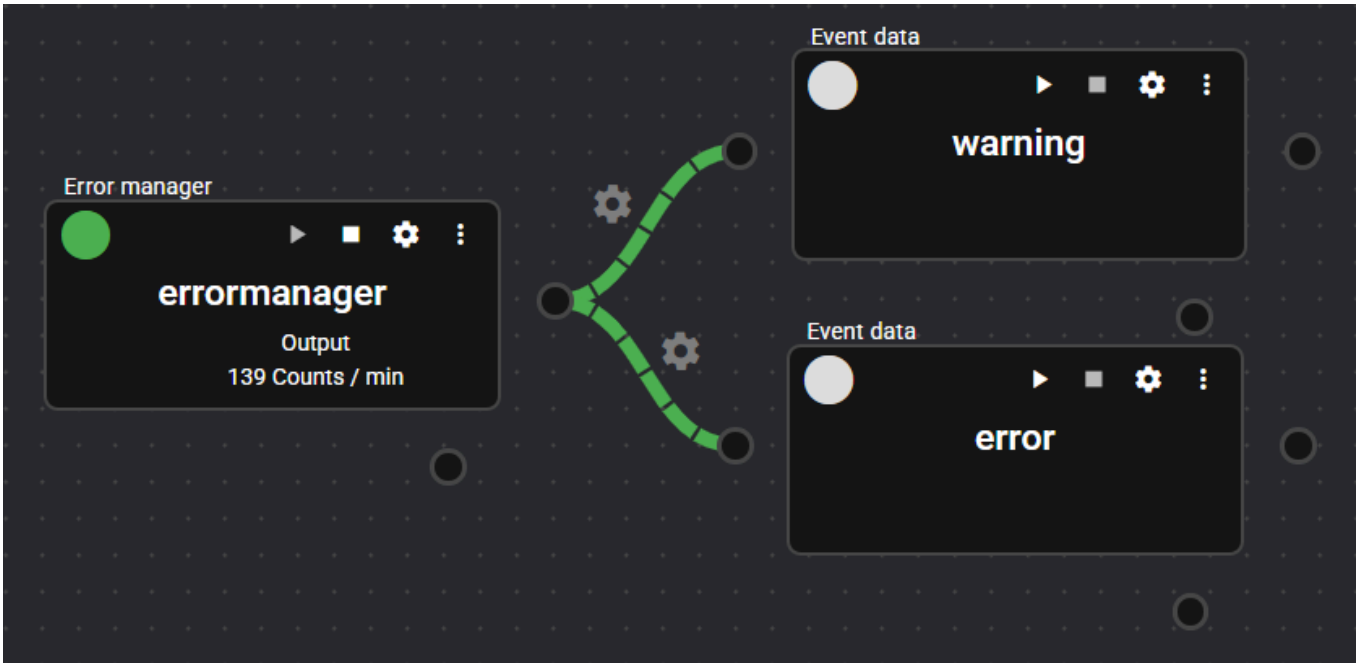
Time to ignore identical events(sec) “300”, data name “latest_warning_component_count”, trigger name “trigger2”, condition type “True”, conditional expression “$VALUE > 0”



Notes on sending email

With this setting, an email is always sent as long as the problem persists, even after 5 minutes (300 seconds) have elapsed. Please note that the amount of notifications can be huge. To send an email only once when an error occurs, set the condition type to “Raise”.

5. Start the event data.

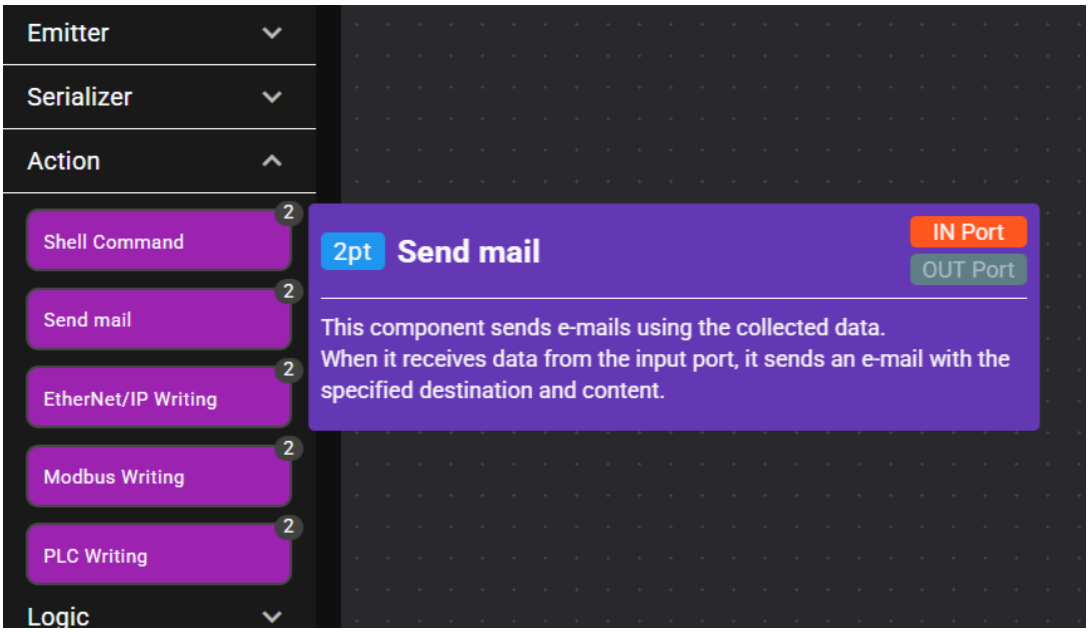


The event data settings are now complete.

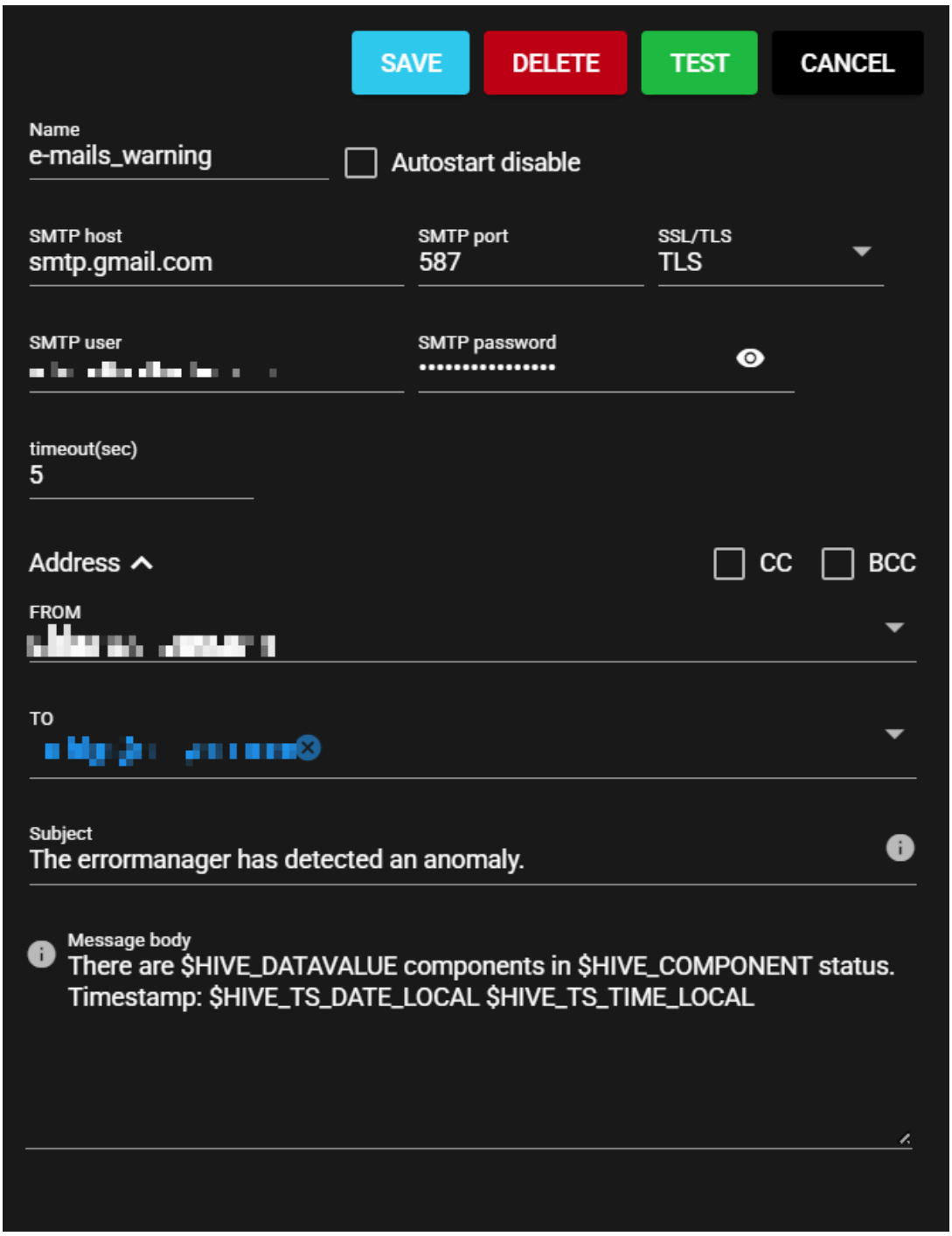
7.3.2.2 Configuring Email Sending

Set the value received from the event data to be sent by email.
In this example, the setting for sending by gmail is explained, as with here. Make sure you have a usable gmail account, set up two-factor authentication, and create an app password.

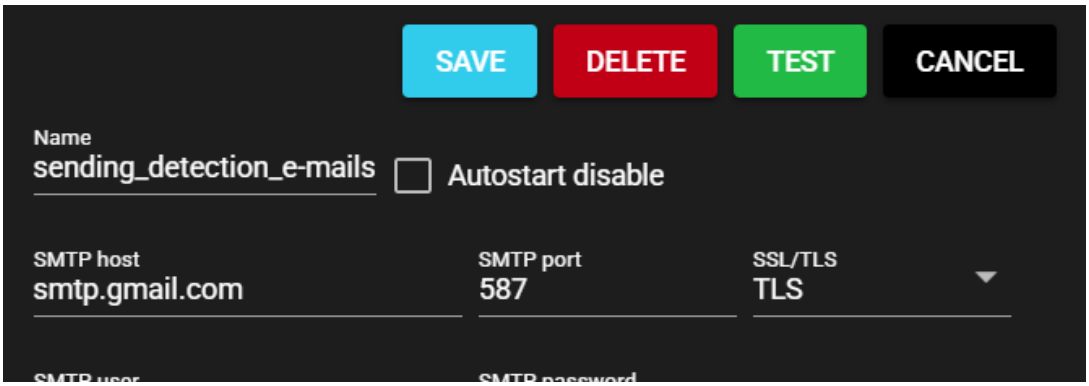
1. Open the System panel and in Edit Mode, drag and drop ComponentBase- > Action- > Send Mail onto the panel.



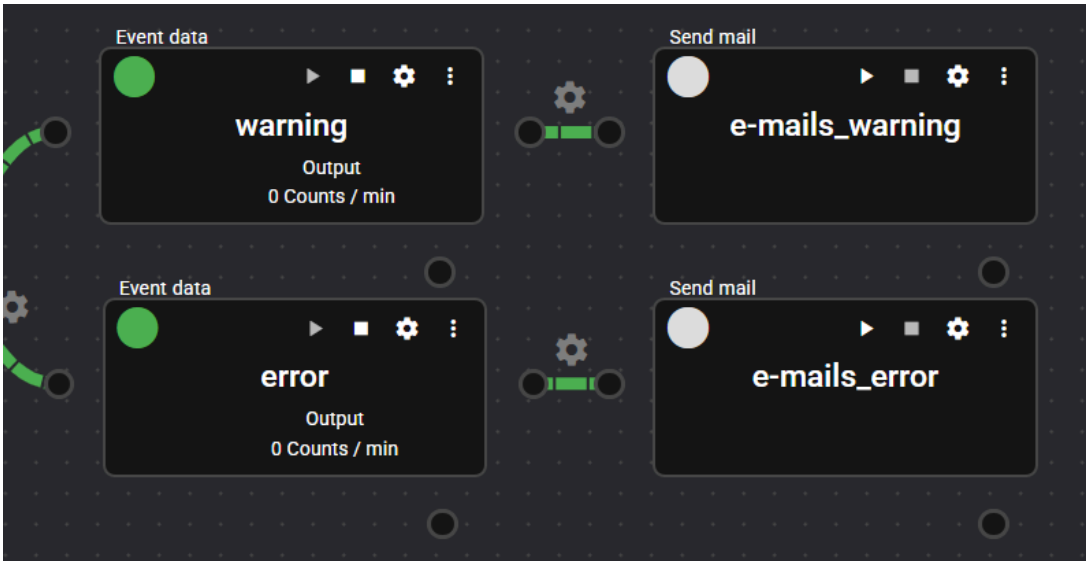
2. Make the necessary settings. For gmail, the SMTP server is "smtp.gmail.com" and the port (TLS) is "587." For the SMTP password, enter the app password that you created.



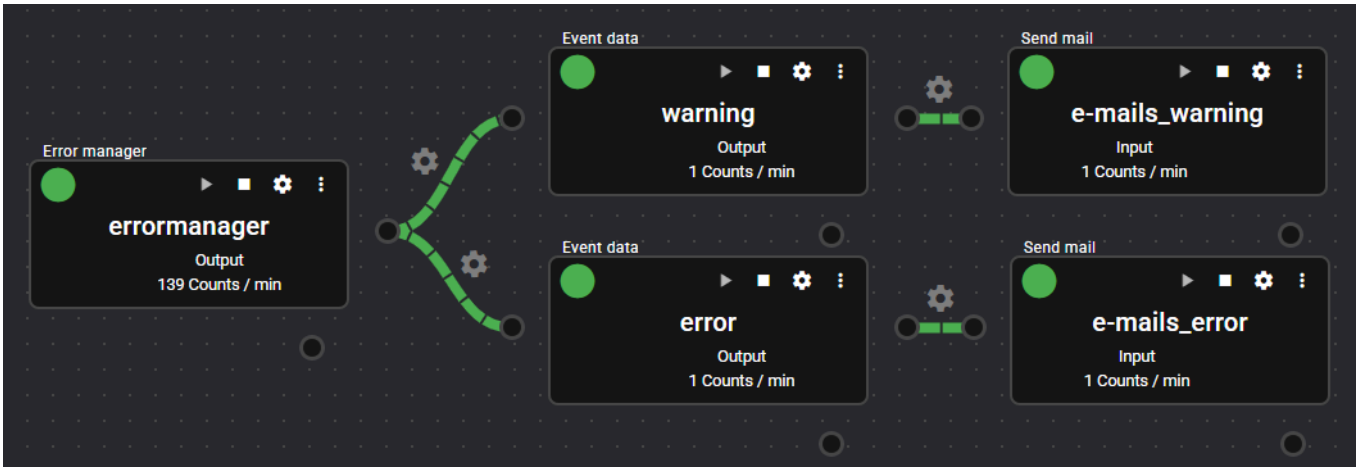
3. Select “Save.”



4. Follow the same steps to create another email sending component. Connect email sending component to the corresponding event data, and start the component.

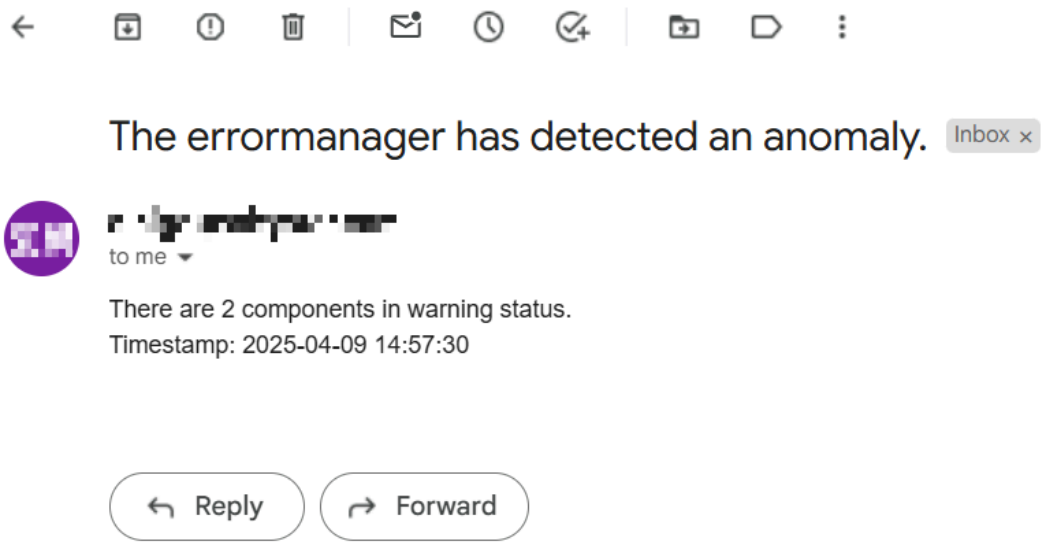


The settings for sending email are now complete.

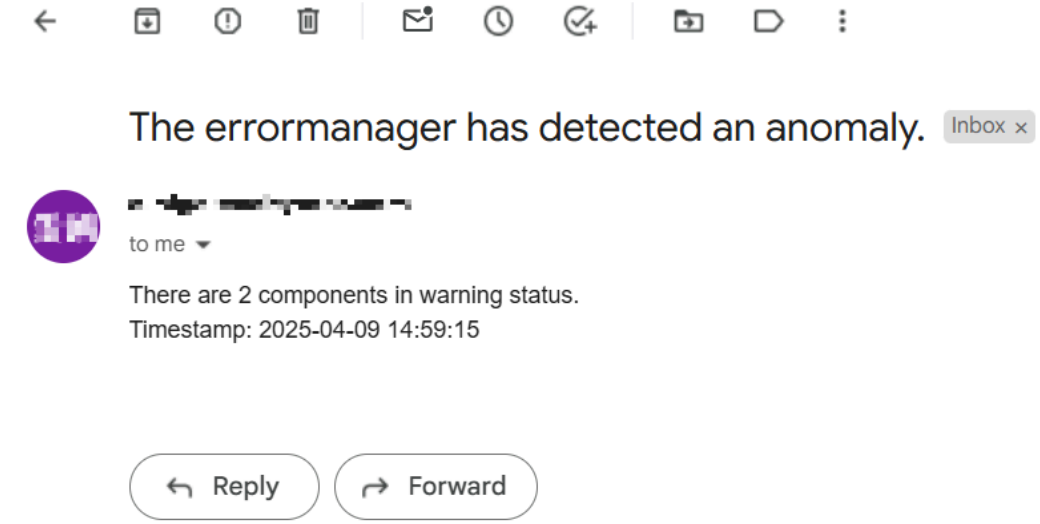


When any component actually fails, the data is sent to the configured email account. Below is an example of an email received.

Warning

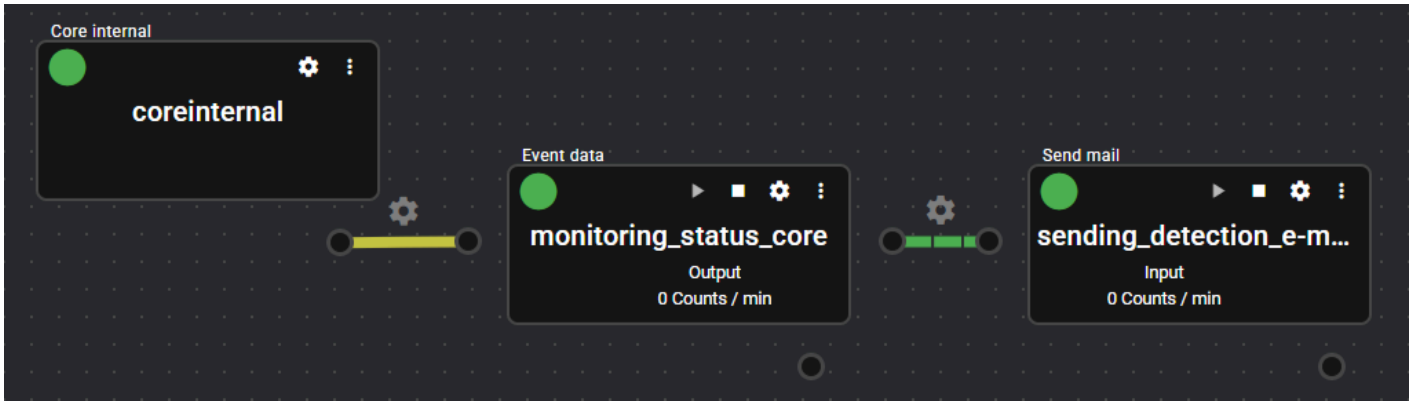


Error



7.3.3 How to Detect Data Loss in a Time Series Database due to Power Outages

In Synapse, if a loss occurs in the time series database used internally due to a power failure in the operating machine, the core internal registers the error as a warning status at the next startup.
By capturing this information at the system port, you can be notified by email if there is a data loss.
This procedure describes how to do this.



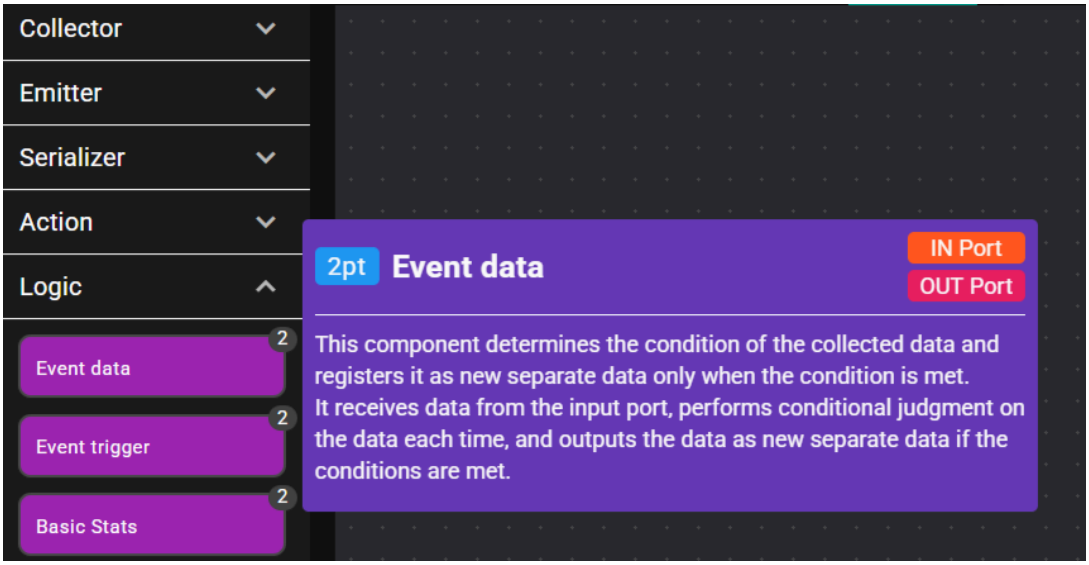
Caution on data loss due to power failure

Note that, although data loss due to power failure can be detected, data cannot be recovered. It is only a function to detect that some data could not be saved.

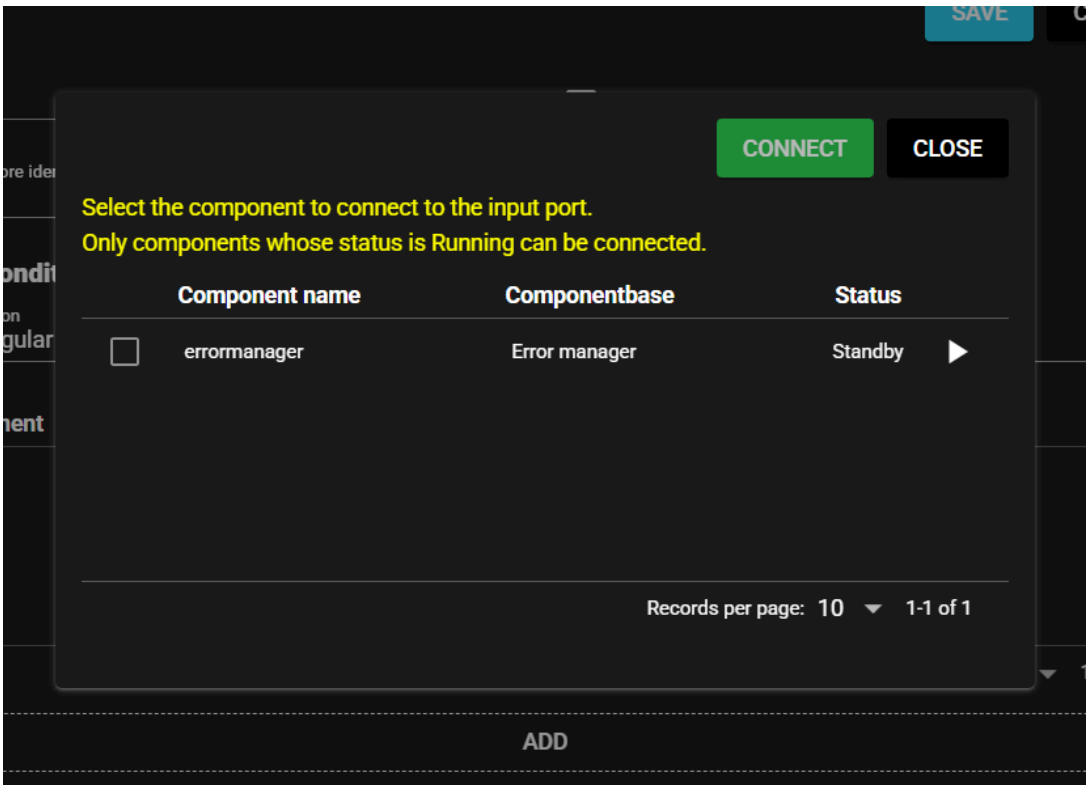
7.3.3.1 Configuring Event Data

Specify the settings so that event data is registered when the value of errors_run_status outputted from the core internal is a value other than Normal or Stopped. When the status lamp indicates a warning or error, an email notification is sent and the power interruption can be detected.

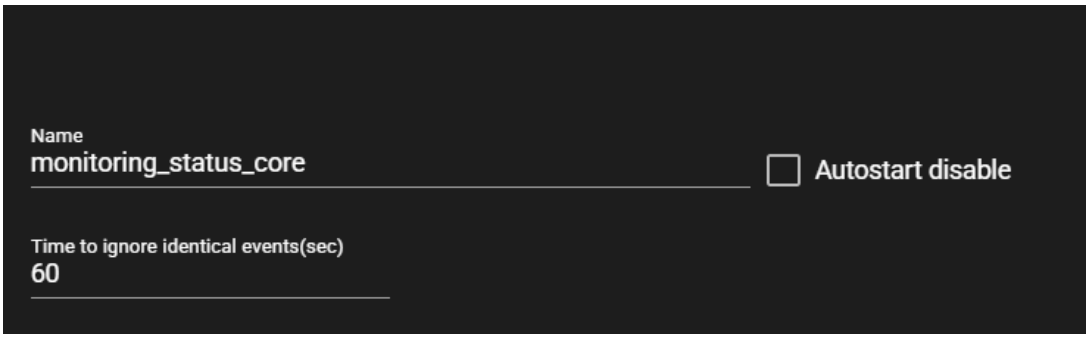
1. Open the System panel and in Edit Mode, drag and drop ComponentBase- > Logic - > Event data onto the panel.



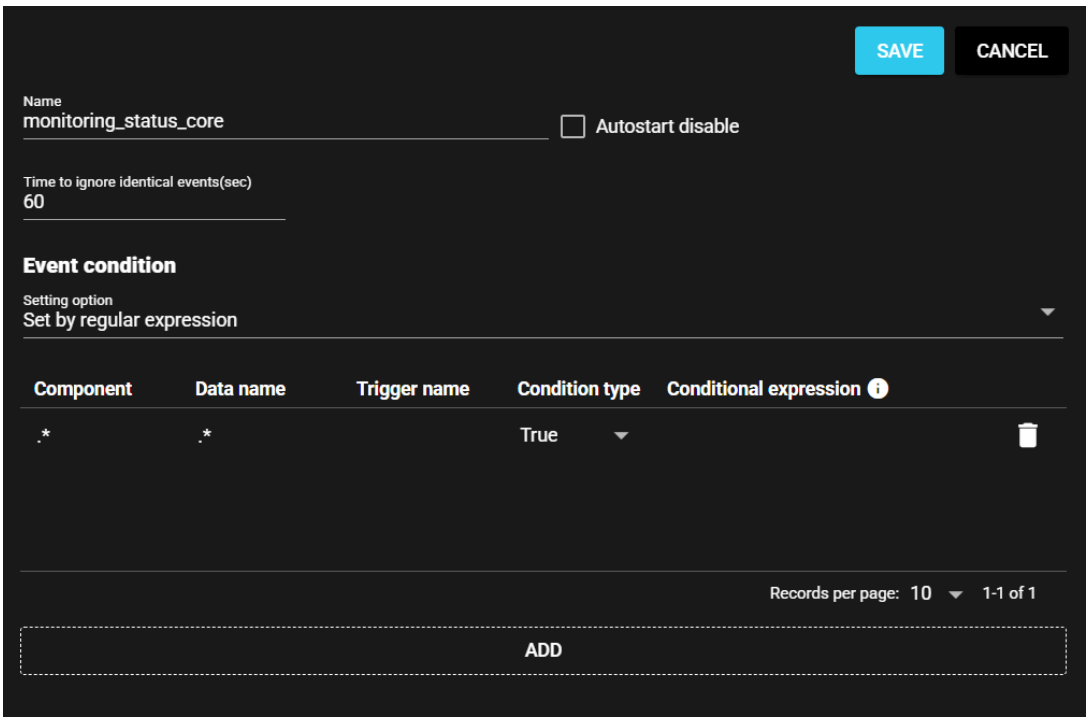
2. The destination of connection menu for the input port opens. Select “Close” this time to connect to the system port.



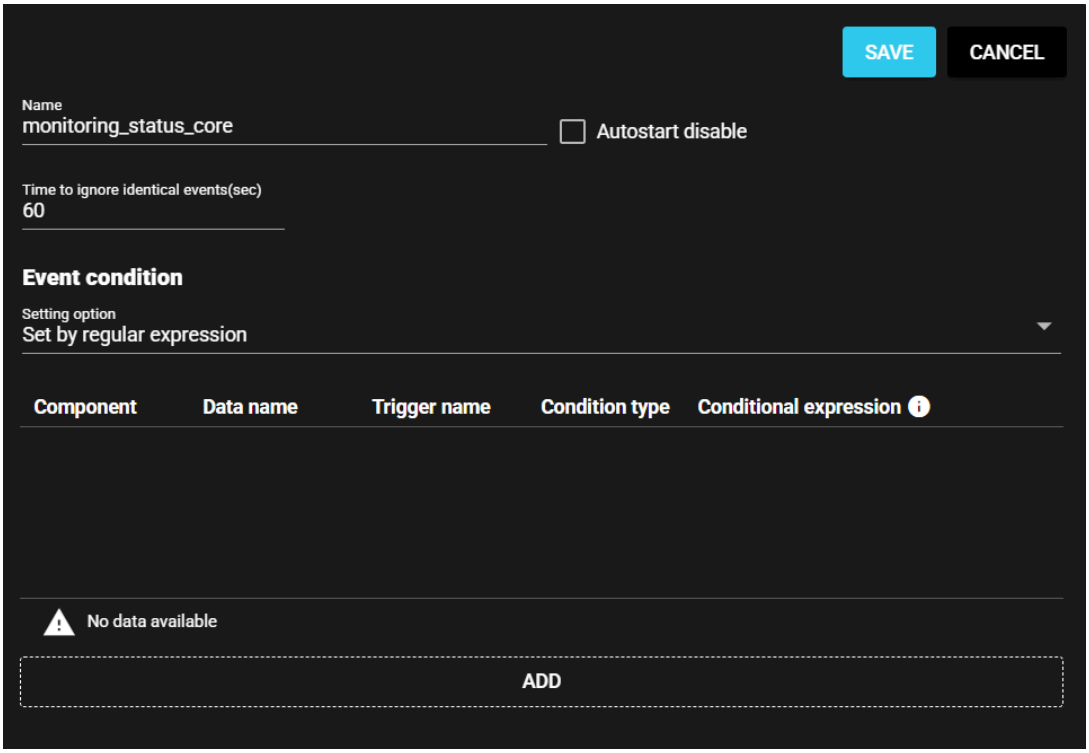
3. Enter any name. In this example, let us assume that the name is “monitoring_status_core”.



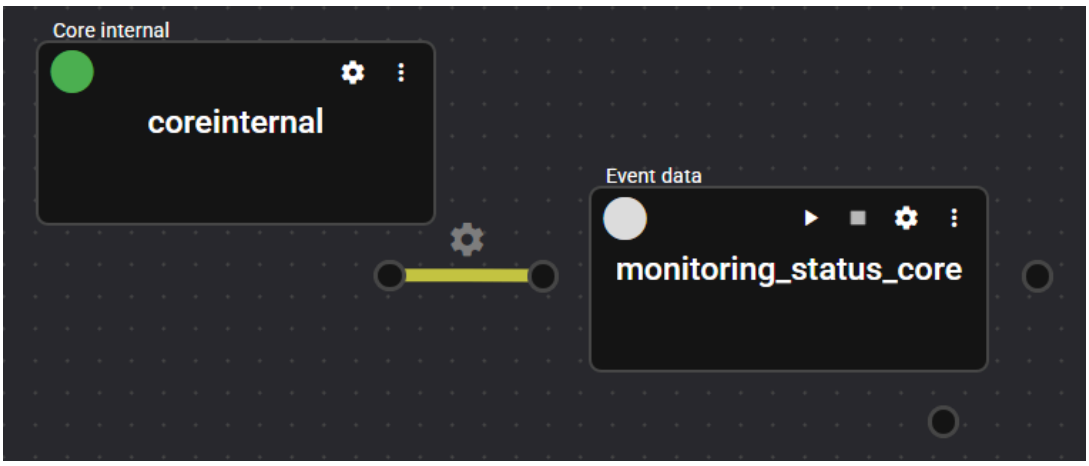
4. Select the trash can mark and delete the data.



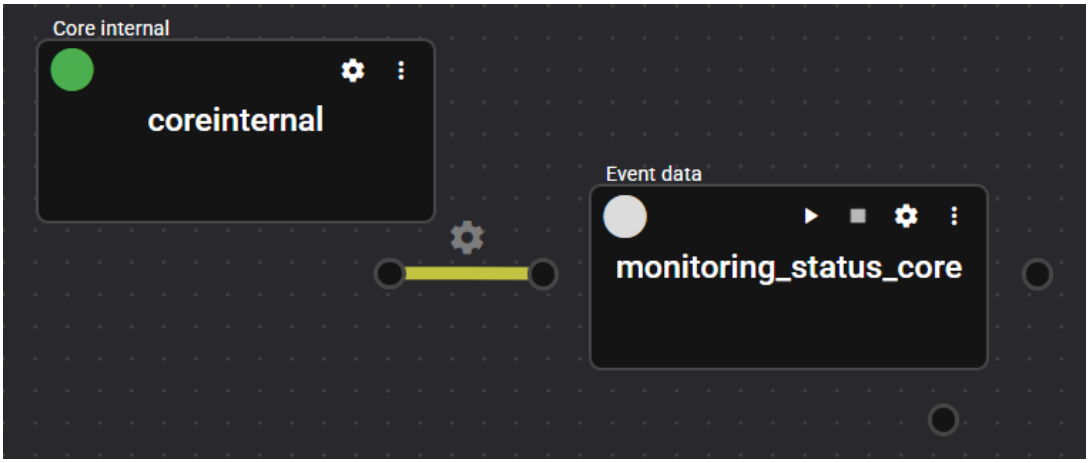
5. Select “Save.”



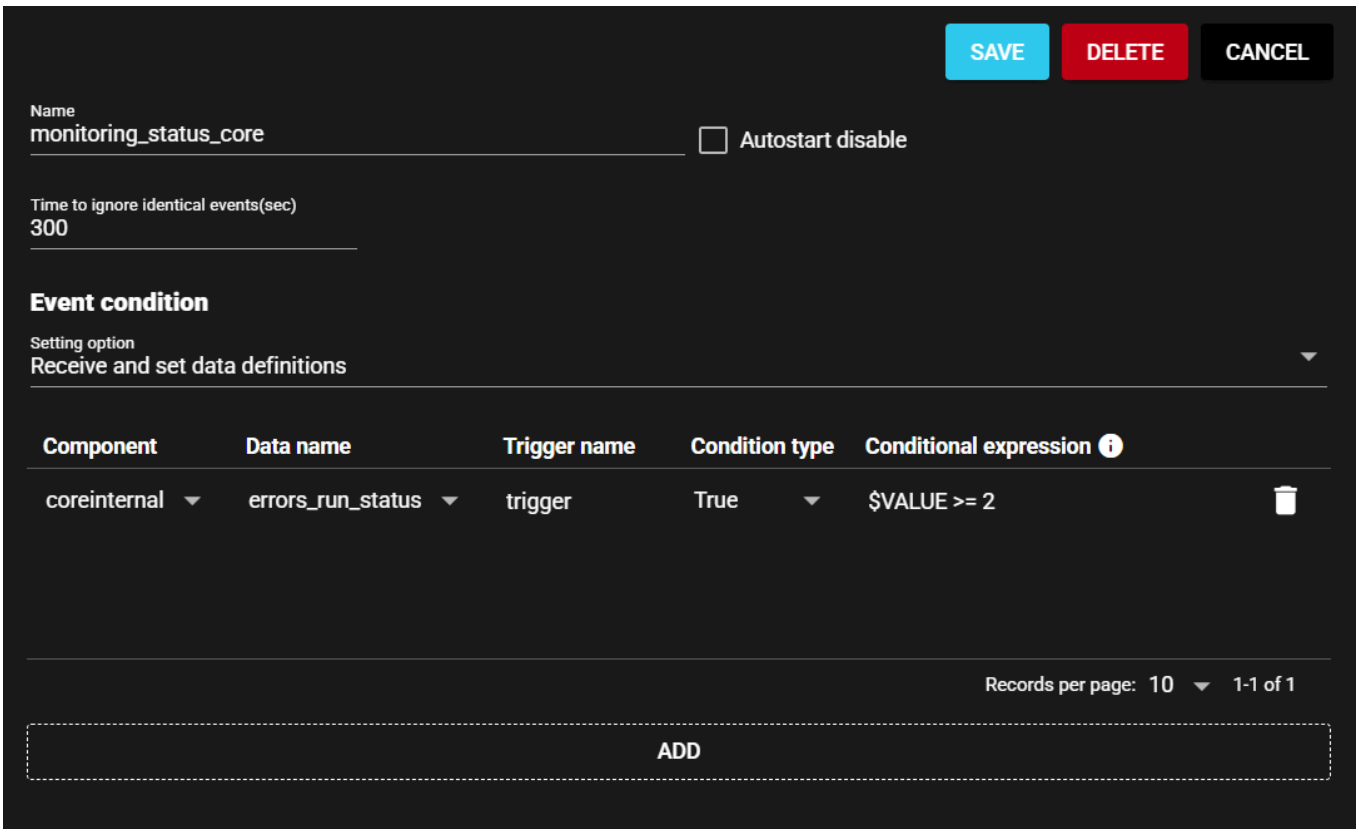
6. Connect event data to the core internal system port.



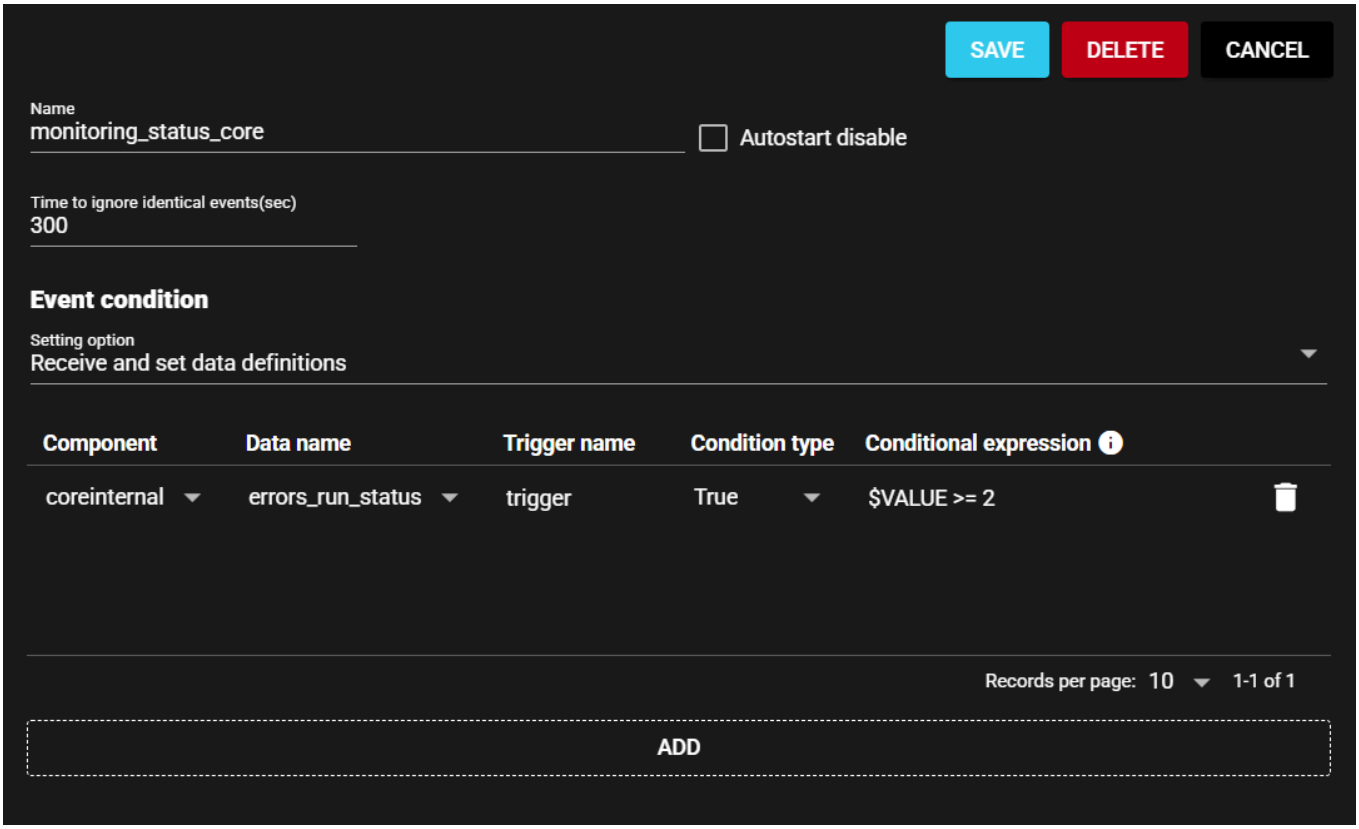
7. Open the Setting Menu of event data.



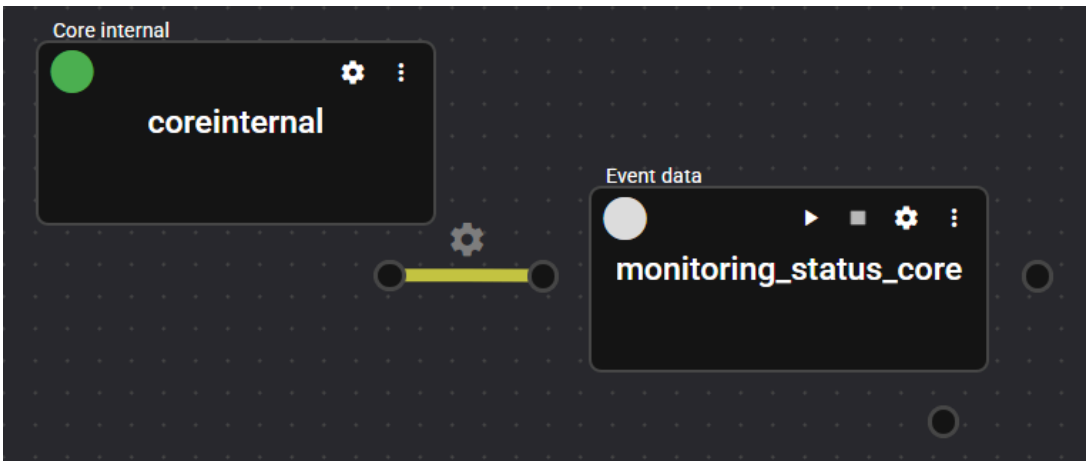
8. Set the event condition. Select "Receive and set data definitions" as the setting method. Set the data name as “errors_run_status”, the trigger name as “trigger”, the condition type as “True”, and the conditional expression as “$VALUE >= 2”.
   Then, set the Time to ignore identical events(sec) to 300.



9. Select “Save.”



10. Start the event data.

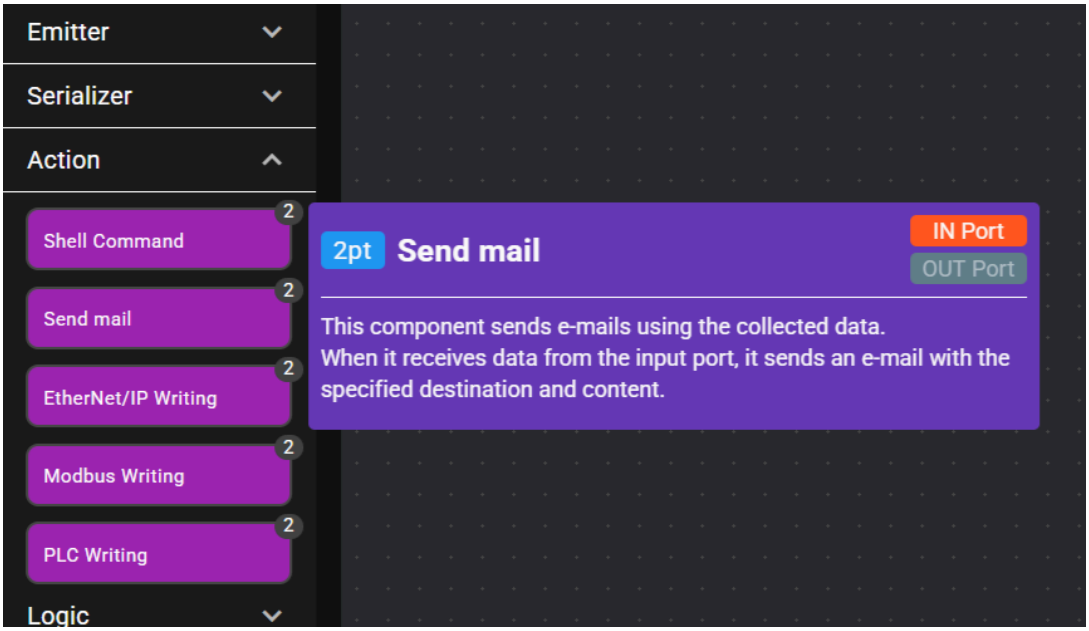


The event data settings are now complete.

7.3.3.2 Configuring Email Sending

Set the value received from the event data to be sent by email.

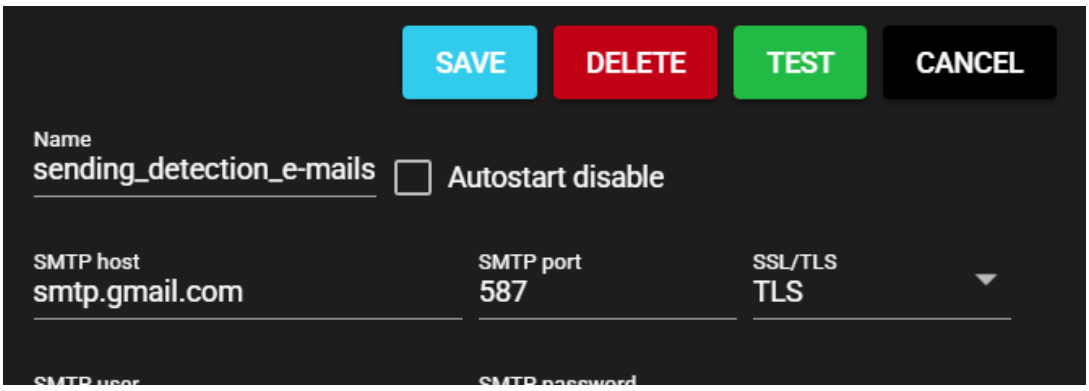
1. Open the System panel and in Edit Mode, drag and drop ComponentBase- > Action- > Send Mail onto the panel.



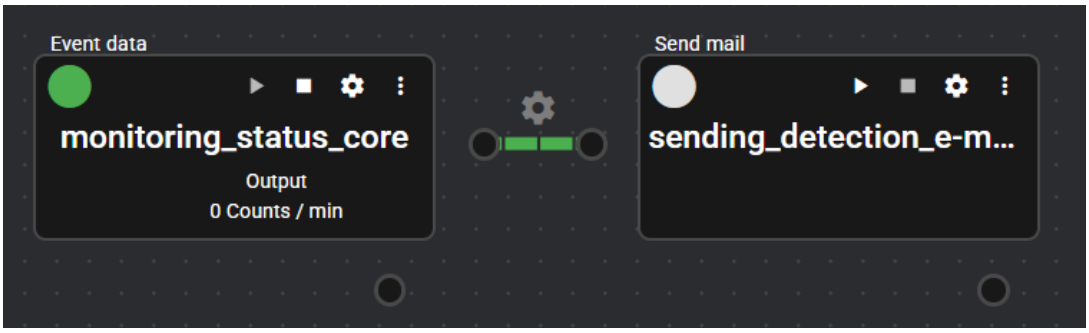
2. Make the necessary settings. For gmail, the SMTP server is "smtp.gmail.com" and the port (TLS) is "587." For the SMTP password, enter the app password that you created.



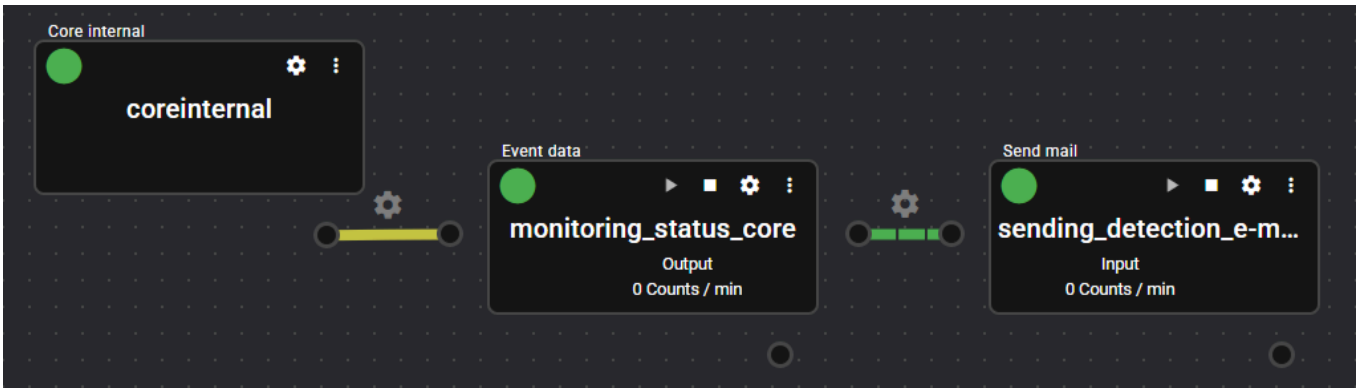
3. Select “Save.”



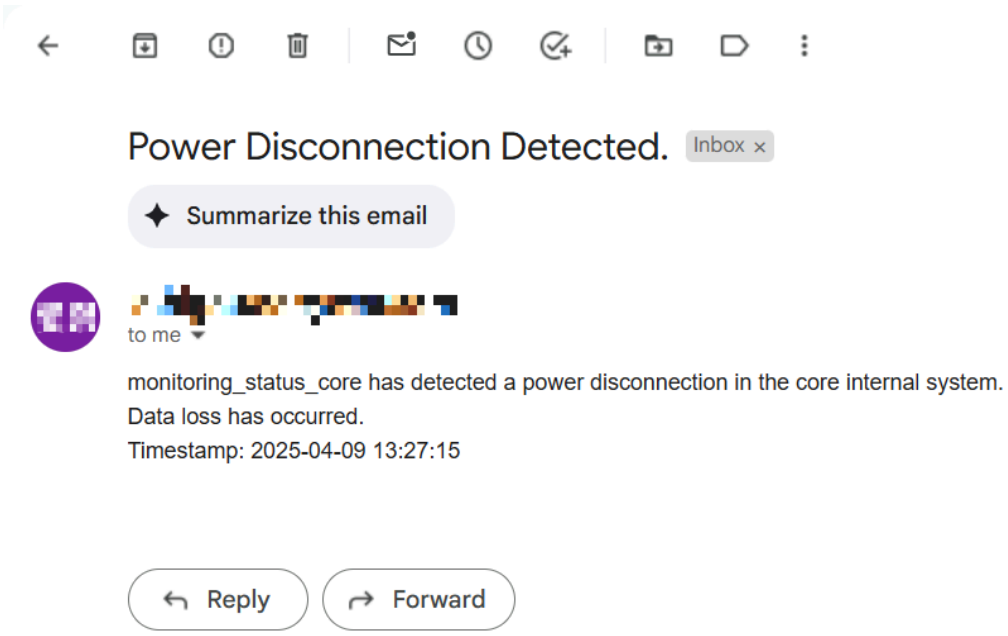
4. Connect email sending to event data and start the component.



The settings for sending email are now complete.



When a power failure actually occurs, the data is sent to the set email account. Below is an example of an email received.



8 Security Guide

This chapter describes how to use SpeeDBee Synapse securely.

8.1 User Management

This section describes how to conduct user management to improve security.

Disable admin user

admin user is the default registered user in SpeeDBee Synapse. This user has all privileges. To prevent unauthorized accesses, we recommend that you disable it unless it is necessary. If you want to disable it, you need to add another user with system administrator privileges beforehand.

For information about how to disable a user, see “User Management”.

Strong password policy

It is desirable to register strong passwords for users. The following is an example of a strong password policy. When using tools that automatically generate passwords, it is recommended that the following conditions be met:

- Must be at least 8 characters in length
- At least one uppercase and lowercase letter, a number and a symbol are all used.
- Must be different from passwords for other accounts.
- Cannot be inferred from names or other personal information.
- English words are not used as they are.
- Do not lay out them in a way that is easy to make analogies and do not use easy combinations of them.

8.2 Using HTTPS

When operating SpeeDBee Synapse, HTTPS is used to encrypt communications. This will increase the level of security.

For details of how to use HTTPS, see "HTTP/HTTPS Communication".

About the certificate to use

When performing encrypted communication using a certificate, please use a proper certificate issued by an official certificate authority.
If you use a self-certificate temporarily for purposes such as verification, create a certificate that includes a SAN (Subject Alternative Name). The certificate used to encrypt each component should also include the SAN.

8.3 Secure Token Management

An access token indicates the access rights to a resource when accessing SpeeDBee Synapse from outside. If the access token is misused by a rogue attacker, the attacker may be able to gain unauthorized access to resources in SpeeDBee Synapse. Therefore, we recommend that you keep access tokens under lock and key to prevent them from falling into the hands of third parties.

8.4 Using Encrypted Communications

Some components can use encrypted communication to provide a higher level of security.

Item	Description
MQTT Collector	You can use encrypted communication by preparing a CA certificate.
Cloud Emitter for Azure	Communication is encrypted with this emitter.
Cloud Emitter for AWS	Communication is encrypted with this emitter.
FTP/FTPS/SFTP Emitter	Communication is encrypted by the use of FTPS or SFTP.
Send Mail	Using TLS provides more secure communication encryption.

8.5 Checking operation logs

By checking operation logs on a regular basis, you can detect suspicious activities at an early stage. For details about how to check operation logs, see “Logs of various types”.

9 Backward Compatibility

This chapter describes backward compatibility.

9.1 Backwards Compatible Target

The idea of backwards compatibility for SpeeDBee Synapse is explained. Backward compatibility applies to the first "SpeeDBee Synapse" released in fiscal 2024. Other products and similar products before this date are not subject to backward compatibility. (Products are not compatible.)

Target	Description
Configuration information	Various types of information set by the user (Includes user management information such as panel and component settings.)
Component	Components provided by SpeeDBee Synapse. Contains custom components added by the user.

9.2 Backwards compatible

The basic concept of backward compatibility is to provide products with backward compatibility in mind.

- Revision, Minor Version Update

You can update the existing configuration and the backward compatibility is maintained. (The structure of the internal settings may be changed by the update process, but the user does not need to be aware of it.)

- Major update

This is an update when a major function expansion is made. In this case, the principle is to try to maintain backward compatibility. But there may be cases where backward compatibility is not maintained. The update procedure in such cases will be detailed at the time of the update. We also provide individual information.

10 Execution License

This chapter describes the SpeeDBee Synapse execution license.

10.1 License Types

Describes the license types.
There are currently three types of licenses for SpeeDBee Synapse.

Item	Description
Development license	This license allows all functions and operations of SpeeDBee Synapse. This license is usually used to build systems. Except for this chapter, the descriptions in this manual assume a development license.
Execution license	This license is for an execution environment in which some of the functions of SpeeDBee Synapse can be used. Under this license, you can take over the configuration built by the development license and run it in multiple environments.
Evaluation license	This is a license to run SpeeDBee Synapse for evaluation purposes. There is a limit of one hour of continuous operation after a component has been started. This is a license used for evaluation purposes before purchasing SpeeDBee Synapse.

About Execution Licenses

An execution license cannot be used by itself. You must operate with at least one development license.

10.2 Operating specifications of the execution license usage environment

This section describes the operating specifications of SpeeDBee Synapse with an execution license.

The table below shows the availability of typical functions for development licenses and execution licenses.

Item	Development License	Execution License
User Management/Authentication	○	○ Note: Development license user information is not taken over.

Item	Development License	Execution License
Grafana Collaboration	○	○ Note: Development license settings are not taken over.
Custom Components (Python, C)	○	○ Note: You must manually configure custom components and external modules
External Control Web-API	○	○
Publish access token	○	○ Note: Development license settings are not taken over.
Encrypt communication	○	○ Note: Development license settings are not taken over.
DB query service	○	○ Note: Configuration is taken over, but token authentication must be manually reconfigured on the Execution license side.
Run Mode	○	○
Edit Mode	○	△ Note: It's available on a limited basis.
Adding, Updating and Deleting Panels	○	×
Register New Component	○	×
Configuration updates for configured components	○	○
Deleting Configured Components	○	×
Adding and Deleting Flow Links	○	×
Configuring Various Ports	○	○

Item	Development License	Execution License
Filter Settings	○	×
Data monitor	○	○
Status Timeline	○	○
Log	○	○
Execution license configuration Download	○	×
Project Settings	○	×

Importing settings for execution licenses between different OSs

If the OS of the development license operating environment and the execution license operating environment are different, some settings are automatically converted when the settings for the execution license are imported. Note, however, that the settings other than those listed below are not converted, so you may need to reconfigure them manually.

- For the case of Linux on the development license side -> Windows on the execution license side

  Converts the path in the configuration file to:

  ```
  /var/speedbeesynapse -> C:\ProgramData\SALTYSTER\SpeeDBeeSynapse
  ```

- For the case of Windows on the development license side -> Linux on the execution license side:

  Converts the path in the configuration file to:

  ```
  C:\ProgramData\SALTYSTER\SpeeDBeeSynapse -> /var/speedbeesynapse
  ```

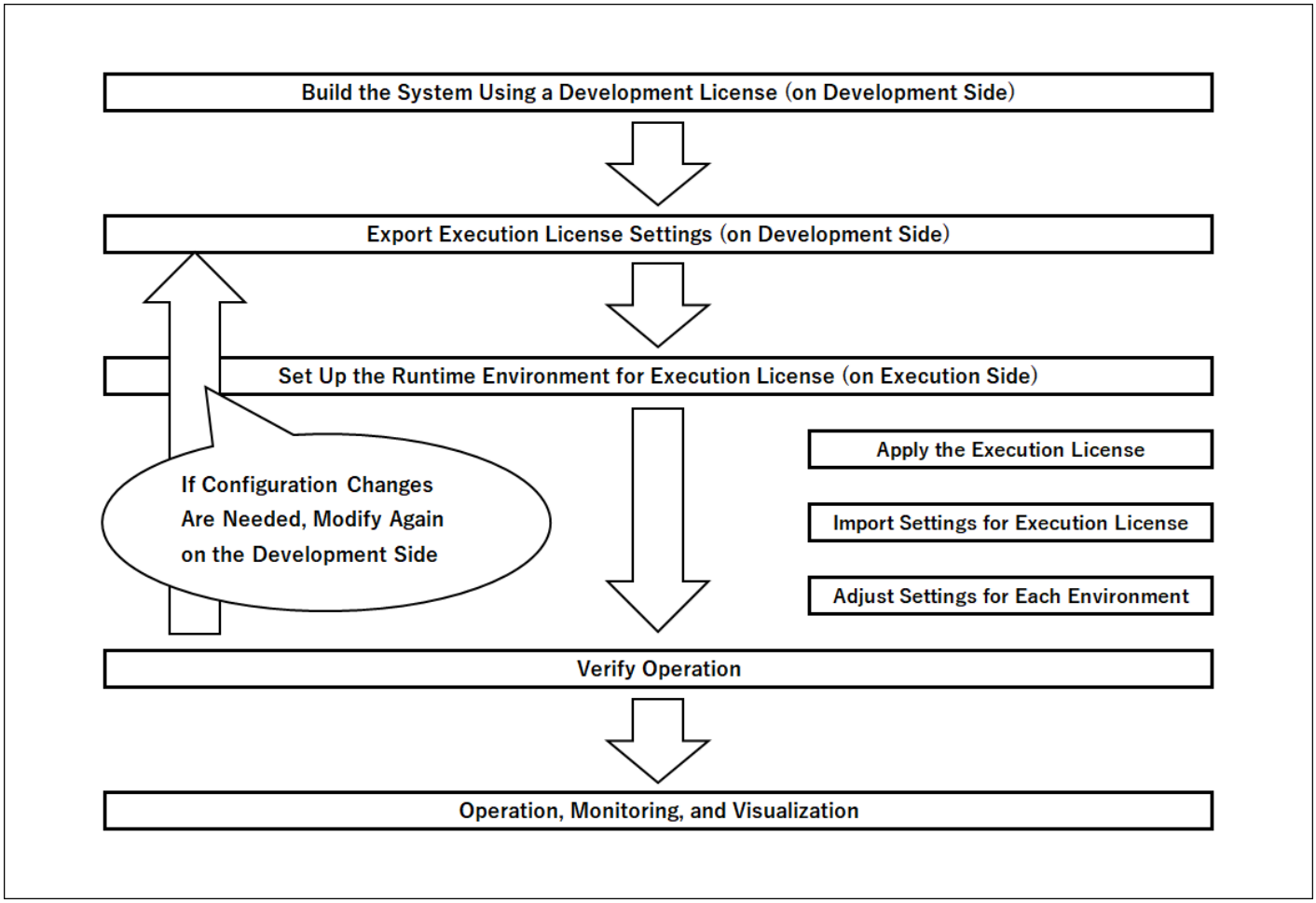
Note: If there is any abnormality after importing the settings, manually reconfigure them.

10.3 Execution license operation method

This section describes how to use execution licenses.

10.3.1 Operation Flow

The following figure shows the operation flow for execution licenses.



(1) System building using a development license (at the side of development license operations)

First, build a system using SpeeDBee Synapse with a development license. Set the data flow such as data collection, output format, data transmission and linkage. The specific settings depend on the purpose of the system, but refer to the Installation flow for reference.

(2) Output of execution license settings (development license side operations)

After setting the data flow to be used with the execution license, output the execution license setting. This operation is only available for SpeeDBee Synapse with a development license.

(3) Execution license Setting up the operating environment (operations on the execution license side)

Set up the operating machine for the execution license and install SpeeDBee Synapse. After installation, apply the purchased execution license. This applies the settings for the execution license, but you might need to specify settings for each execution license operating environment. For example, it may be necessary to set the IP address of an external device according to the usage environment.

(4) Operation check

After setting up the operating environment for Execution license, actually start the system. Make sure that the settings of the external device are correct. Note that you cannot use this operation to create a new data flow by adding a new component. If you need to reconfigure the system, you need to (1) build the system using the development license, and (2) output the execution license configuration again to import the settings for the execution license.

(5) Operation, monitoring and visualization

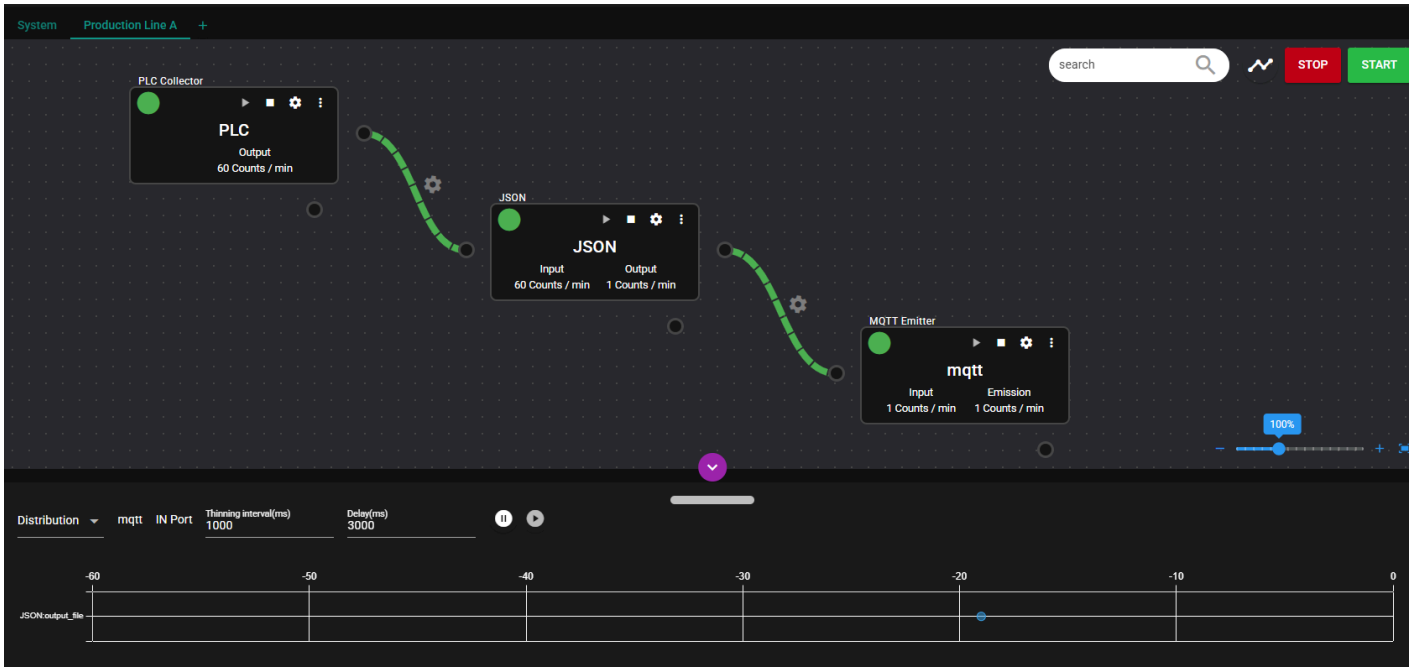
Start up the operation in the operating environment of execution license. If it becomes necessary to change the data flow, perform the operation on the development license side again, and change the settings according to this procedure.

10.3.2 Quick Tutorial

This section describes the procedure for creating an actual execution license operating environment. This tutorial assumes that the machine on which the development license is running and the machine on which the execution license is running are located in the same network. It is assumed that SpeeDBee Synapse has already been installed.

The data flow used in this tutorial is as follows:

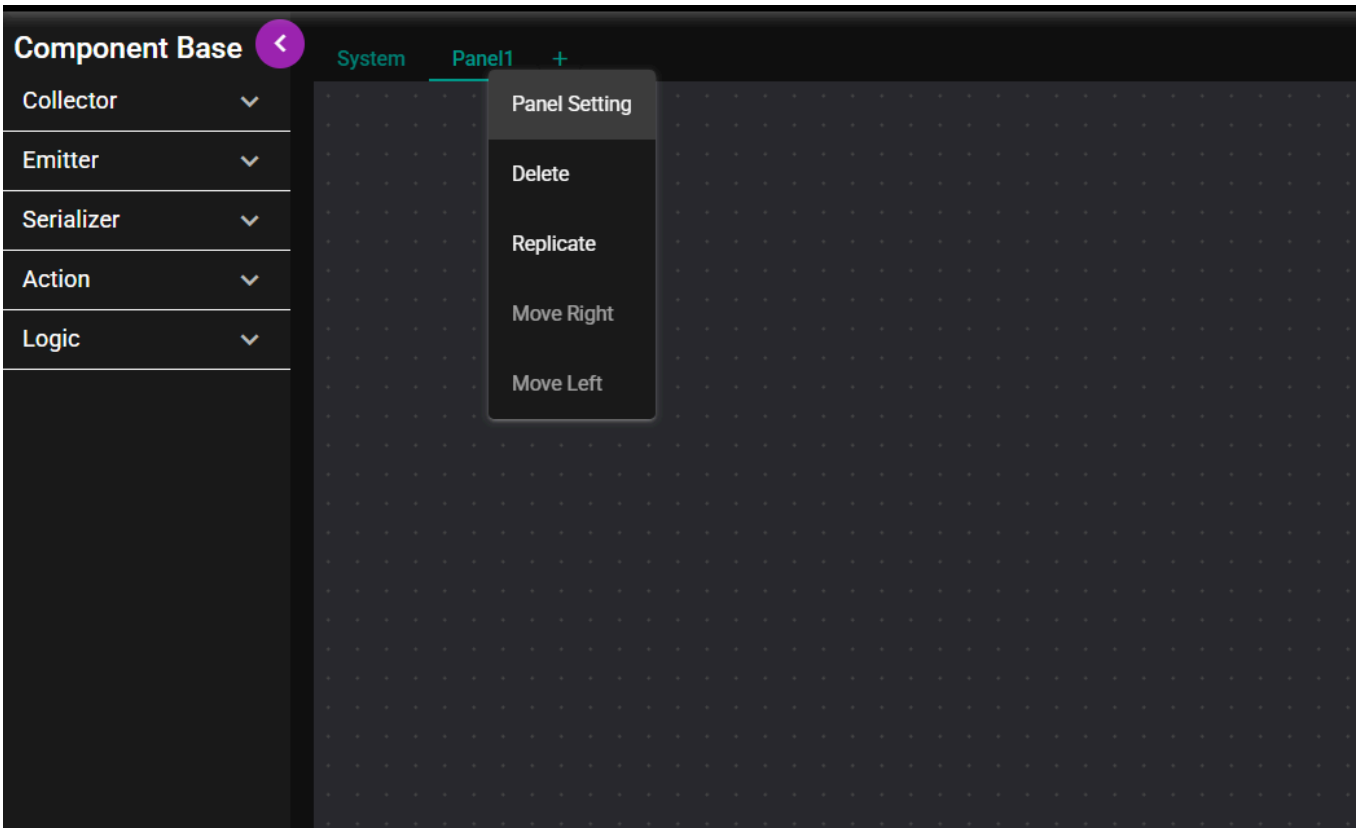
Item	Description
Production Line A	Data collected by PLC collector is sent to outside by MQTT in JSON format.



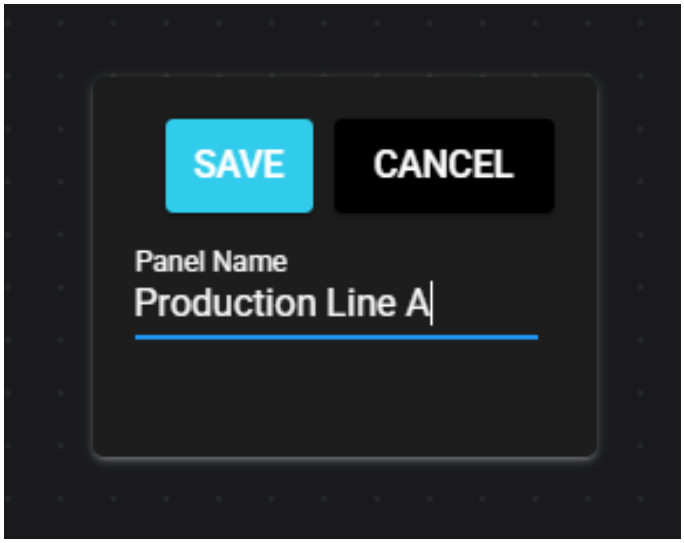
10.3.2.1 Building a System Using a Development License

This operation is only available for SpeeDBee Synapse under a development license. Use the following instructions as a reference to configure the PLC collector, JSON serializer and MQTT emitter.

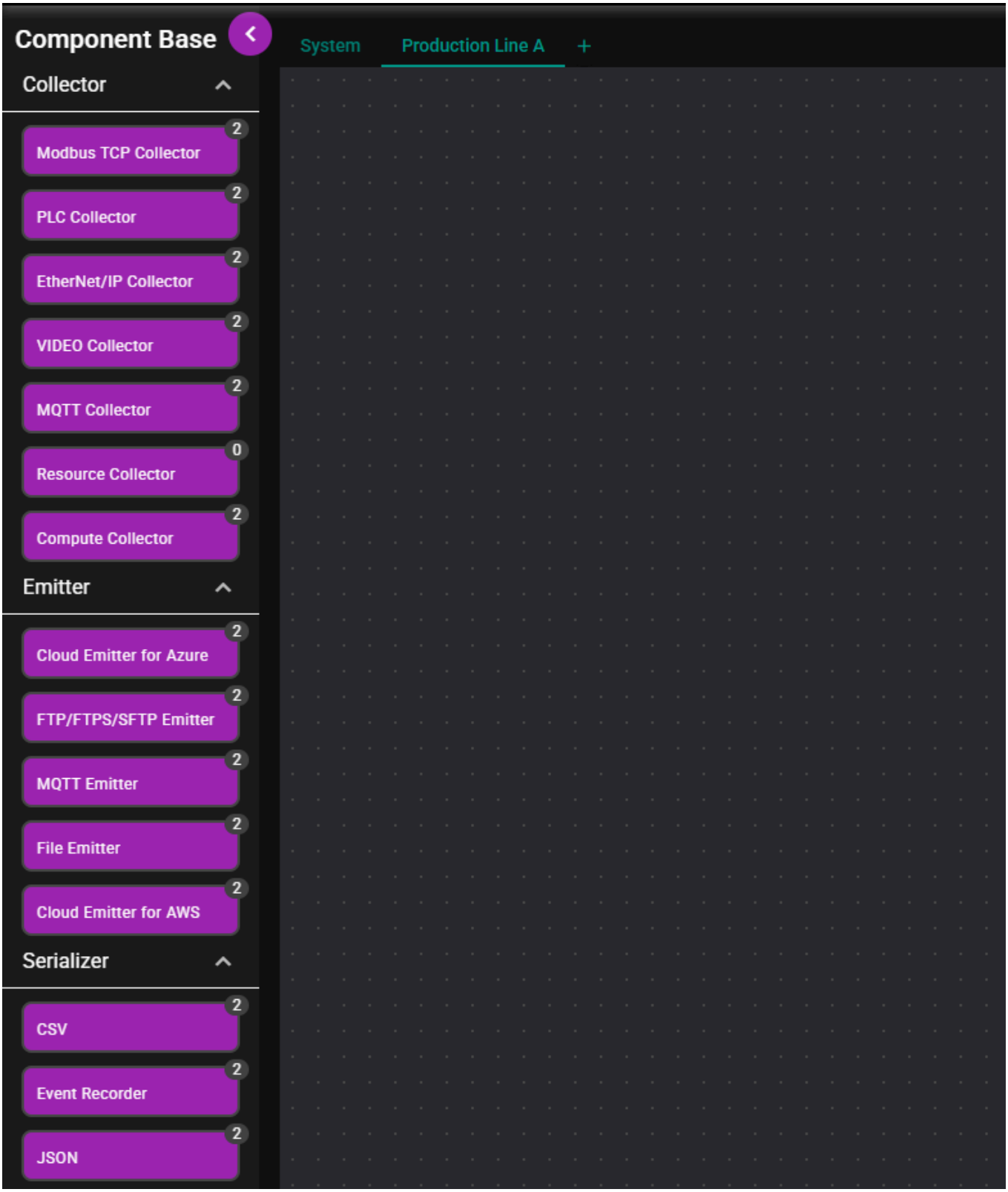
1. Select “Panel 1” and click “Panel Setting” on the right-click menu.



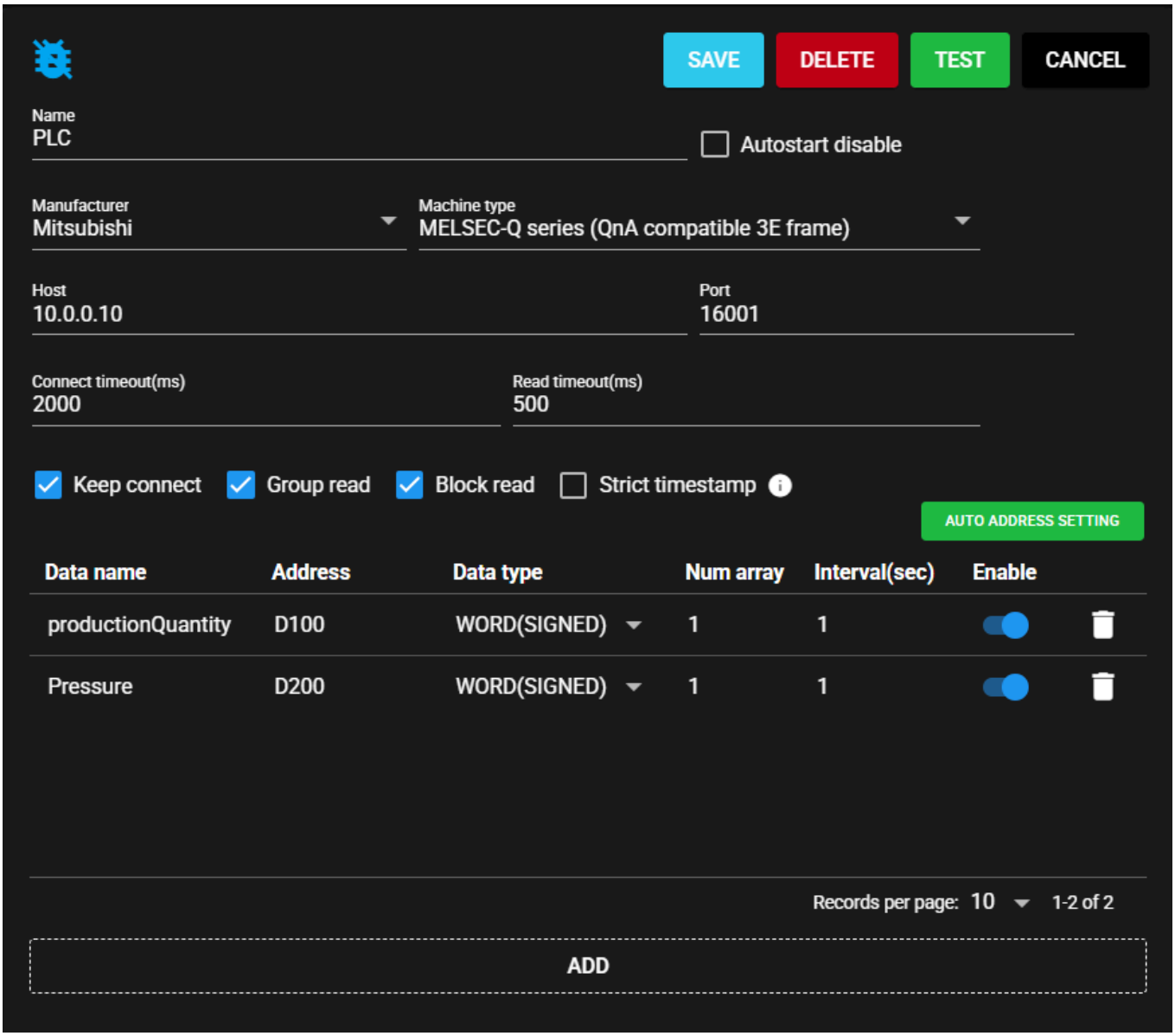
2. Change "Panel 1" to "Production Line A" and save.



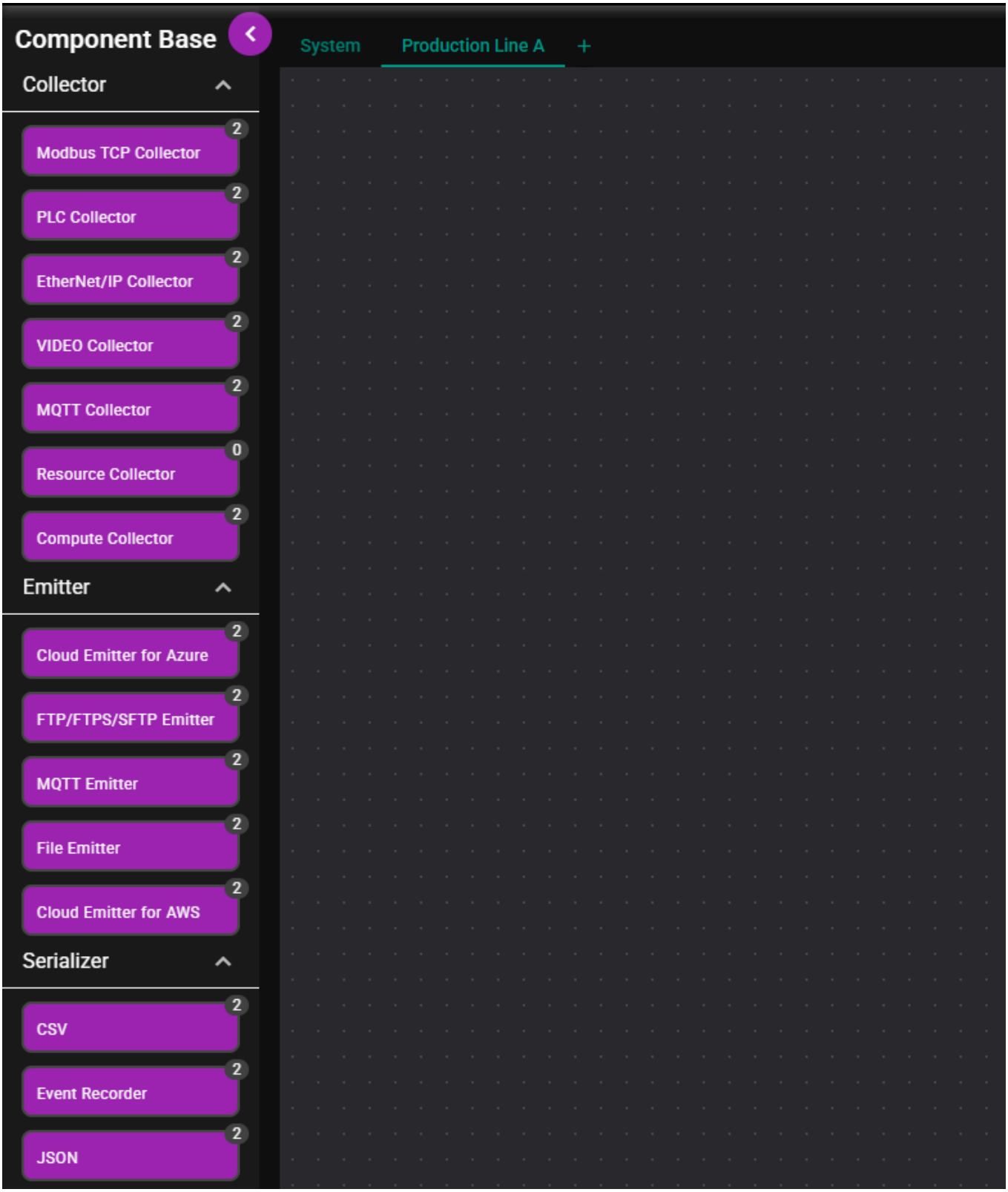
3. Configure the PLC collector settings. Drag and drop “PLC Collector” from the left side of the Flow Editor to the center of the Flow Editor.



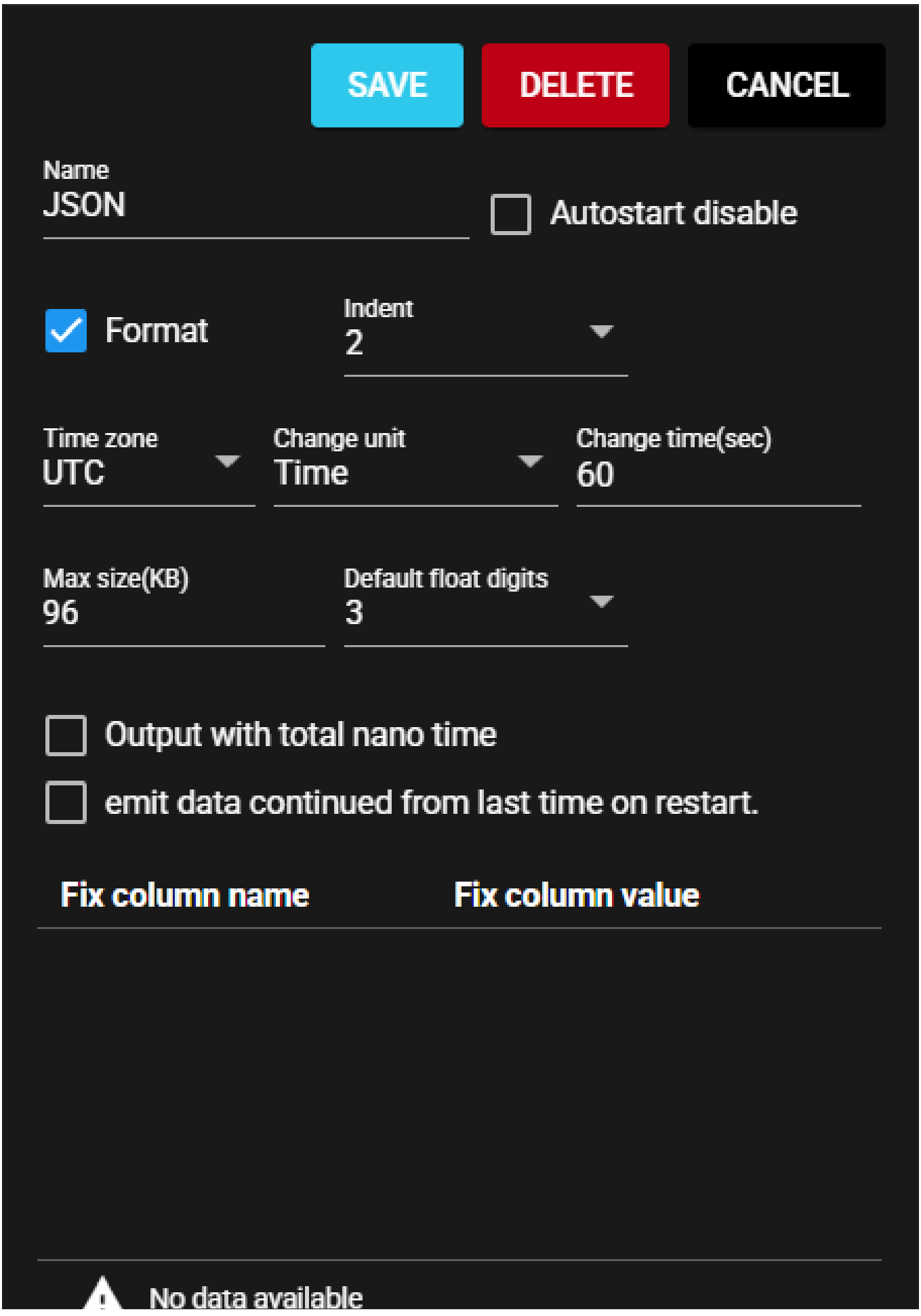
4. Enter settings for the PLC collector and click “Save.” Please adjust the IP address, port and data according to your environment.
   (Note: If the PLC collector is not available, replace it with a resource collector.)



5. Configure the serializer. Drag and drop "JSON" from the left side of the Flow Editor to the center of the Flow Editor.

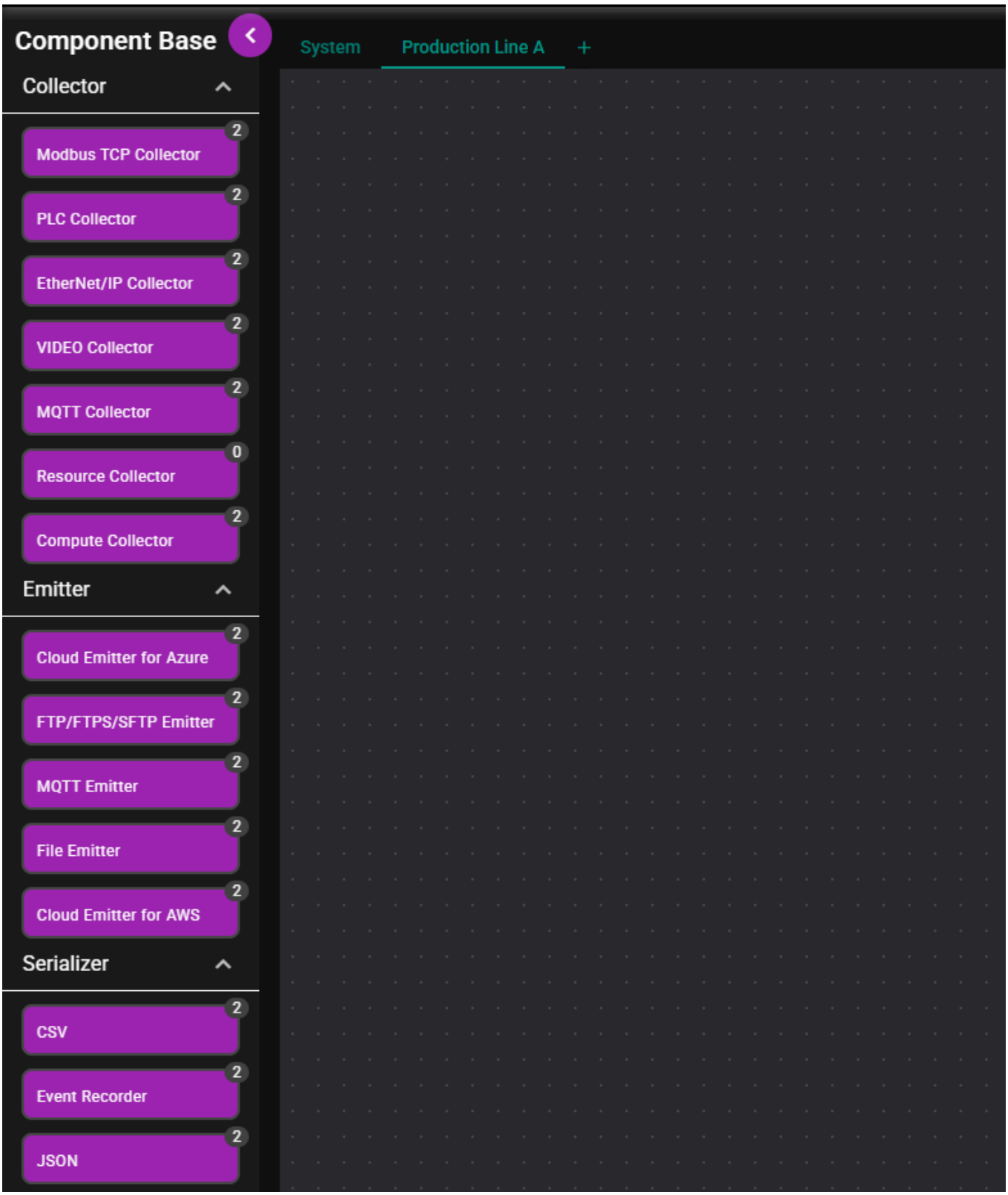


6. Enter the JSON settings and click “Save.”

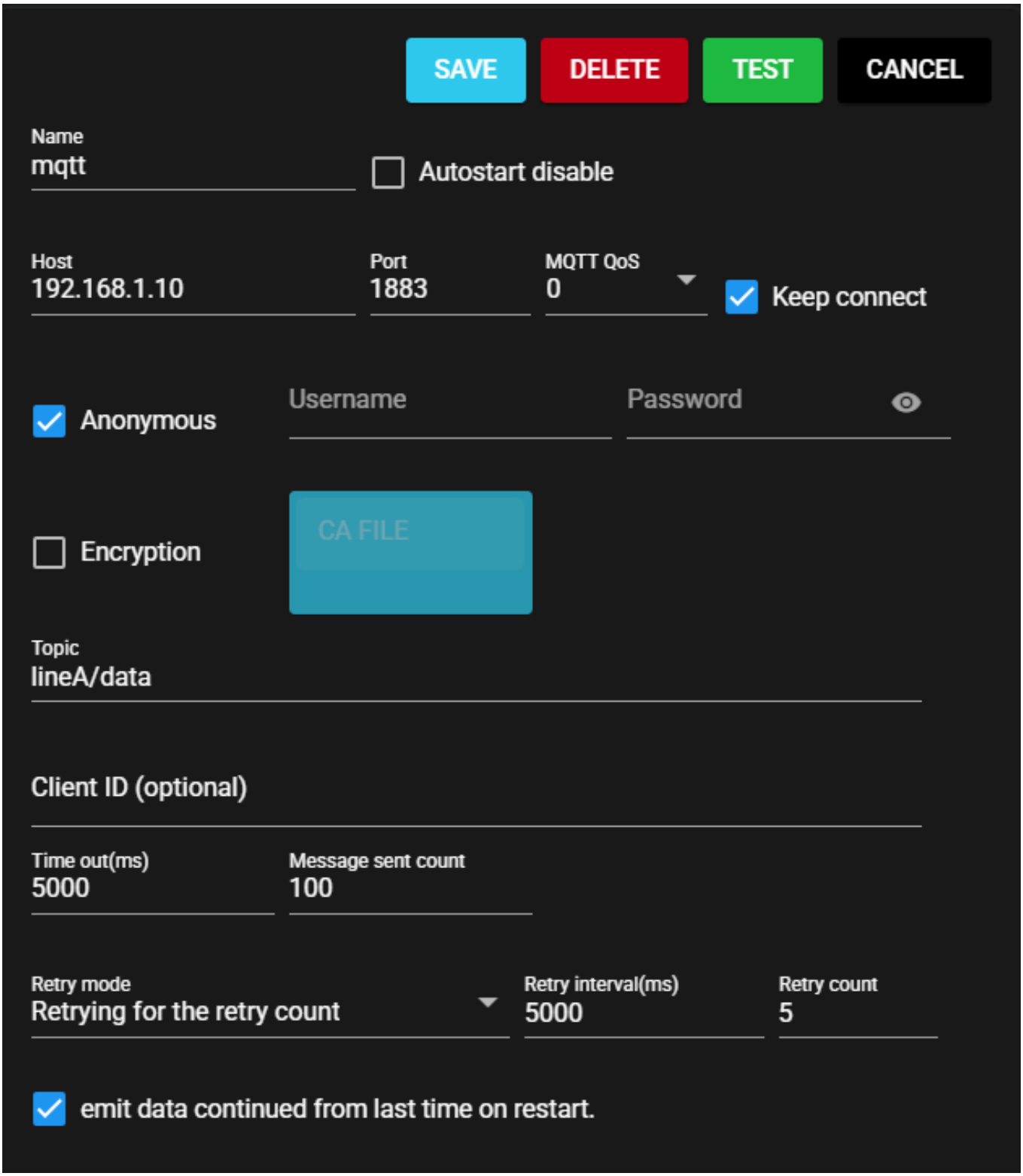

SAVE
DELETE
CANCEL
Name
JSON
Autostart disable
Format
Indent
2
Time zone
UTC
Change unit
Time
Change time(sec)
60
Max size(KB)
96
Default float digits
3
Output with total nano time
emit data continued from last time on restart.
Fix column name
Fix column value
No data available



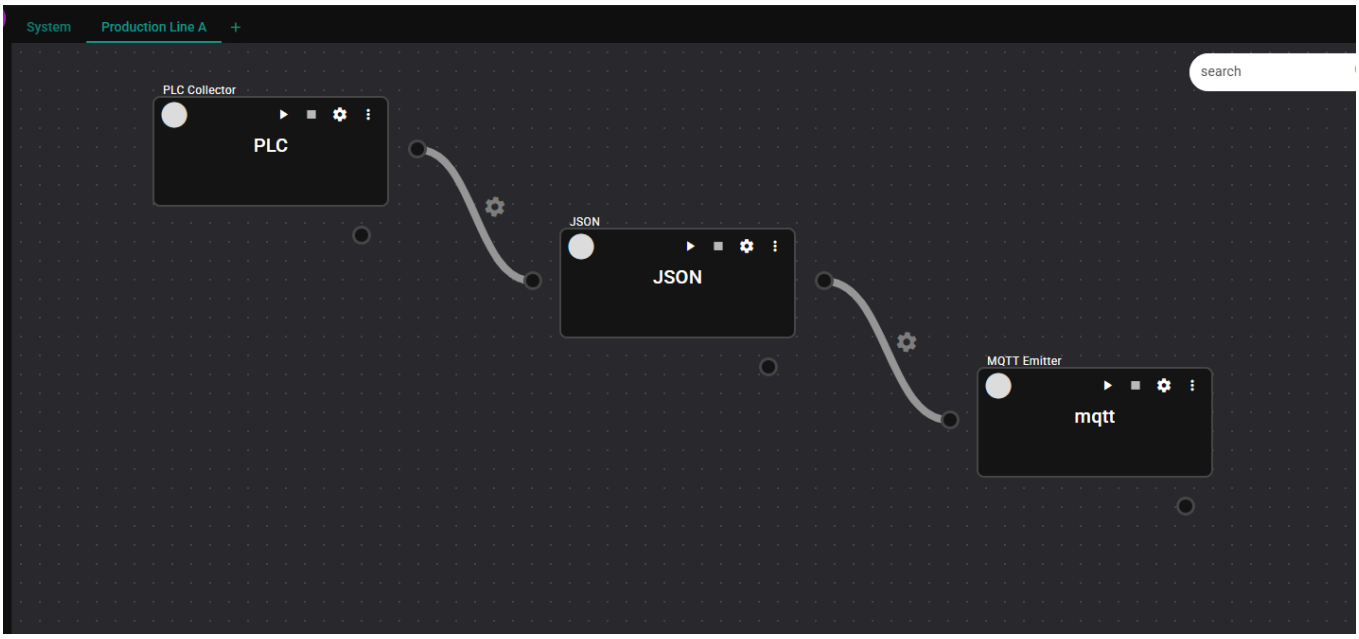
7. Configure the MQTT emitter. Drag and drop the “MQTT Emitter” from the left side of the Flow Editor to the center of the Flow Editor.


Component Base
System
Production Line A
+
Collector
Modbus TCP Collector
2
PLC Collector
2
EtherNet/IP Collector
2
VIDEO Collector
2
MQTT Collector
2
Resource Collector
0
Compute Collector
2
Emitter
Cloud Emitter for Azure
2
FTP/FTPS/SFTP Emitter
2
MQTT Emitter
2
File Emitter
2
Cloud Emitter for AWS
2
Serializer
CSV
2
Event Recorder
2
JSON
2

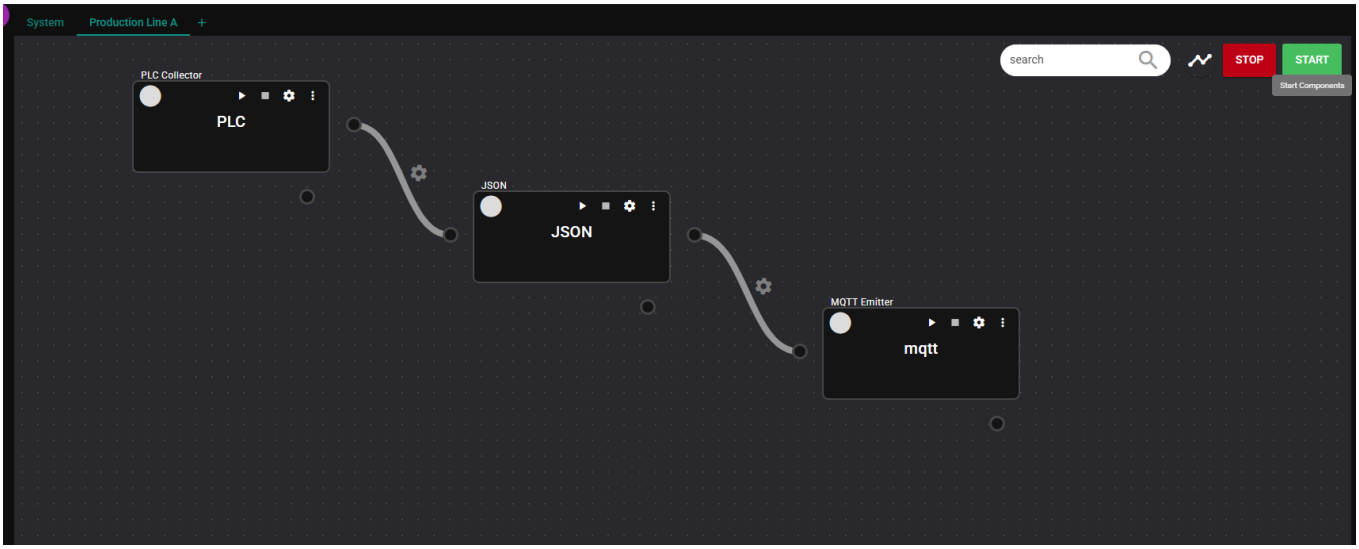
8. Enter the MQTT emitter settings and click “Save.” Please adjust the IP address, port and so on according to your environment.
   (Note: If the MQTT emitter is not available, replace it with a File emitter and so on.)



9. Connect flow links.

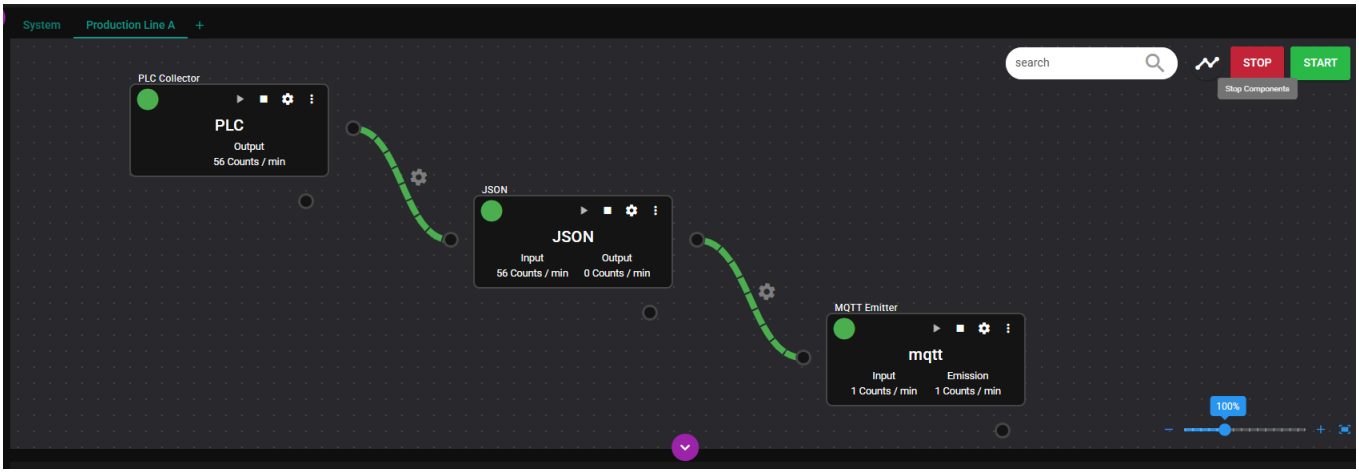


10. Make the component work. Verify that each component works properly. Let it operate for a certain period of time and check whether any abnormality occurs.





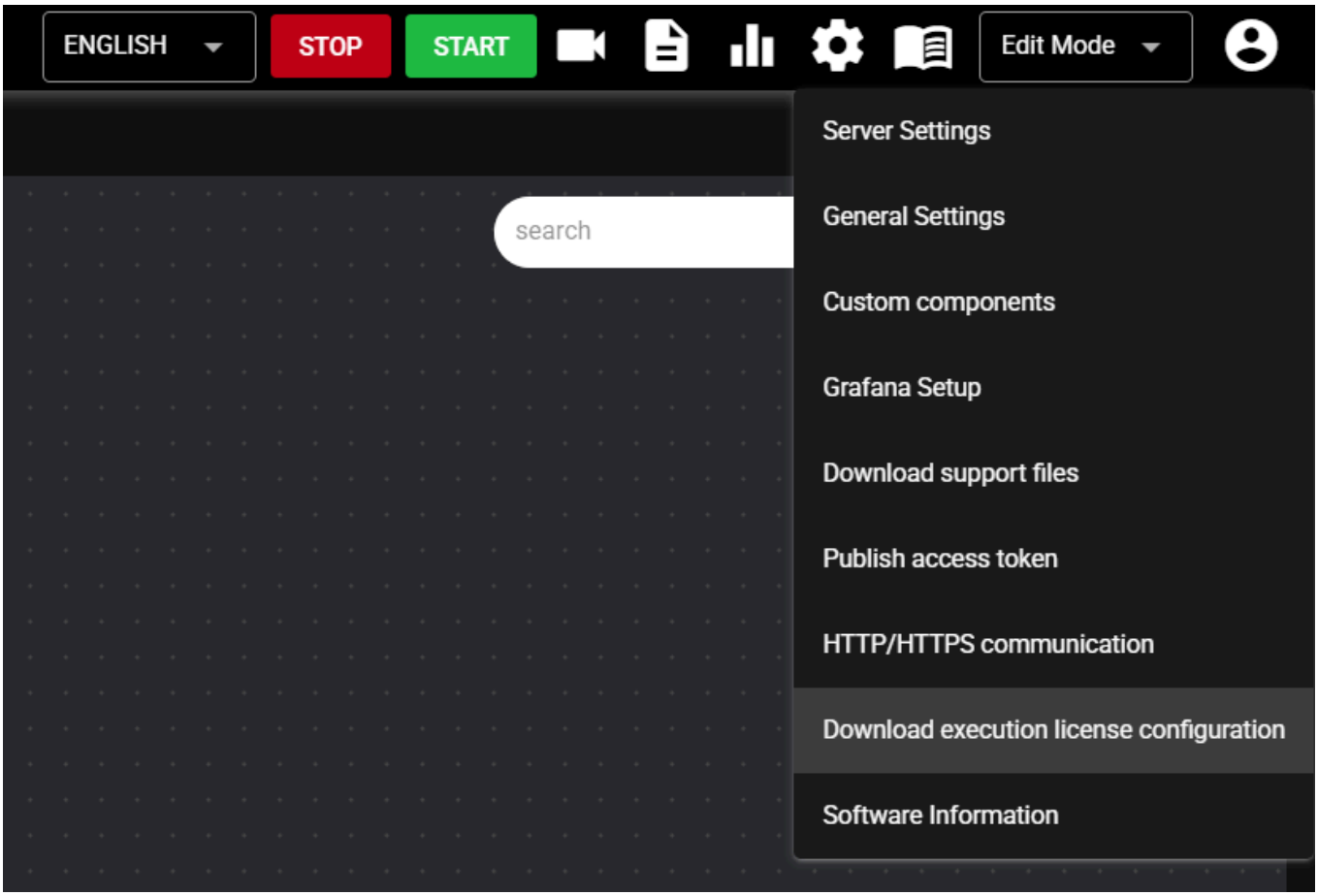
11. When you have verified that everything is working properly, stop the components.



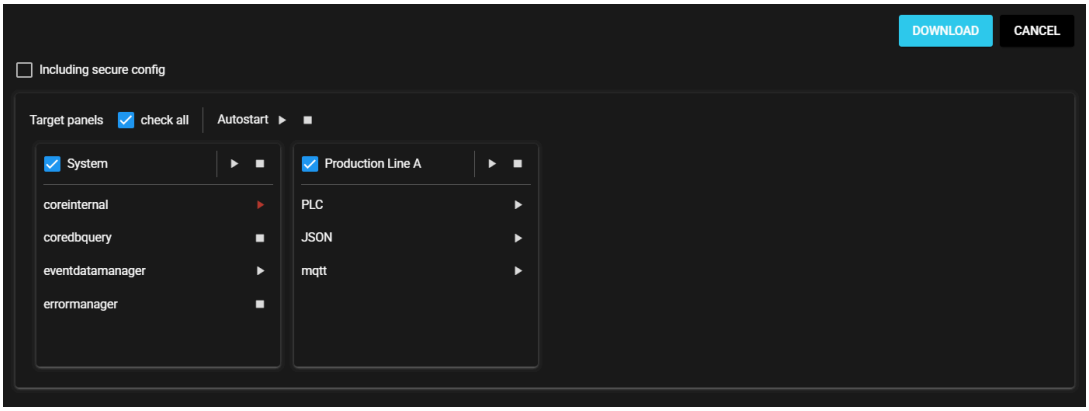
10.3.2.2 Output of execution license settings

After setting up in the development license operating environment, output the settings for the execution license.

1. Click “Download execution license configuration” in the Setting menu. This file is used in the execution license operating environment.



2. In the confirmation dialog box, click “Download.” If you want to include confidential information, check “Including secure config” before clicking “Download.” If you want to include only selected panels, check on the panels to include. You can also specify the initial Autostart settings. ▶(Autostart) ■ (Autostart disable)



> **About confidential information**
>
> When you perform Download execution license configuration that includes confidential information (For example, Azure connection strings, user names such as MQTT, and passwords), be aware that confidential information is included in the output file.
> The certificate file set for each component and the file attached to the custom component are also treated as confidential information. If you do not include confidential information, it is not outputted, so you must set confidential information individually on the machine running the execution license.

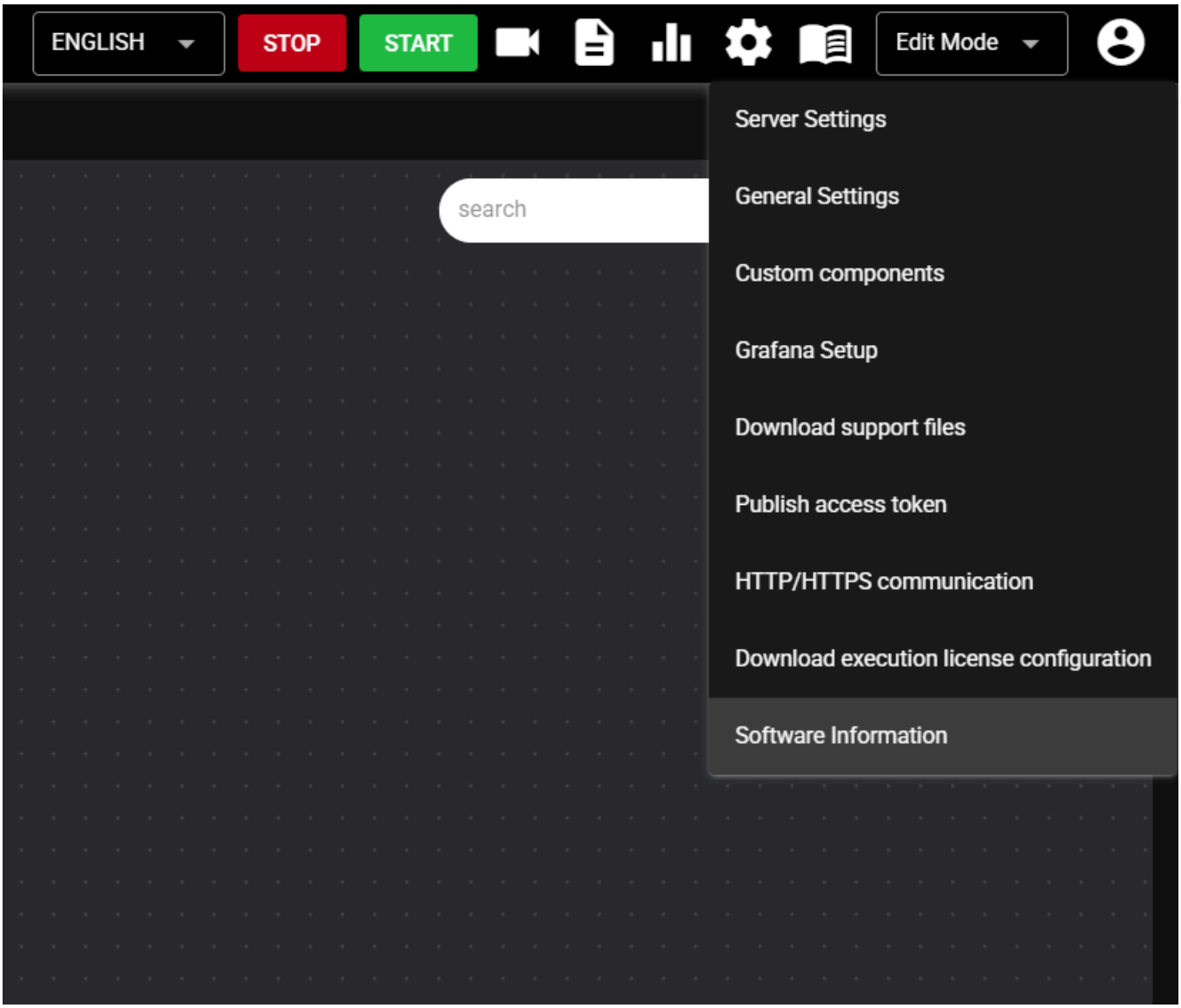
3. A file similar to the following is downloaded: The file name format is "execution_license_package_year_month_date.pkg."

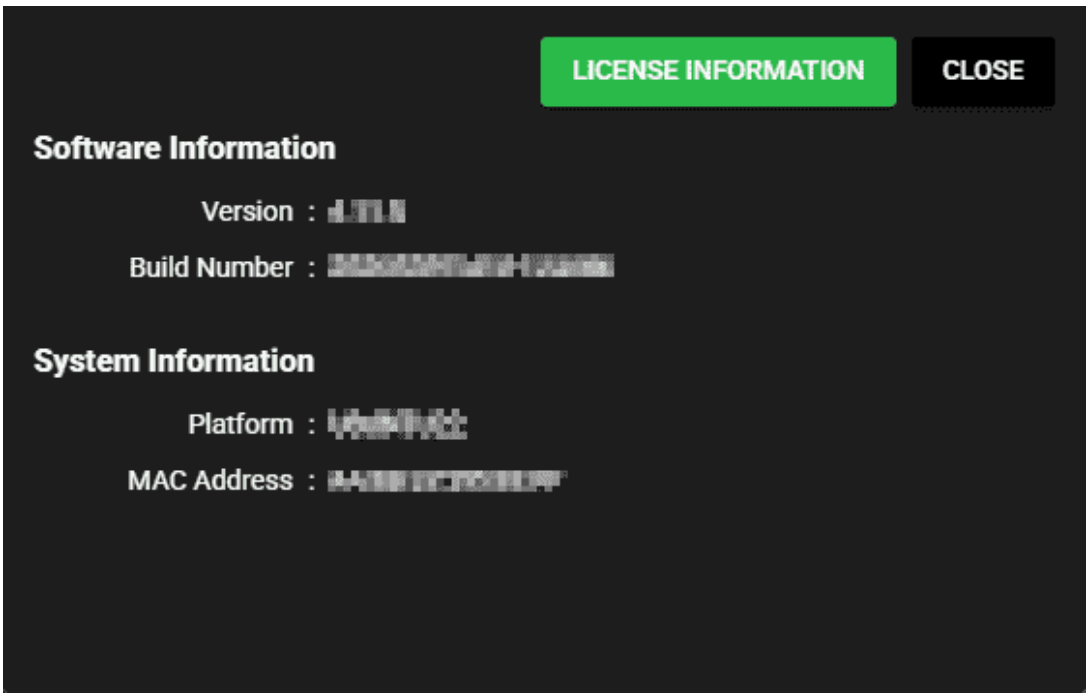


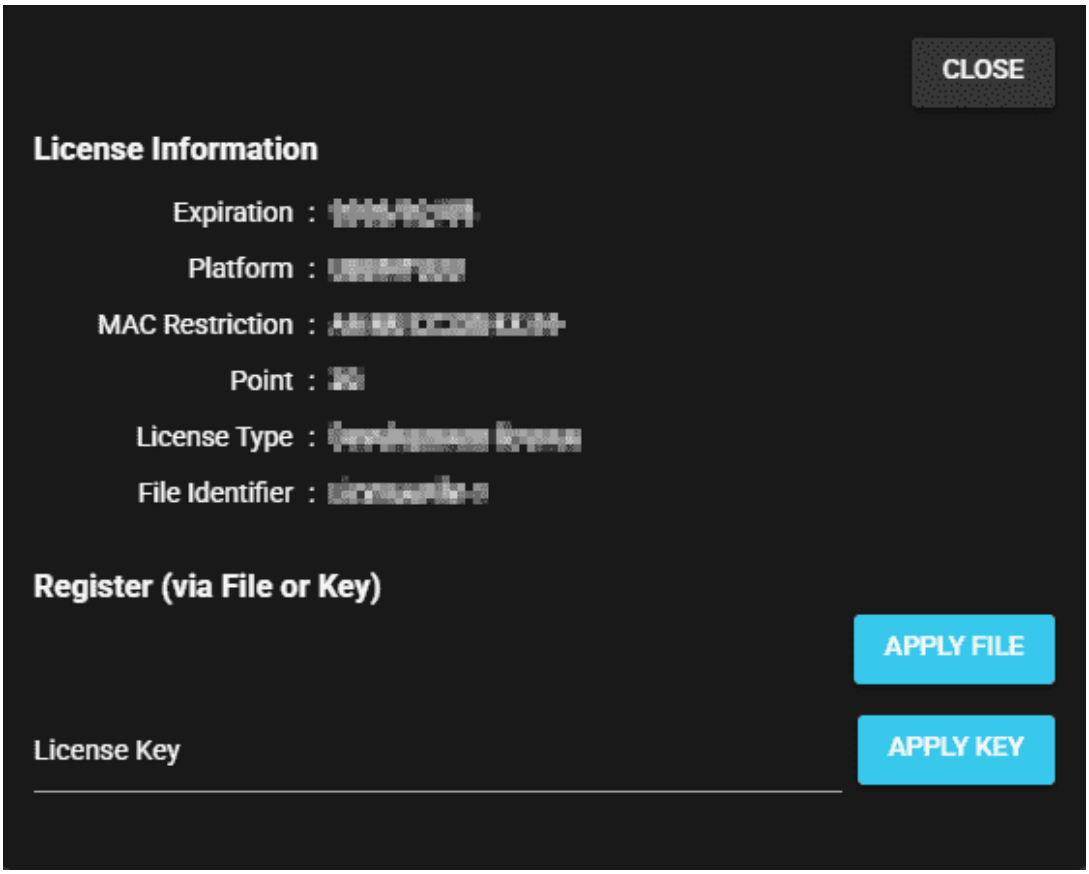
10.3.2.3 Setting up of the execution license operating environment

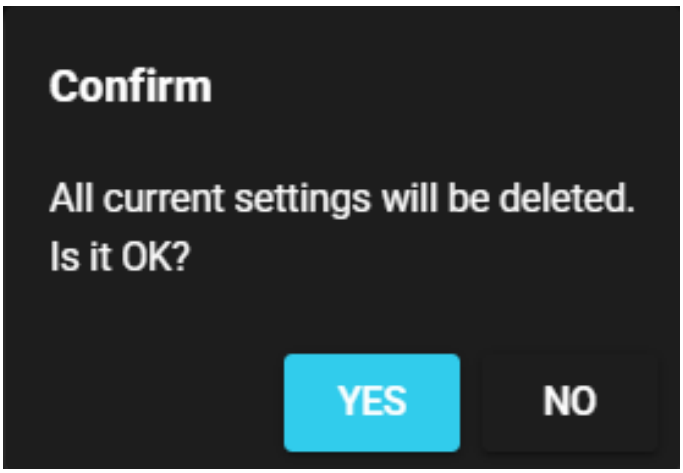
The settings for the execution license operating environment are performed. Set the SpeeDBee Synapse to Edit Mode.

1. Register your license from the “Software Information” in the Settings menu.



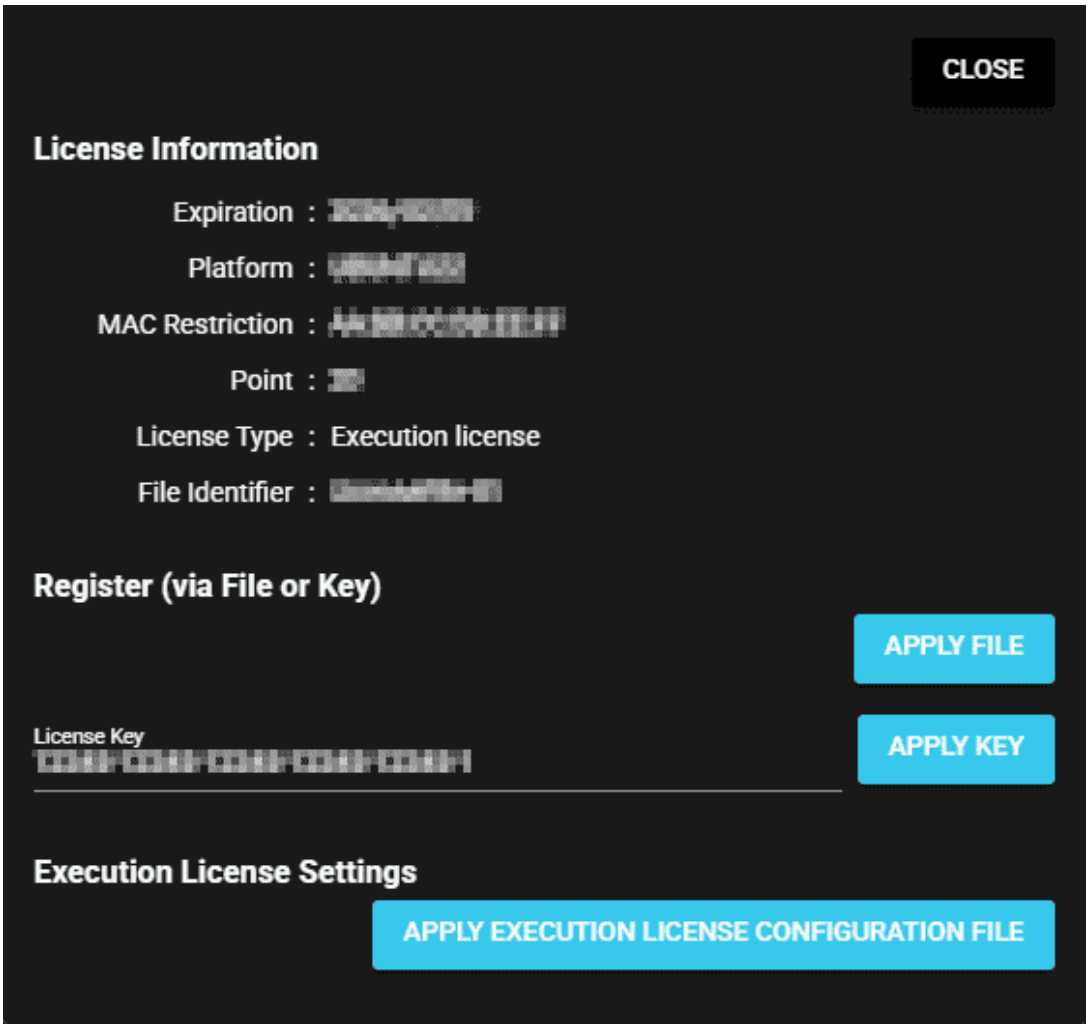

LICENSE INFORMATION
CLOSE
Software Information
Version :
Build Number :
System Information
Platform :
MAC Address :


CLOSE
License Information
Expiration :
Platform :
MAC Restriction :
Point :
License Type :
File Identifier :
Register (via File or Key)
APPLY FILE
License Key
APPLY KEY

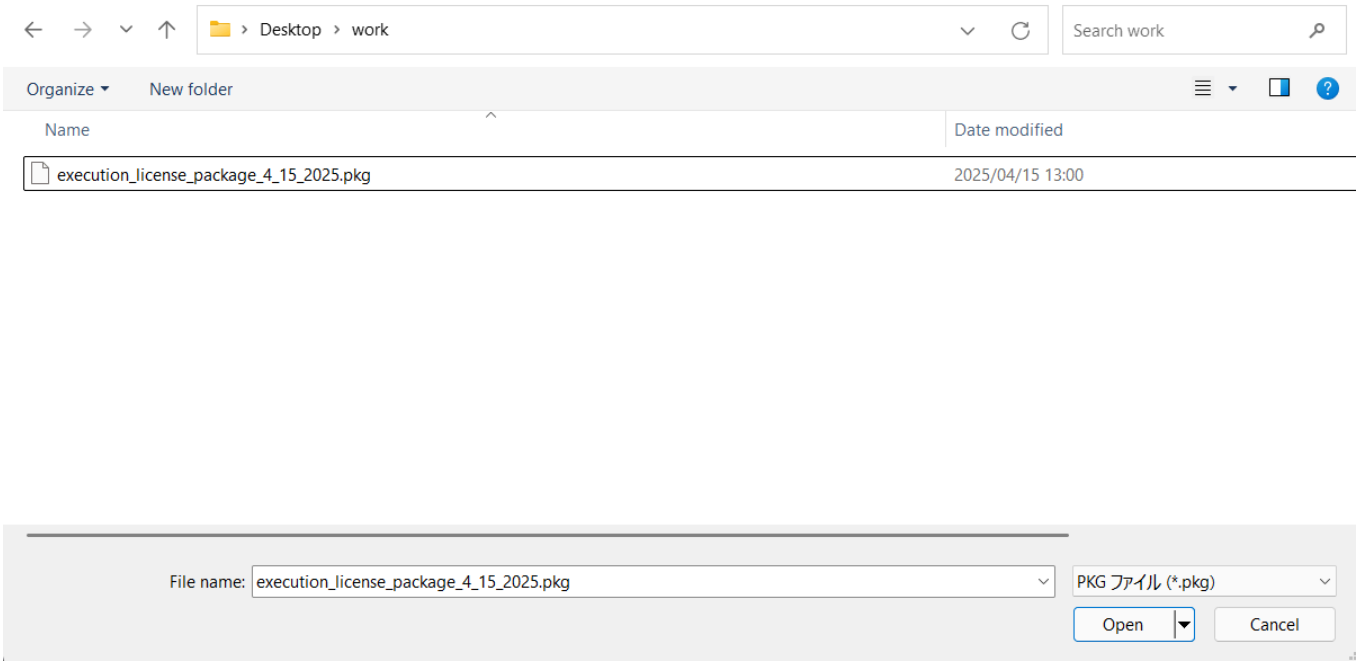


2. Review the message and, if acceptable, click Yes.

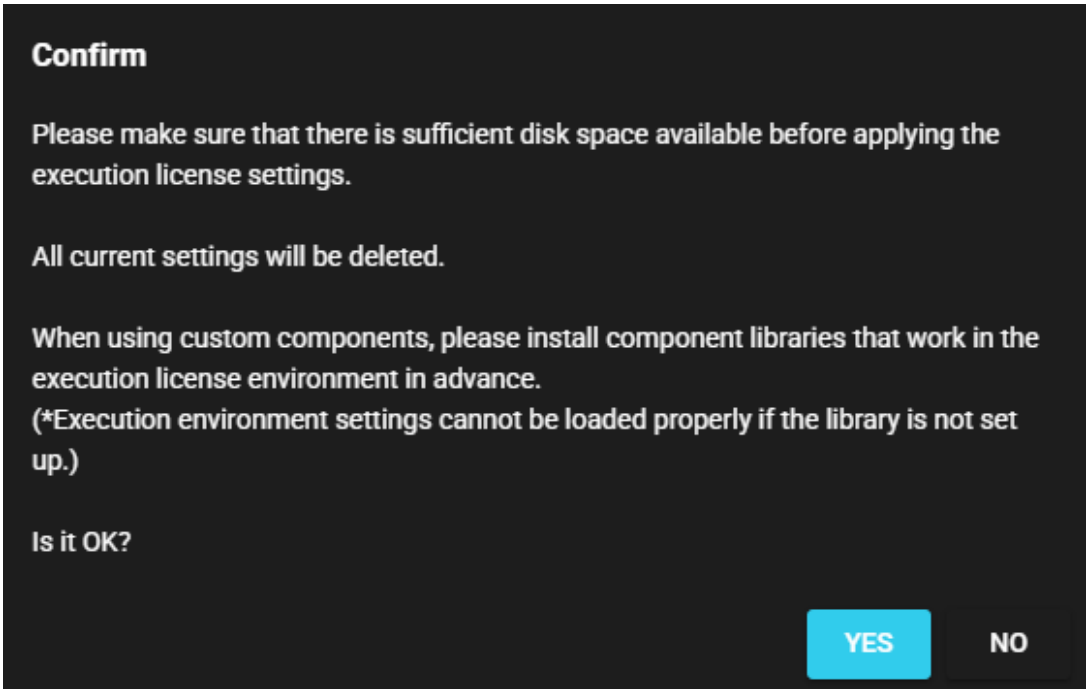
3. Import the execution license configuration file. Click “Apply Execution License Configuration File”



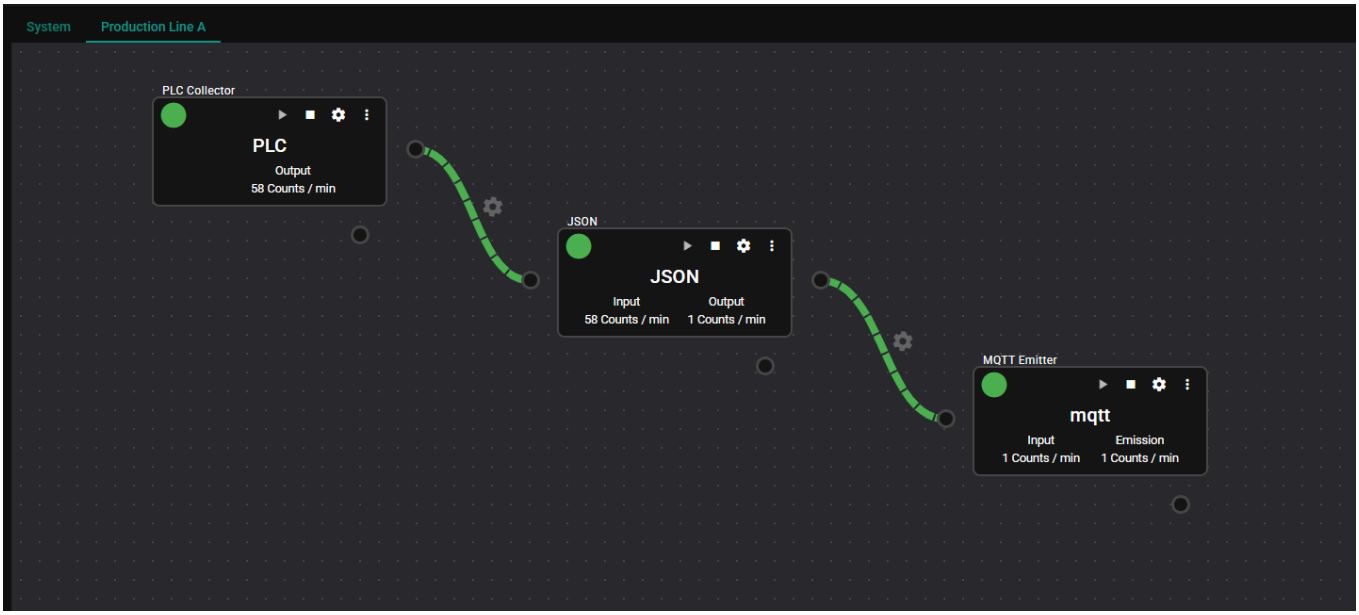
4. Select the configuration file for the execution license.



5. Review the message and, if acceptable, click Yes.

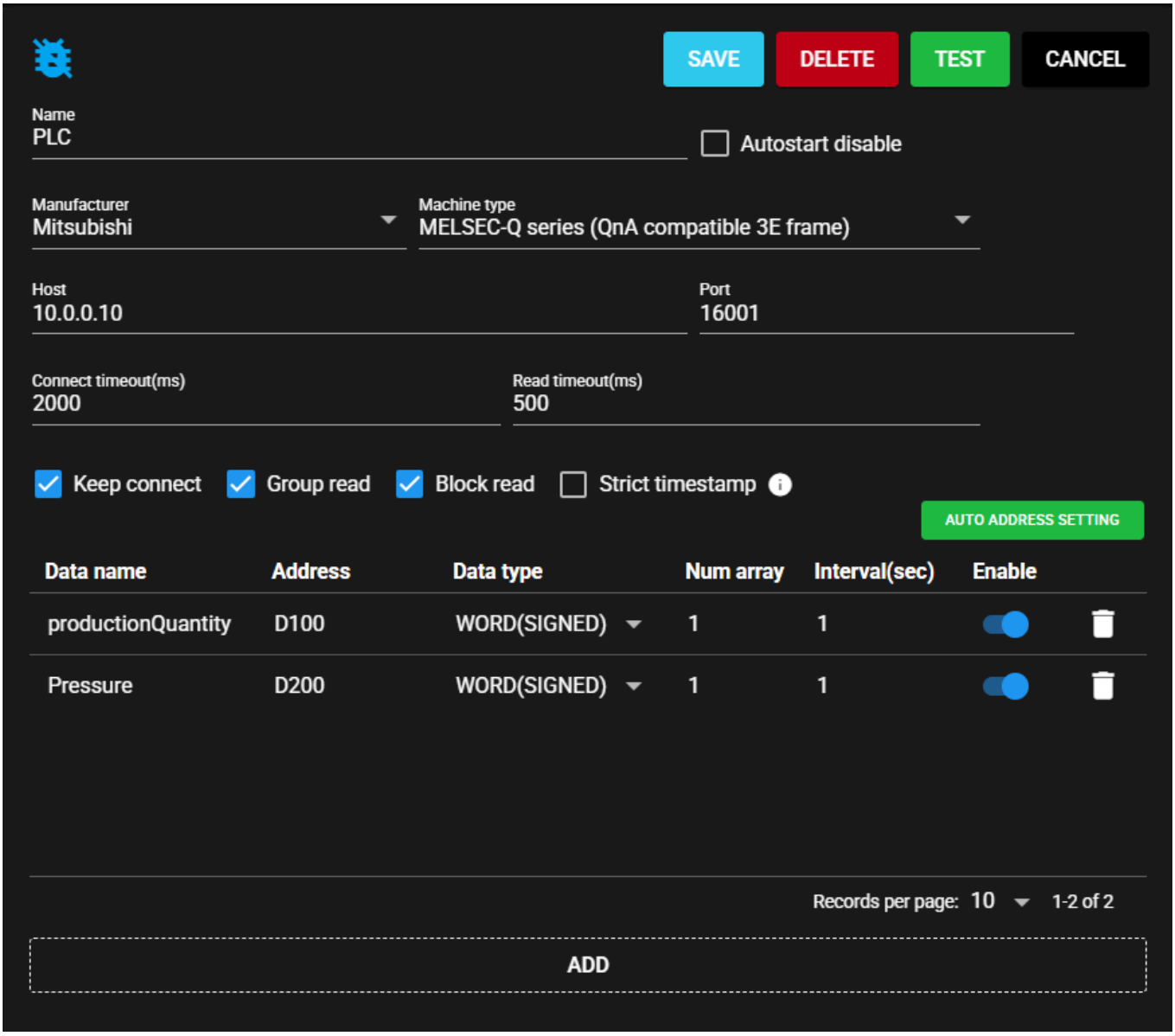


6. After a certain amount of time, the component starts.
   (Note: To check the display, close the "Software Information" window.)

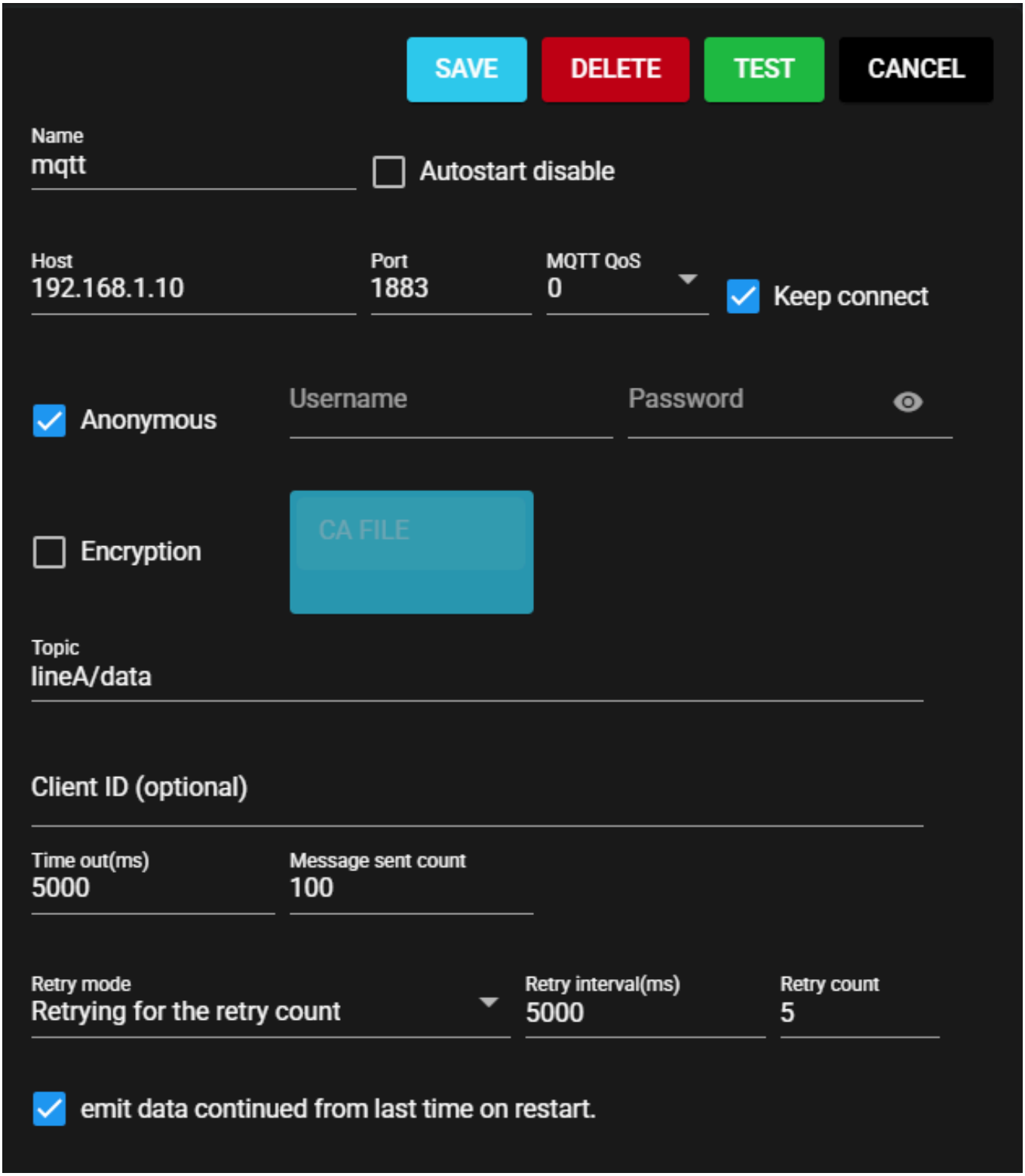


> **About components containing confidential information**
>
> If you import settings for an execution license that do not contain confidential information, those confidential information are not included, and thus components that require confidential information are disabled for automatic startup. For confidential information and attached files, reconfigure them individually.

7. Modify the PLC Collector settings as desired. For example, if the connection destination is different between the machine on which the development license is used and the machine on which the execution license is used, specify the settings.
   (Note: Since the tutorials assume the same network environment, you do not need to change the settings if the development license side is stopped.)



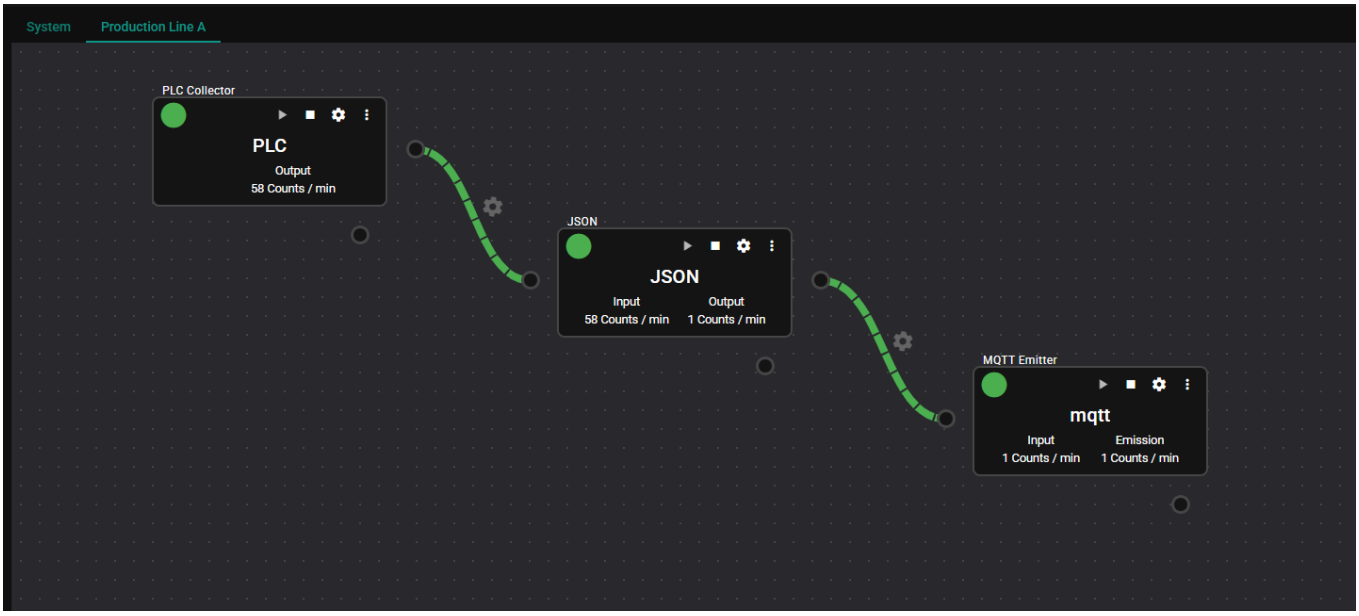
8. Modify the MQTT collector settings as needed. For example, if the connection destination to MQTT broker is different between the machine on which the development license is used and the machine on which the execution license is used, specify the settings.
   (Note: Since the tutorials assume the same network environment, you do not need to change the settings if the development license side is stopped.)



10.3.2.4 Operation Check

Confirm that the settings up to the previous section work correctly.

1. Check the component lights. Verify that there are no warnings or errors. Review the settings if there are any problems.



10.3.2.5 Operation, Monitoring and Visualization

When it is confirmed that normal operation is possible with the settings up to the previous paragraph, start operation. See the Introduction flow for monitoring and visualization.
This concludes the tutorial on operating an execution license.

10.4 Notes on the execution license operating environment

Notes on the execution license operating environment are shown below.

Notes

- When you apply an execution license, the operating mode is in accordance with the execution license and the available functions are restricted. (Reference: Operating specifications of the execution license operating environment)
- When you import the settings for the execution license, the previous settings of the selected project are deleted. (Custom components and external modules are not deleted.)
- When switching to the execution license, all projects that are not selected are deleted. Export the projects you want to keep beforehand. (Reference: Export)
- The access token settings of the development license side are not taken over. Also, the access token generated on the execution license side is not deleted even when the settings for the execution license are imported.
- The settings of the DB query service are taken over by the execution license side. However, since the access token used for token authentication is not taken over, they must be configured on the execution license side.
- If the execution license configuration uses custom components or Python external modules, they must be configured before importing the execution license configuration.
- Before importing the settings for the execution license, check whether there is enough free storage space on the operating machine. If the storage capacity is insufficient, acquisition of settings for the execution license will fail.
- To perform user management in an execution license operating environment, you must perform user management for each operating environment. Development license settings are not taken over.

11 i-Trigger (Expansion Custom Package)

11.1 i-Trigger (Expansion Custom Package)

This section explains i-Trigger, an Expansion Custom Package for SpeeDBee Synapse.

The details of the description are as follows:

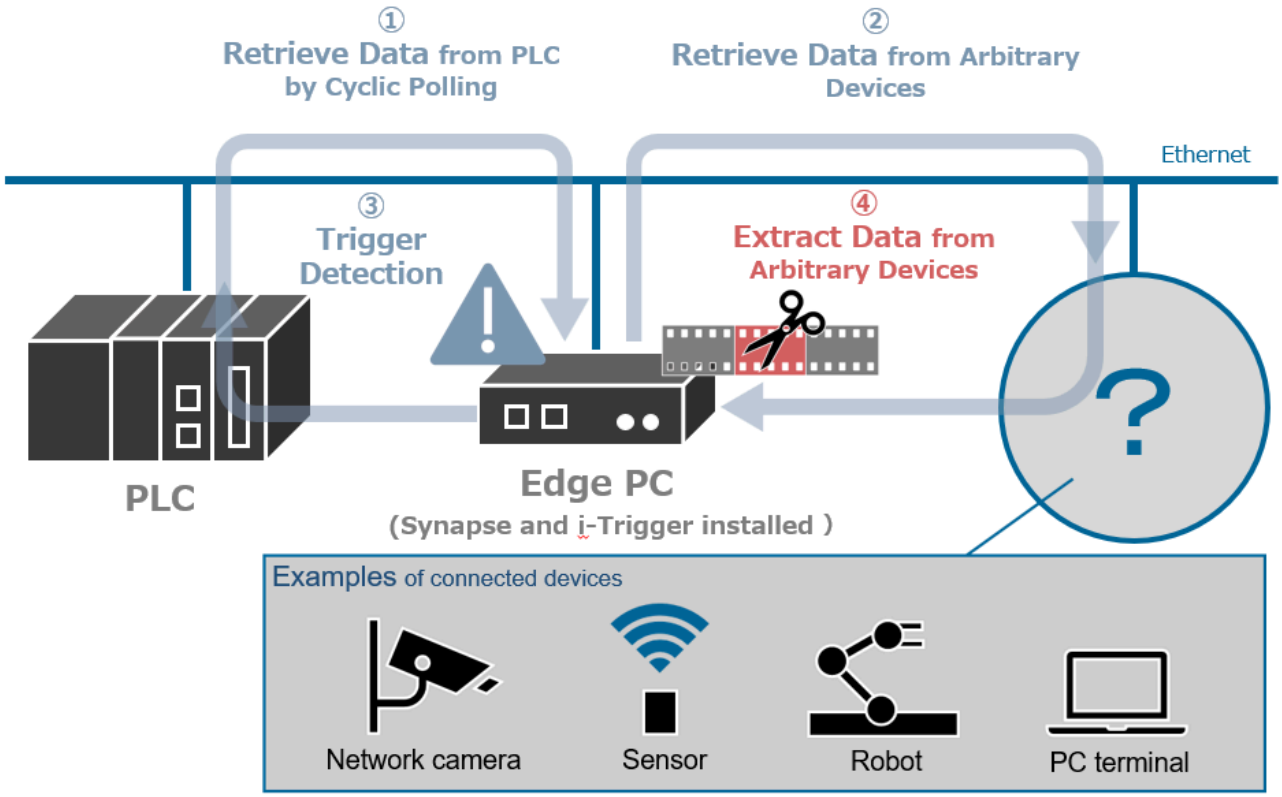
1. i-Trigger Overview
2. Cameras That Can Be Used
3. Installation
4. Video Viewer for Video Playback
5. Components
6. Tutorial
7. Recording Specifications for i-Trigger

11.2 i-Trigger Overview

This section explains an overview of i-Trigger.

11.2.1 i-Trigger Features

i-Trigger is an Expansion Custom Package for SpeeDBee Synapse to monitor a short-time breakdown in manufacturing.
It serves a function similar to a dashboard camera in the facility. This allows you, when an error occurs in the production line, to automatically extract the numerical data before and after the error occurred and the video data captured by the camera based on the data collected by the various types of components, and save them in a specified file format.



Processing Overview

① Collect data from PLC with interval reference

Get the data for trigger detection and recording from the PLC.

② Get data from any device

In addition to the PLC, data collected from each component can also be used for trigger detection in the same way as the PLC.

③ Trigger detection

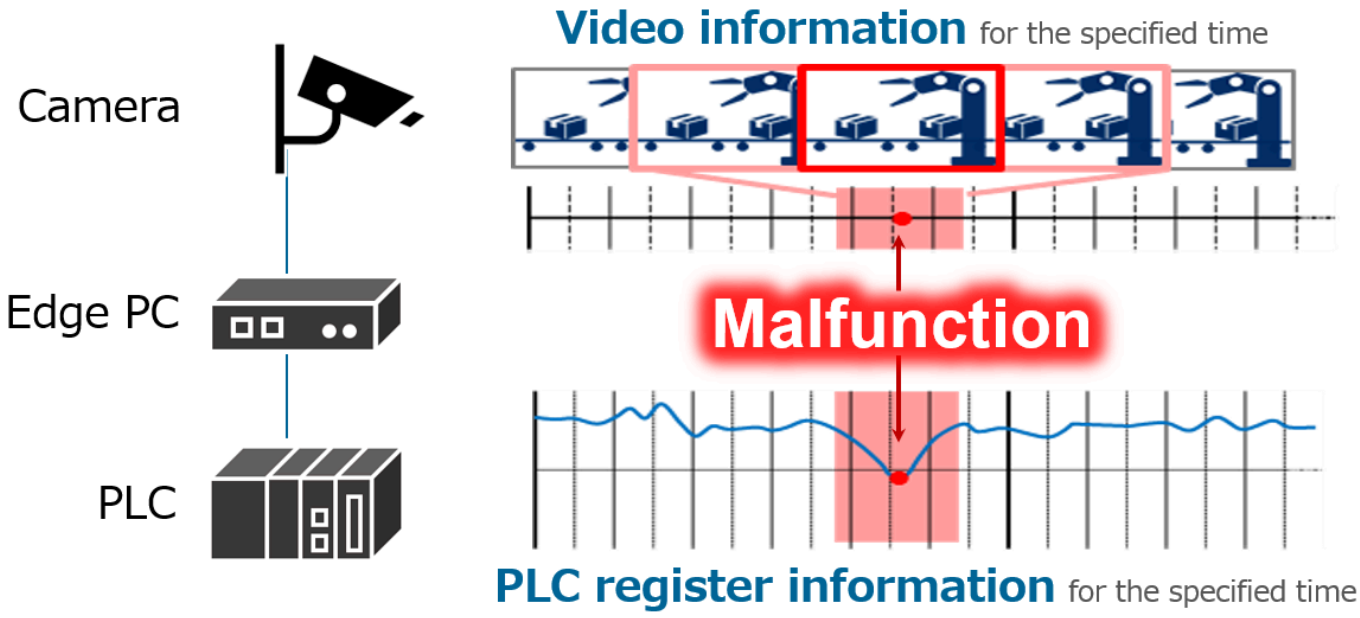
When the collected data meets the preset conditions, the extraction process is executed as a trigger detection.

④ Extract any device data

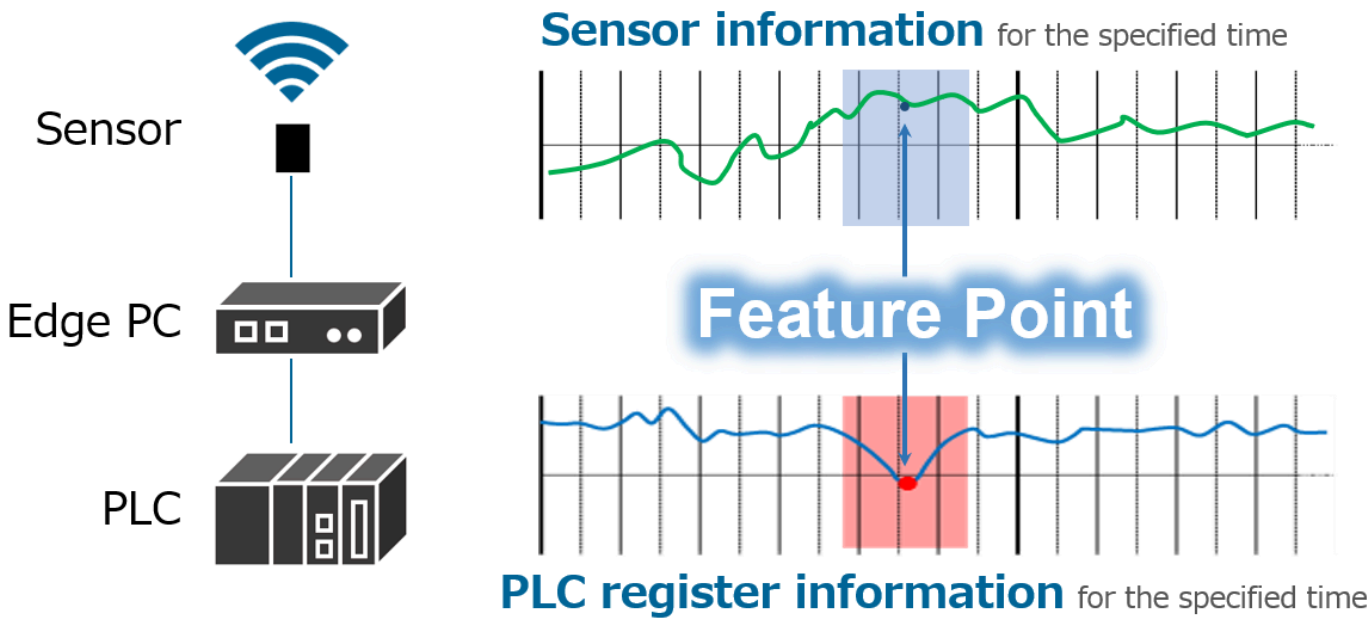
After detecting a trigger, data from the target device (data collected by a network camera or PLC) is extracted according to the preset record range. You can view the extracted data in a dedicated Video Viewer.

Dedicated Video Viewer

i-Trigger is configured with a dedicated Video Viewer for viewing extracted data. As shown in the figure below, it features the ability to simultaneously play back video and collected data in the event of a failure.



The video and the various data collected can be visualized at the same time, which can be useful in analyzing the relationship between data and events.



11.2.2 Points to keep in mind when using i-Trigger

Since i-Trigger is an Expansion Custom Package for SpeeDBee Synapse, you will need to prepare SpeeDBee Synapse itself to use it. Therefore, it operates in the same environment as SpeeDBee Synapse. For more information, see Execution Environment.

11.3 Cameras That Can Be Used

This section explains the cameras that can be used with i-Trigger.

11.3.1 Camera Specifications That Can Be Used/Supported Codecs

The following cameras can be used with i-Trigger.

Type	Supported Video Codecs	Operation Verified Profiles
Network cameras that support Real-Time Streaming Protocol (RTSP)	H.264	• Baseline Profile • Main Profile • High Profile

11.3.2 Number of Cameras in Use

The number of cameras that can be used simultaneously with i-Trigger is limited. Your SpeeDBee Synapse license includes a number of cameras, and you may not use more than that number of cameras to record simultaneously.

Item	Number of Cameras
Standard number of cameras in use	4
Maximum number of cameras in use	30 Note: You will need a license for the number of cameras used.

Please purchase a license that grants you the number of cameras you need.

> In the Data Flow Controller and OMRON-SBS for Windows versions, the number of cameras that can be connected with the initial license is fixed at 30.

11.4 Installation

This section explains the installation and uninstallation procedures.

11.4.1 Installation and Updates

You can install i-Trigger by following the steps below. Please be sure to use the same version of SpeeDBee Synapse and i-Trigger. The same procedure is used for updates.
On Windows, it is installed at the same time as SpeeDBee Synapse is installed. See Installation (Windows).

> In the Data Flow Controller version, i-Trigger is pre-installed by default, so no installation is required.

Placing Files

1. Place the i-Trigger files in the target device to be installed (in the /tmp directory). The following are some examples of placing them with the scp command from Windows Power Shell.

If the device OS is Ubuntu20(amd64)

```
PS >scp speedbeesynapse_ubuntu20_x.x.x-r0_amd64.deb (Account name)@(IP address of the
device):/tmp
PS >scp speedbeesynapse-itrigger_ubuntu20_x.x.x-r0_amd64.deb (Account name)@(IP address of
the device):/tmp
```

Note: If the same version of SpeeDBee Synapse is already installed, you do not need to transfer SpeeDBee Synapse.
Note: Where x.x.x is the version number.

If the device OS is Ubuntu22(amd64)

```
PS >scp speedbeesynapse_ubuntu22_x.x.x-r0_amd64.deb (Account name)@(IP address of the
device):/tmp
PS >scp speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_amd64.deb (Account name)@(IP address of
the device):/tmp
```

Note: If the same version of SpeeDBee Synapse is already installed, you do not need to transfer SpeeDBee Synapse.
Note: Where x.x.x is the version number.

If the device OS is Ubuntu22(arm64)

```
PS >scp speedbeesynapse_ubuntu22_x.x.x-r0_arm64.deb (Account name)@(IP address of the
device):/tmp
PS >scp speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_arm64.deb (Account name)@(IP address of
the device):/tmp
```

Note: If the same version of SpeeDBee Synapse is already installed, you do not need to transfer SpeeDBee Synapse.
Note: Where x.x.x is the version number.

If the device OS is Raspberry Pi OS(bookworm 64-bit)

```
PS >scp speedbeesynapse_raspi_bookworm_x.x.x-r0_arm64.deb (Account name)@(IP address of the
device):/tmp
PS >scp speedbeesynapse-itrigger_raspi_bookworm_x.x.x-r0_arm64.deb (Account name)@(IP address
of the device):/tmp
```

Note: If the same version of SpeeDBee Synapse is already installed, you do not need to transfer SpeeDBee Synapse.
Note: Where x.x.x is the version number.

If the device OS is debian12(bookworm arm64)

```
PS >scp speedbeesynapse_debian12_x.x.x-r0_arm64.deb (Account name)@(IP address of the
device):/tmp
PS >scp speedbeesynapse-itrigger_debian12_x.x.x-r0_arm64.deb (Account name)@(IP address of
the device):/tmp
```

Note: If the same version of SpeeDBee Synapse is already installed, you do not need to transfer SpeeDBee Synapse.
Note: Where x.x.x is the version number.

Installation and Start Up

1. Use ssh to log in to the device. The following is an example of using the ssh command to log in from Windows Power Shell.

   ```
   PS >ssh (Account name)@(IP address of the device)
   ```

2. Run the following command to install i-Trigger.

If the device OS is Ubuntu20(amd64)

- When installing (updating) SpeeDBee Synapse at the same time

```
$ sudo apt install /tmp/speedbeesynapse_x.x.x-r0_amd64.deb /tmp/speedbeesynapse-itrigger_ubuntu20_x.x.x-r0_amd64.deb
```

- When installing (updating) i-Trigger only

```
$ sudo apt install /tmp/speedbeesynapse-itrigger_ubuntu20_x.x.x-r0_amd64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22(amd64)

- When installing (updating) SpeeDBee Synapse at the same time

```
$ sudo apt install /tmp/speedbeesynapse_ubuntu22_x.x.x-r0_amd64.deb /tmp/speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_amd64.deb
```

- When installing (updating) i-Trigger only

```
$ sudo apt install /tmp/speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_amd64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Ubuntu22(arm64)

- When installing (updating) SpeeDBee Synapse at the same time

```
$ sudo apt install /tmp/speedbeesynapse_ubuntu22_x.x.x-r0_arm64.deb /tmp/speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_arm64.deb
```

- When installing (updating) i-Trigger only

```
$ sudo apt install /tmp/speedbeesynapse-itrigger_ubuntu22_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

If the device OS is Raspberry Pi OS(bookworm 64-bit)

- When installing (updating) SpeeDBee Synapse at the same time

```
$ sudo apt install /tmp/speedbeesynapse_raspi_bookworm_x.x.x-r0_arm64.deb /tmp/speedbeesynapse-itrigger_raspi_bookworm_x.x.x-r0_arm64.deb
```

- When installing (updating) i-Trigger only

```
$ sudo apt install /tmp/speedbeesynapse-itrigger_raspi_bookworm_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

If the device OS is debian12(bookworm arm64)

- When installing (updating) SpeeDBee Synapse at the same time

```
$ sudo apt install /tmp/speedbeesynapse_debian12_x.x.x-r0_arm64.deb /tmp/speedbeesynapse-itrigger_debian12_x.x.x-r0_arm64.deb
```

- When installing (updating) i-Trigger only

```
$ sudo apt install /tmp/speedbeesynapse-itrigger_debian12_x.x.x-r0_arm64.deb
```

Note: Where x.x.x is the version number.

The above command will restart the service in the SpeeDBee Synapse itself.

11.4.2 Uninstallation

You can uninstall i-Trigger from the device by following the steps below. For Windows, see Uninstallation (Windows).

1. Use ssh to log in to the device. The following is an example of using the ssh command to log in from Windows Power Shell.

```
PS >ssh (Account name)@(IP address of the device)
```

2. Run the following command to uninstall i-Trigger.

```
$ sudo apt remove speedbeesynapse-itrigger
```

or

```
$ sudo apt purge speedbeesynapse-itrigger
```

There is no difference in the behavior of the uninstallation process when either of the above commands is used.

The above command will stop the service in the SpeeDBee Synapse itself. If you want to start the service again, please refer to SpeeDBee Synapse Control to start the service.

Cautions on uninstallation

If you start the service in SpeeDBee Synapse while i-Trigger is uninstalled, any remaining configurations related to i-Trigger, such as VIDEO Collector, will be deleted.

If you want to install i-Trigger again, follow the same steps as described in Installation.

11.5 Video Viewer for Video Playback

This section explains the Video Viewer for playing back recorded video.

11.5.1 Video Viewer Access

You can open the Video Viewer for video playback in the following ways:

1. URL direct typing

   You can access the following URL from your browser to open the Video Viewer for video playback.

   - `http://{Host name or IP address of the device}:8120/video-viewer`

     Example) `http://192.168.1.3:8120/video-viewer`

2. Transition from SpeeDBee Synapse screen

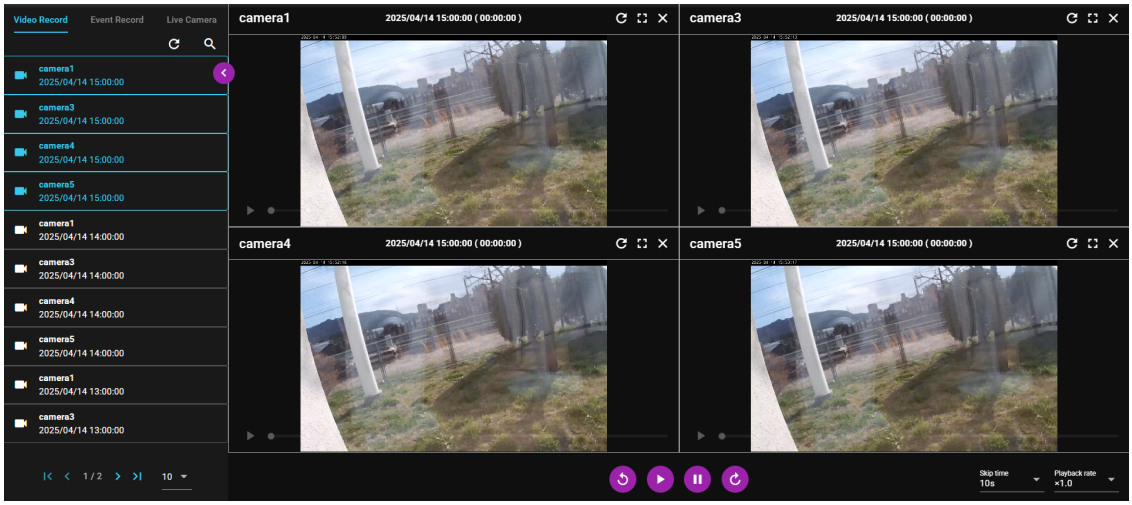
   You can open the Video Viewer for video playback by pressing the transition icon to the Video Viewer for video playback in the header of SpeeDBee Synapse. This icon is displayed only when i-Trigger is enabled.



When the screen is opened, the Video Record Play screen is displayed first. By selecting a tab from the menu on the left, you can switch between the screens you are viewing.

11.5.2 Video Record Play

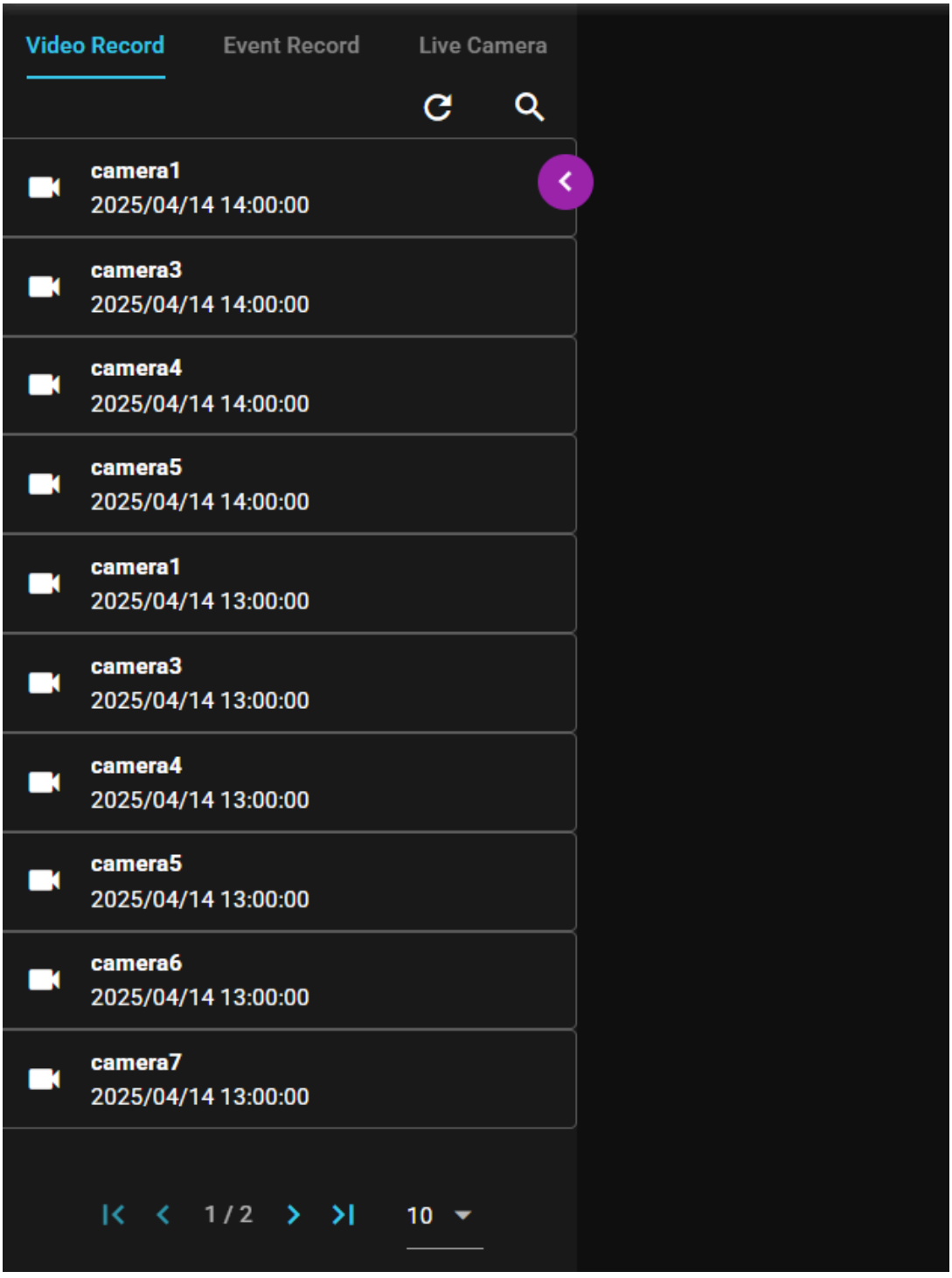
A Video Record is a video that is constantly being recorded while the VIDEO Collector is running. You can play back a Video Record.



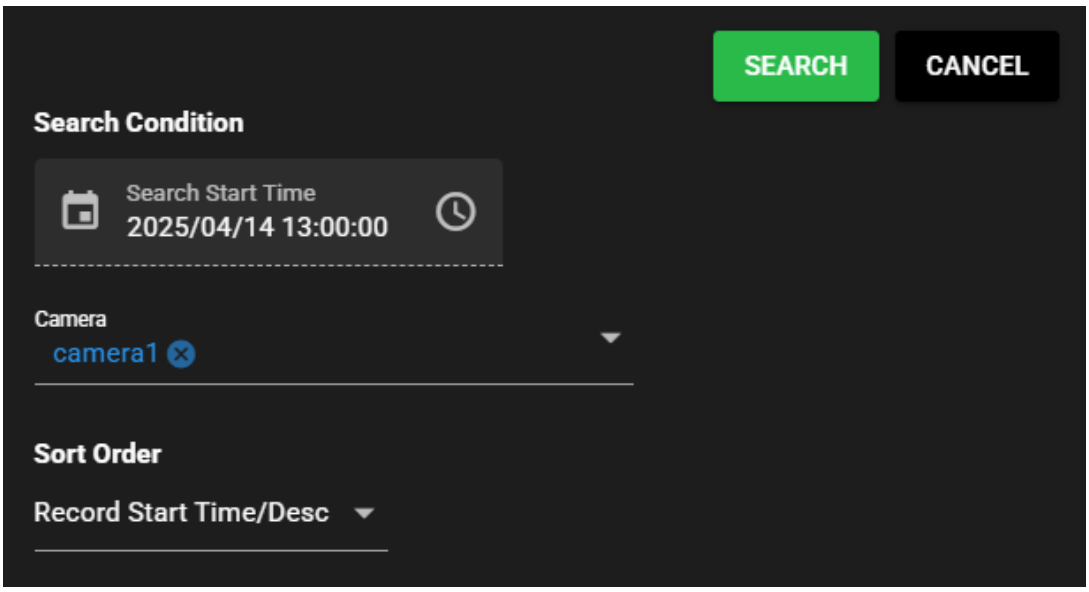
11.5.2.1 List

Video Records are listed on the left side of the screen. By default, all video recordings prior to the date and time you open the screen will be displayed. You can also perform a search by following the steps below.

1. Press the magnifying glass icon.

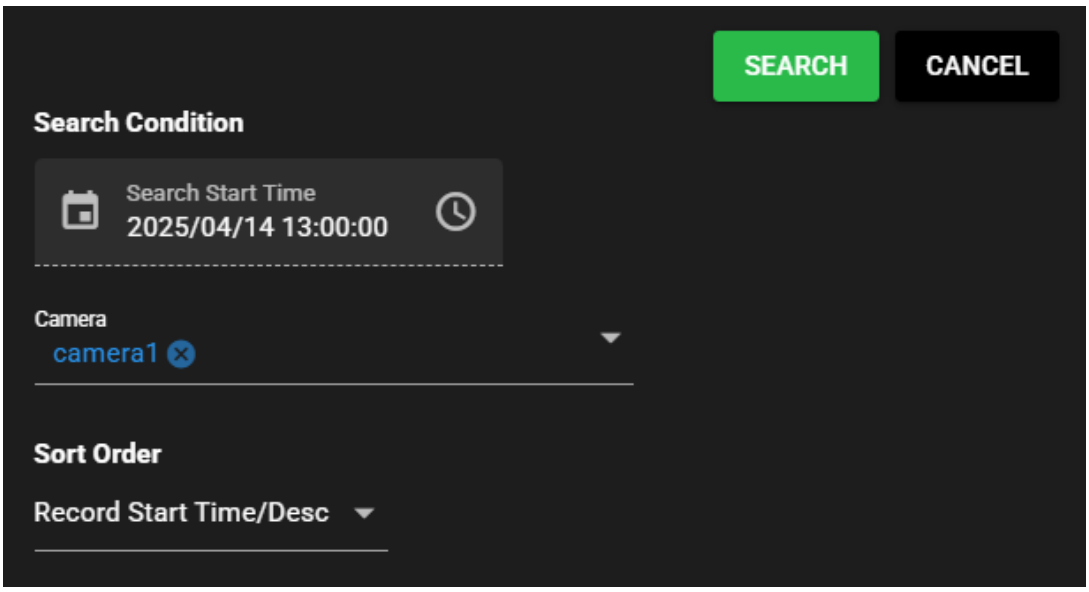


2. Input the following items.

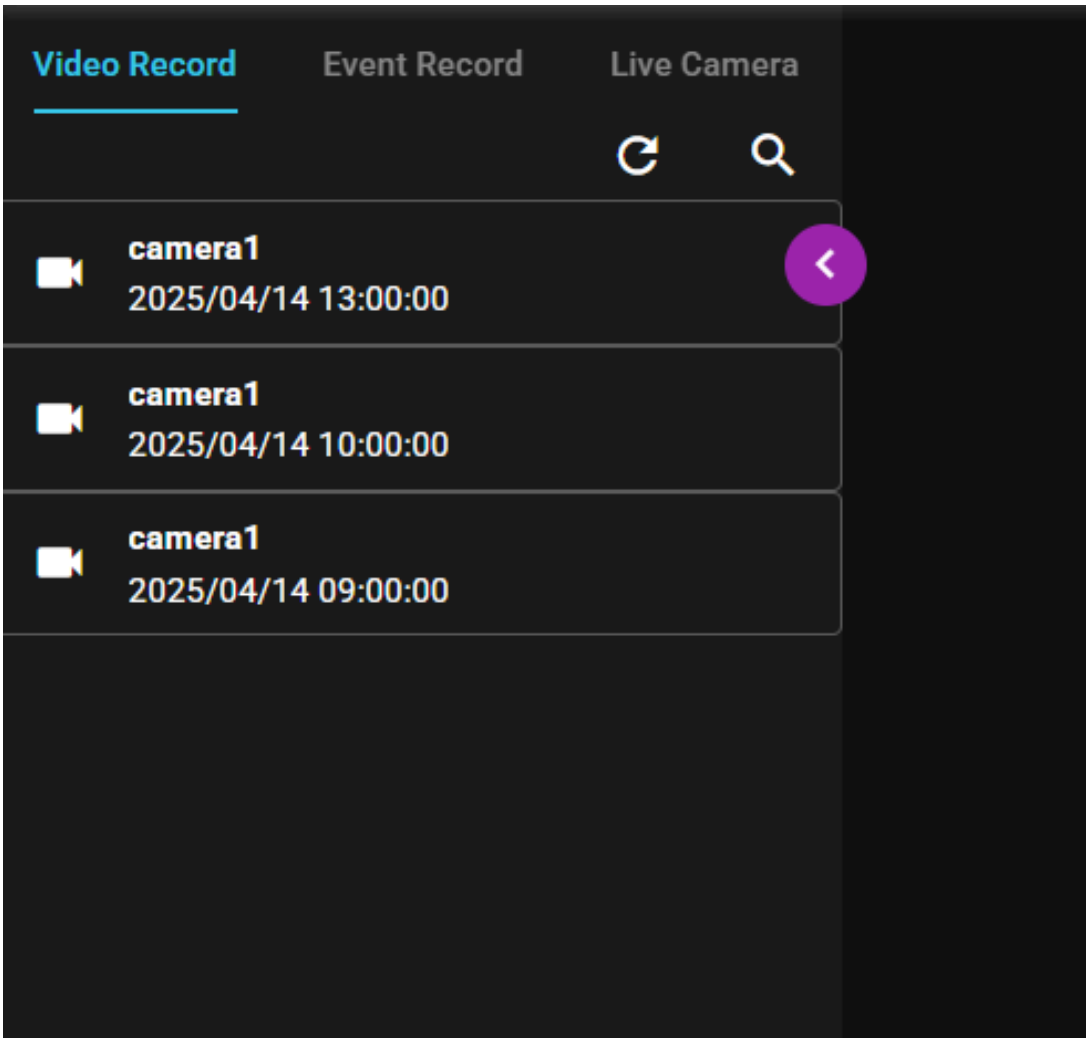


Item	Description
Search Start Time	The time range to search for. Note: You can search for video recordings up to the time you input.
Camera	Name of the VIDEO Collector to search for
Sort Order	Combination of sort key and order Note 1: The sort key is selected from the following. - Record Start Time - Camera Name Note 2: The sort order is selected from the following. - Asc - Desc

3. Press Search.



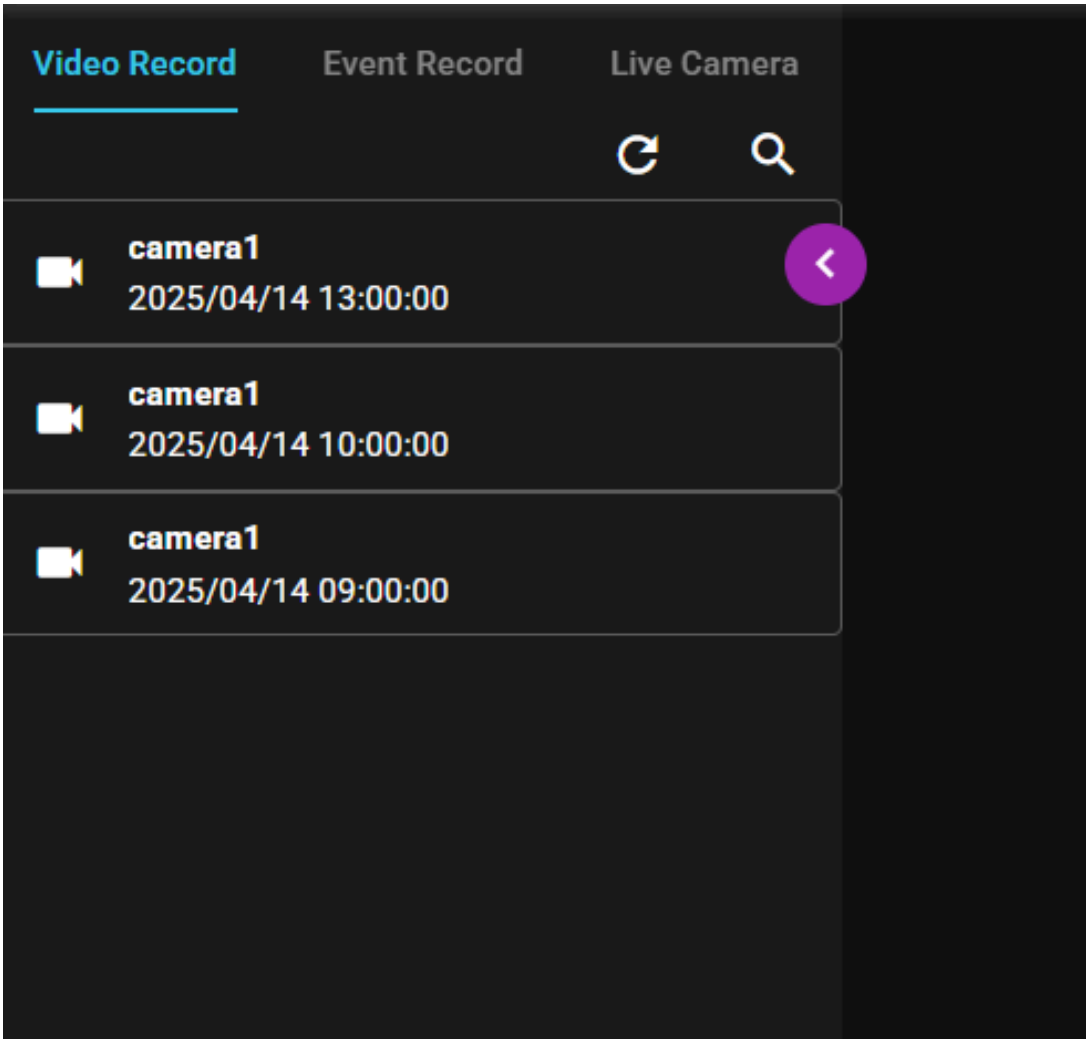
4. The results of your search will be displayed.



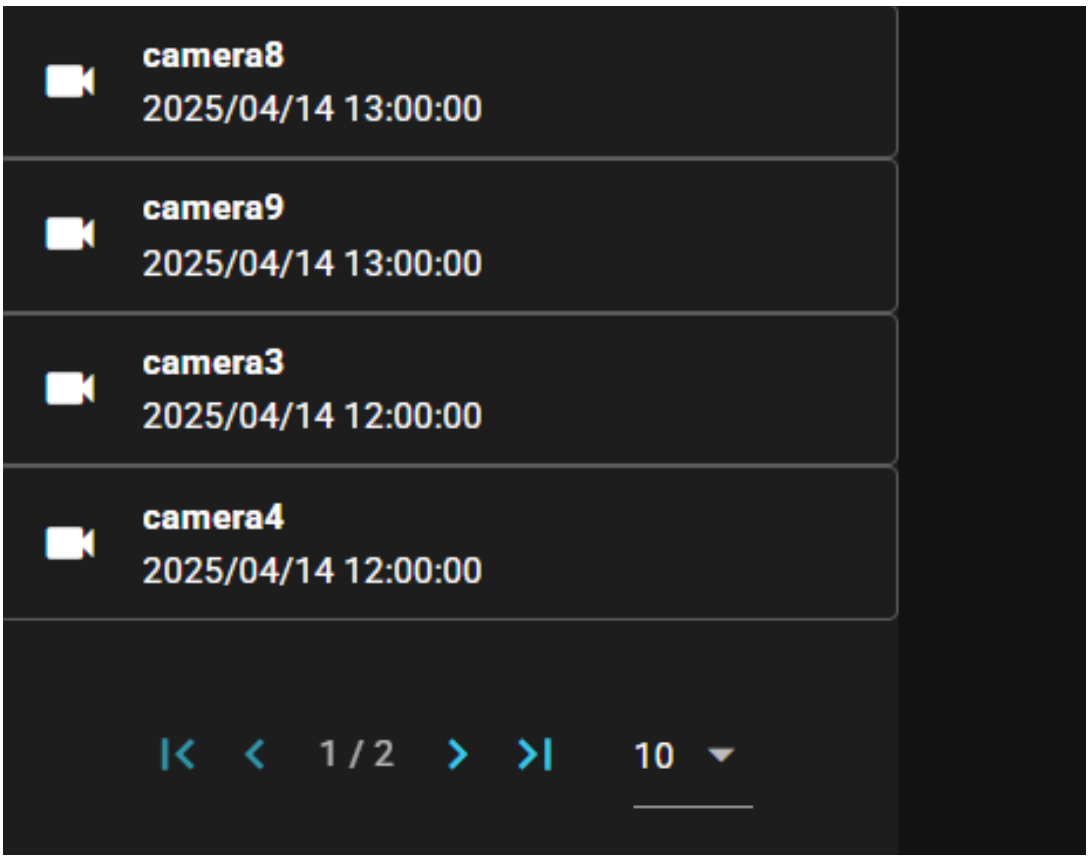
The items displayed are as follows:

- VIDEO Collector name
- Record Start Time

5. Press the refresh icon to search again with the same criteria. In addition, refresh is performed automatically at an interval of once every 30 seconds.



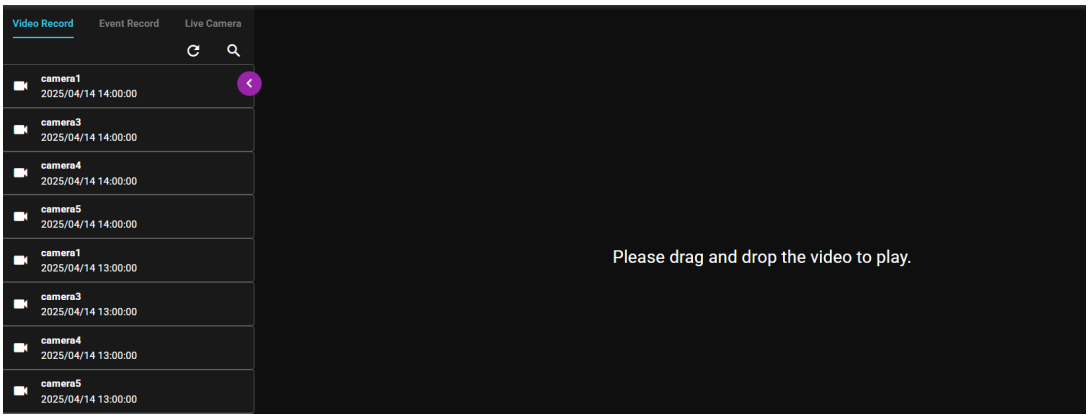
6. If there are many items to be displayed, they will be displayed on multiple pages. You can select the number of items to be displayed per page from 10, 100, or 1000.



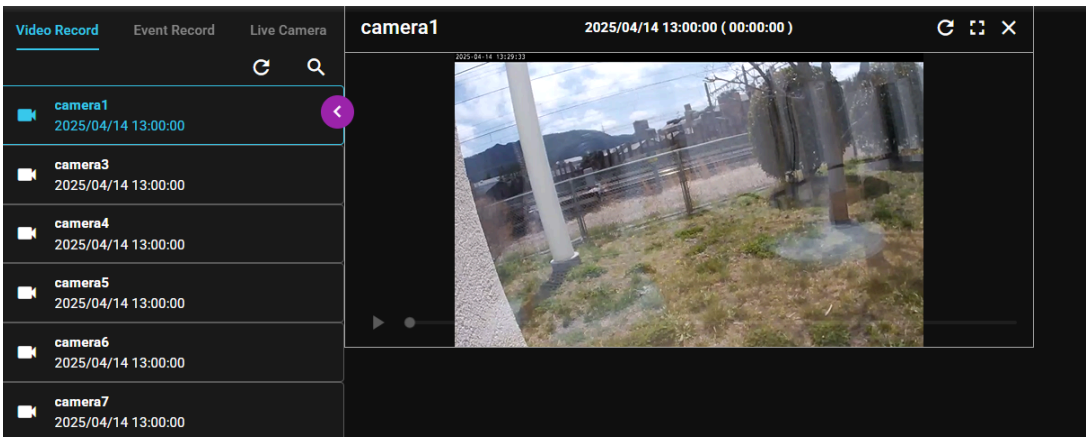
11.5.2.2 Play

You can play back a Video Record image by following the steps below. It can also simultaneously play back 10 recorded videos.

1. Drag a Video Record image in the list and drop it on the right side.



2. The Video Record image will be displayed. After the video loading is complete, you can play the video by pressing the Play icon.



The screen items are as follows:



Number	Description
①	Displays the name of the VIDEO Collector.
②	Displays the recording start date and time.
③	Reloads the video.
④	Press the icon to display the video in full screen.
⑤	Press the icon to hide the video.
⑥	Displays the Video Record image.
⑦	When pressed, toggles between video playback and stop.

Number	Description
⑧	Displays the current playback position. Also, by dragging and dropping the current position icon, you can change the playback position.

3. You can control multiple videos simultaneously by operating the bottom area on the right side of the screen. The operation items are as follows:



Item	Description
①	Press this button to move the playback position back by the specified skip time.
②	Press this button to play all videos displayed.
③	Press this button to stop all videos displayed.
④	Press this button to advance the playback position by the specified skip time.
⑤	You can select the number of seconds for skip forward/skip back from the following: - 10 seconds - 30 seconds - 60 seconds

Item	Description
⑥	You can select the video playback rate from the following: - 0.5x speed - 0.75x speed - 1x speed - 1.25x speed - 1.5x speed - 2x speed

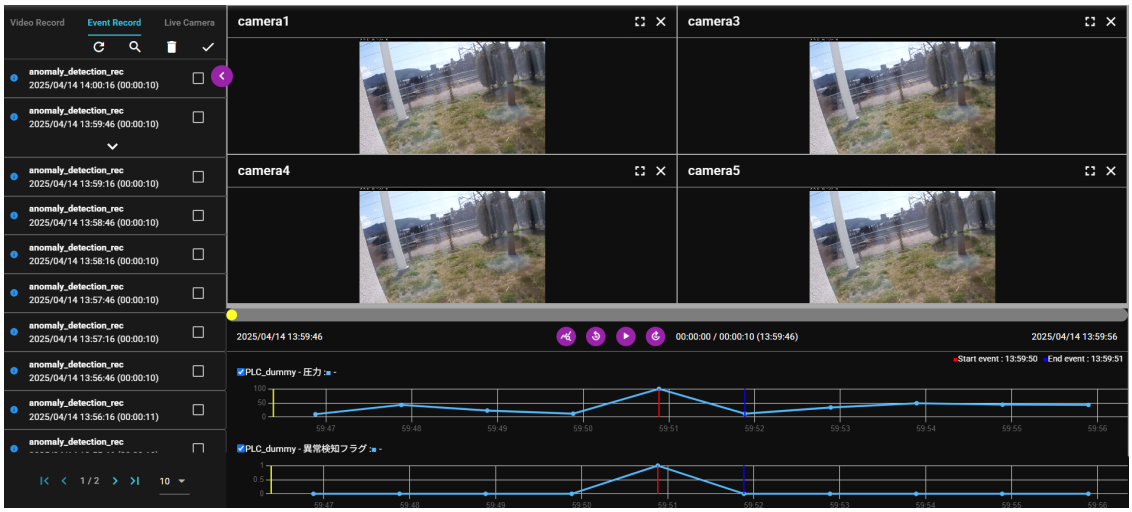
Notes on Video Record Play

In Video Record Play, the videos to be played back are displayed in one-hour units. With the ability to play multiple videos simultaneously, you can use the following playback ways:

- Playback of videos from the same time period by different cameras (Example: simultaneous playback of videos from the same time period by cameras at different angles)
- Playback of videos from different time periods by the same camera (Example: videos in normal and error conditions can be viewed and compared together.)

11.5.3 Event Record Play

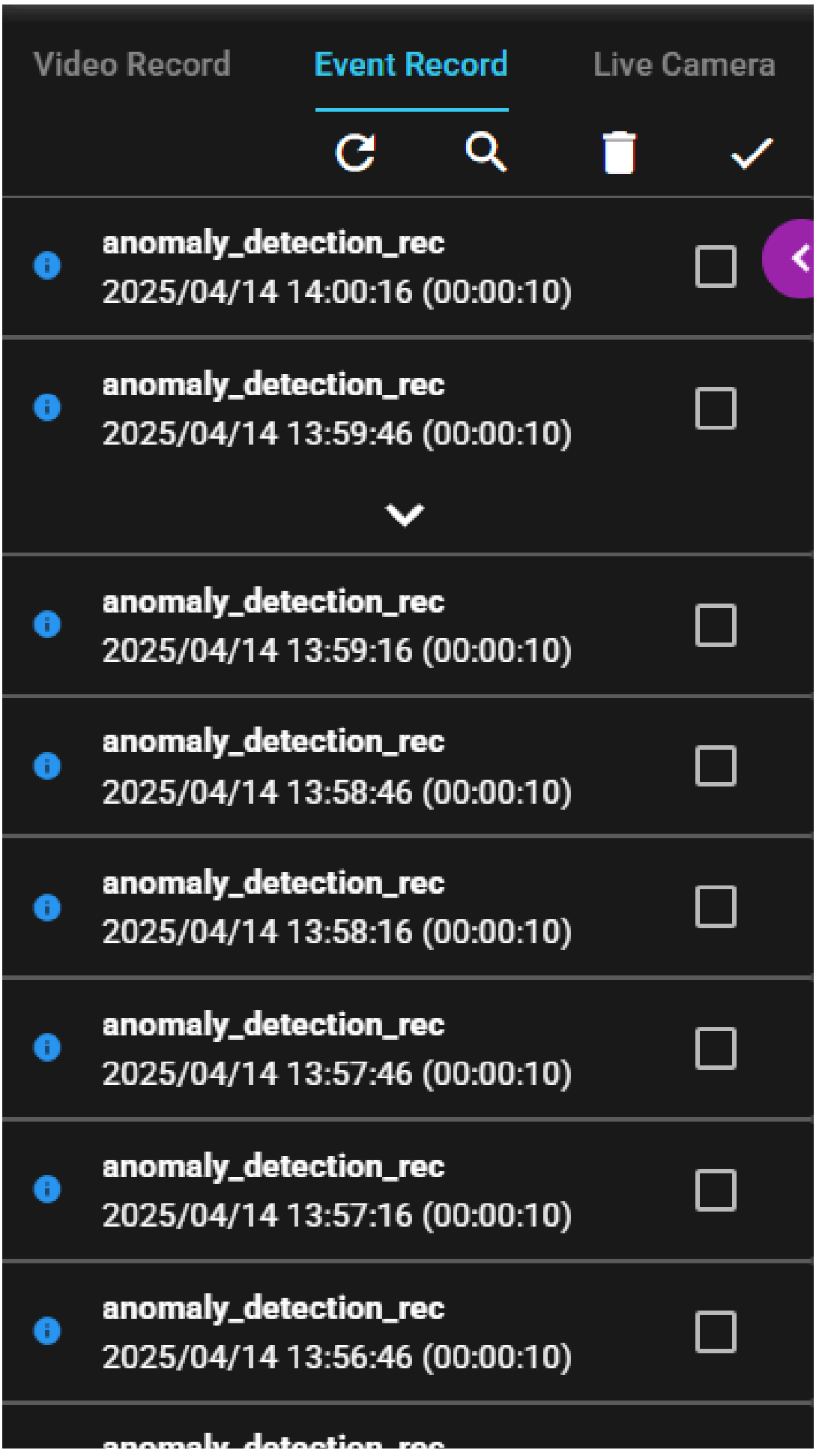
You can confirm the events generated by the Event Recorder in SpeeDBee Synapse. Video and data from the camera at the time of the event can be viewed.



11.5.3.1 List

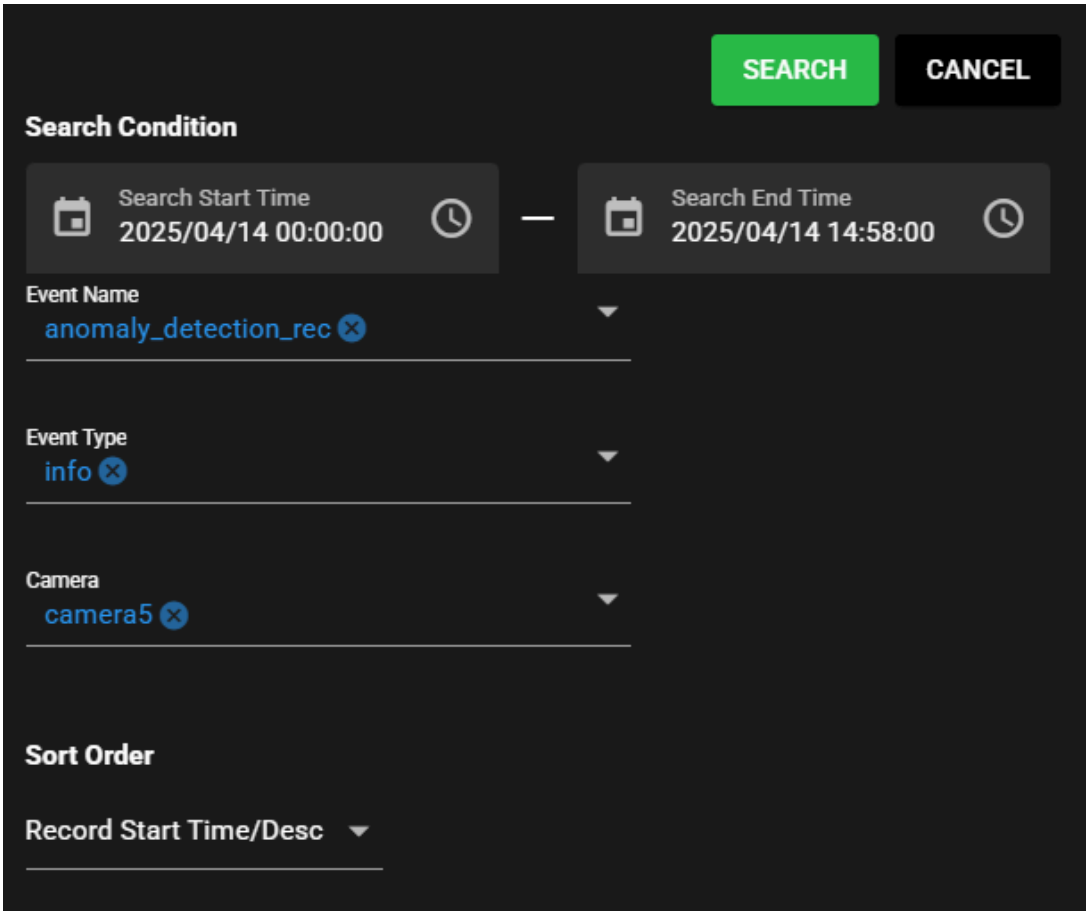
Events are listed on the left side of the screen. By default, all events that occurred between 00:00:00 and 24:00:00 on the day you open the screen will be displayed. You can also search for events by following the steps below.

1. Press the magnifying glass icon.


Video Record
Event Record
Live Camera
anomaly_detection_rec
2025/04/14 14:00:16 (00:00:10)
anomaly_detection_rec
2025/04/14 13:59:46 (00:00:10)
anomaly_detection_rec
2025/04/14 13:59:16 (00:00:10)
anomaly_detection_rec
2025/04/14 13:58:46 (00:00:10)
anomaly_detection_rec
2025/04/14 13:58:16 (00:00:10)
anomaly_detection_rec
2025/04/14 13:57:46 (00:00:10)
anomaly_detection_rec
2025/04/14 13:57:16 (00:00:10)
anomaly_detection_rec
2025/04/14 13:56:46 (00:00:10)



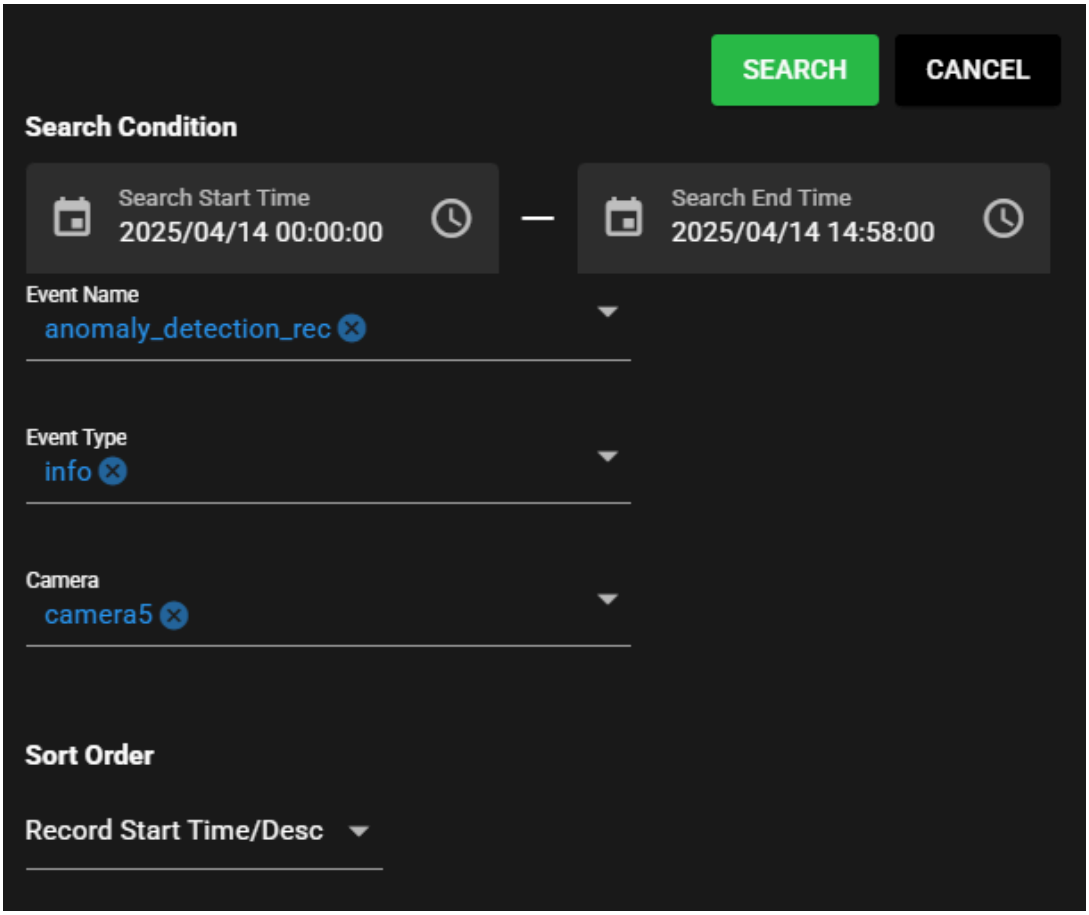
2. Input the following items.



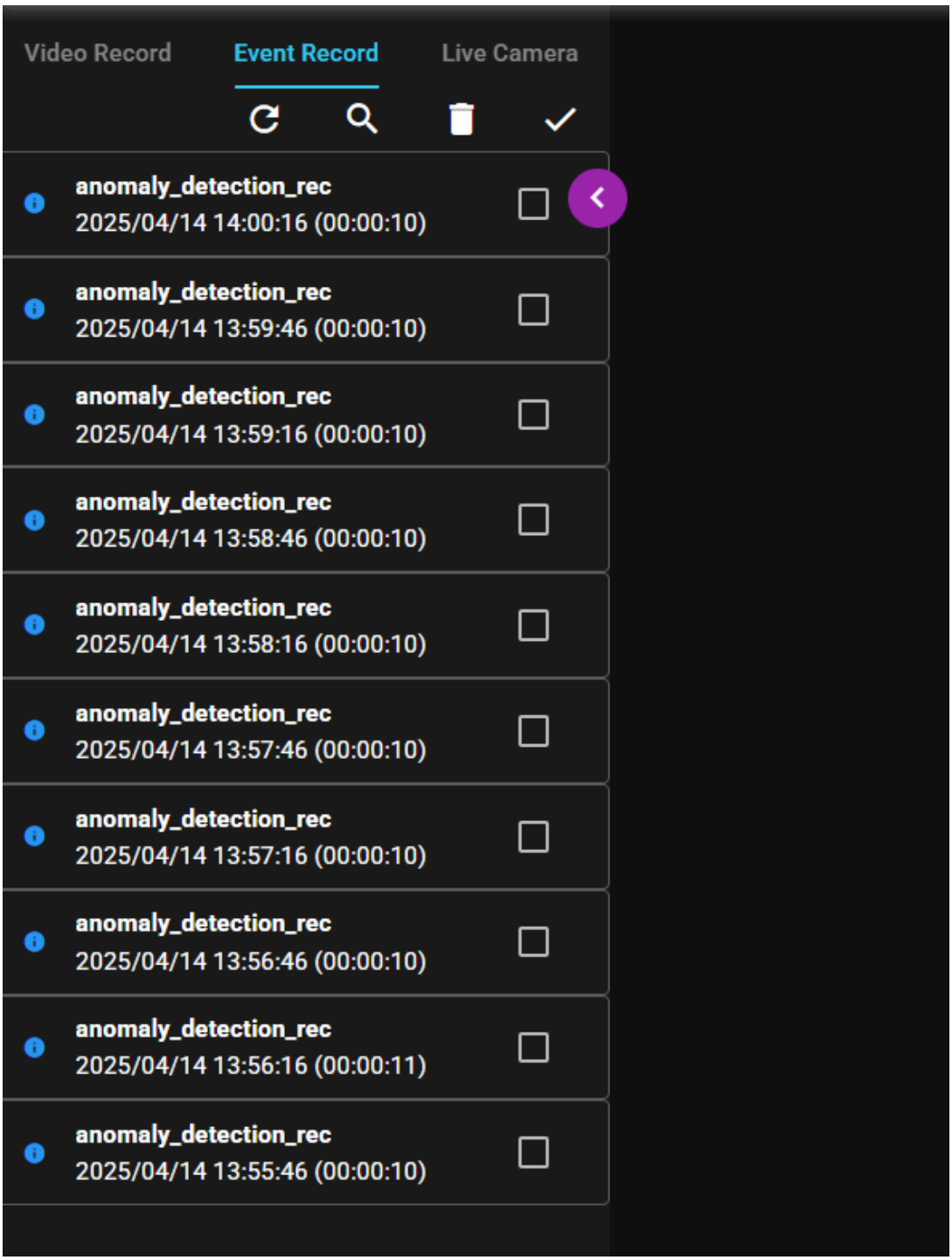
Item	Description
Search Start Time/Search End Time	The time range to search for. Note: The events for which a start event occurred during the time range you input will be searched.
Event Name	Name of the event to search for

Item	Description
Event Type	Event Kind to search for
Camera	Name of the camera associated with the event you are searching for
Sort Order	Combination of sort key and order Note 1: The sort key is selected from the following. - Record Start Time - Event Name - Event Kind Note 2: The sort order is selected from the following. - Asc - Desc

3. Press Search.



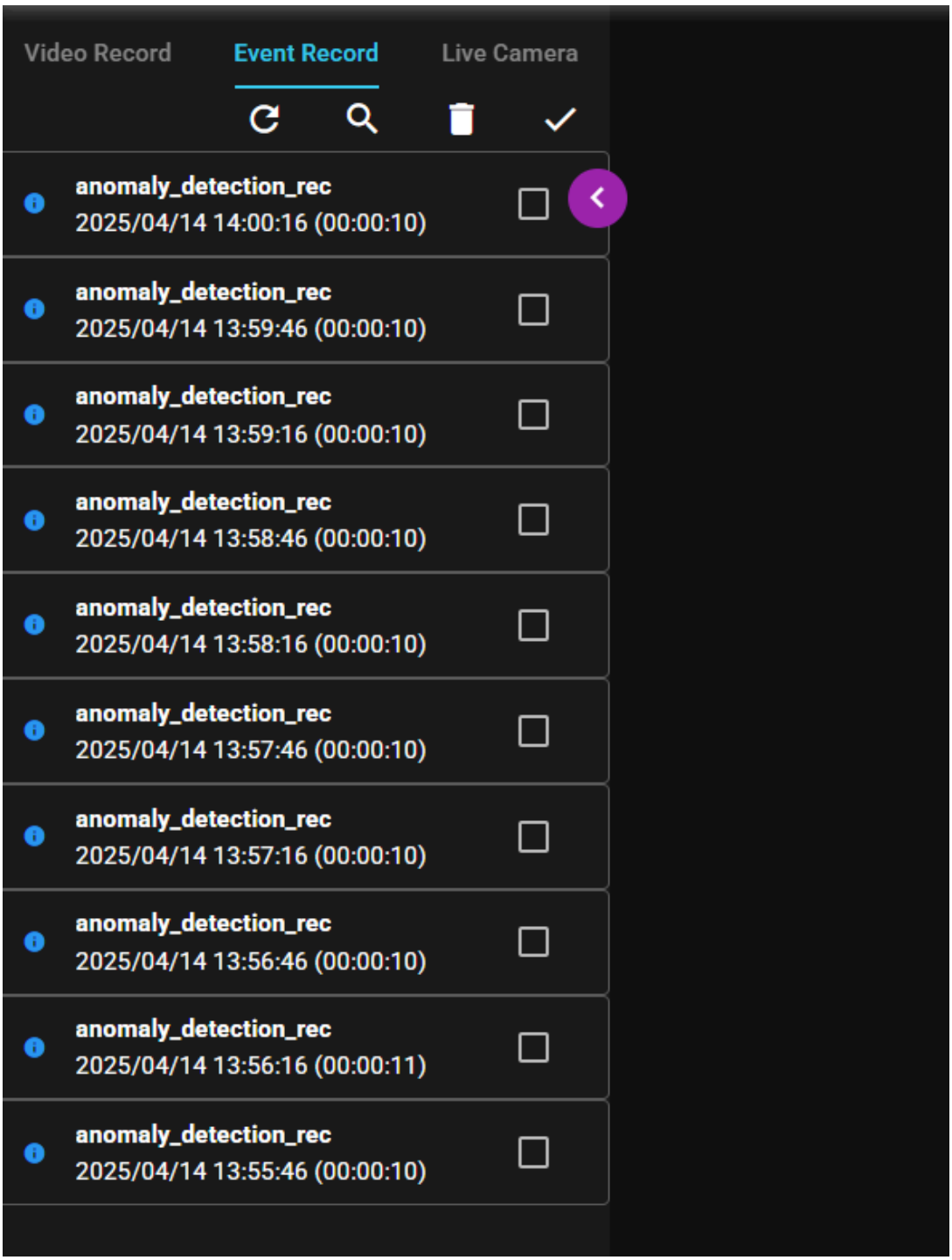
4. The results of your search will be displayed.



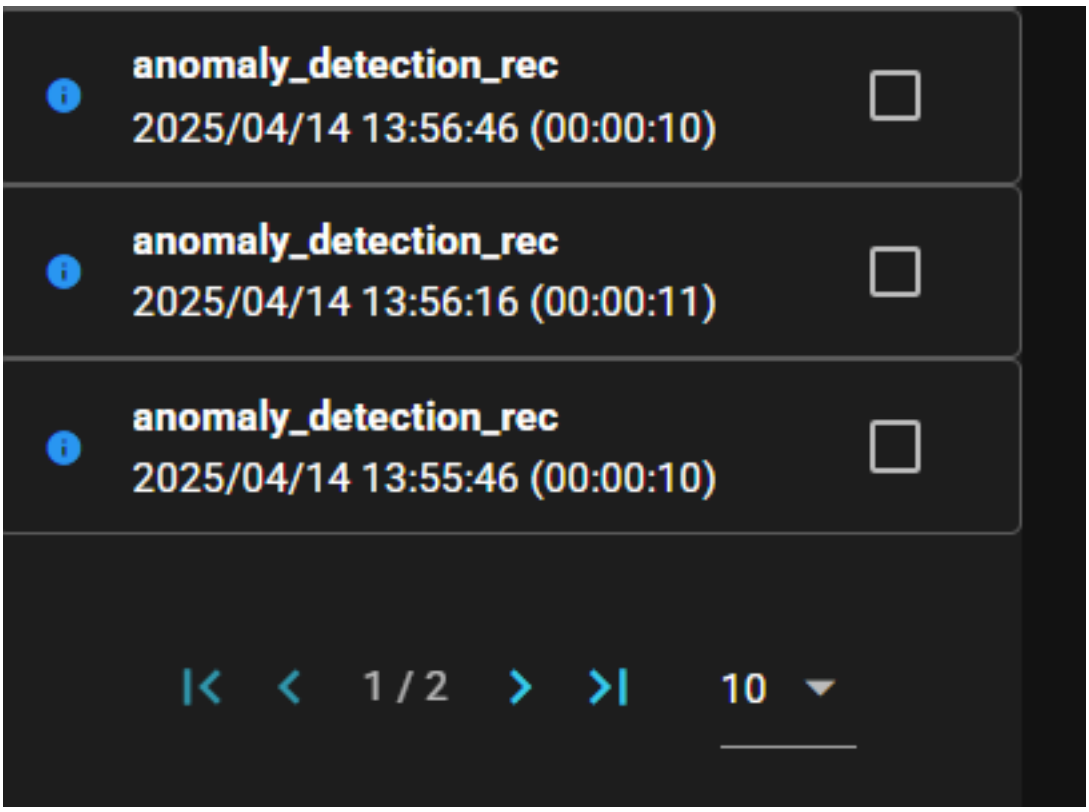
The items displayed are as follows:

- Icon indicating the event type
- Event Name
- Date and time of start event occurrence
- Record time of the event

5. Press the refresh icon to search again with the same criteria. In addition, refresh is performed automatically at an interval of once every 30 seconds.



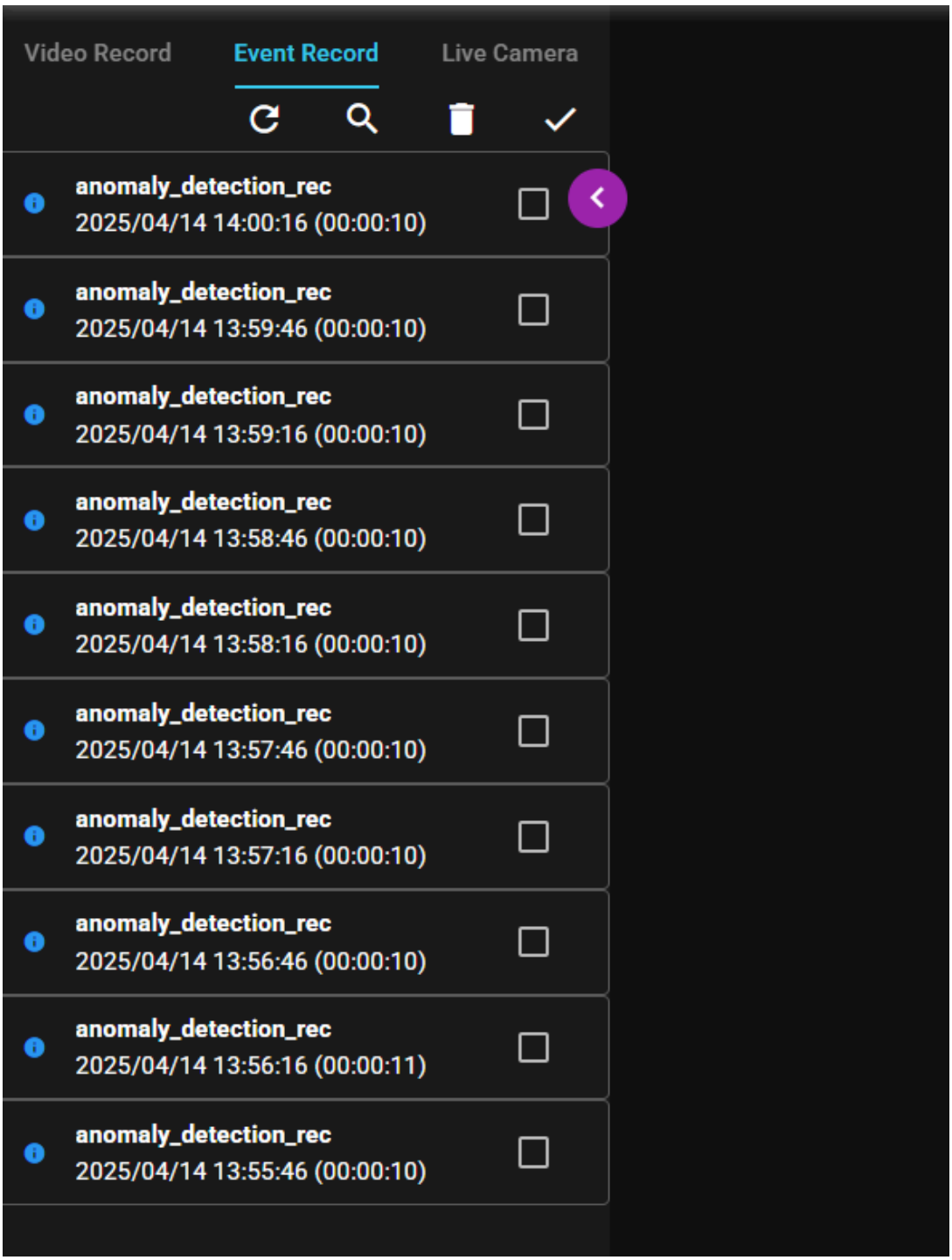
6. If there are many items to be displayed, they will be displayed on multiple pages. You can select the number of items to be displayed per page from 10, 100, or 1000.



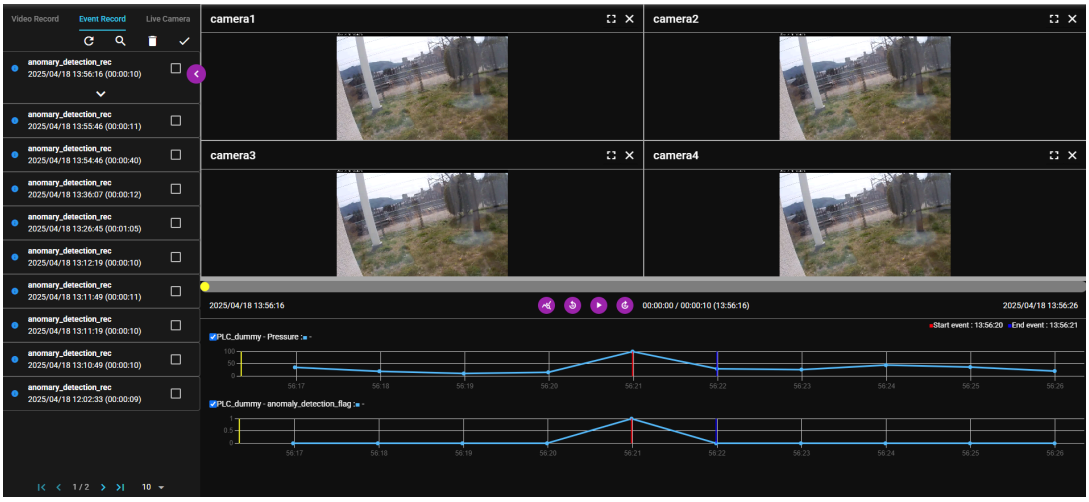
11.5.3.2 Play

You can play back a graph of the data in the component and the video from the camera at the time of the event by following the steps below. Cameras can be played simultaneously, up to 4.

1. Select an event.



2. Video and data from your selected event will be displayed on the right side of the screen.



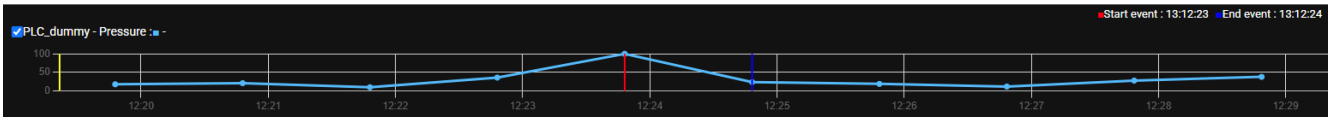
3. When you select the Play button or video, the video and graph associated with the event will play. The screen items are as follows:
   - Video section



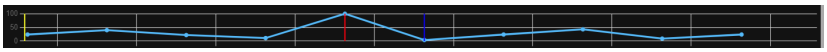
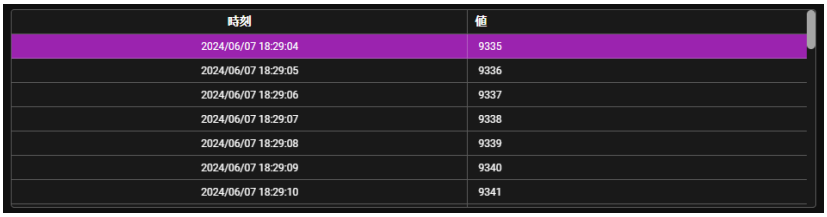
Number	Description
①	Displays the name of the VIDEO Collector.
②	Press the icon to display the video in full screen.
③	Press the icon to hide the video.
④	Displays the video from the camera registered in the event at the time of the event. When pressed, toggles between event playback and stop.
⑤	Displays the current playback position of the video. Also, by dragging and dropping a point at the current position, you can change the playback position. You can also adjust the viewing area of the video by dragging the area between this section and the video and moving it up and down.

Number	Description
⑥	Displays the Record Start Time of the event.
⑦	Press this button to toggle the graph display status from one of the following. - Overall Graph ... Displays all data in the graph. - Detail Graph ... Displays data at the current playback position at about 30 seconds.
⑧	Press this button to skip the playback position before 10 seconds.
⑨	Press this button to start playing the video and graph.
⑩	Press this button to skip the playback position after 10 seconds.
⑪	Displays the event record time and the elapsed playback time.
⑫	Displays the Record End Time of the event.

- Graph section



Item	Description
①	Displays graphs with check marks and hides graphs without check marks.
②	Displays the name of the component from which the data is emitted and the data name.
③	Displays the line color of the data and the value of the current playback position.
④	Displays the time and line color of the start and end events.

Item	Description
⑤ (for numerical data)	Displays the data of the component registered in the event at the time of the event in a time series. The X-axis indicates time and the Y-axis indicates value. The following lines will also be displayed: - Yellow line ... Moves in conjunction with the playback of the video and indicates the current playback position. - Red line ... Indicates the time of the start event. - Blue line ... Indicates the time of the end event. It is displayed only on the end event occurrence. 
⑤ (for character string data)	Displays the data of the component registered in the event at the time of the event in a table consisting of two columns of date/time and value. The following will also be displayed: - Purple row ... Moves in conjunction with the playback of the video and indicates the current playback position. - Red mark... Displayed next to the time and indicates the start event. - Blue mark... Displayed next to the time and indicates the end event. It is displayed only on the end event occurrence. 

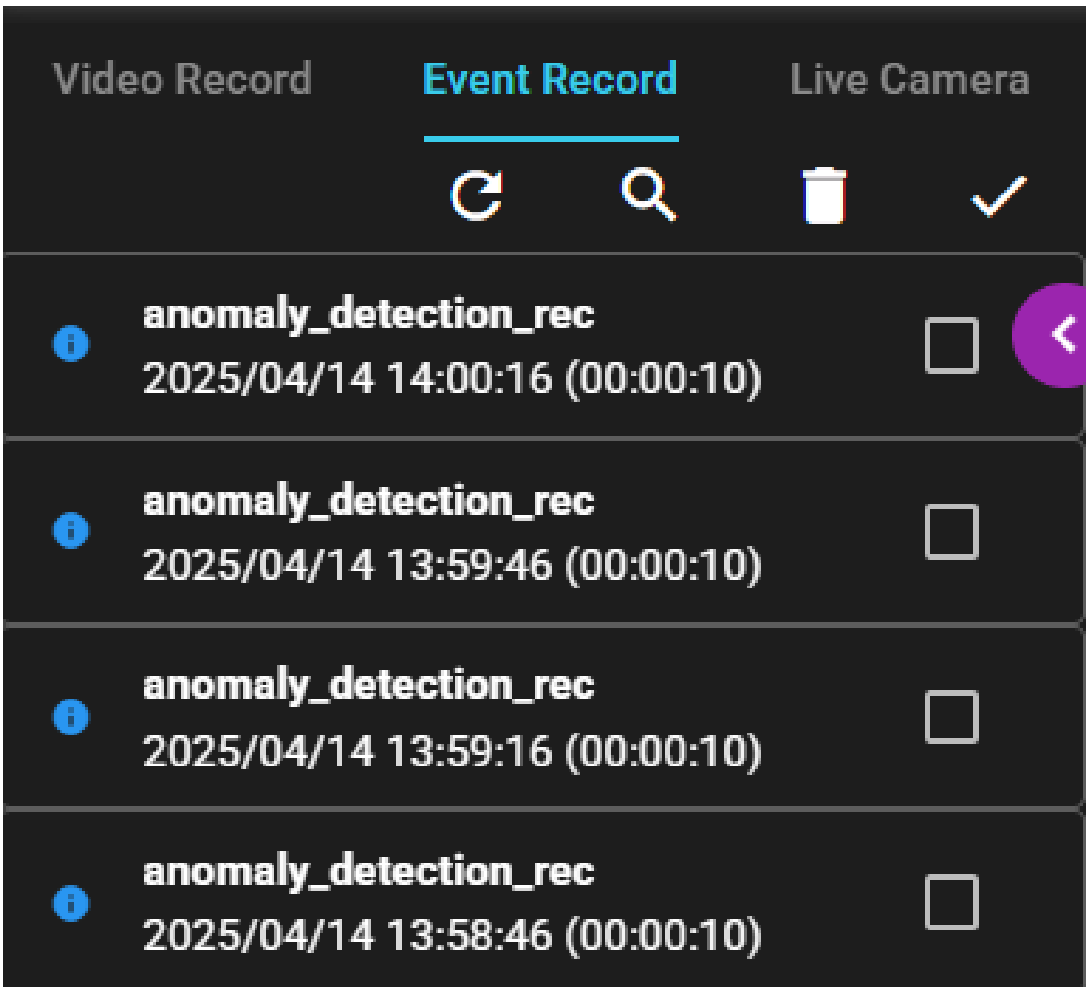
4. You can choose to show or hide the video by toggling the toggle switch next to the Camera Name, which is displayed on the selected event in the list.



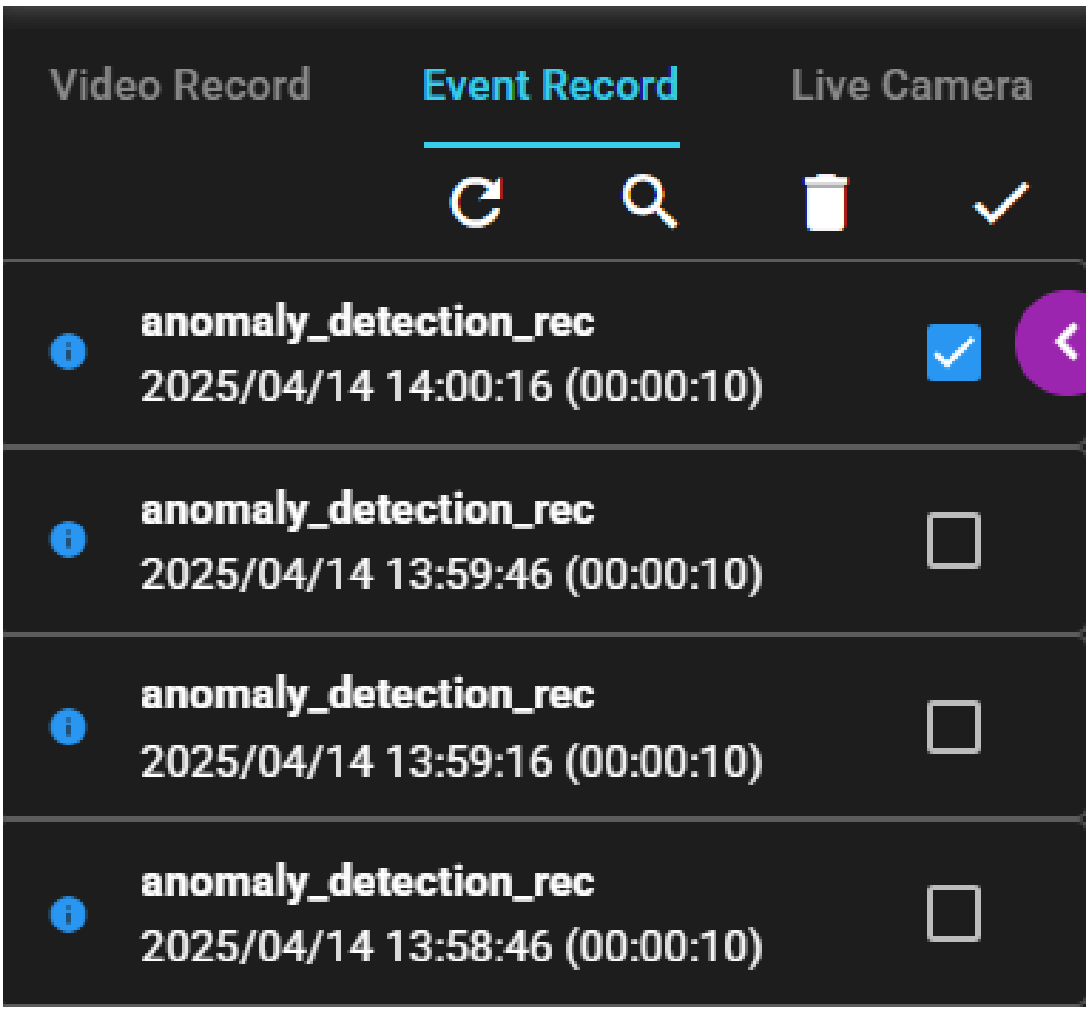
11.5.3.3 Delete

You can delete an event by following the steps below.

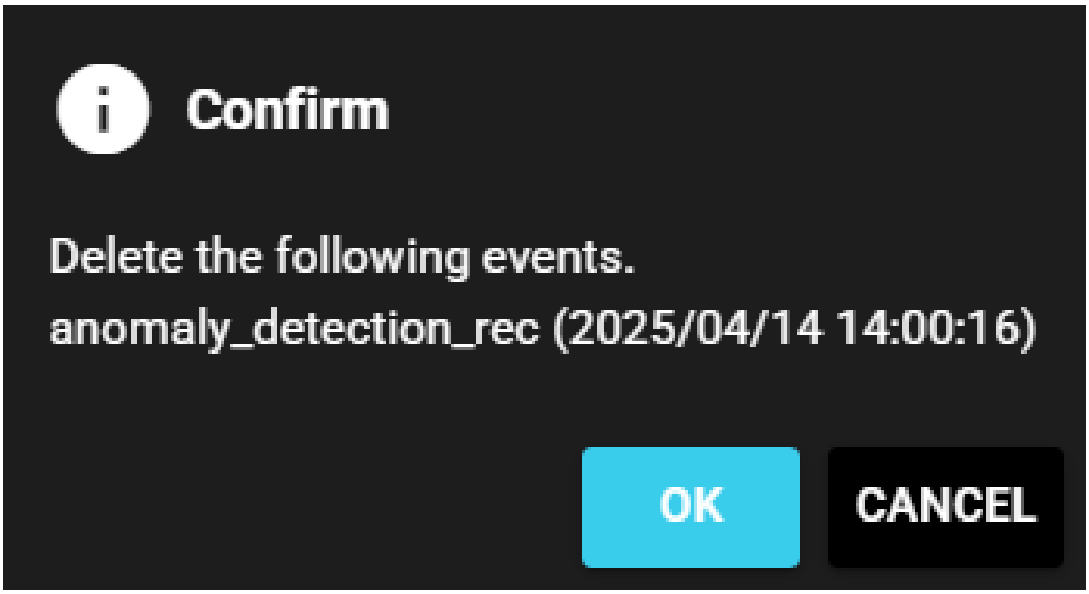
1. Place a check mark in the checkbox for the event. By pressing the check icon in the header, you can place a check mark in all events displayed in the list.



2. Press the Delete icon in the header.



3. Select OK.



4. The event with a check mark will be deleted.

11.5.3.4 Event Display by Specifying URL

By specifying the following URL, which includes the Record Start Time (YYYY/MM/DD hh:mm:ss), you can directly access the events generated by the Event Recorder in SpeeDBee Synapse. If multiple events occur at the same time, an ID is appended to the last of the URL.

- `http://{Host name or IP address of the device}:8120/video-viewer/event_record/{YYYY-MM-DDThhmmss}`

  Example 1)

  - `http://192.168.1.3:8120/video-viewer/event_record/2025-03-28T165209`

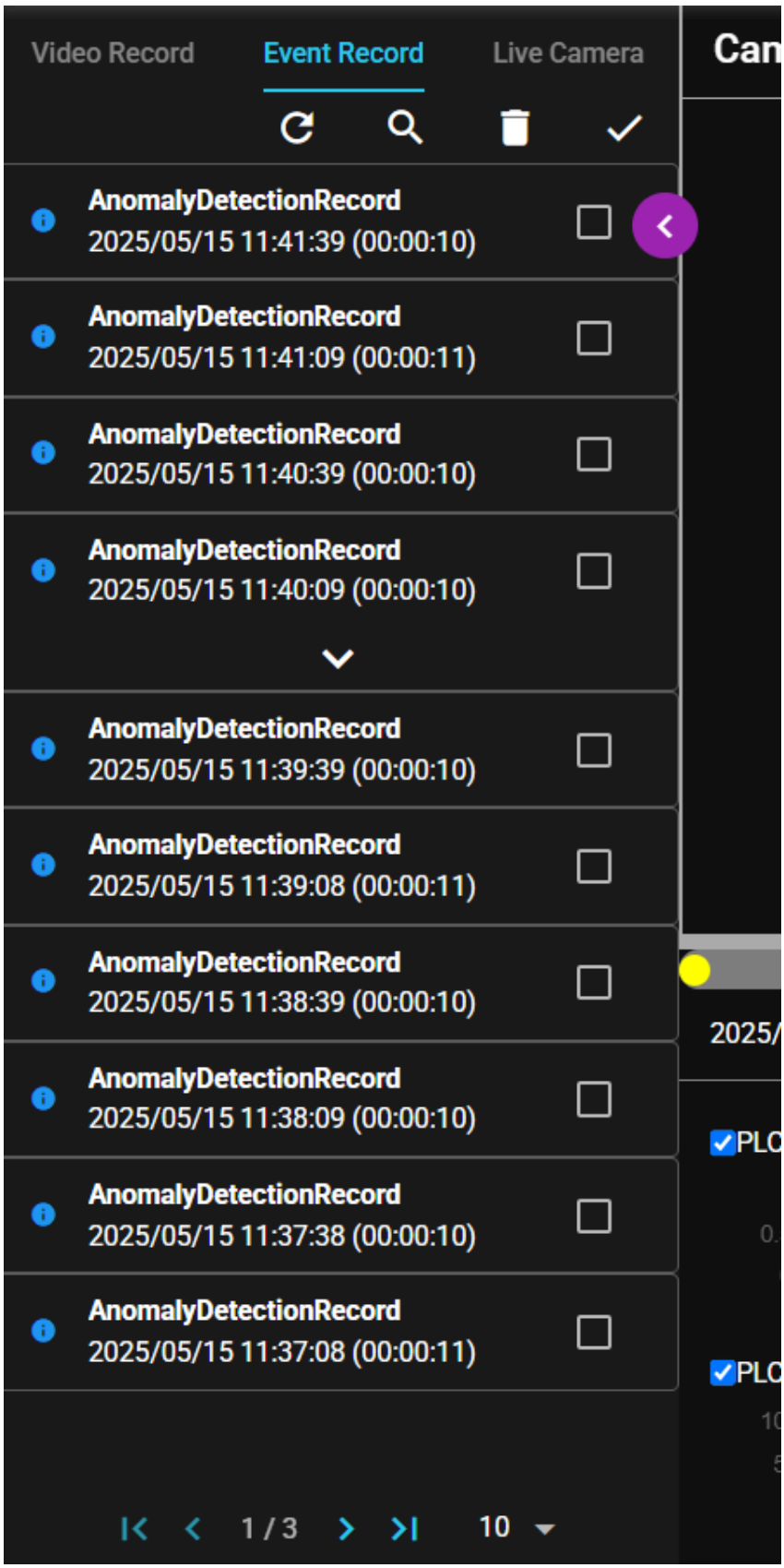
  Example 2) Note: When events with overlapping times occur

  - `http://192.168.1.3:8120/video-viewer/event_record/2025-03-28T165210/1`
  - `http://192.168.1.3:8120/video-viewer/event_record/2025-03-28T165210/2`

The URL is generated by converting the format of the Record Start Time of the event to "YYYY-MM-DDThhmmss".



Press each event to automatically update the URL.

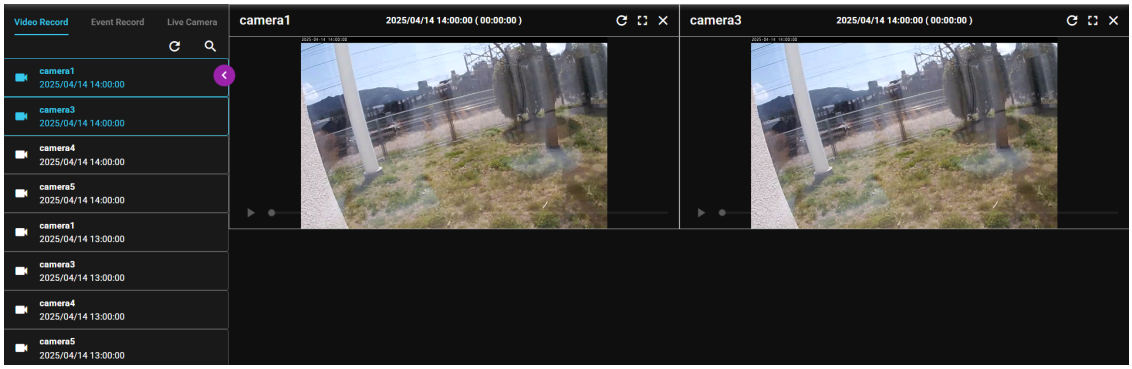


When you access an event by specifying its URL, events that occurred within 2 minutes before and after the record start time of the target event will be listed. When you access the event list, the maximum number of events displayed per page will be 100, and the target event will be selected and displayed.



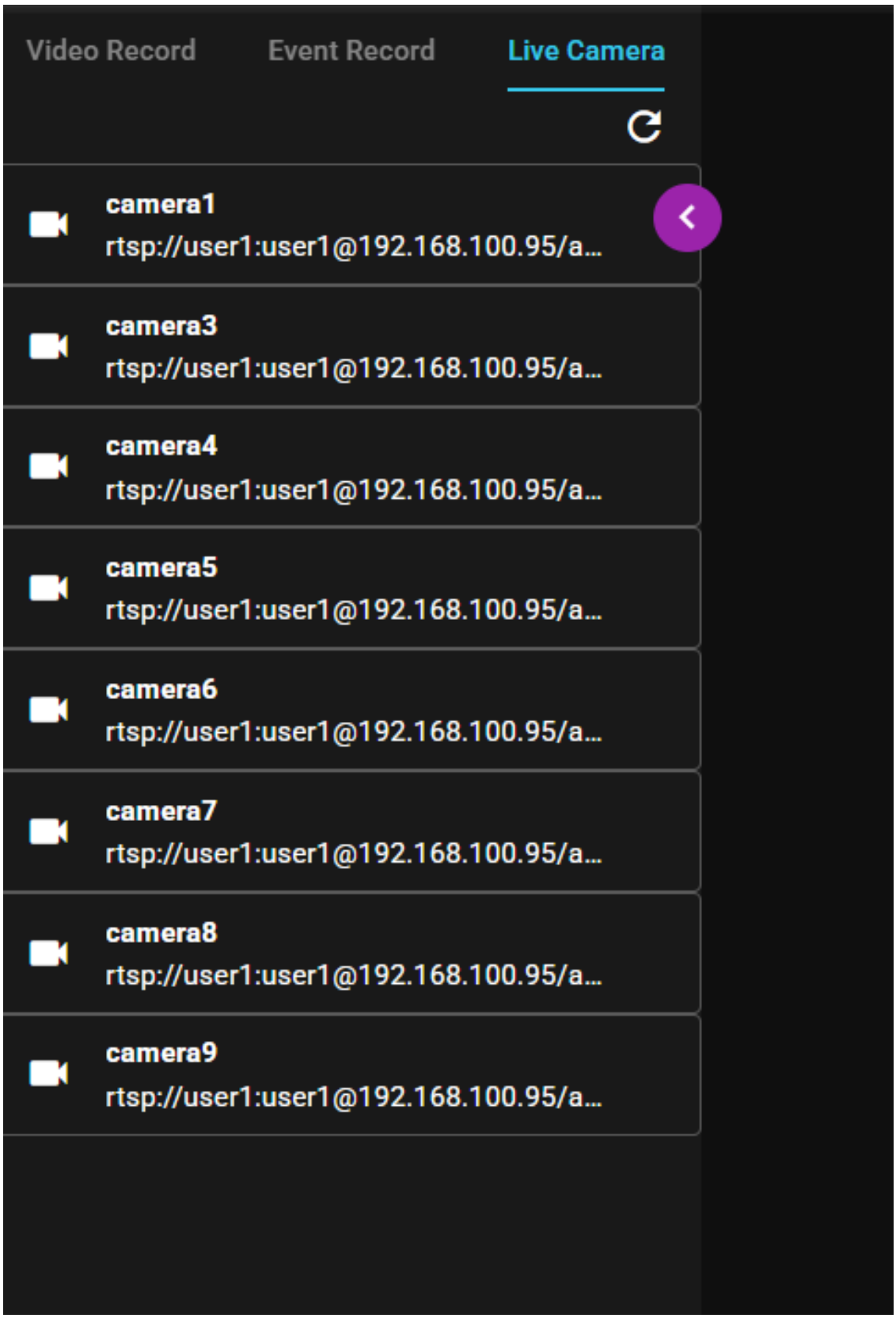
11.5.4 Live Camera

You can play back the video (current camera video) of the VIDEO Collector running in SpeeDBee Synapse.



11.5.4.1 List

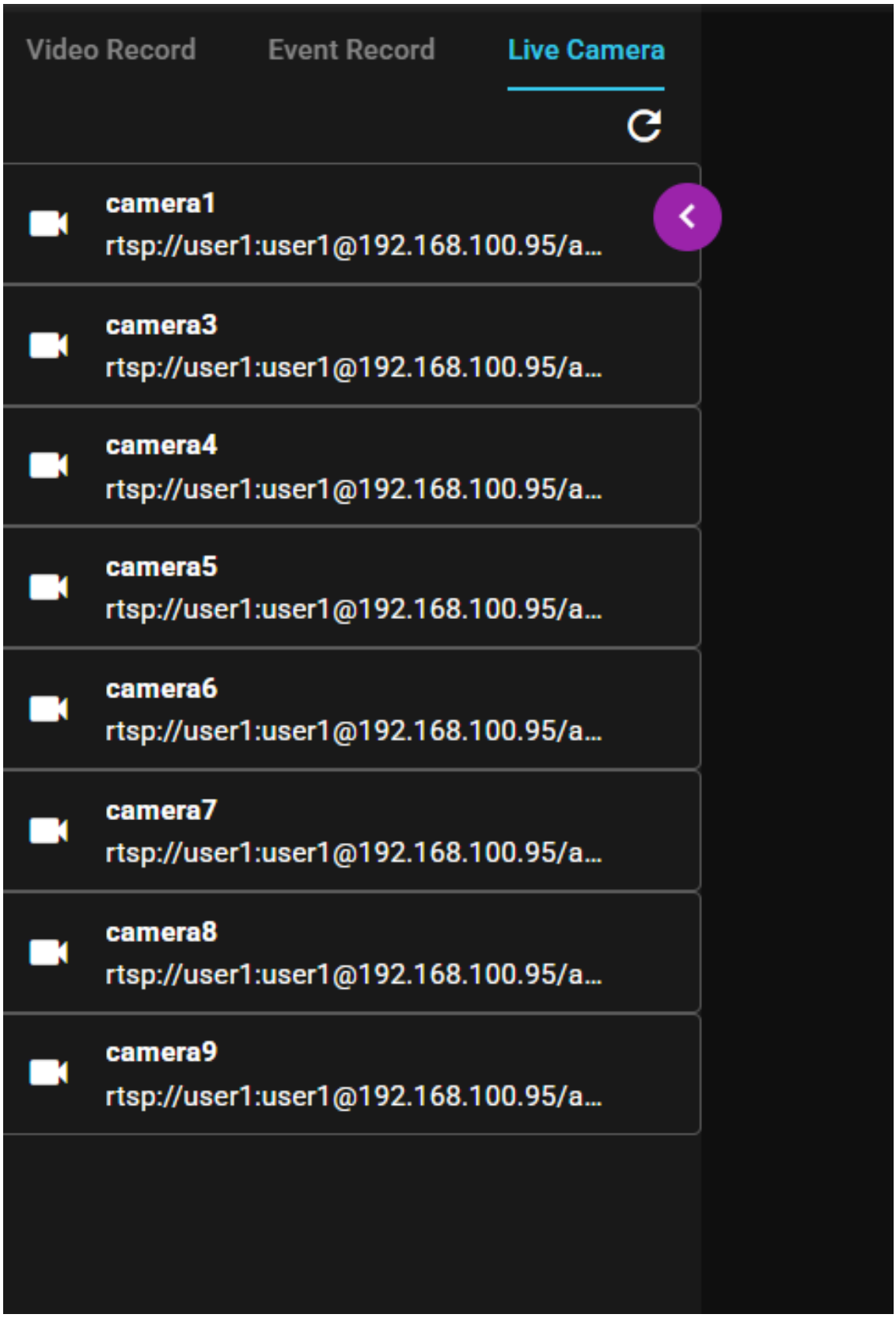
VIDEO Collectors are listed on the left side of the screen.



The items displayed are as follows:

- VIDEO Collector name
- The RTSP URL of the camera registered in the VIDEO Collector

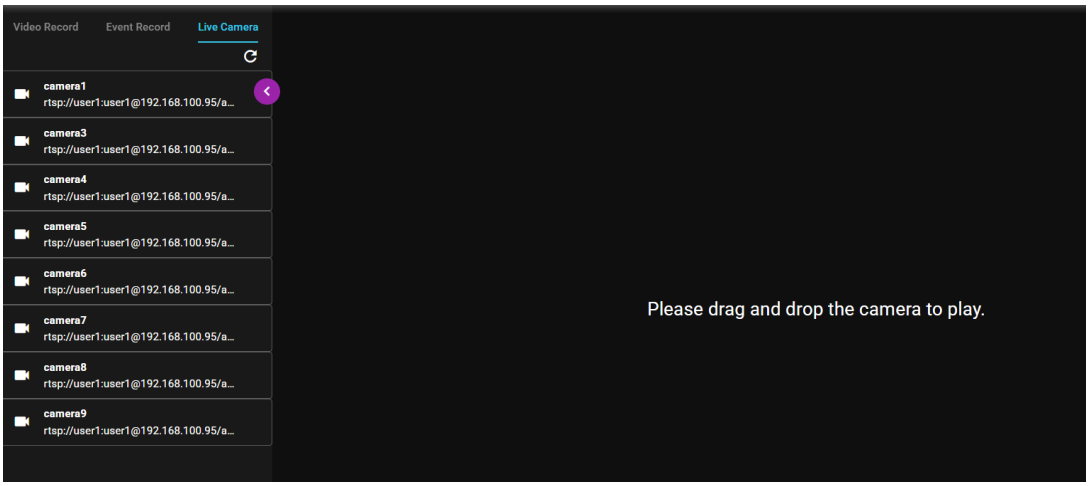
Also, when you press the refresh icon, the information in the list will be updated to reflect the latest status of the VIDEO Collector (Registered, updated, and deleted items will be applied.).



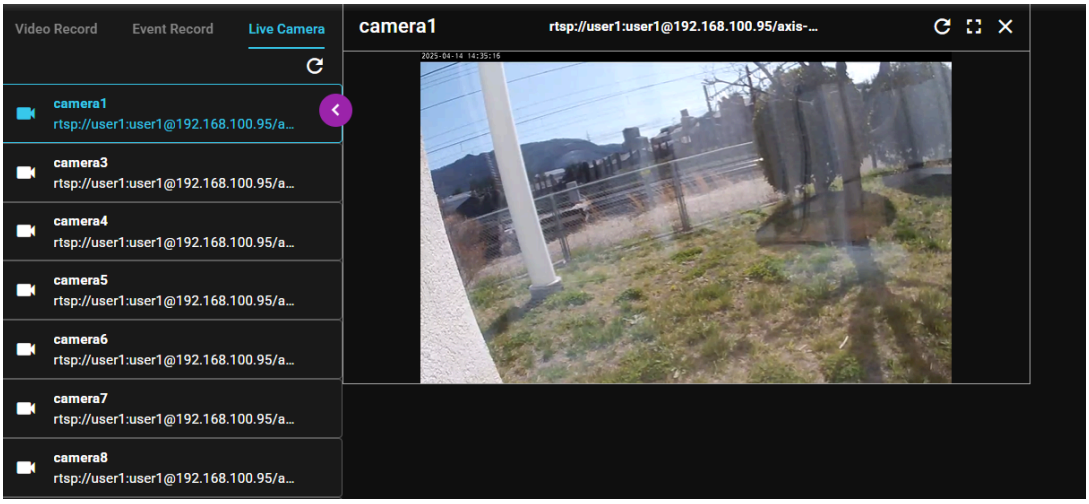
11.5.4.2 Play

You can play back the current video from the camera by following the steps below. It can play back 10 cameras simultaneously.

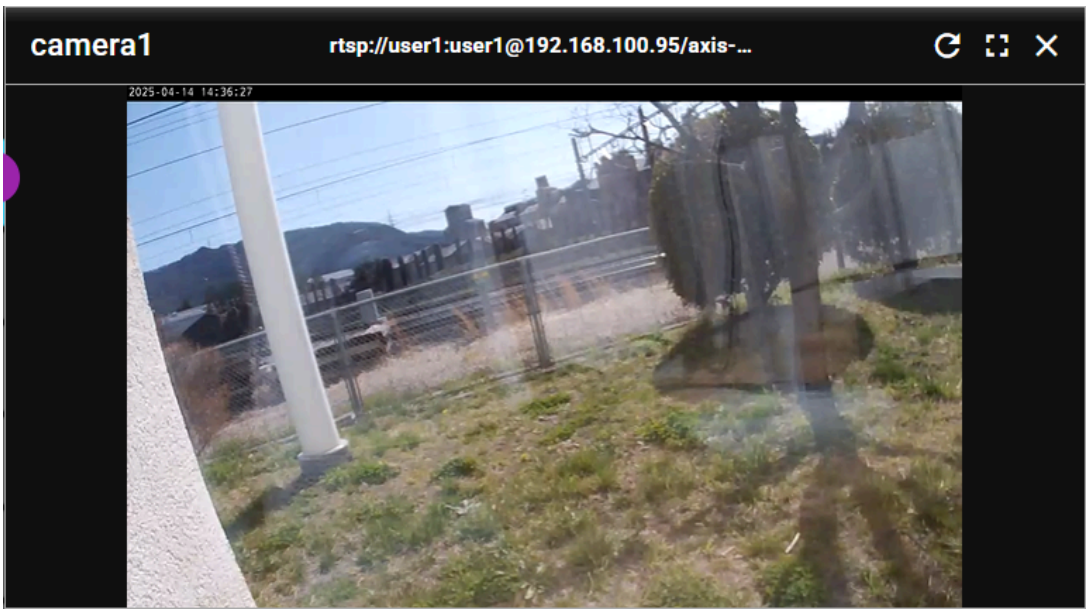
1. Drag a camera video in the list and drop it on the right side.



2. The current video from the camera will play.



The screen items are as follows:



Number	Description
①	Displays the name of the VIDEO Collector.
②	Displays the RTSP URL of the camera registered in the VIDEO Collector.
③	Press the icon to reconnect to the camera.
④	Press the icon to display the camera video in full screen.
⑤	Press the icon to hide the camera.
⑥	Plays the live video from the camera.

If storage is full

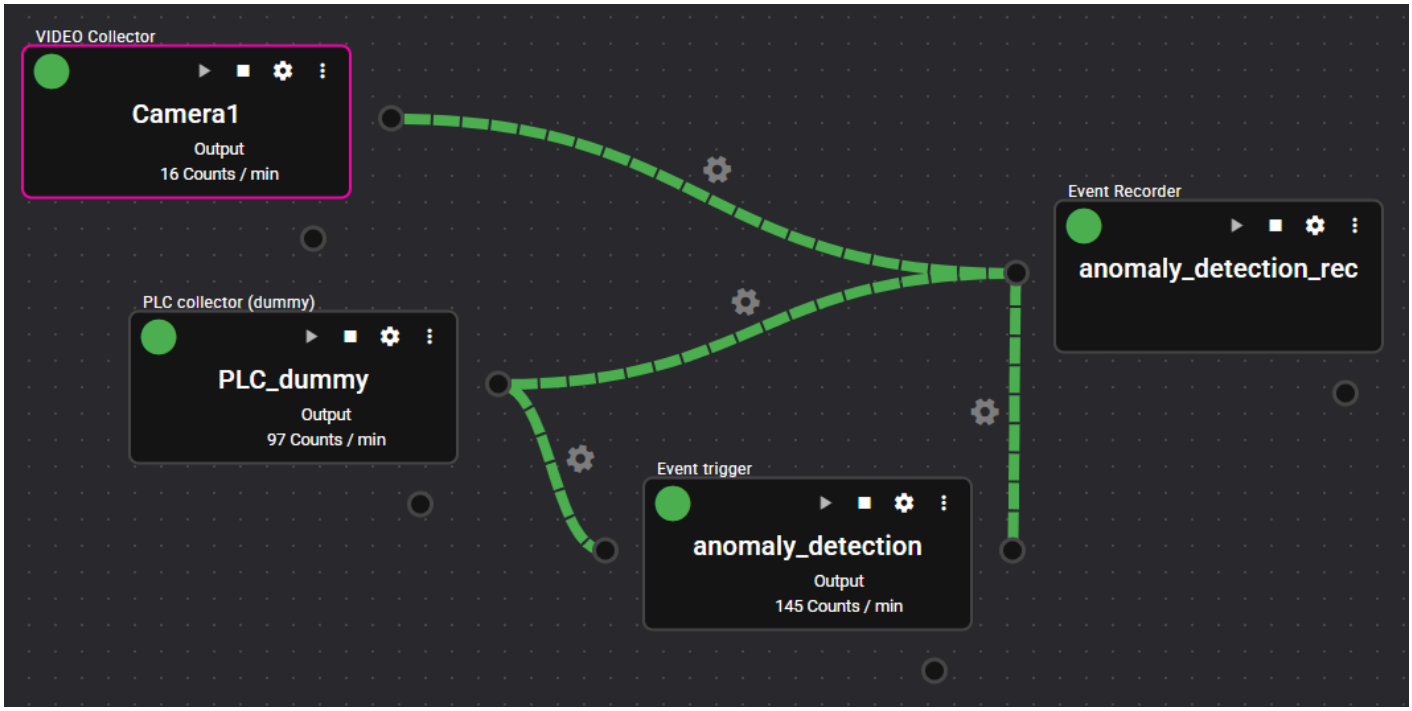
Live Camera function cannot be used if the machine on which i-Trigger is running is full of storage space.

11.6 Components

This section explains the components used in i-Trigger.

11.6.1 VIDEO Collector

This component receives images from network cameras distributed by RTSP and saves recorded files.
Recorded files are saved as ts files in a specified directory and can optionally be output in mp4 format.
Saved video files are retained for a certain period of time and can be played back on the Video Viewer screen while retained.
This component does not support communication through a Proxy Server.



11.6.1.1 Basic Information

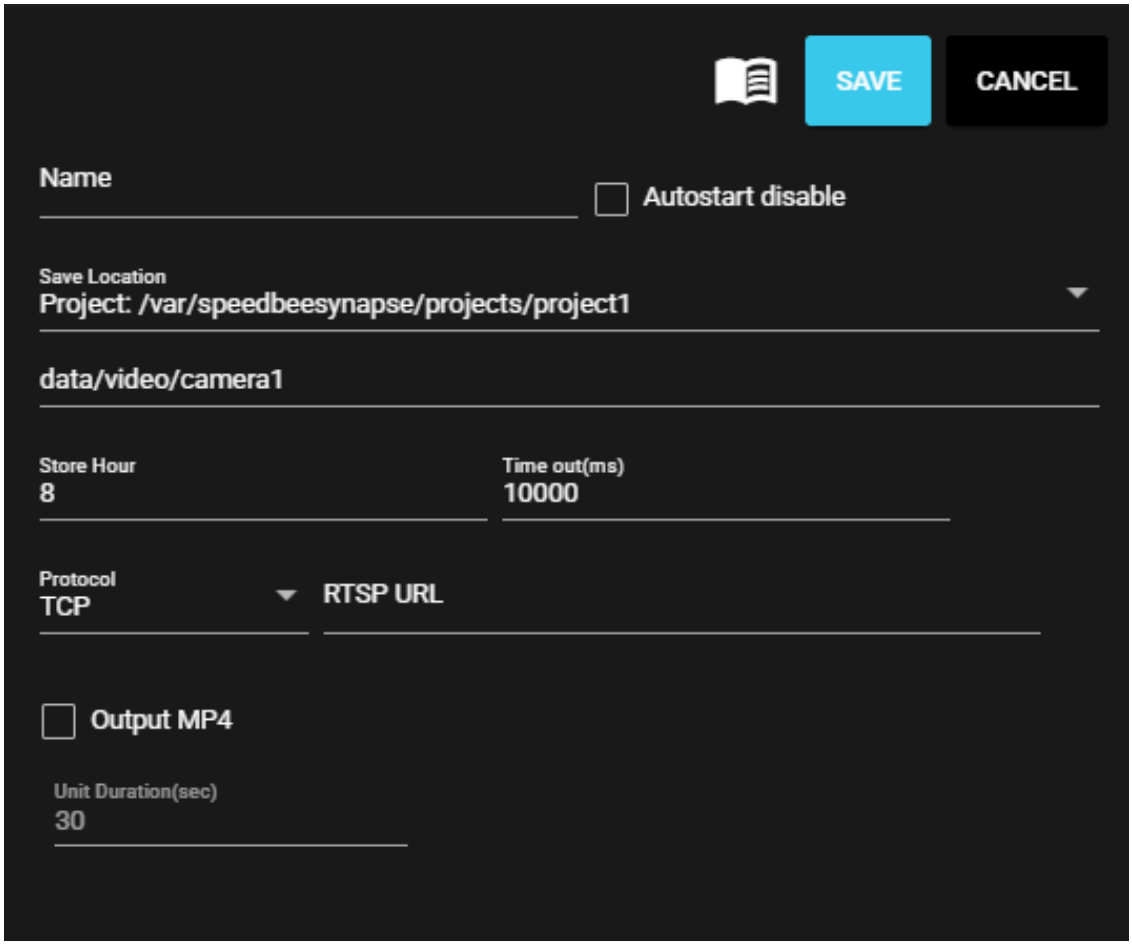
Item	Description
Base Name	VIDEO Collector
Description	It receives video from network cameras and saves TS files (MPEG2-TS) in a specified directory.
Consumption Point	2

Item	Description
Port	Input: 0 Output: 1 Received video is output to a TS file as recorded data. You can play this file in the Video Viewer. For more information on output data, see Output Data Items. System: 1 Outputs the status of component operation and error information. For more information on output error information, see Error Code List.

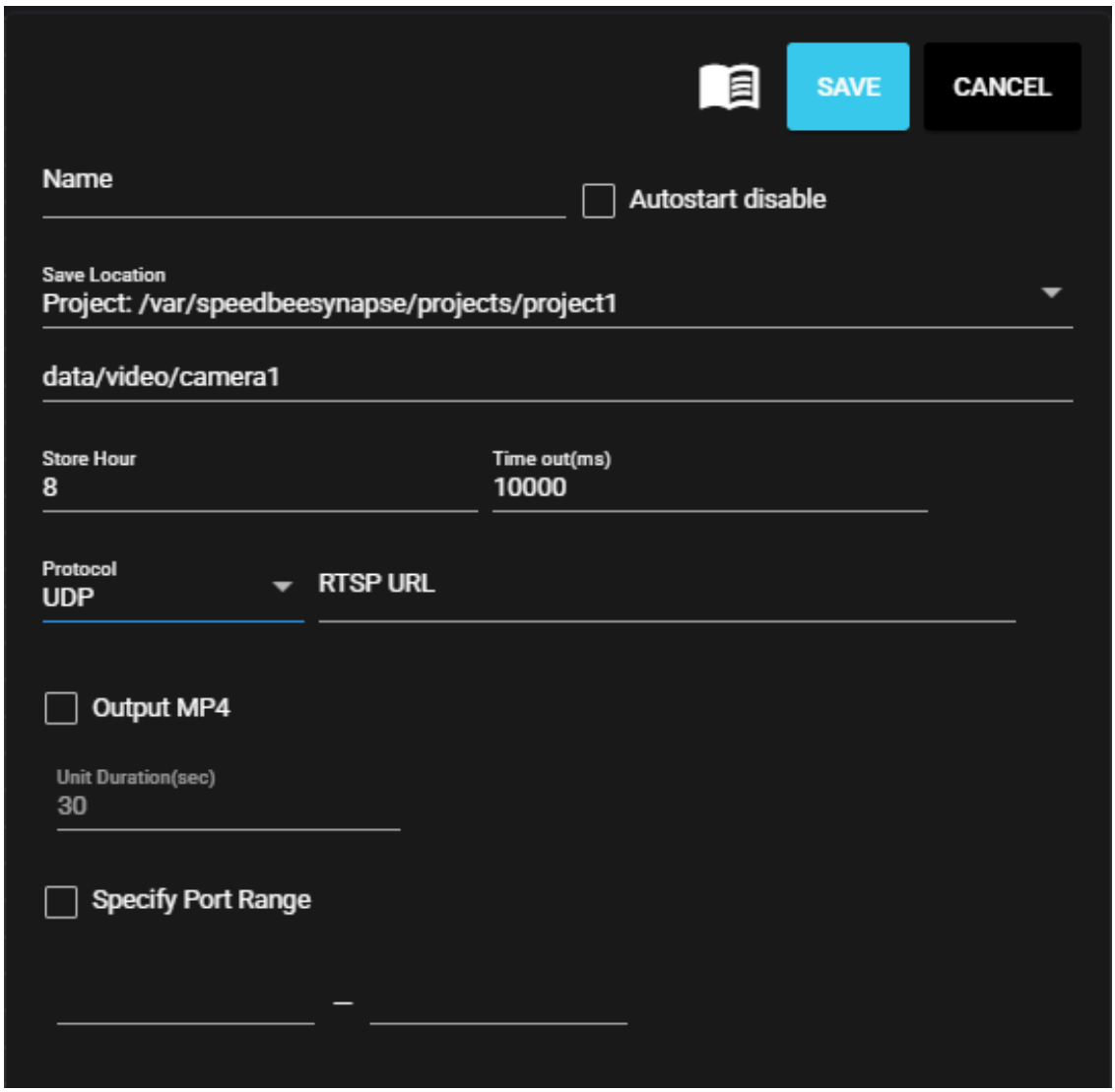
11.6.1.2 Configuration Items

The VIDEO Collector configuration items are as follows:

- When TCP protocol is selected



- When UDP protocol is selected



Item	Description
Name	Input the name of the component. Note: Names that duplicate other component names are not allowed.
Autostart disable	Used to disable autostart of the component.
Save Location	A directory where video files are saved. The base directory and its relative path are specified. The base directory can be selected from the options below. • Project path (Files are saved under the project directory. If the project is duplicated, the save destination will be under the new project.) Note: In some environments, this setting cannot be changed/edited. • Custom path (You can directly specify an absolute path. Unlike the project path, this setting remains unchanged even if the project is duplicated. Normally, a relative path to the base directory is specified, but you can also use an absolute path.

Item	Description
Store Hour	Retention period for video files. It is specified on an hourly basis.) Note: Videos will be automatically deleted when the Store Hour is exceeded.
Time out(ms)	Input the time in milliseconds that the communication time times out.
Protocol	Protocol to be used for transport. • TCP • UDP Note: In some environments, this setting cannot be changed/edited to UDP.
RTSP URL	Live stream URL for the network camera
Output MP4	When enabled, MP4 video files can be generated at regular intervals. Note: When disabled, only TS files are output.
Unit Duration	When Output MP4 is enabled, the recording time per MP4 file can be specified in seconds. Note: The video length may not exactly match the specified setting.
Specify Port Range	When UDP is selected, a range of UDP ports for receiving data is specified.

11.6.1.3 Output Data Items

The VIDEO Collector outputs the following data:

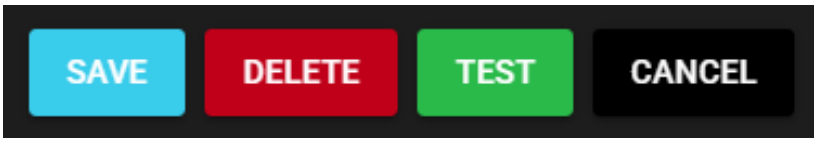
Item	Column Name	Data Type	Description
TS File	$ss_movie_file	FILE	This column outputs TS files (MPEG2-TS). The media type is fixed to "video/MP2T".
Container File	container_file	FILE	This column outputs MP4 files when Output MP4 is enabled. The media type is fixed to "video/mp4".

11.6.1.4 Test Connect

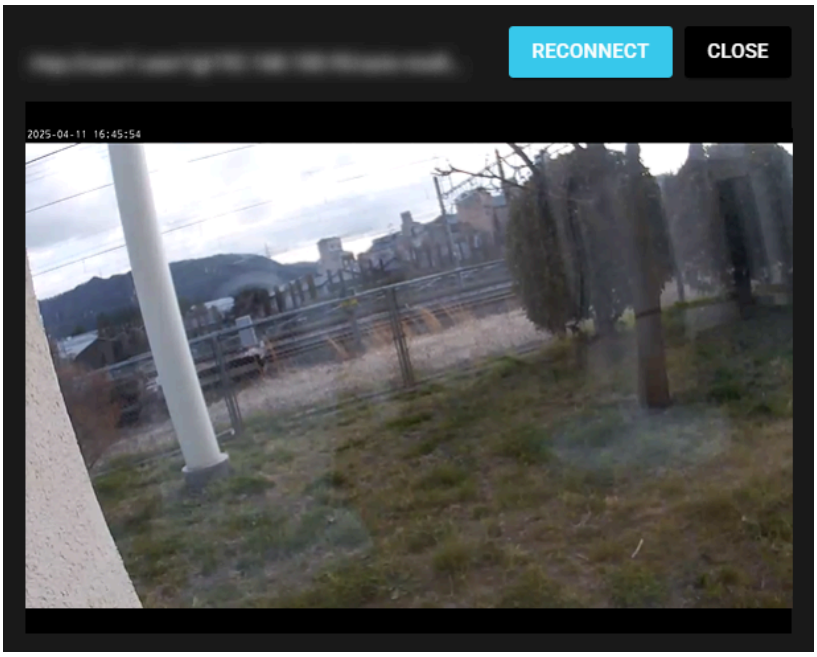
By following the steps below, you can perform a test connection to the camera you have configured.

1. Open the configuration screen of the registered VIDEO Collector.

2. Press Test.



3. A test connection to the configured camera is performed, and if the connection is successful, the camera image can be played back live.



Caution for test connection of the camera

- If you cannot connect to the camera

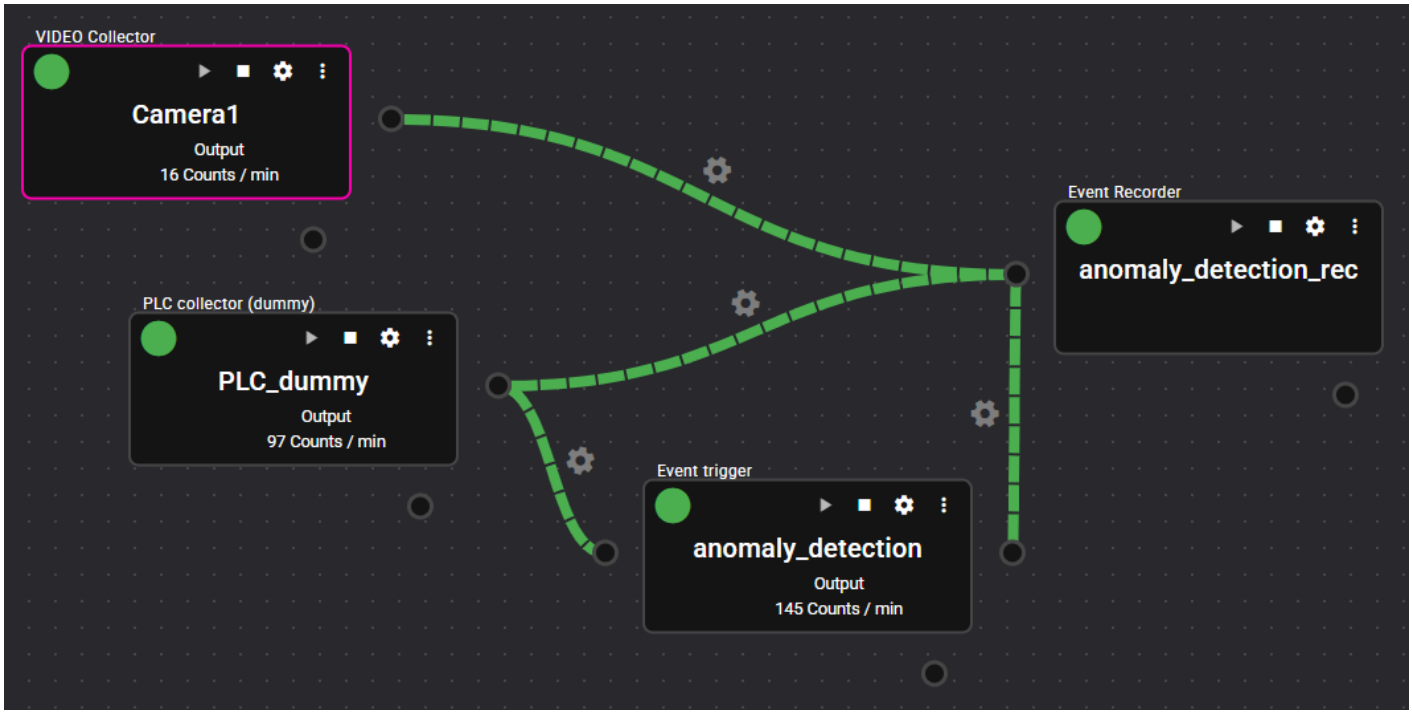
Make sure that the RTSP URL and other settings are correct. Also, make sure that the network camera you are using is the camera that is supported for use. Even if the camera is supported for use, if the configuration on the camera side is not compatible with the supported video codecs, you will need to change the configuration on the camera side.
Reference: See the cameras that can be used.
Also, increasing the setting for timeout may resolve the problem.

- Number of simultaneous connections

Depending on the camera type, the number of simultaneous connections for the same streaming will vary. Low-specification cameras may support only one connection for simultaneous playback, so please check your camera's instruction manual or the camera's product website.

11.6.2 Event Recorder

This component collects data in the event section and outputs it in JSON format that can be referenced by the event record function in the Video Viewer screen.
If the VIDEO Collector is connected to the IN Port, video files in the section can also be collected.
Since the event section is determined by the event trigger component, an event trigger must be connected to the IN Port.



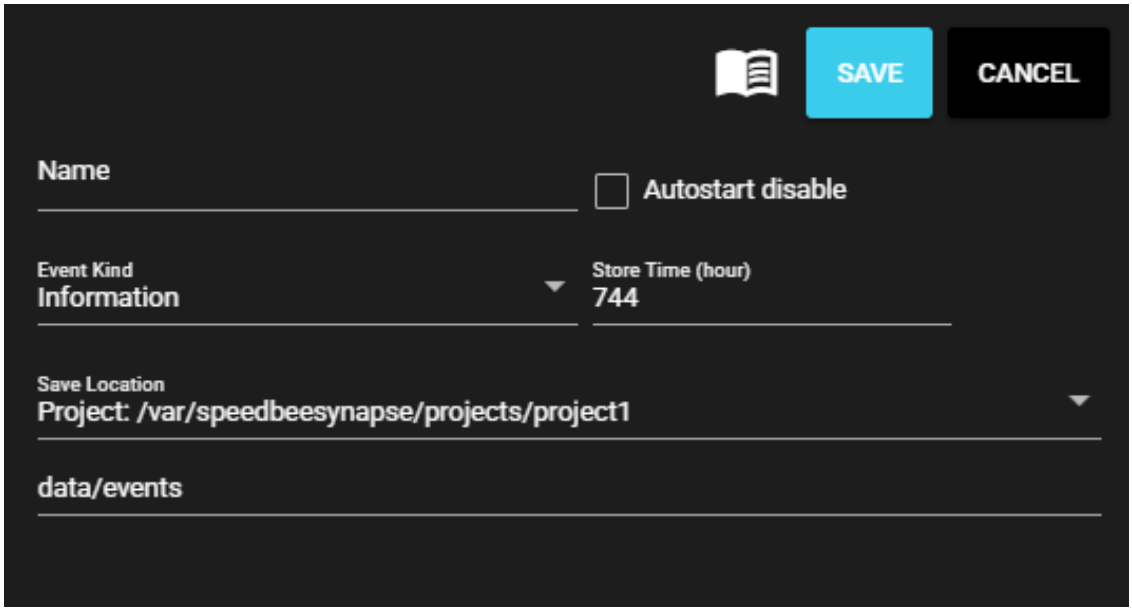
11.6.2.1 Basic Information

Item	Description
Base Name	Event Recorder
Description	It collects data after an event is detected and outputs the collected data in JSON. If there are video files in the same section, they will be collected together as well. By saving the collected data to a specified directory, you can play it back in the Video Viewer.
Consumption Point	2
Port	Input: 1 Receives data from other components. Output: 1 Outputs the data in the section to be recorded to a JSON file. For more information on output data, see Output Data Items.

Item	Description
	System: 1 Outputs the status of component operation and error information. For more information on output error information, see Error Code List.

11.6.2.2 Configuration Items

The Event Recorder configuration items are as follows:



Item	Description
Name	Input the name of the component. Note: Names that duplicate other component names are not allowed.
Autostart disable	Used to disable autostart of the component.
Event Kind	A classification that can be configured for events. You can optionally specify this option according to the event content. • Information • Warning • Error
Store Hour	Retention period for recorded data and videos. It is specified on an hourly basis. Note: Videos will be deleted by the event data manager when the Store Hour is exceeded.

Item	Description
Save Location	A directory for saving data and videos in the section where an event is detected. The base directory and its relative path are specified. The base directory can be selected from the options below. • Project path (This is saved under the project. If a project is replicated, the save folder will be under the new project.) Note: In some environments, this setting cannot be changed/edited. • Custom path (Normally, a relative path to the base directory is specified, but you can also specify any absolute path directly. Even if the project is duplicated, the specified path will remain as the save location.)

11.6.2.3 Output Data Items

The Event Recorder outputs the following data:

Item	Column Name	Data Type	Description
Event File	event_file	FILE	This column outputs a JSON file for event recording. The media type is fixed to "application/json".

11.6.3 Event Trigger

This is a component included in the SpeeDBee Synapse console. For more information, see Event Trigger.

11.6.4 Event data manager

This is a component included in the SpeeDBee Synapse console. For more information, see Event Data Manager.

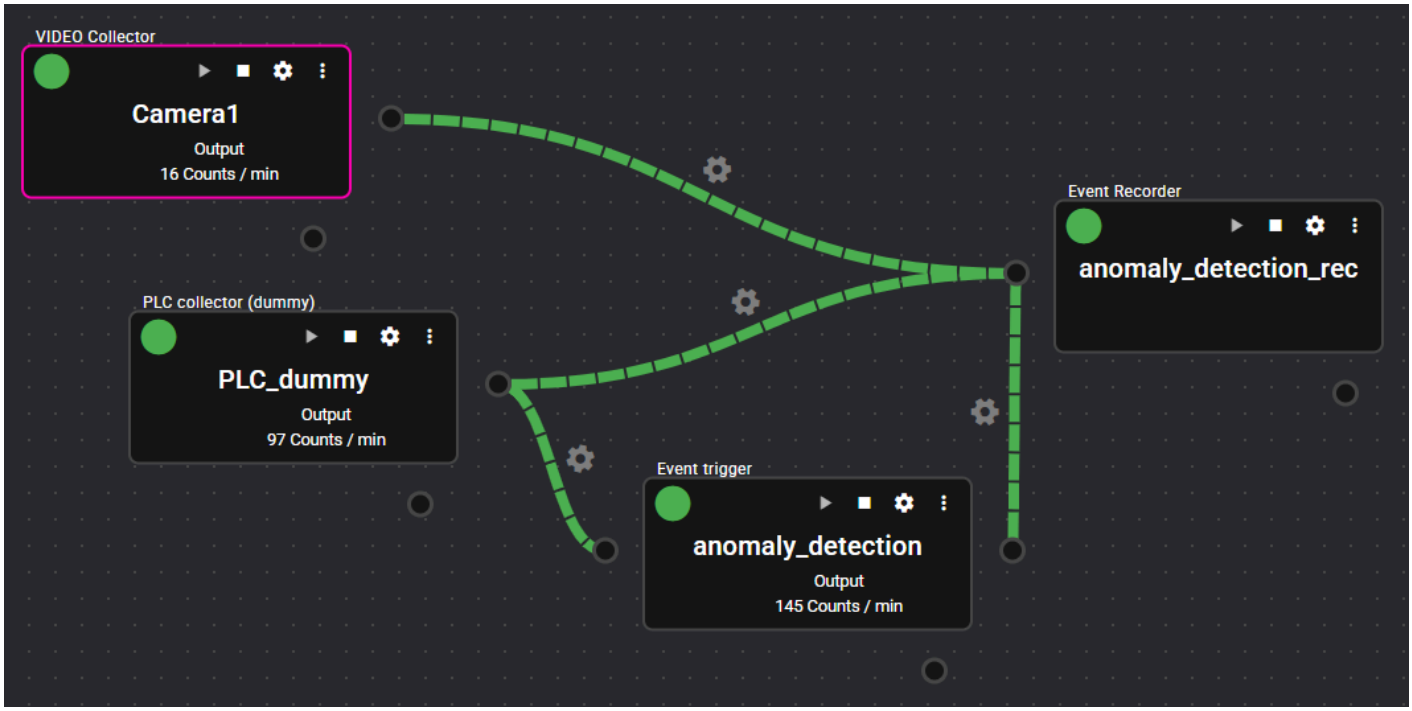
11.7 Tutorial

This section provides a simple tutorial on how to use i-Trigger.

11.7.1 Assumed Use Cases

In this tutorial, error detection will be performed according to the following use cases.

When an error is detected in the data collected from the PLC, confirm the video and collected data before and after the error in the Video Viewer. (Note: This is an example of using custom components to simulate a PLC Collector.)



1. Collect the following data from the PLC:
   - Error detection flag
   - Pressure
2. Record video with a single network camera.
3. When the error detection flag is turned ON, event recording starts.
   - Record 5 seconds before the error detection flag is detected and 5 seconds after the error detection flag is turned OFF.
   - The data displayed in the Video Viewer includes the error detection flag obtained from the PLC, the pressure, and the video taken in that section.

11.7.2 Custom Component Configuration (Simulated PLC Collector)

First, configure a custom component to collect error detection and pressure data.

11.7.2.1 Custom Component Code

Copy the following code, paste it into a text editor such as Notepad, and save it on your PC as a file named dummy_plc.py. Specify UTF-8 as the character encoding.

```
from speedbeesynapse.component.base import HiveComponentBase, HiveComponentInfo, DataType
import time
import json


@HiveComponentInfo(uuid='12318c73-fe45-fded-150a-b95c894f3692', name='PLC Collector(Dummy)',
inports=0, outports=1)
class HiveComponent(HiveComponentBase):
    def __init__(self):
        # called at creating component instance
        pass
    def __del__(self):
        # called at deleting component instance
        pass

    def premain(self, param):
        # called before main()
        # create a column to register dummy data
        self.data = self.out_port1.Column('Pressure', DataType.INT16)
        self.flag = self.out_port1.Column('anomaly_detection_flag', DataType.BOOLEAN)

    def postmain(self, param):
        # called after main() finished
        pass

    def main(self, param):
        value = False
        count = 0
        while self.is_runnable():
            ts = self.get_timestamp()
            pressure = 50
            count += 1
            # register error data every 30 counts
            if (count % 30) == 0:
                value = True
                pressure = 100
            else:
                value = False
            # register dummy data
            self.flag.insert(value, ts)
            self.data.insert(pressure, ts)

            time.sleep(1.0)

    def notify_stop(self):
        # called before the component received stop request from system
        pass
```

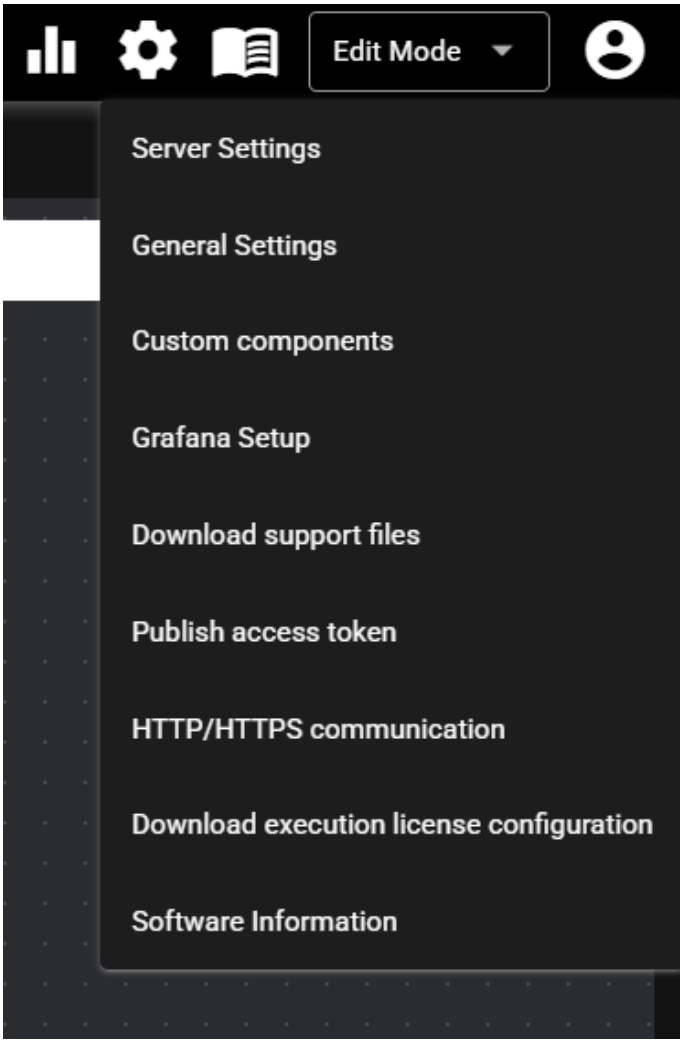
The above code executes the following process:

- At approximately 30-second intervals, set the error detection flag to True and register data.
- Register pressure at 50 when normal.
- When an error occurs, register the pressure value at 100.

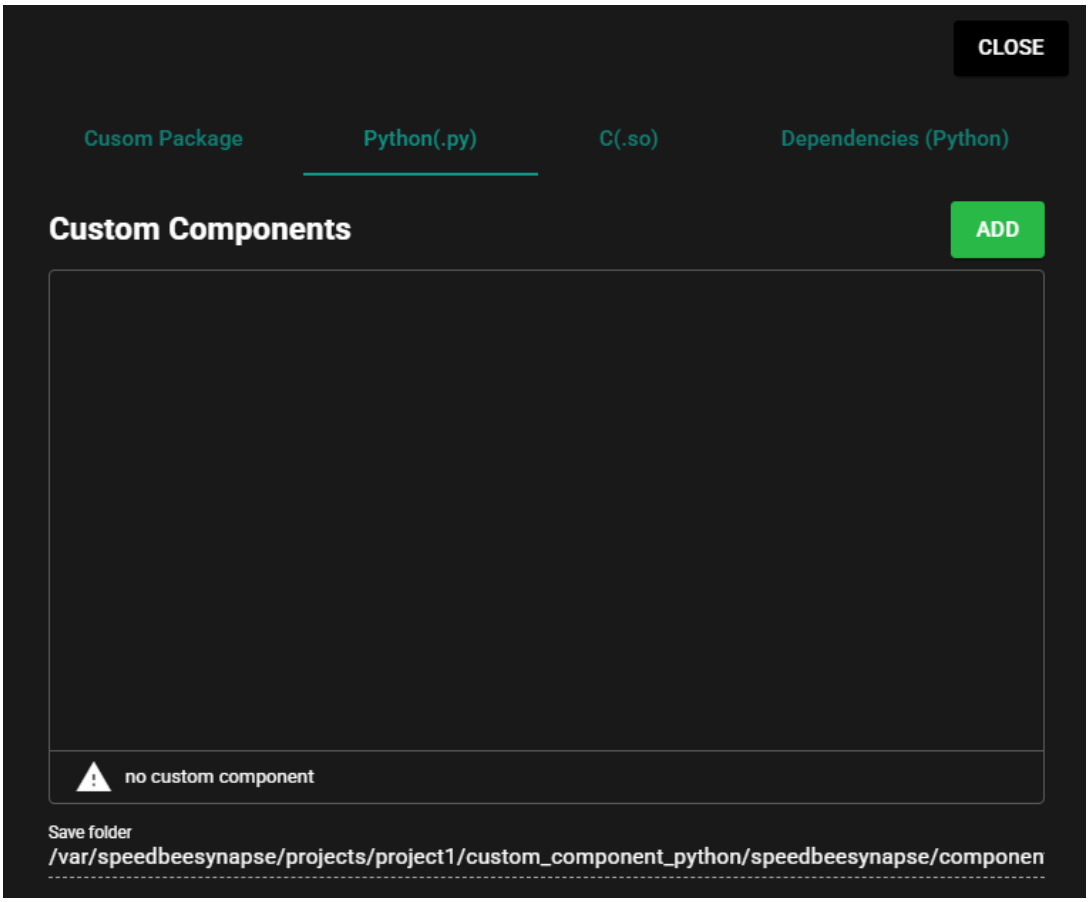
11.7.2.2 Register Custom Components

Configure the saved Python source code for use in SpeeDBee Synapse.

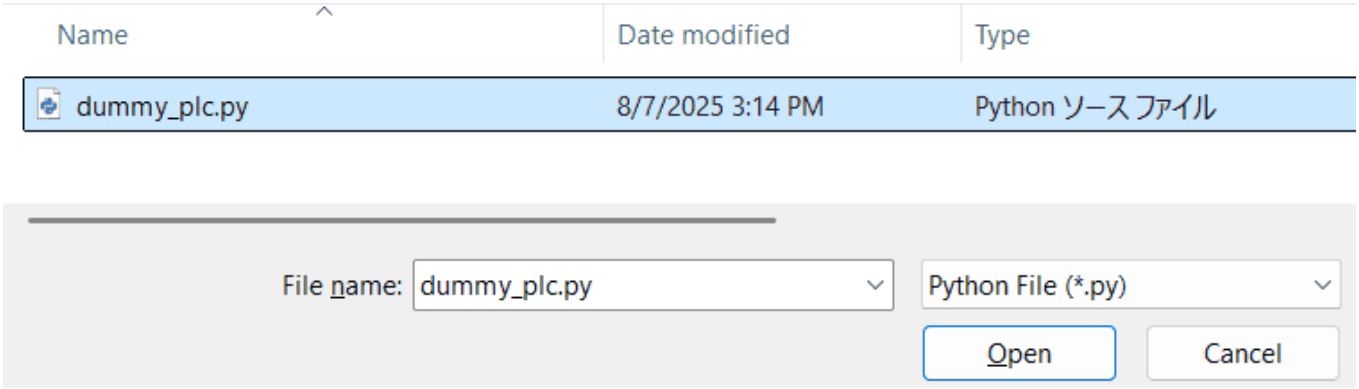
1. Click on Custom Components in the Server Settings Icon.



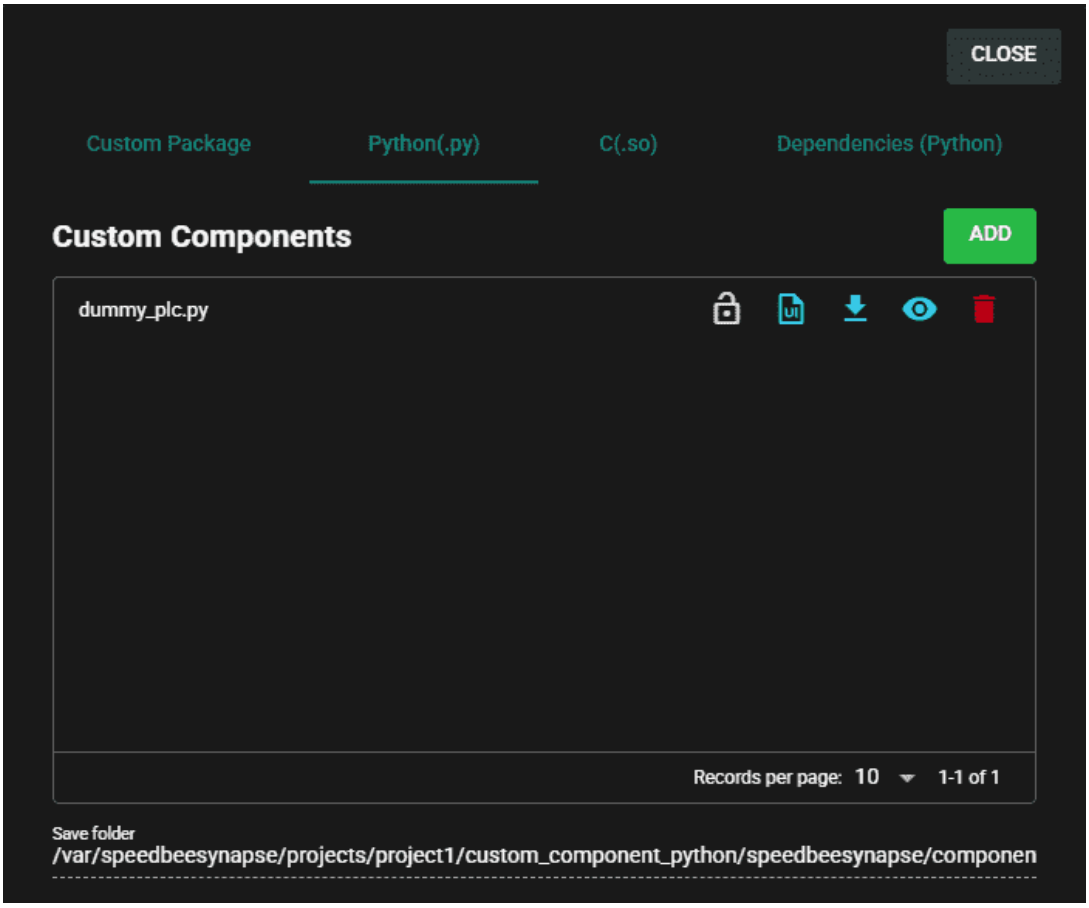
2. Click on Add for Custom Components.



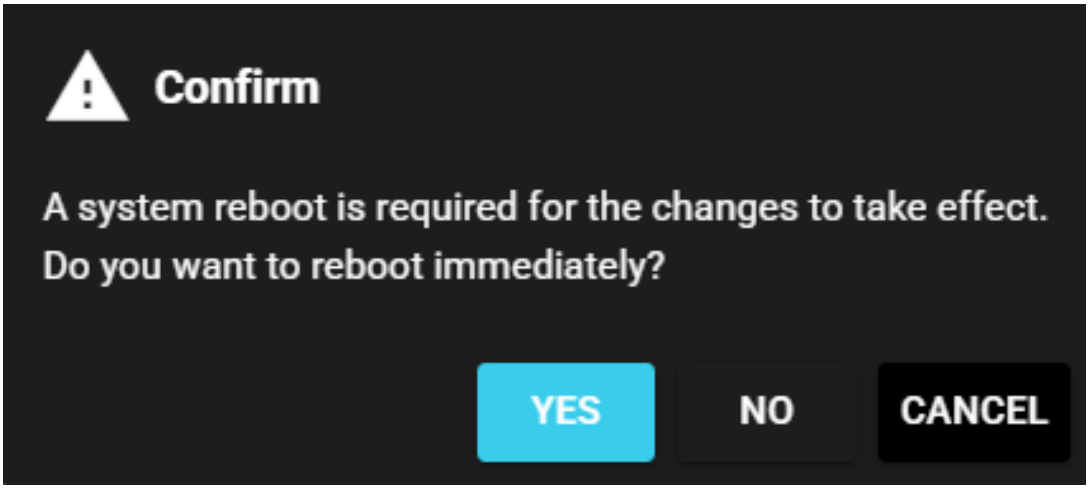
3. Select the dummy_plc.py you just saved.



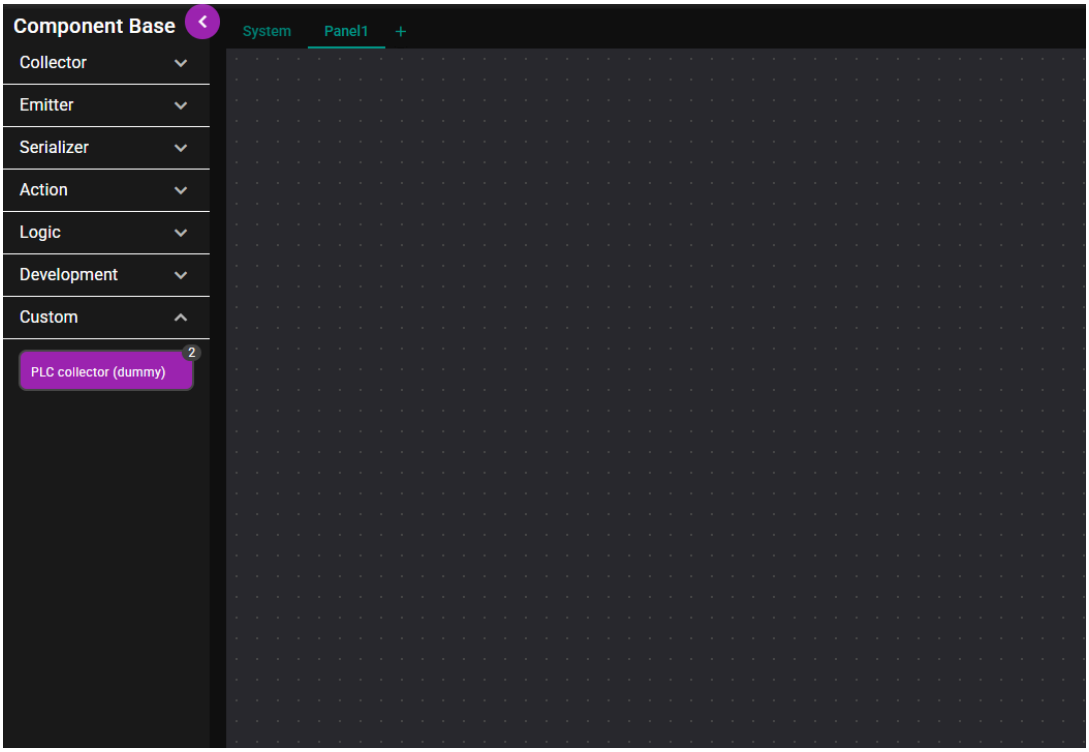
4. Verify that the custom component is displayed on the screen.
   When dummy_plc.py is displayed, click Close on the dialog.



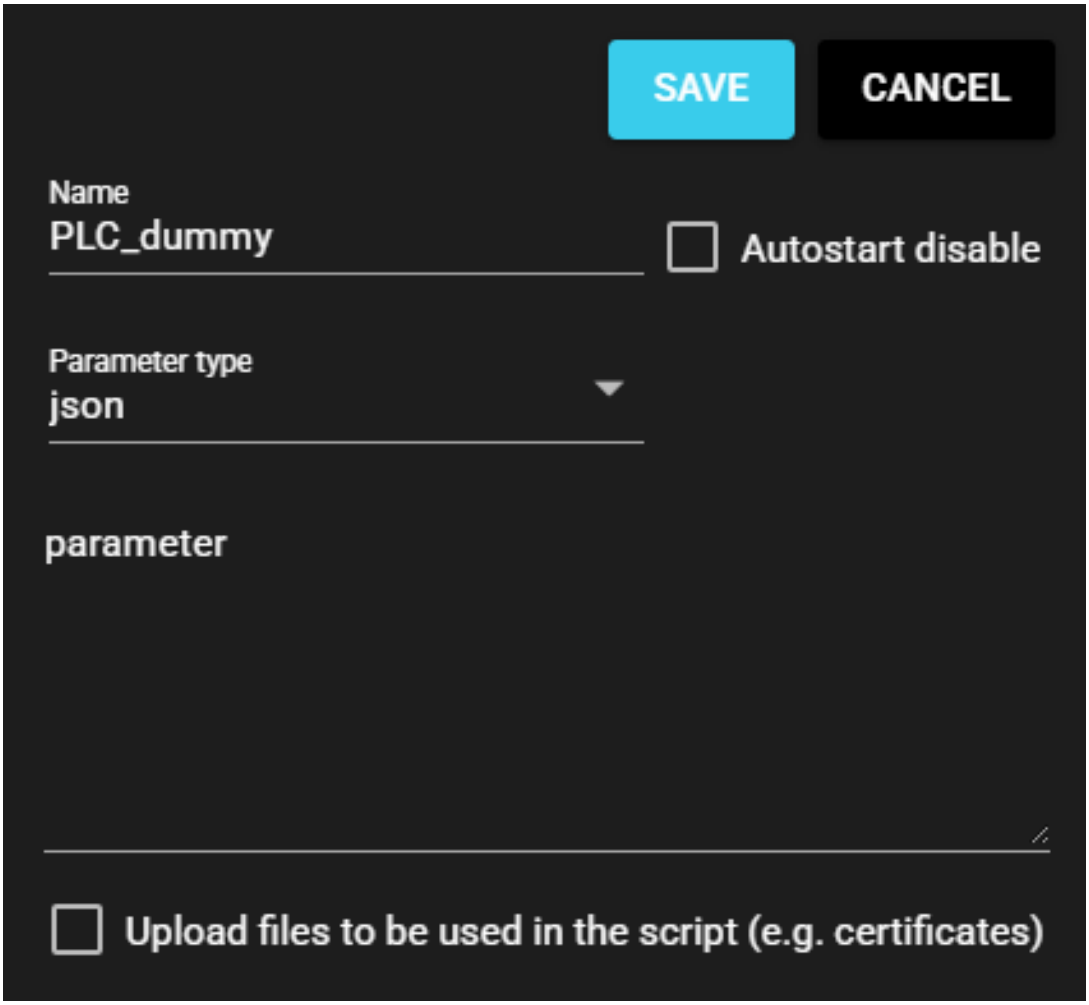
5. When prompted to restart, click Yes.



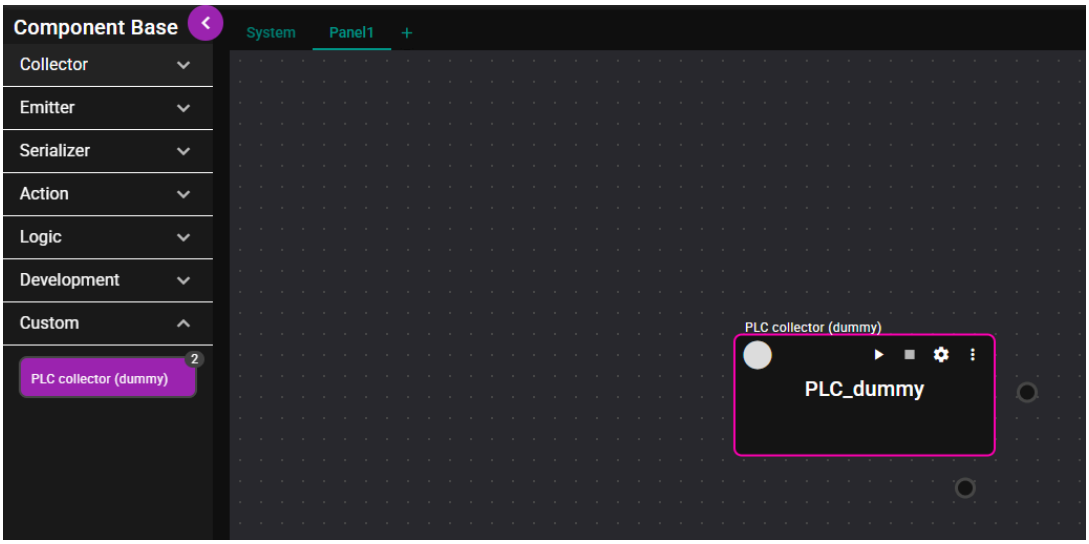
6. Drag and drop PLC Collector (Dummy) from Custom on the left side of the Flow Editor to the center of the editor.



7. Set the name to PLC_dummy and click Save.



8. The custom component is ready.



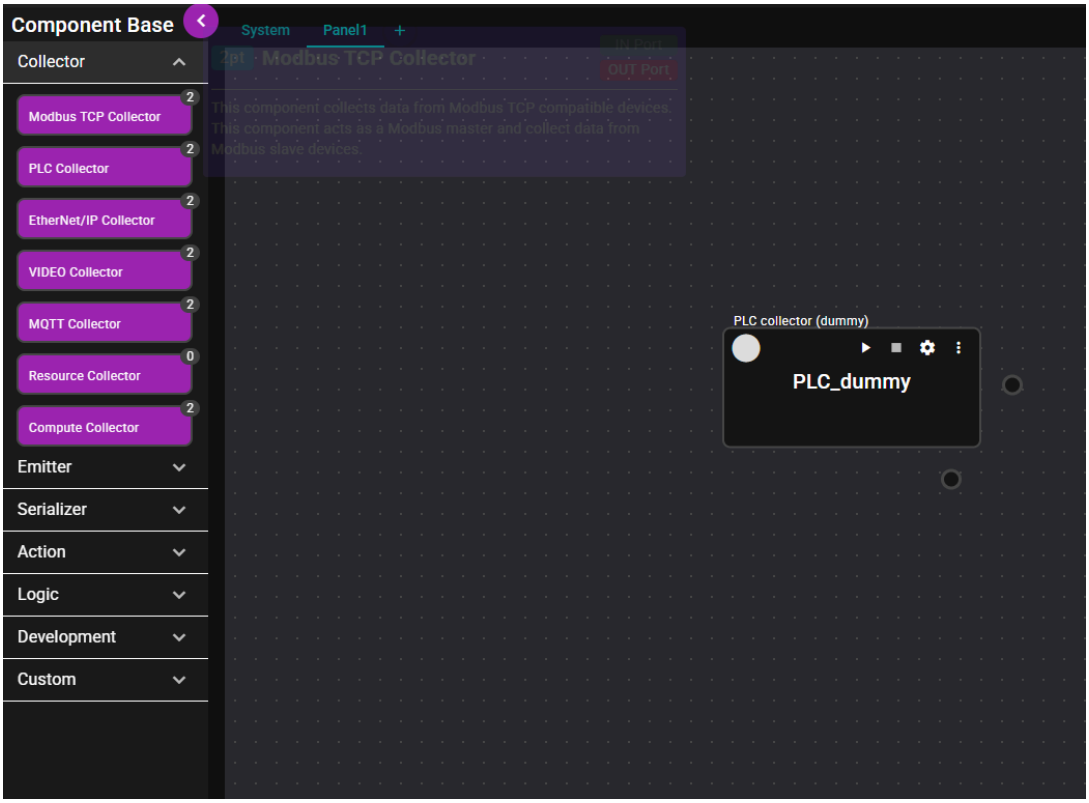
11.7.3 Component Placement and Configuration

Here you can configure various components for using i-Trigger.

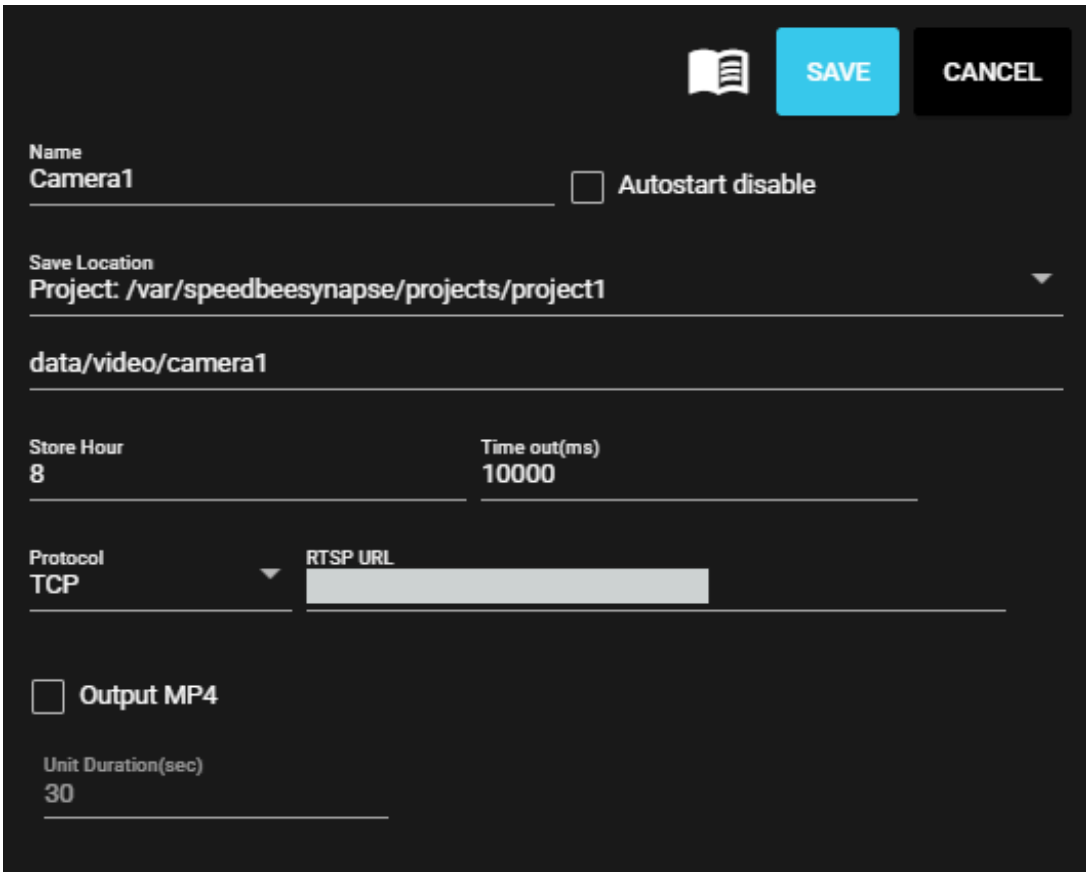
11.7.3.1 VIDEO Collector Configuration

Configure the VIDEO Collector to record video.
If there are no network cameras available, you may skip this configuration step.

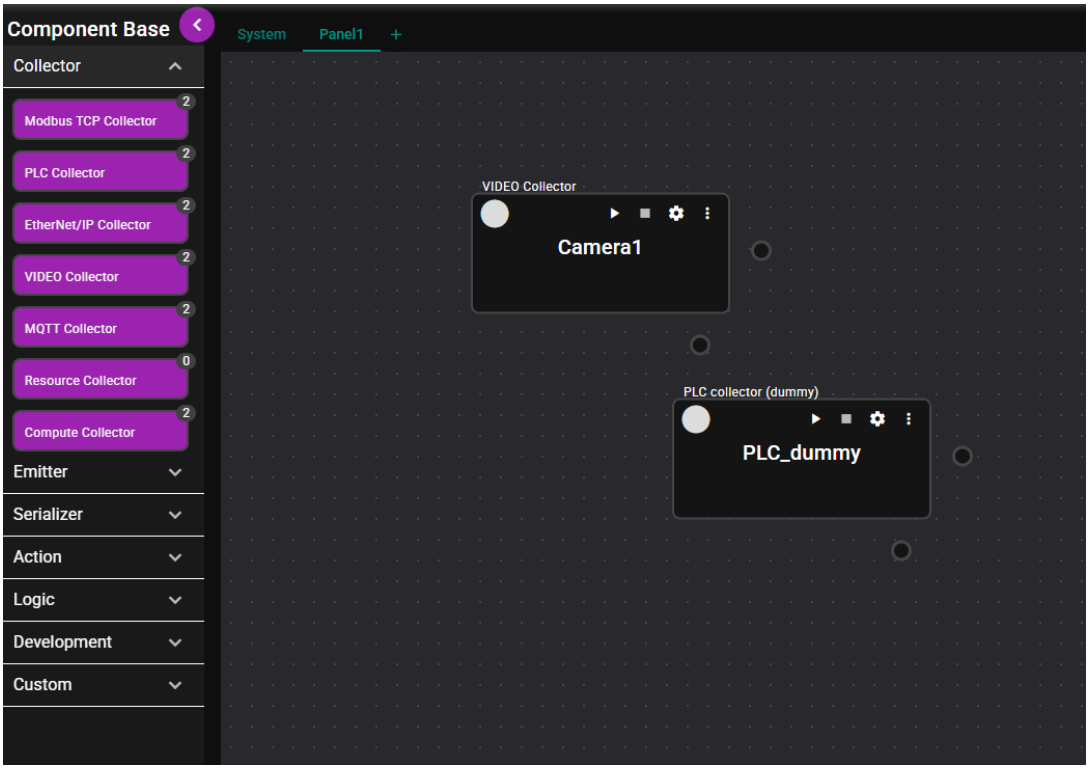
1. Drag and drop VIDEO Collector from Collector on the left side of the Flow Editor to the center of the editor.



2. Set the name to Camera1 and input the specified URL of the camera in the RTSP URL field. Then click Save.



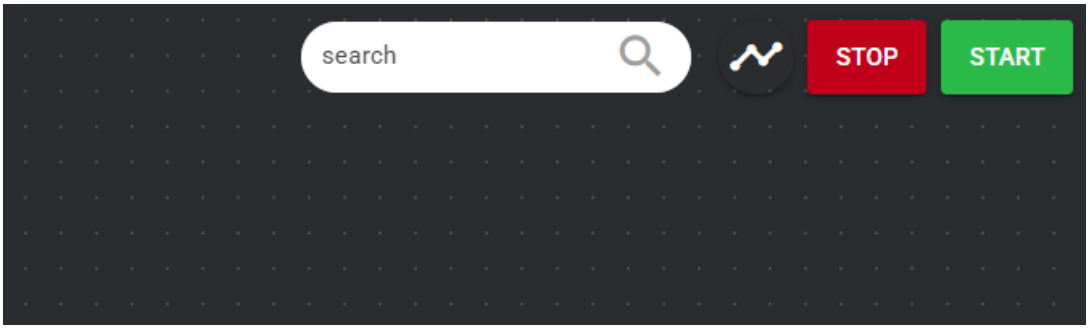
3. The VIDEO Collector Camera1 is ready.



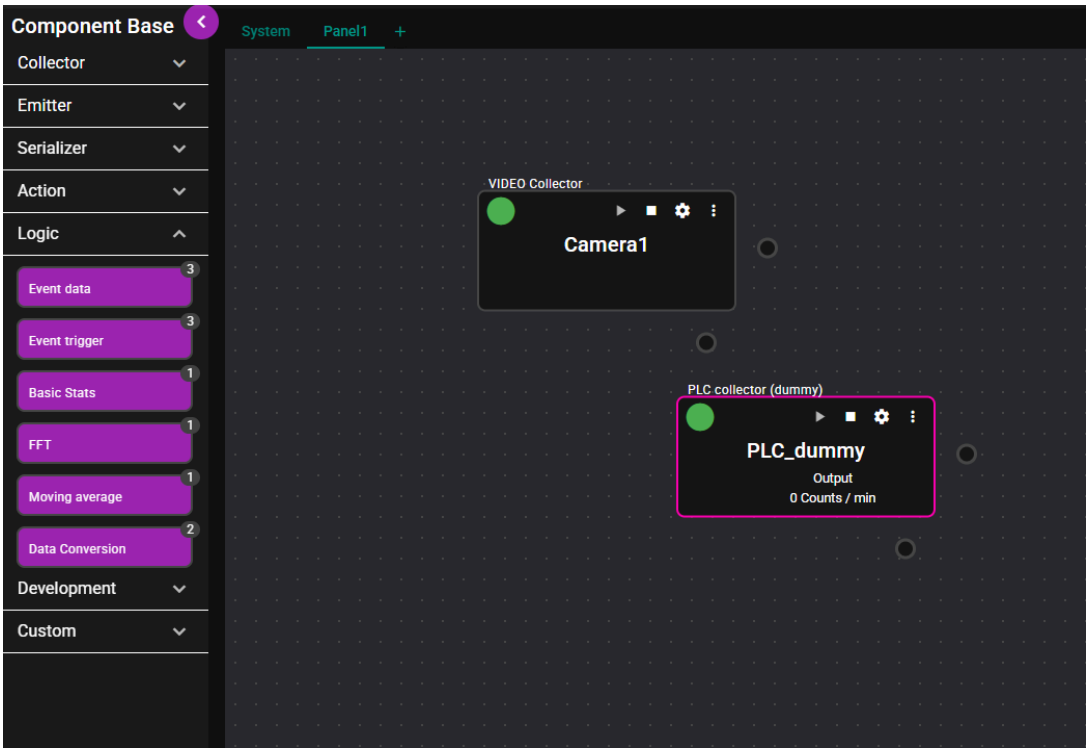
11.7.3.2 Event Trigger Configuration

Event triggers allow you to set conditions for error detection.

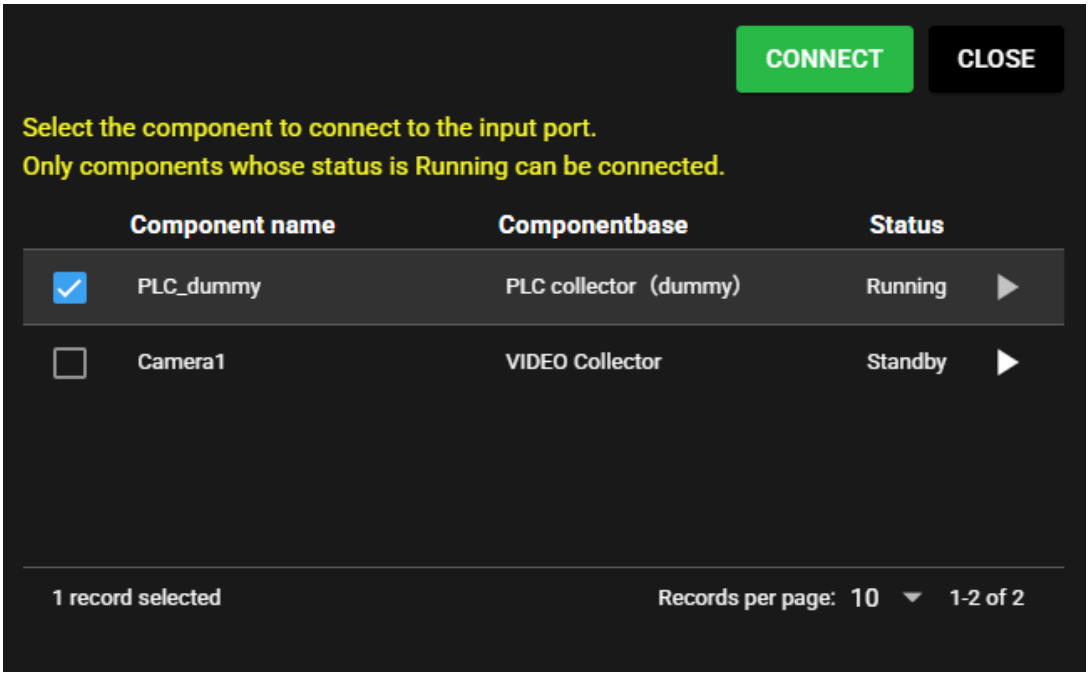
1. Start PLC_dummy and Camera1 before configuring the event trigger. Click Start in the upper right of the panel.



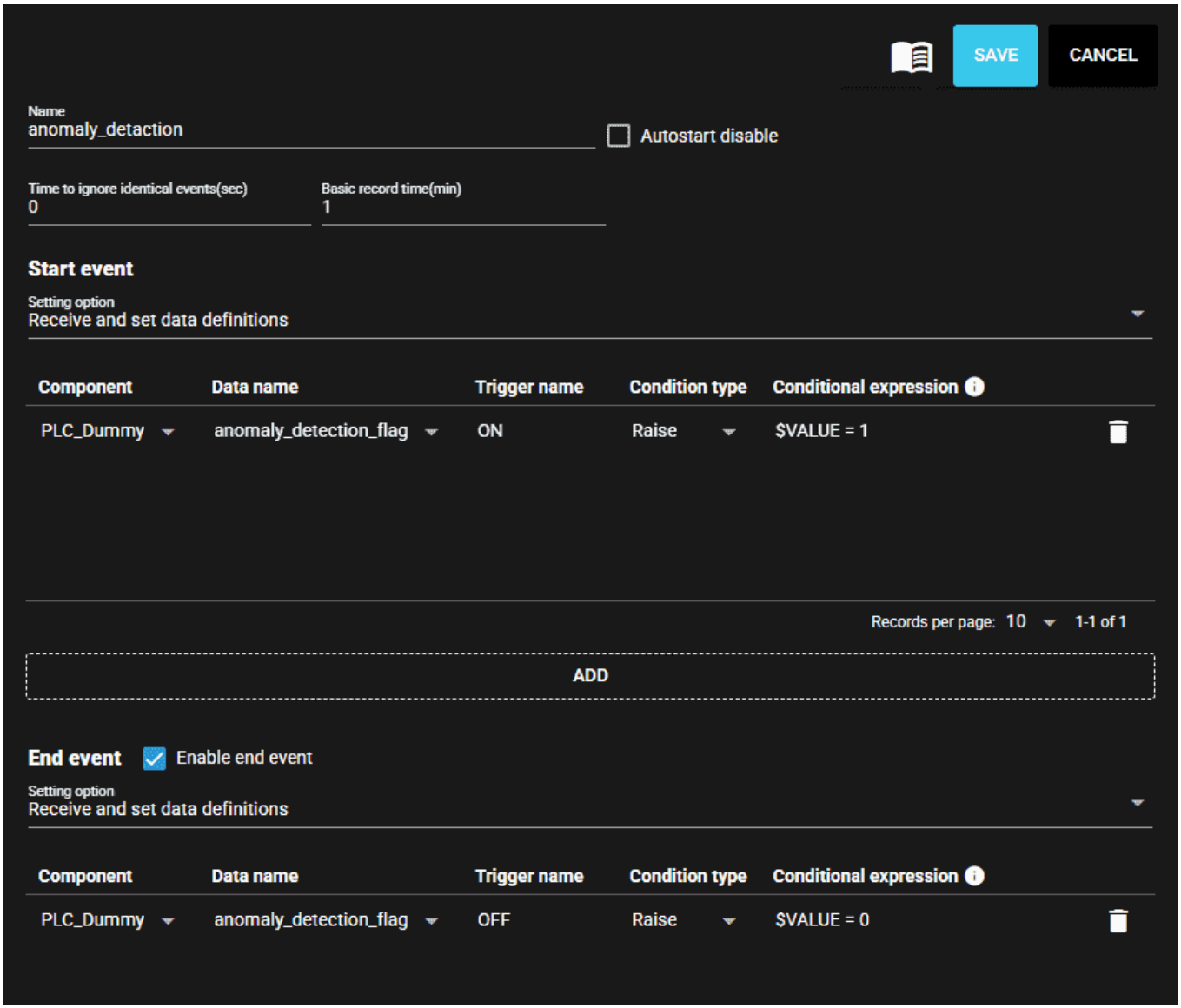
2. Drag and drop Event Trigger from Logic on the left side of the Flow Editor to the center of the editor.



3. For the component to be connected to the IN Port, place a check mark in PLC_dummy and click Connect.



4. Set the name to anomaly_detection. Input other items by referring to the following and click Save.



Item	Value	Supplementary Information
Time to ignore identical events(sec)	0	Set to 0 since it is ignored this time and no setting is required.
Basic record time(min)	1	
Start event	-	-
Component	PLC_dummy	
Data name	Error detection flag	

Item	Value	Supplementary Information
Trigger name	ON	
Condition type	Raise	
Conditional expression	$VALUE = 1	Raise condition for flag to be True.
End event	-	-
Component	PLC_dummy	
Data name	Error detection flag	
Trigger name	OFF	
Condition type	Raise	
Conditional expression	$VALUE = 0	Falling condition for flag to be False.

5. The event trigger anomaly_detection configuration is complete.



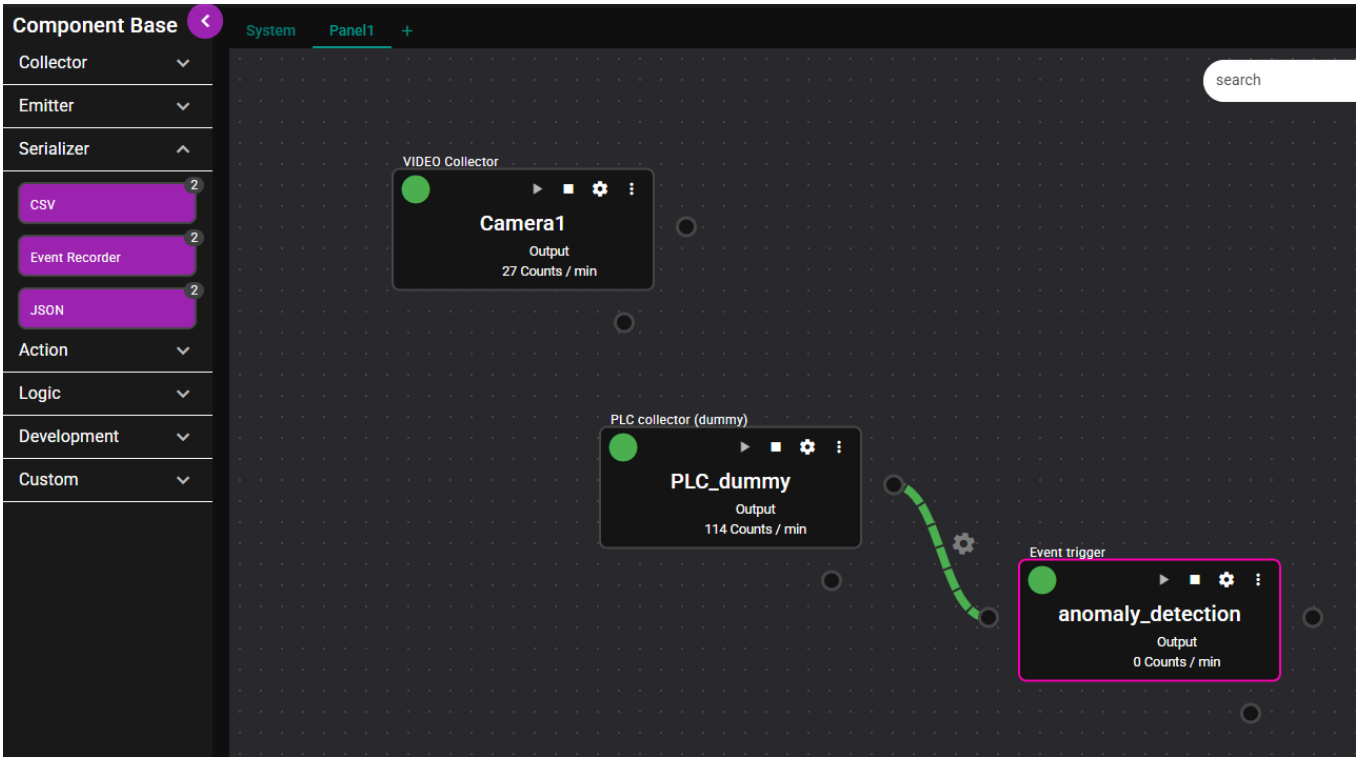
11.7.3.3 Event Recorder Configuration

Finally, configure the Event Recorder settings. You can also configure here the settings of the record interval in the event of an error.

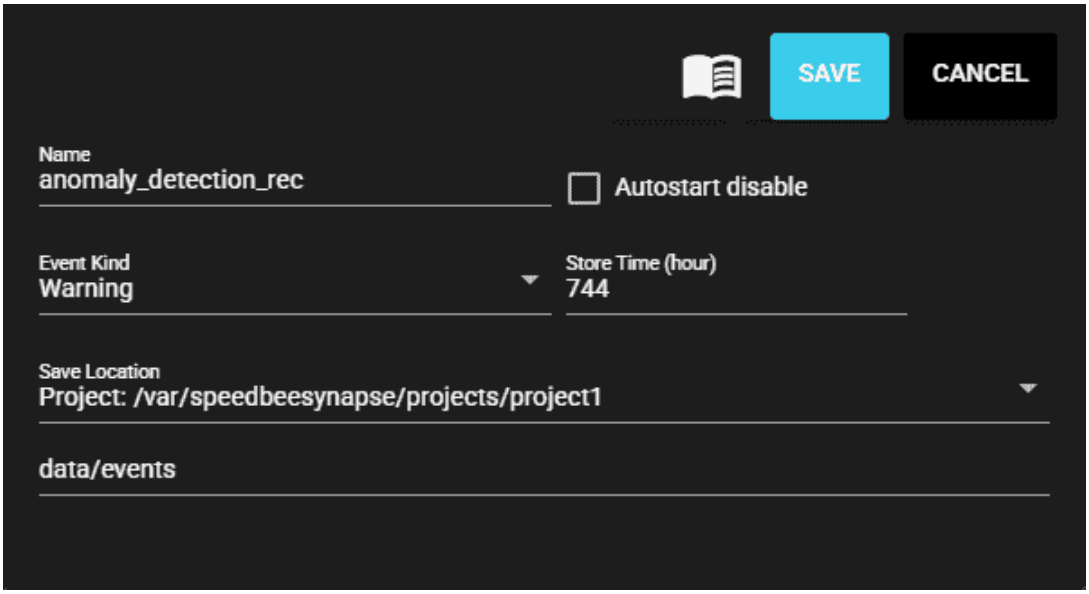
1. Start the event trigger anomaly_detection. Click Start Icon.



2. Drag and drop Event Recorder from Serializer on the left side of the Flow Editor to the center of the editor.



3. Set the name to anomaly_detection_rec. Select Warning for Event Kind. Then click Save.



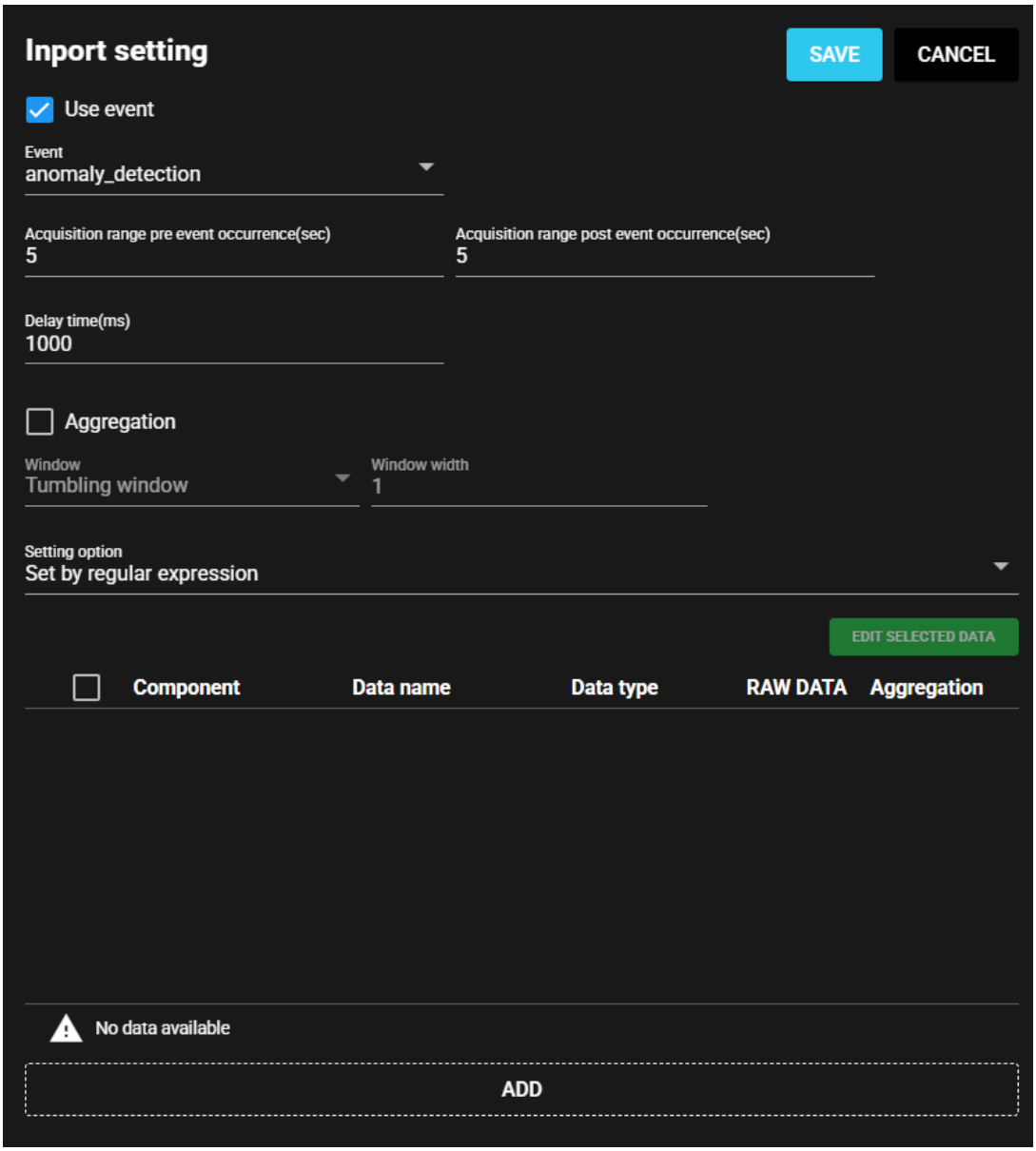
4. Connect the flowlinks of PLC_dummy, Camera1, and anomaly_detection to the anomaly_detection_rec.



5. Open the IN Port settings for anomaly_detection_rec.



6. Configure the IN Port settings and click Save. Please refer to the following to configure the input value settings.



Item	Value	Supplementary Information
Use event	Check ON	If you do not configure this setting, the range of the event occurrence will not be recorded in the Event Recorder.
Event	anomaly_detection	If you do not configure this setting, the range of the event occurrence will not be recorded in the Event Recorder.
Acquisition time pre event occurrence(sec)	5	

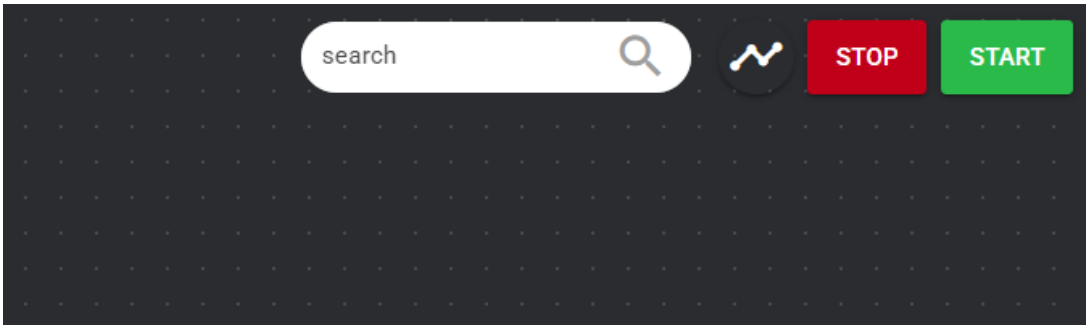
Item	Value	Supplementary Information
Acquisition time post event occurrence(sec)	5	

7. This completes the configuration of all event record settings.

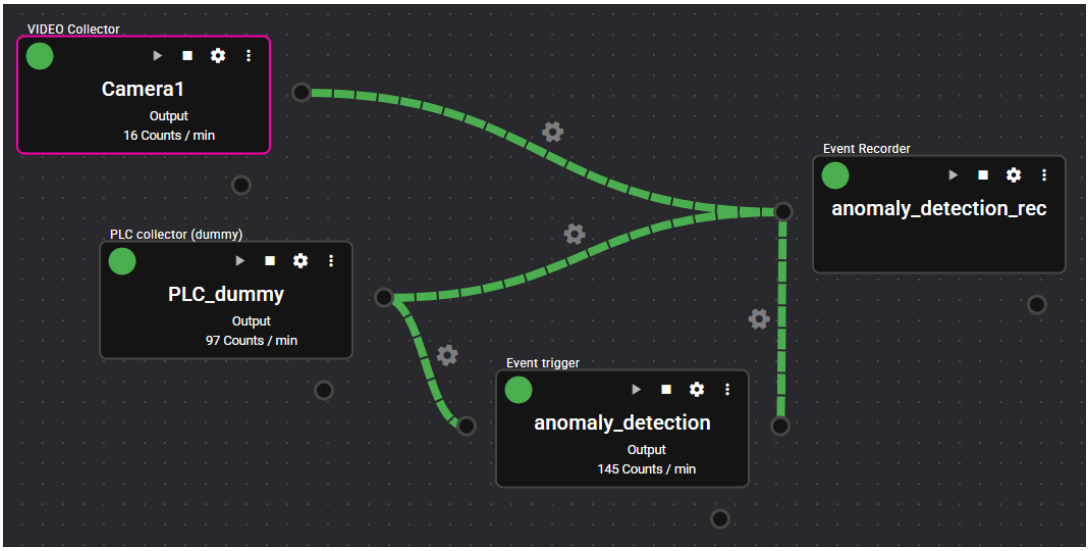
11.7.4 Confirmation of Operation

Operate the data flow for which you have configured settings so far.

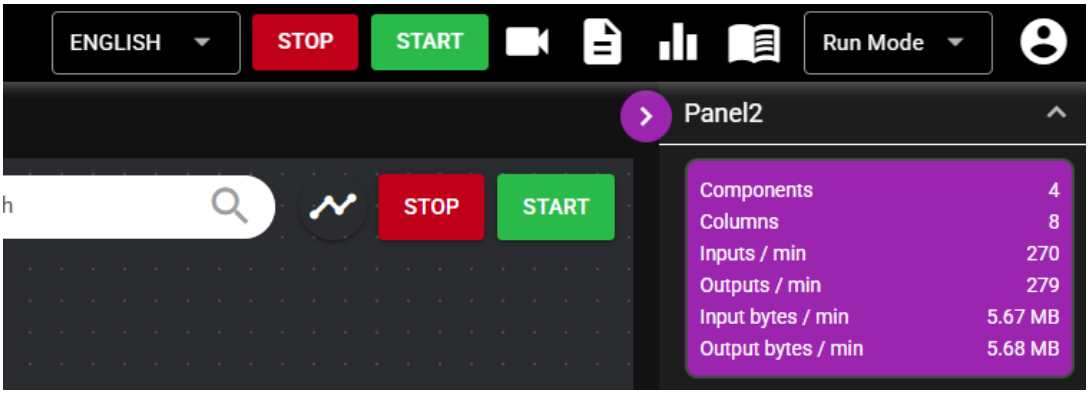
1. Start the configured component.



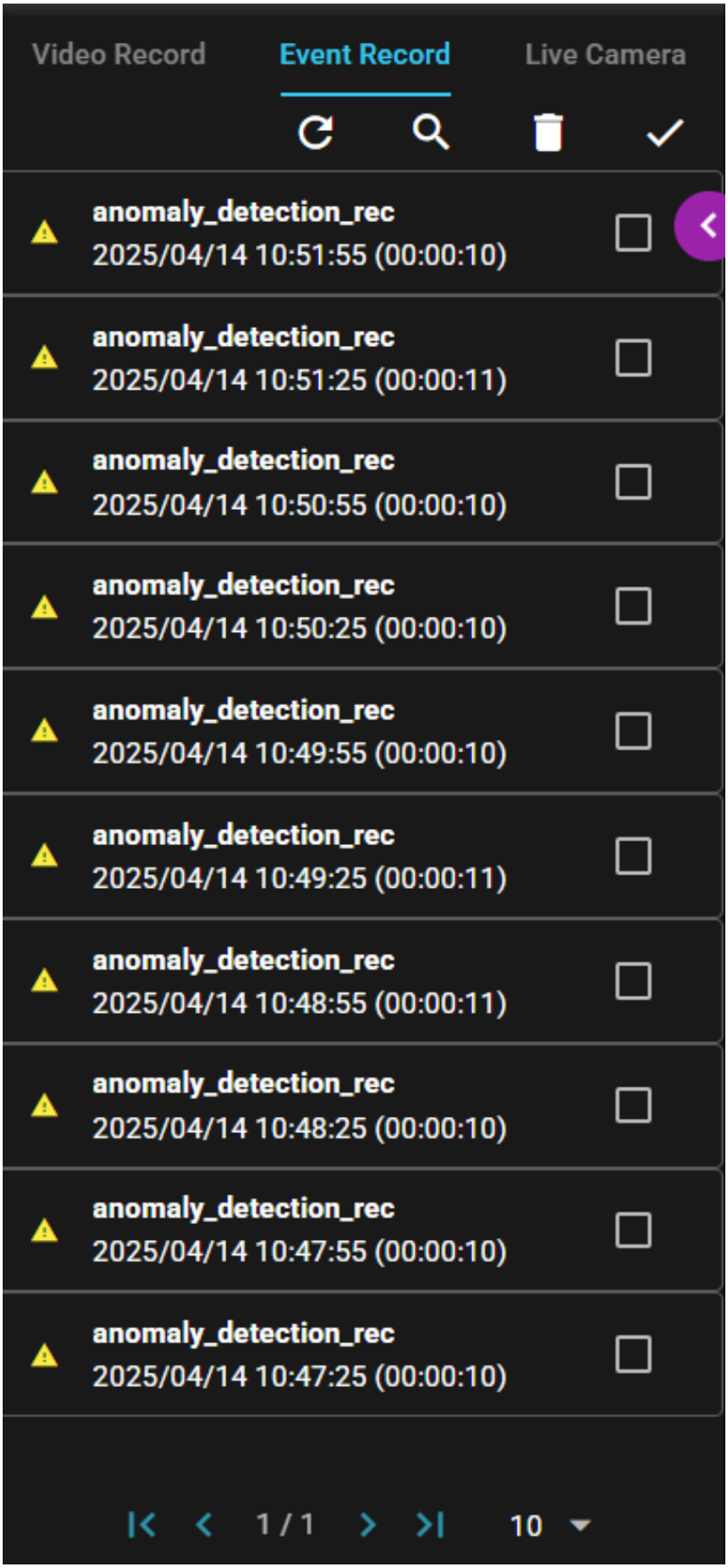
2. Verify that the placed component starts normally. If there are errors in the component lamps, check and, if necessary, revise the settings.



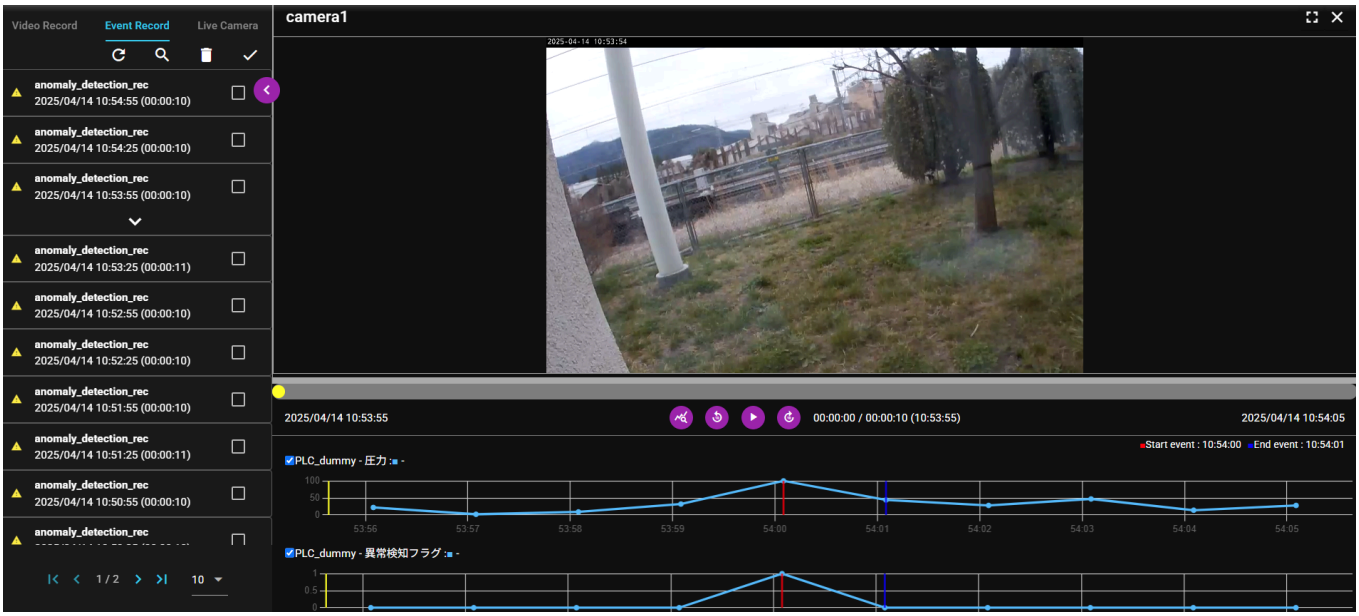
3. After running for more than a few minutes, check the Video Viewer. Click Video Viewer Icon.



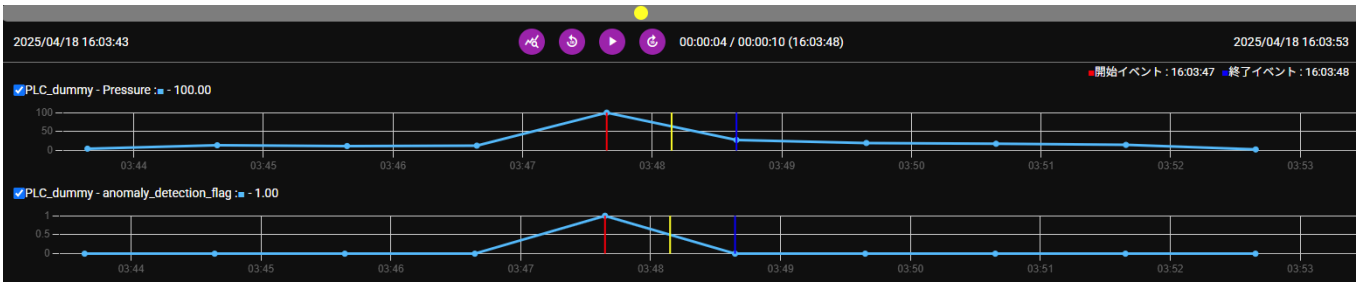
4. Click on Event Record in the left part of the Video Viewer. Verify that the anomaly_detection_rec appears in the list.



5. Click on any anomaly_detection_rec and verify that the General Video Viewer is displayed.



6. Click on the Play Icon to play back the event record. If the pressure is confirmed to be 100 at the timing when the error detection is set to 1 simultaneously with the event recording for approximately 10 to 11 seconds, the system is working normally.



7. This concludes the Tutorial.

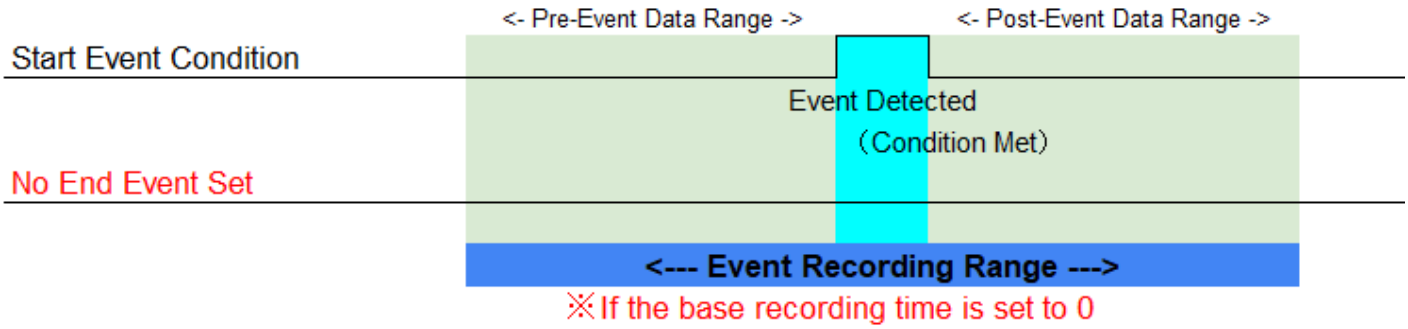
11.8 Recording Specifications for i-Trigger

This section explains the recording specifications for i-Trigger. The recording specifications are somewhat complex, so please refer to this section if you want to review the details.

11.8.1 When only Start Event is configured in Event Trigger

When only the start event is configured in an event trigger, the record interval is considered as follows.

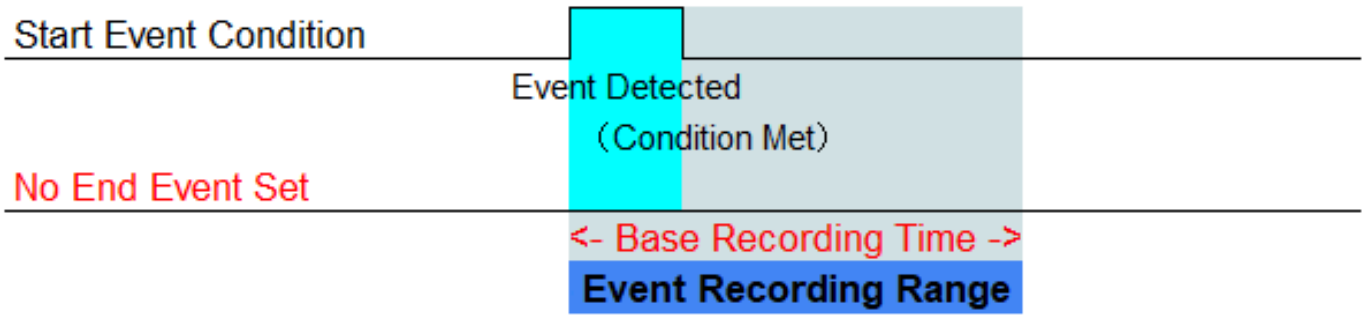
Pattern 1



- Basic record time is set to 0 minutes.
- Specify 5 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends, respectively.
- The start event condition will not be satisfied consecutively unless the condition is satisfied once as a Raise and then the condition will no longer be satisfied.

The range recorded in this pattern is the Event Record Range (blue background section). Since the record time before and after the event occurs is set to 5 seconds, the event is recorded for approximately 10 seconds.

Pattern 2

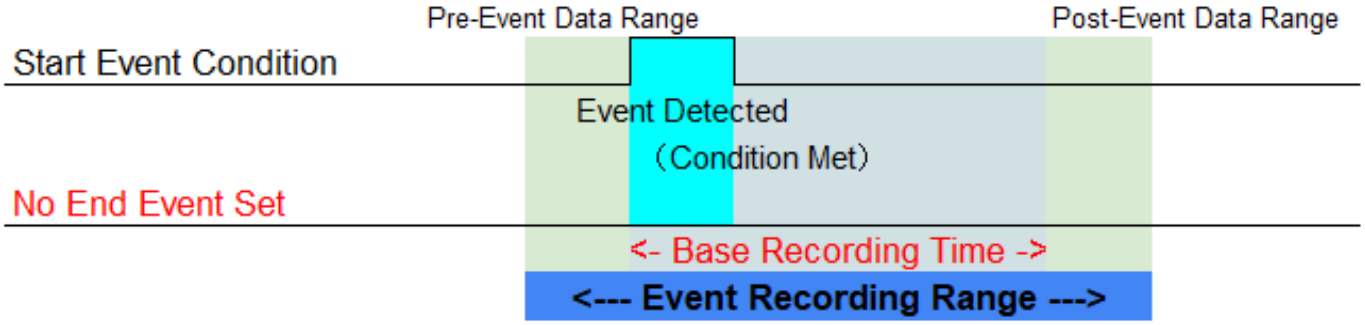


- Basic record time is set to 1 minute.

- Specify 0 seconds for the range to be collected before the event occurs and 0 seconds for the range to be collected after the event ends, respectively.
- The start event condition will not be satisfied consecutively unless the condition is satisfied once as a Raise and then the condition will no longer be satisfied.

The range recorded in this pattern is the Event Record Range (blue background section). The event record range is approximately 1 minute after the event occurred.

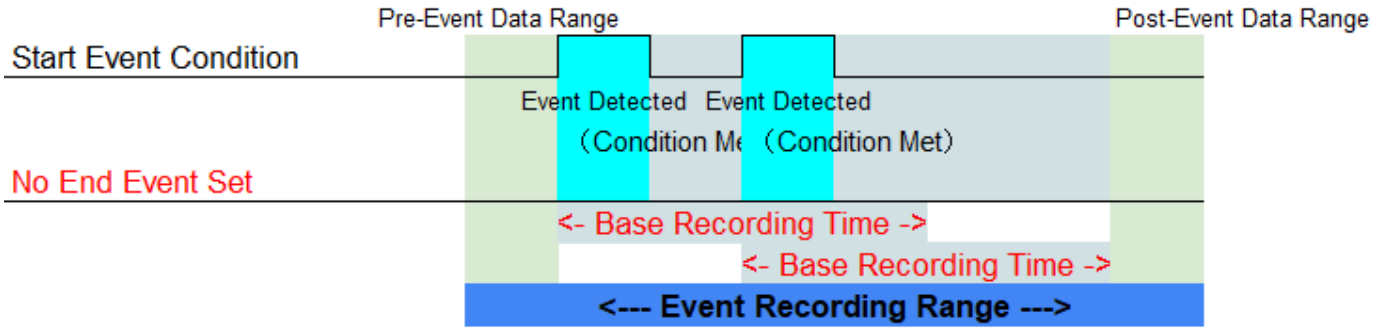
Pattern 3



- Basic record time is set to 1 minute.
- Specify 5 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends, respectively.
- The start event condition will not be satisfied consecutively unless the condition is satisfied once as a Raise and then the condition will no longer be satisfied.

The range recorded in this pattern is the Event Record Range (blue background section). In this case, the event record time is approximately 70 seconds, since the range before and after the event occurs is recorded in addition to the basic record time.

Pattern 4 (Event Extension)



- Basic record time is set to 1 minute.
- Specify 5 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends, respectively.

- The start event condition will not be satisfied consecutively unless the condition is satisfied once as a Raise and then the condition will no longer be satisfied.
- Thirty seconds after the first event is established, the event is established again.

The range recorded in this pattern is the Event Record Range (blue background section). If the event occurs again in the section of the basic record time, the event is extended. In this case, the section in which the first event is established is 30 seconds, +and the basic record time (60 seconds) after the second event is established, +and 10 seconds before and after the occurrence, thus the event record time is approximately 100 seconds.

Event Extension

There is a maximum limit of 15 minutes for extending an event. If an event is extended for 15 minutes or longer, the subsequent event is recorded as a new event.

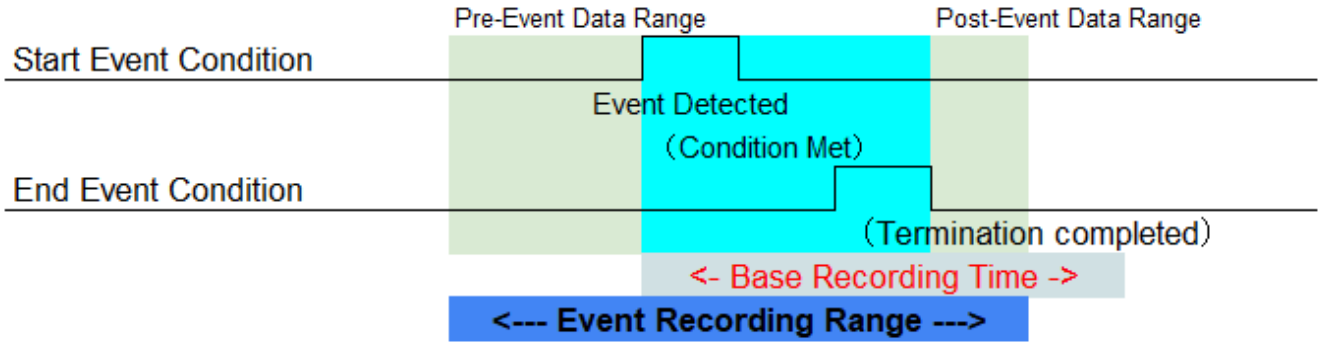
Event Condition Type

For all of the events described so far, the condition type of the event is Raise. In this case, the condition is satisfied when the value raises, so the behavior of the pattern is as described. Note that if you have set the condition type to True or some other condition, the event will always be extended while the condition is satisfied.

11.8.2 When Start Event and End Event are configured respectively in Event Trigger

When a start event and an end event are configured respectively in an event trigger, the record interval is considered as follows. Basically, it is not much different from the case where only the start event is configured.

Pattern 5

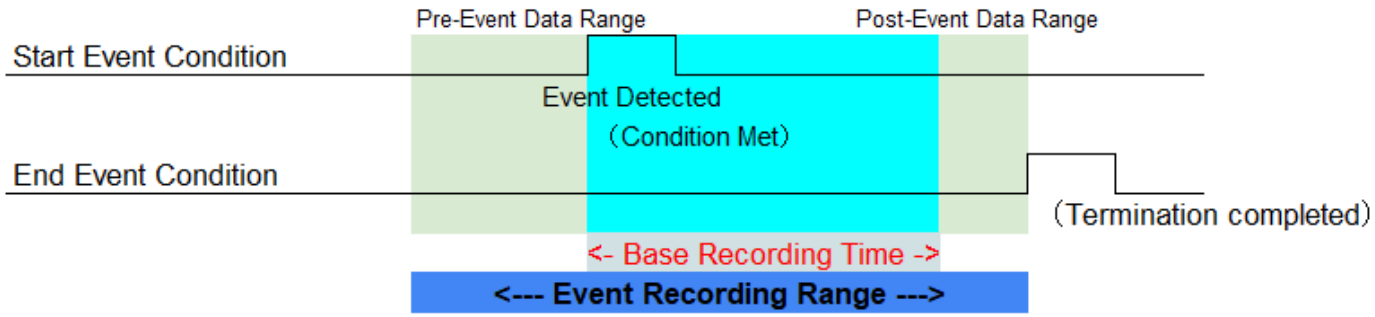


- Basic record time is set to 1 minute.
- Specify 10 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends.

- The start event condition and end event condition will not be satisfied consecutively unless the conditions are satisfied once as a Raise and then the conditions will no longer be satisfied.
- Thirty seconds after the start event is established, the end event is established.

The range recorded in this pattern is the Event Record Range (blue background section). In addition to the 10 seconds before the event occurs +and the 5 seconds after it ends, the time from the start of the event to its end is 30 seconds, thus the event record time is approximately 45 seconds. Since the end event is established before the basic record time is reached, the record time is less than 1 minute.

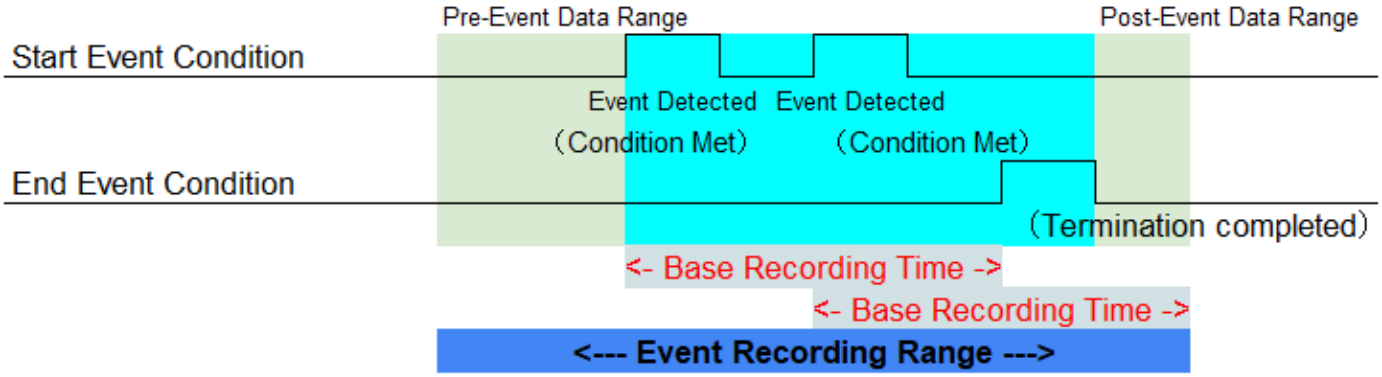
Pattern 6



- Basic record time is set to 1 minute.
- Specify 10 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends.
- The start event condition and end event condition will not be satisfied consecutively unless the conditions are satisfied once as a Raise and then the conditions will no longer be satisfied.
- The end event has not been established even after 1 minute has elapsed after the start event is established.

The range recorded in this pattern is the Event Record Range (blue background section). In addition to the 10 seconds before the event occurs +and the 5 seconds after it ends, time elapses from the start of the event until the basic record time is reached, thus the event record time is approximately 75 seconds. Since the end event has not been established within the basic record time, the record is aborted. In this case, the end event is ignored even if it is established after the record interval has been exceeded.

Pattern 7 (Event Extension)



- Basic record time is set to 1 minute.
- Specify 10 seconds for the range to be collected before the event occurs and 5 seconds for the range to be collected after the event ends.
- The start event condition and end event condition will not be satisfied consecutively unless the conditions are satisfied once as a Raise and then the conditions will no longer be satisfied.
- Thirty seconds after the start event is established, the start event is established again, and after 35 seconds have elapsed from the time of the establishment, the end event is established.

The range recorded in this pattern is the Event Record Range (blue background section). In addition to the 10 seconds before the event occurs +and the 5 seconds after it occurs, time elapses for 30 seconds in the section where the first event is established, and an additional 35 seconds after the second event is established, thus the event record time is approximately 80 seconds. Since the start event is established again within the basic record time, the basic record time is extended.

Event Extension

There is a maximum limit of 15 minutes for extending an event. If an event is extended for 15 minutes or longer, the subsequent event is recorded as a new event.

Event Condition Type

For all of the events described so far, the condition type of the event is Raise. In this case, the condition is satisfied when the value raises, so the behavior of the pattern is as described. Note that if you have set the condition type to True or some other condition, the event will always be extended while the condition is satisfied.

12 Release Notes

This chapter is the release notes for SpeeDBee Synapse.

12.1 v.4.11.4 (2026/4/17)

- Changes and Improvements
  - Updated the version of links to the official Grafana documentation in the manual to the version used in the examples.
  - Added information to the manual about characters that cannot be used in column names.
  - Added an explanation to the manual about the "Window" criterion for switching between JSON/CSV serializers.
  - Added an explanation to the manual about protocols that support "Auto address setting" in the PLC Collector and PLC Writing features.

12.2 v.4.11.2 (2026/3/31)

- Changes and Improvements
  - Added download function for Development files.
  - Added byte swap function for STRING type data in PLC Collector and PLC Writing.
  - Improved EtherNet/IP Collector and EtherNet/IP Writing so that hexadecimal values (e.g., 0xFF) in CIPObject settings are displayed as entered after saving.
  - Added display of numeral notation (DEC/HEX) in address input fields for PLC Collector and PLC Writing.
  - Increased the upper limit of Instance ID and Attribute ID in CIPObject settings for EtherNet/IP Collector and EtherNet/IP Writing to 65535 (0xffff).
  - Added warning dialog in Grafana dashboard settings to notify that manually edited content in Grafana will be overwritten.
  - Added "RTU over TCP" as a connection method for Modbus Collector and Modbus Writing.
  - Replaced Windows installer image in the English manual with an English version.
  - Changed the default value of "Basic record time(min)" in Event Trigger to 0.
  - Increased the upper limit of the acquisition range before and after event occurrence in In port setting to 300.
  - Added a mechanism to prevent the management directory from becoming incomplete during initial setup.
- Fixed Issues
  - When using the SQL statement aggregation function in the DB query component, the calculation result for ranges where there is no data may be 0.
  - When performing an overwrite installation of Synapse for Windows, the same Python dependency module may appear multiple times with different versions.

- Even when the changes to a component have not been applied, the icon indicating unapplied changes may disappear.
- When creating an event with an Event Recorder registered by copy & paste, the event name may remain the same as the source, resulting in duplicate event names.
- During project import, the API may be locked for a long time while installing Python dependencies.
- Even if you execute application webui log deletion from server settings data deletion, the logs may not be deleted.
- The process to supplement missing event videos did not work correctly on the Windows version of Synapse.

12.3 4.11.1 (2026/2/27)

- Changes and Improvements
  - Overall performance improvement of the Web screen
  - Support for displaying images in file-type columns in Data Monitor
  - Addition of function to hide specific component bases in Component Base List
  - Changed window width setting unit from 1 second to 50 milliseconds in Input Port Settings Aggregation
  - Support for encrypted communication in MQTT Client
  - Enable retry settings in Cloud Emitter for AWS
  - Support for data aggregation function in JSON Deserializer
  - Support for license activation via license file
  - PLC Collector and PLC Writing support "Panasonic MEWTOCOL".
- Fixed Issues
  - Corrected typos and omissions in the manual.
  - Updated used-OSS list in the manual
  - Fixed incorrect status lamp display in VIDEO Collector
  - Error occurs when sending/receiving binary data in MQTT Client
  - Buffer overflow may occur when using certain protocols in PLC Collector
  - TLS versions other than TLS1.2 cannot be used in MQTT Collector and Cloud Emitter (for AWS, for Azure)
  - Calculation processing may be performed even if no input data is received in the Compute Collector
  - Large amounts of database logs may be generated when the Compute Collector is running
  - If the network is disconnected while the EtherNet/IP Collector is running, the connection may not be restored even after recovery
  - Structure padding is not taken into account in EtherNet/IP writing
  - STRING type cannot be written in a structure in the EtherNet/IP writing
  - The correct value is not written when the value type is "Component" in EtherNet/IP writing

- Unsigned integers may not be written in EtherNet/IP writing
- Core processes may continue to run even after the Synapse service is stopped

12.4 v.4.10.1 (2025/12/18)

- Fixed Issues
  - The `serve` subcommand does not work in SCCDE.
  - If communication failure continues when transmitting persisted serializer output from an emitter, data may be lost even within the persistence period.
  - Existing Modbus Collector and Modbus Writing may fail to start after updating from v.4.9.9 or earlier to v.4.10.0.

12.5 v.4.10.0 (2025/11/28)

- Changes and Improvements
  - Overall revamp of Linking with Grafana feature.
  - Addition of List Mode feature.
  - Addition of General JSON Serializer component.
  - Addition of JSON Deserializer component.
  - Addition of MQTT Client component.
  - Addition of Auto-Install of Dependent modules feature for SCCDE Package(SCCPKG) registration.
  - Support multilingual definition of SCCDE package information file(scc-info.json).
  - Custom UI(HTML) supports multiple language switching.
  - PLC Collector and PLC Writing support "MITSUBISHI MELSEC 3C-Frame" and "KEYENCE Upper-link".
  - Modbus Collector and Modbus Writing support "RTU".
- Fixed Issues
  - Corrected typos and omissions in the manual.
  - The EtherNet/IP collector cannot handle strings longer than 255 bytes correctly.
  - Exporting a project may fail in the case of large number of python dependent modules.
  - When a file cannot be read from the input file column in the FTP emitter, the file is repeatedly retried for resending.

12.6 v.4.9.11 (2025/12/9)

- Fixed Issues

- Corrected typos and omissions in the manual.
- The EtherNet/IP collector cannot handle strings longer than 255 bytes correctly.
- Exporting a project may fail in the case of large number of python dependent modules.
- When a file cannot be read from the input file column in the FTP emitter, the file is repeatedly retried for resending.
- The save path is truncated in the settings screen of the VIDEO Collector, File Emitter, and Event Recorder.
- Support files may not include DB log.
- Protected custom components were included in the downloaded support file.

12.7 v.4.9.10 (2025/11/11)

- Changes and Improvements
  - Support for acquiring variables with names starting with `_` in EtherNet/IP Collector.
  - Added characters that cannot be used in column names when creating columns in custom components to the manual.
  - Removed information about unsupported data types from custom component Python-API reference.
  - Improved JavaScript code for custom UI (HTML) samples using Vue and Quasar.
- Fixed Issues
  - Corrected typos and omissions in the manual.
  - When using FTP Emitter, outputting the execution license configuration file without confidential information will result in failure.
  - Even if persistence is enabled in the system port settings, it reverts to OFF when the settings are reopened.
  - The system administrator account cannot modify the permissions of certain users.
  - When using simulation mode, the component status lamp may not turn blue.
  - When changing the individual log settings, some parameter information of custom components may be lost.
  - Starting the component may fail if a component is given a name in Japanese with 14 or more characters.
  - Even after changing a component's name and restarting it, the alias name used by DB Query are not updated.
  - On some platforms, multibyte characters in DB Query responses are escaped as `\uXXXX`.

12.8 v.4.9.9 (2025/9/12)

- BlackBear OS v3.0.10 support.
- Fixed Issues
  - Unused columns of the output port may not be invalidated, resulting in excessive memory allocation.
  - The process to delete actual files in file-type columns may not work.

12.9 v.4.9.8 (2025/9/10)

- Changes and Improvements
  - Removed the 10,000 limit on the number of response data items in DB Query, and improved memory usage efficiency.
- Fixed Issues
  - When the configuration file for the execution license is loaded, certain project information may not be inherited, which could result in multiple components failing to start.

12.10 v.4.9.5 (2025/9/2)

- Changes and Improvements
  - Improved behavior when accessing web pages via https reverse proxy.
  - Improved the disk usage information on the web screen to display even if tmpfs is set as the data directory in the general settings.
  - Added supplementary information about Data Flow Controller and OMRON-SBS for Windows to the manual.
  - Parameter screens created with custom HTML in the SCCDE package can now be displayed even with operator privileges.
- Fixed Issues
  - Event Recording DB files may not be included when downloading support files.
  - If the disk becomes full while Event Recorder is running, the event recording data becomes inconsistent.
  - When using the custom UI functionality for custom components using HTML, if there is a large amount of parameter information, data transfer to the custom HTML may fail.
  - When the group-based retention period setting is OFF, the retention time for file-type data output by the serializer becomes extremely short.
  - Using custom HTML in an SCCDE package, applying package protection will prevent the settings from being displayed.

12.11 v.4.9.4 (2025/8/28)

- Fixed Issues
  - Sending an SQL query containing special characters such as line breaks in a DB query may cause the core to stop or restart.
  - If the data directory is changed in the general settings and the data directory becomes full, inconsistencies may occur in the event recording data list.

12.12 v.4.9.3 (2025/8/25)

- Fixed Issues
  - In the case of multiple conditions are set for the same column in the Event Trigger, Event Data, Basic Statistics, Moving Average, or FFT components, a deadlock may occur when the component starts.
- Manual Enhancement
  - Corrected the installation manual and troubleshooting instructions.

12.13 v.4.9.2 (2025/8/18)

- Fixed issue of that memory may be exhausted when executing a DB query to retrieve a large amount of data.

12.14 v.4.9.1 (2025/8/8)

- Changes and Improvements
  - The latest UI design has been reflected in the manual's screenshots.
  - Bash-dependent syntax has been removed from the Linux installation setup script.
- Fixed Issues
  - Memory may be exhausted when executing a DB query to retrieve a large amount of data.

12.15 v.4.9.0 (2025/7/31)

- Changes and Improvements
  - Added proxy support for Cloud Emitter for Azure, Cloud Emitter for AWS, MQTT Collector and MQTT Emitter.
  - Added proxy support when installing dependent modules of Python custom components.
  - The Conversion Configuration function supports inputting formulas to convert BCD format data to normal numbers.
  - UI improvements for Storage Configuration, Conversion Configuration and Registration Condition in OUT Port setting.
  - UI improvements to allow IN port settings, OUT port settings, System port settings, and Flowlink settings to be configured without right-clicking.
  - Added support for Mitsubishi MELSEC iQ-R series in PLC Collector and PLC Writing.
- Fixed Issues
  - Memory may be exhausted when downloading large support file.
  - On Windows, when upgrading from v4.7.2, some component configuration data may not be carried over.
  - Access tokens issued after upgrading from v4.7.2 cannot be used with DB Query.

- Adding an extra string to the API key in the request header for the DB Query will result in an error.

12.16 v.4.8.0 (2025/6/30)

- Changes and Improvements
  - Addition of Project Features
  - Addition of English version of manual
  - Addition of size cap setting for time series database (where data collected by Synapse is stored)
  - Supports for log display and log level switching for individual components
  - Addition of Data Conversion Component
  - Improved destination settings for VIDEO collector, File emitter and Event Recorder
  - Addition of ability to delete logs, temporary files, etc.
  - EtherNet/IP collector supports acquisition of `SHORT STRING` type data
  - Addition of a link icon to the manual in each component's setting screen
  - Change of collection interval units for PLC Collector and ModbusTCP from seconds to milliseconds
  - Addition of custom UI functionality for custom components via HTML
  - Support for SQL INSERT syntax in DB query component
  - Improved touch panel control related operations in the flow editor (iPad +Safari)
- Fixed Issues
  - Attachment is not available for custom component parameter settings (Windows version only).
  - PLC/MODBUS/EtherNetIP collector with collection interval less than 100 ms keeps memory shorter than the memory retention period set by the server.
  - When the type of the target data is changed in the input port setting, output port setting or filter setting, the data is excluded from the target data.
  - On Windows, the EtherNet/IP Collector does not automatically reconnect after the network is restored following a disconnection during execution.
  - The mail sending component continues running even if an authentication error occurs, instead of stopping.
  - The timeout value configured in the MQTT Emitter is not applied during the test connection.
- Internal change
  - Change of internal data directory structure due to addition of project function
  - Performance improvements when starting components that create large numbers of columns
  - Improvement of Web screen operation performance when creating many components

12.17 v.4.7.6 (2025/3/19)

- Changes and Improvements
  - EtherNet/IP writing component supports writing to CIPObject.
  - Addition of Custom Component Protection Functions
  - Addition of Synapse custom component development environment
  - Addition of a function to notify on the screen when an abnormality is detected in the system time in which Synapse operates.
  - Addition of Simulation Modes
- Fixed Issues
  - EtherNet/IP Collector may fail to acquire CIPObject STRING Type.
  - When the same variable name is specified more than once by the EtherNet/IP collector, some cannot be obtained.
  - An error occurs if a null value is received by the MQTT collector.
- Revising the chapter structure of the manual
- Internal change
  - Improved data transition process from previous versions
  - Changing the Directory Configuration of Internal Data

12.18 v.4.7.5 (2025/3/13)

- Fixed Issues
  - Installation of BlackBear version fails under certain conditions

12.19 v.4.7.3 (2025/2/1)

- BlackBear support
- Changes and Improvements
  - Virtual interfaces are excluded from MAC addresses used for license matching.

12.20 v.4.7.2 (2025/1/31)

- Update v.4.7. 0 Changes to Other System Environments
- Fixed Issues
  - After searching for components in the Flow Editor, components selected from the search field are not displayed in the center of the screen.

12.21 v.4.7.0 (2024/12/20)

Target System Requirements: Ubuntu22 (arm64)

- Changes and Improvements
  - Improved tooltip display for component based list
  - Support for multiple selection of components in the Flow Editor
  - Improved context menu for components in the Flow Editor
  - Addition of General Settings Dialog (Time Zone Settings Only)
  - Support for rearranging and duplicating panels
  - Addition of variables available in the email sending component
  - Addition of settings for the time stamp assignment method in the PLC collector, EtherNet/IP collector and Modbus TCP collector
  - Improved component behavior when custom collector fails to be imported
- Fixed Issues
  - FTP Emitter fails to save file to specific FTP Server
  - Invalid operation result is outputted when data inputted byCompute Collector is BOOL type.

12.22 v.4.6.2 (2024/10/17)

Supported system requirements: Ubuntu20 (amd64), Ubuntu22(amd64, arm64), RaspberryPiOS (bookworm 64bit), Windows10 (64bit 22H2 or later), Windows11 (64bit 23H2 or later), Special environments (Arm32 bit)

- Changes and Improvements
  - Improved error notification in case of transmission failure in email sending component
- Fixed Issues
  - Cloud Emitter for Azure, Cloud Emitter for AWS and FTP/SFTP Emitter fail to start under certain conditions
  - An error occurs when some received data is missing in the EtherNet/IP writing component.
  - When RAW_DATA is OFF and aggregation is performed in the tumbling window of the input port setting, the aggregation result with the specified width is not obtained.
  - EtherNet/IP writing fails to write some structures

12.23 v.4.6.0 (2024/9/30)

Supported system requirement: Windows 10 (64bit version 22H2 or later), Windows 11 (64bit version 23H2 or later)

- Windows Support
- Changes and Improvements

    - Changed the maximum number of characters that can be entered in the destination to 200 in the VIDEO collector.
    - The directory structure of the C Custom Components sample is changed.
    - Improved the ability to specify the URL to be set as a Grafana data source from the Synapse screen.
    - In the EtherNet/IP collector, if a BOOL type array is specified only as a variable name, the first element of the tuple is registered if the number of arrays is 0 or 1.
- Fixed Issues
    - In Compute Collector, if a calculation expression triggered by multiple input data is set, the collector stops abnormally.
    - In Compute Collector, if only one trigger is set when multiple input data are operated, the operation result becomes invalid.
    - SFTP emitters may place files in a different directory than intended.
    - SSH connections continue to grow while SFTP Emitter Is running.
    - MQTT Emitter may drop messages at the time when network disconnect -> resume, even with QoS 1 or 2.

12.24 v.4.5.1 (2024/9/20)

Target system requirements: Special environments (Arm32 bit)

- Added retrieval of data by specifying array element to EtherNet/IP collector
- Fixed Issues
    - EtherNet/IP writing cannot write BOOL arrays
    - EtherNet/IP collector cannot read BOOL arrays
    - Changing event trigger settings may cause triggers not to run.
    - Long time EtherNet/IP collector data acquisition causes the cycle to be out of sync.

12.25 v.4.5.0 (2024/9/6)

Target system requirements: Special environments (Arm32 bit)

- Addition of OMRON Sysmac NJ/NX for PLC collector and PLC writing
- Addition of prototype for EtherNet/IP Writing
- Fixed Issues
    - If array data enters the input port of a python component, it will all have the same value as the 0th element.
- Internal change
    - WebAPI address setting is changed to 127.0.0.1.
    - Change of the internal behavior of the core when changing ports

- Obsolete settings of system_config are removed.
- Change of event trigger behavior
- Addition of Port Number check

12.26 v.4.4.0 (2024/7/31)

Supported system requirements: Ubuntu20 (amd64), Ubuntu22(amd64, arm64), RaspberryPiOS (bookworm 64bit)

- Addition of the function to select the target panel and whether to start automatically when exporting the execution license setting output
- Addition of the latest value fetching function to Python-API for custom components
- Addition of address/port setting items used by WebUI to HTTP/HTTPS communication setting screen
- Displays MAC address information for license judgment on the software information screen.
- Improved error display when an invalid license key is entered on the software information screen
- Fixed Issues
  - Core ends abnormally when Get pre-reboot data is ON in status timeline.
  - When certain components are connected to a CSV/JSON serializer, the serializer hangs with an error.
- Documentation improvement
  - Additional notes on Python custom component filenames
  - Additional noteson Modbus address ranges
  - Proofreading (Correction of typographical errors and omissions, improvement of sentence expressions that are difficult to understand, etc.)
- Internal change
  - Core library configuration changes
  - Improved logging and error determination for installation scripts
  - Change of the default value for the output interval of persistent data (from 1 second to 20 seconds)

12.27 v.4.3.0 (2024/8/2)

Target system requirements: Special environments (Arm32 bit)

- Ported to special environments with some modifications since 4.2

12.28 v.4.2.2(2024/6/28)

Supported system requirements: Ubuntu20 (amd64), Ubuntu22(amd64, arm64), RaspberryPiOS (bookworm 64bit)

- Addition of Synapse System Requirements: RaspberryPiOS (bookworm 64bit)
- Addition of i-Trigger-related functions
  - VIDEO Collector
  - Event Recorder
  - Video Playback Viewer
- Shortest data acquisition cycle of PLC collector reduced from 100 ms to 10 ms
- Addition of AWS IoT Core (Custom Authentication) Settings to MQTT Collector
- Procedure for receiving data from a cloud service with the MQTT collector is added to the manual.
- Correction of defects
  - Improved operation when disk space is insufficient
  - Correction of the defect where Python external modules for custom components are removed at reinstallation time of Synapse

12.29 v.4.1.0 (2024/6/17)

Target System Requirements: Ubuntu22 (arm64)

- Addition of the Execution License Function
- Support for Grafana service account functionality in Grafana linkage
- Support for registering and installing whl files from the custom component configuration screen to run Python custom components
- Support for defining nested data structures in custom UI
- Support for setting screen size specification in custom UI
- Proofreading for SpeeDBee Synapse (Correction of typographical errors and omissions, improvement of sentence expressions that are difficult to understand, etc.)
- Correction of defects
  - Improved operation when disk space is insufficient
  - Improved display of error information in status timeline
  - Correction of a bug that prevents a persisted database from being deleted after its retention period under certain conditions
  - Correction of defect where, in PLC Collector MELSEC protocol, writing 0 to bit register is prevented
  - Correction of a bug that prevented disabling the admin user
  - Correction of a bug that caused the system to restart when using the PLC collector FINS protocol
  - Correction of a defect that causes access to an address different from the specified address when using the ZR register of the PLC collector MELSEC (3E) protocol
  - Improved processing at core process startup

12.30 v.4.0.2 (2024/4/1)

Supported system requirements: Ubuntu20 (amd64), Ubuntu22(amd64, arm64)

- We have released a new product "SpeeDBee Synapse" which is an evolution of SpeeDBee Hive.

SpeeDBee and SpeeDBee Synapse are registered trademarks of Saltyster Inc.
The “SpeeDBee Synapse” logo is a trademark pending.

Microsoft Azure, Microsoft Defender and Windows are either registered trademarks or trademarks of Microsoft Corporation in Japan, the United States and/or other countries.

Other company and product names mentioned here are the trademarks or registered trademarks of their respective companies.
Or,
Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Contact the vendor from which you purchased the product (SpeeDBee Synapse).

15 Appendix

15.1 Appendix

1. Error Code List
2. Custom Component Python-API Reference
3. Custom Component C-API Reference
4. Custom UI JS-API Reference
5. Open Source Used
6. Index
7. DB Query Service Web-API Specification

15.2 Error code list

This section provides a list of error identifiers that are displayed in the Status Timeline, including the following items for each identifier.

Item	Description
Message	Message assigned to the error identifier
Content	Details of the error
Action	How to resolve the error

15.2.1 Component default

15.2.1.1 NO_MEMORY

Item	Description
Message	Low memory
Content	Not enough memory on the machine running SpeeDBee Synapse.
Action	Stop components not in use. In addition, make sure that unnecessary data collection and other wasteful settings are not configured. If the error still persists, consult your System Administrator and add more memory or to adjust with other applications.

15.2.1.2 PARAMETER_PARSE_ERROR

Item	Description
Message	Parameter parsing error
Content	The configuration file has not been saved in the correct format. The settings may be corrupted.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.3 PARAMETER_FORMAT_ERROR

Item	Description
Message	Parameter error
Content	The configuration file contains incorrect parameters.
Action	In the Compute Collector, this error occurs if no variable or expression is set. In other components, it may also occur due to poor configurations, such as an incorrect Azure connection string. Check the configuration of the relevant component. If the error persists, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.4 FRAMEWORK_ERROR

Item	Description
Message	Framework api error
Content	An error occurred in the internal framework used by the relevant component.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.5 DISK_FULL

Item	Description
Message	Disk full
Content	Not enough storage space on the machine running SpeeDBee Synapse.
Action	You need to take some action, such as deleting unnecessary files. Consult your System Administrator and free up storage space.

15.2.1.6 IO_ERROR

Item	Description
Message	IO error
Content	Failed in I/O processing on the machine running SpeeDBee Synapse. (E.g., storage read/write failure)
Action	Consult your System Administrator and make confirmation on the storage status and the like. (E.g., possibility of failure)

15.2.1.7 NETWORK_ACCESS_ERROR

Item	Description
Message	Network access error
Content	A communication error occurred in the relevant component.
Action	(Action 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Action 2) Make sure that the device or service to which the relevant component communicates is enabled for communication.

15.2.1.8 PROCESSING_DELAY

Item	Description
Message	Processing delay
Content	Processing is delayed on the relevant component.
Action	(Action 1) Reduce the count of data items sent and received by the relevant component. (Action 2) Stop or remove unnecessary components that are running on SpeeDBee Synapse. (Action 3) Check whether no delay has occurred in the network of the machine running SpeeDBee Synapse.

15.2.1.9 INVALID_REGEX

Item	Description
Message	Regex error
Content	A setting of the relevant component has the input of a regular expression in an incorrect format.
Action	1. Open the configuration screen of the relevant component. 2. Check the settings to identify the one with the input of a regular expression in an incorrect format. 3. For the setting confirmed in step 2, enter a regular expression in the correct format, and save it. 4. Restart the relevant component.

15.2.1.10 COMPONENT_DATA_MISMATCH

Item	Description
Message	No data for the component matching the specified parameters Note: This message includes the following information. ▪ Component ▪ Data
Content	No data corresponds to the contents specified in the settings. In this state, no action is taken according to the relevant configuration.
Action	1. Confirm the component and data included in the message. 2. Right-click the IN Port of the relevant component. 3. Specify "Receive and set data definitions" for the "Setting option" setting, and check the data list. 4. Open the configuration screen of the relevant component. 5. For the setting with the data confirmed in step 1 specified, enter the data confirmed in step 3, and save it. Note: If you use a regular expression to specify data, enter the regular expression that contains at least one item of data confirmed in step 3. 6. Restart the relevant component.

15.2.1.11 STATISTICS_INIT_ERROR

Item	Description
Message	Statistical processing Initialization error

Item	Description
Content	Failed to initialize statistical processing.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.12 STATISTICS_INSERT_ERROR

Item	Description
Message	Statistics registration error
Content	Failed to register a statistic value.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.13 UNAVAILABLE_DATA_TYPE

Item	Description
Message	An invalid type was specified in the conditional expression Note: This message includes the following information. ▪ Component ▪ Data
Content	The specified type cannot be used for data in a conditional expression. For example, strings and arrays cannot be used in conditional expressions.
Action	1. Confirm the component and data included in the message. 2. Open the configuration screen of the relevant component. 3. For the setting with the data type confirmed in step 1 specified, specify the appropriate data type, and save it. 4. Restart the relevant component.

15.2.1.14 UNKNOWN_ERROR

Item	Description
Message	Unknown error
Content	An unknown error occurred.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.1.15 INTERNAL_ERROR

Item	Description
Message	Unexpected Error
Content	An error occurred that does not correspond to other error identifiers.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2 Resource Collector

15.2.2.1 CPUUSAGE_OPENREAD_ERROR

Item	Description
Message	CPU usage open error
Content	Unable to access the following file containing CPU usage information. /proc/stat
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.2 CPUUSAGE_FORMAT_ERROR

Item	Description
Message	CPU usage acquisition error
Content	Failed to retrieve CPU usage information from the following file. /proc/stat
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.3 CPUFREQ_OPENREAD_ERROR

Item	Description
Message	CPU frequency open error
Content	Unable to access the following file containing CPU frequency information. /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq
Action	The CPU frequency is not acquirable depending on your machine or platform. If the above file exists and this error occurs, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.4 CPUFREQ_FORMAT_ERROR

Item	Description
Message	CPU frequency acquisition error
Content	Failed to retrieve CPU frequency information from the following file. /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq
Action	The CPU frequency is not acquirable depending on your machine or platform. If the above file exists and this error occurs, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.5 CPUTEMP_OPENREAD_ERROR

Item	Description
Message	CPU temperature open error
Content	Unable to access the following file containing CPU temperature information. /sys/class/thermal/thermal_zone0/temp
Action	The CPU temperature is not acquirable depending on your machine or platform. If the above file exists and this error occurs, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.6 CPUTEMP_FORMAT_ERROR

Item	Description
Message	CPU temperature acquisition error
Content	Failed to retrieve CPU temperature information from the following file. /sys/class/thermal/thermal_zone0/temp
Action	The CPU temperature is not acquirable depending on your machine or platform. If the above file exists and this error occurs, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.7 IOSTAT_NOT_FOUND

Item	Description
Message	IO device error
Content	The following file containing IO information does not exist. /dev/{"I/O device" setting}
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.8 IOSTAT_OPENREAD_ERROR

Item	Description
Message	IO open error
Content	Unable to access the following file containing IO information. /dev/{"I/O device" setting}
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.9 IOSTAT_FORMAT_ERROR

Item	Description
Message	IO acquisition error
Content	Failed to retrieve IO information from the following file. /dev/{"I/O device" setting}
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.10 IOSTAT_STORAGE_MOUNT_POINT_ERROR

Item	Description
Message	IO unacquirable (unable to get mount point) Note: This message includes the following information. ▪ Device name
Content	Mount point information is not defined correctly in the configuration file.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.11 IOSTAT_STORAGE_INFO_ERROR

Item	Description
Message	IO unacquirable (unable to get information with statvfs) Note: This message includes the following information. ▪ Device name ▪ Mount point ▪ Status
Content	Failed to get IO information using the statvfs command for the mount point defined in the configuration file.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.12 MEM_OPENREAD_ERROR

Item	Description
Message	Memory open error
Content	Unable to access the following file containing memory information. /proc/meminfo
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.13 MEM_FORMAT_ERROR

Item	Description
Message	Memory acquisition error
Content	Failed to retrieve memory information from the following file. /proc/meminfo
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.14 NETWORK_OPENREAD_ERROR

Item	Description
Message	Network open error
Content	Unable to access the following file containing network information. /proc/net/dev
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.2.15 NETWORK_FORMAT_ERROR

Item	Description
Message	Network acquisition error
Content	Failed to retrieve network information from the following file. /proc/net/dev
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.3 PLC Collector

15.2.3.1 CONNECTION_REFUSED

Item	Description
Message	Disconnected
Content	The PLC has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the configuration on the PLC side for the host and port capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component.

15.2.3.2 NETWORK_TIMEOUT

Item	Description
Message	Timeout Note: This message includes the following information. ▪ Address ▪ Port number ▪ Setting timeout value ▪ Operation type
Content	(When the operation type is connect or close) Failed to connect the PLC within the time specified for the "Connect timeout" setting. (When the operation type is receive or send) Failed to receive data from the PLC within the time specified for the "Read timeout" setting.
Action	1. Confirm the operation type included in the message. 2. Open the configuration screen of the relevant component. 3. (When the operation type confirmed in step 1 is connect or close) Increase the "Connect timeout" setting. (When the operation type confirmed in step 1 is receive or send) Increase the "Read timeout" setting. 4. Restart the relevant component.

15.2.3.3 SERIAL_CONNECTION_REFUSED

Item	Description
Message	Disconnected (Serial Communication)
Content	The PLC has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the PLC settings for the correct "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit". 2. Open the configuration screen of the relevant component. 3. Enter and save the values confirmed in step 1 for the "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit" settings. 4. Restart the relevant component.

15.2.3.4 SERIAL_NETWORK_TIMEOUT

Item	Description
Message	Timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Failed to receive data from the PLC within the time specified for the "Read timeout" setting.
Action	1. Open the configuration screen of the relevant component. 2. Increase the "Read timeout" setting. 3. Restart the relevant component.

15.2.3.5 SERIAL_NETWORK_ACCESS_ERROR

Item	Description
Message	Communication error Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Error number ▪ Operation type
Content	(When the operation type is connect) Failed to connect to the PLC. (When the operation type is receive or send) Failed to send or receive data to/from the PLC.
Action	1. Confirm the operation type included in the message. (If the operation type confirmed in step 1 is connect) 2. Open the configuration screen of the relevant component. 3. Check the PLC settings for the correct "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit", and enter and save the confirmed settings. 4. Restart the relevant component. (If the operation type confirmed in step 1 is receive or send) 2. Check for any connection issues with the PLC.

15.2.3.6 SERIAL_LOCK_TIMEOUT

Item	Description
Message	Connection exclusive control timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Could not connect to the PLC within the time specified for the "Connect timeout" setting because another component using the same device is currently connected.
Action	1. Open the configuration screen of the relevant component. 2. Increase the value of the "Connect timeout" setting. 3. Restart the relevant component.

15.2.3.7 REGISTER_ADDRESS_ERROR

Item	Description
Message	Specified register is invalid Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address not supporting bit specification has been entered in the "Address" setting with bit specification.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that supports bit specification, and save it. 4. Restart the relevant component.

15.2.3.8 REGISTER_ADDRESS_RANGE_ERROR

Item	Description
Message	Incorrect register address Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address that cannot be handled by the PLC protocol has been entered in the "Address" setting.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that can be handled by the PLC protocol, and save it. 4. Restart the relevant component. Note: Refer to the manual of the relevant device and others for usable addresses and their range.

15.2.3.9 REGISTER_ADDRESS_DATA_TYPE_ERROR

Item	Description
Message	Incorrect data type Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A data type different from the configuration on the PLC side has been specified for the "Data type" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the PLC configuration for the data type of the register confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Data type" setting of the register confirmed in step 1, specify the data type confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.3.10 REGISTER_ADDRESS_SIZE_ERROR

Item	Description
Message	Incorrect number of contiguous regions Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	The number of contiguous regions different from the configuration on the PLC side has been entered for the "Num array" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the PLC configuration for the number of contiguous regions confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Num array" setting of the register confirmed in step 1, enter the number of contiguous regions confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.3.11 REGISTER_READ_ERROR

Item	Description
Message	Register acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed to get the register.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.3.12 GROUP_REGISTER_READ_ERROR

Item	Description
Message	Register batch acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed in register batch acquisition.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.4 Modbus Collector

15.2.4.1 CONNECTION_REFUSED

Item	Description
Message	Disconnected
Content	The device has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the configuration on the device side for the host and port capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component.

15.2.4.2 NETWORK_TIMEOUT

Item	Description
Message	Timeout Note: This message includes the following information. ▪ Address ▪ Port number ▪ Setting timeout value ▪ Operation type
Content	(When the operation type is connect or close) Failed to connect to the device within the time specified for the "Connect timeout" setting. (When the operation type is receive or send) Failed to receive data from the device within the time specified for the "Read timeout" setting.
Action	1. Confirm the operation type included in the message. 2. Open the configuration screen of the relevant component. 3. (When the operation type confirmed in step 1 is connect or close) Increase the "Connect timeout" setting. (When the operation type confirmed in step 1 is receive or send) Increase the "Read timeout" setting. 4. Restart the relevant component.

15.2.4.3 SERIAL_CONNECTION_REFUSED

Item	Description
Message	Disconnected (Serial Communication)
Content	The device has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the device settings for the correct "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits". 2. Open the configuration screen of the relevant component. 3. Enter and save the values confirmed in step 1 for the "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits" settings. 4. Restart the relevant component.

15.2.4.4 SERIAL_NETWORK_TIMEOUT

Item	Description
Message	Timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Failed to receive data from the device within the time specified for the "Read timeout" setting.
Action	1. Open the configuration screen of the relevant component. 2. Increase the "Read timeout" setting. 3. Restart the relevant component.

15.2.4.5 SERIAL_NETWORK_ACCESS_ERROR

Item	Description
Message	Communication error Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Error number ▪ Operation type
Content	(When the operation type is connect) Failed to connect to the device. (When the operation type is receive or send) Failed to send or receive data to/from the device.
Action	1. Confirm the operation type included in the message. (If the operation type confirmed in step 1 is connect) 2. Open the configuration screen of the relevant component. 3. Check the device settings for the correct "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits", and enter and save the confirmed settings. 4. Restart the relevant component. (If the operation type confirmed in step 1 is receive or send) 2. Check for any connection issues with the device.

15.2.4.6 SERIAL_LOCK_TIMEOUT

Item	Description
Message	Connection exclusive control timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Could not connect to the device within the time specified for the "Connect timeout" setting because another component using the same device is currently connected.
Action	1. Open the configuration screen of the relevant component. 2. Increase the value of the "Connect timeout" setting. 3. Restart the relevant component.

15.2.4.7 REGISTER_ADDRESS_ERROR

Item	Description
Message	Specified register is invalid Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address not supporting bit specification has been entered in the "Address" setting with bit specification.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that supports bit specification, and save it. 4. Restart the relevant component.

15.2.4.8 REGISTER_ADDRESS_RANGE_ERROR

Item	Description
Message	Incorrect register address Note: This message includes the following information.

Item	Description
	▪ Register name ▪ Register address ▪ Operation type
Content	A register address that cannot be handled by the device protocol has been entered for the "Address" setting.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that can be handled by the device protocol, and save it. 4. Restart the relevant component. Note: Refer to the manual of the relevant device and others for usable addresses and their range.

15.2.4.9 REGISTER_ADDRESS_DATA_TYPE_ERROR

Item	Description
Message	Incorrect data type Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A data type different from the configuration on the device side has been specified for the "Data type" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the configuration on the device side for the data type of the register confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Data type" setting of the register confirmed in step 1, specify the data type confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.4.10 REGISTER_ADDRESS_SIZE_ERROR

Item	Description
Message	Incorrect number of contiguous regions Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	The number of contiguous regions different from the configuration on the device side has been entered for the "Num array" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the configuration on the device side for the number of contiguous regions confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Num array" setting of the register confirmed in step 1, enter the number of contiguous regions confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.4.11 REGISTER_READ_ERROR

Item	Description
Message	Register acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed to get the register.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.4.12 GROUP_REGISTER_READ_ERROR

Item	Description
Message	Register batch acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed in register batch acquisition.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.5 EtherNet/IP Collector

15.2.5.1 TYPE_MISMATCH

Item	Description
Message	Type mismatch Note: This message includes the following information. ▪ Tag name ▪ Setting type ▪ Read type
Content	A data type different from the configuration on the device side has been specified for the "Data type" setting.
Action	1. Confirm the tag name, the setting type, and the read type included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Data type" setting of the tag confirmed in step 1, specify the read tag type confirmed in step 1, and save it. 4. Restart the relevant component.

15.2.5.2 UNKNOWN_TAG

Item	Description
Message	Specified tag unreadable Note: This message includes the following information. ▪ Tag name
Content	Data for the specified tag is unavailable. Note: This error occurs when the "Setting option" setting is "Variable".
Action	1. Confirm the tag name included in the message. 2. Check the configuration on the device side for the correct tag name. 3. Open the configuration screen of the relevant component. 4. For the "Variable name" setting of the tag confirmed in step 1, enter the tag name confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.5.3 CIP_SETTING_ERROR

Item	Description
Message	CIP object setting error Note: This message includes the following information. ▪ Class ID ▪ Instance ID ▪ Attribute ID
Content	Data for the specified CIP object unavailable. Note: This error occurs when the "Setting option" setting is "CIP object".
Action	1. Confirm the class ID, instance ID, and attribute ID included in the message. 2. Check the configuration on the device side for the correct class ID, instance ID, and attribute ID. 3. Open the configuration screen of the relevant component. 4. For the "Class ID", "Instance ID", and "Attribute ID" settings of the CIP object confirmed in step 1, enter the information confirmed in step 2, and save them. 5. Restart the relevant component. Note: Refer to the manual of the relevant device and others for accessible CIP objects.

15.2.5.4 ARRAYSIZE_MISMATCH

Item	Description
Message	Array size error Note: This message includes the following information. ▪ Column name
Content	The received data array size is less than the number of arrays specified for the "Array size" setting.
Action	1. Confirm the column name included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Array size" setting of the column confirmed in step 1, enter a smaller number, and save it. 4. Restart the relevant component.

15.2.5.5 READ_TIME_OVER

Item	Description
Message	Read time over Note: This message includes the following information. ▪ Time taken to load
Content	Reading data is taking longer than the interval specified for the "Get Interval" setting.
Action	1. Confirm the time taken to read data included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Get Interval" setting, enter a time longer than the time confirmed in step 1, and save it. 4. Restart the relevant component.

15.2.6 Compute Collector

15.2.6.1 EXPRESSION_FORMAT_ERROR

Item	Description
Message	Incorrect formatting of formulas Note: This message includes the following information. ▪ Output name
Content	An expression in an incorrect format has been entered in the "Expression" setting.

Item	Description
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.2 SOME_VARIABLE_NOT_SPECIFIED

Item	Description
Message	Some variables in the formula are not specified Note: This message includes the following information. ▪ Output name ▪ Variable name
Content	The "Expression" setting contains undefined variables.
Action	(Workaround 1) 1. Confirm the output name and variable name included in the message. 2. Open the configuration screen of the relevant component. 3. Add input data whose setting "Variable name" is the same as the variable name confirmed in step 1. 4. For the "Trigger variable" setting of the output data confirmed in step 1, specify the variable added in step 3, and save it. 5. Restart the relevant component. (Workaround 2) 1. Confirm the output name and variable name included in the message. 2. Open the configuration screen of the relevant component. 3. From the "Expression" setting of the output data confirmed in step 1, remove the variable confirmed in step 1, and save it. 4. Restart the relevant component.

15.2.6.3 NO_TRIGGER_SPECIFIED

Item	Description
Message	No trigger is set on any of the variables Note: This message includes the following information. ▪ Output name
Content	The "Trigger variable" setting has no variable entries.

Item	Description
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.4 EVALUATION_ERROR

Item	Description
Message	Formula Evaluation Error Note: This message includes the following information. ▪ Output name ▪ Error status
Content	Failed to evaluate the "Expression" setting.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.5 NO_VARIABLE_SPECIFIED

Item	Description
Message	Not a single variable is specified in the formula. Note: This message includes the following information. ▪ Output name
Content	The "Expression" setting does not contain the variable specified in the "Trigger variable" setting.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.6 COMPUTE_ERROR

Item	Description
Message	Error in arithmetic operation Note: This message includes the following information. ▪ Output name
Content	An error occurred during execution of the "Expression" setting.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.7 OUTCOLUMN_CREATE_ERROR

Item	Description
Message	Failed to create output column Note: This message includes the following information. ▪ Output name
Content	Failed to generate an output column for registering the execution result of the "Expression" setting.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.8 OUTCOLUMN_INSERT_ERROR

Item	Description
Message	Failed to register data in output column Note: This message includes the following information. ▪ Output name
Content	Failed to register the execution result of the "Expression" setting in the output column.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.6.9 VARIABLE_NOT_UNIQUE_ERROR

Item	Description
Message	Variable settings are duplicated and variable names cannot be identified Note: This message includes the following information. ▪ Component name ▪ Data name ▪ Statistics type
Content	The "Input Data" setting contains variables with duplicated definitions.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.7 MQTT Collector

15.2.7.1 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to MQTT Broker.
Action	(When turning OFF the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side for the correct user information (user name and password). 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to OFF. 4. For the "User name" and "Password" settings, enter the user name and password confirmed in step 1, and save them. 5. Restart the relevant component. (When turning ON the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side to make sure that anonymous access is allowed. 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to ON, and save it. 4. Restart the relevant component.

15.2.7.2 MESSAGE_PARSE_ERROR

Item	Description
Message	Incoming message parsing error
Content	Data in a different format than expected was received.
Action	Configure the setting on the publishing device side to send data in the following format. { 'records' : [{ 'timestamp': 'Timestamp (ISO 8601 format)' }, { 'Data name': 'Value' }]} Note 1: Define data as an array in JSON format for the records key. Note 2: The array should contain {'data name': 'Value' } as a data definition. Note 3: The Timestamp setting is optional. If setting it, use {'timestamp': 'Timestamp (ISO 8601 format)' } to define the timestamp. If not set, timestamp is automatically assigned. In this event, the system time of the machine running SpeeDBee Synapse is used.

15.2.7.3 UNSUPPORTED_DATA_TYPE

Item	Description
Message	Unsupported data type
Content	The data received contained a value of a type that the MQTT collector does not support.
Action	Configure the setting on the publishing device side to send data of the following types. ▪ Integer value ▪ Decimal value ▪ Character ▪ True/false value

15.2.7.4 TIMESTAMP_ERROR

Item	Description
Message	Timestamp error Note: This message includes the following information. ▪ Last registration time ▪ Error registration time
Content	The currently received data timestamp is older than the previously received data timestamp.

Item	Description
Action	1. Confirm the last registration time included in the message. 2. Configure the setting on the publishing device side to send data with a timestamp that is newer than the last registration time confirmed in step 1.

15.2.7.5 TIMESTAMP_RANGE_ERROR

Item	Description
Message	Timestamp tolerance error Note: This message includes the following information. ▪ Timestamp ▪ Minimum time allowed ▪ Maximum time allowed included
Content	The difference between the timestamp defined in the received data and the actual timestamp is larger than the value set in the "Timestamp error tolerance" setting.
Action	(Action 1) Configure the setting on the publishing device side to send data including no timestamp. (Action 2) 1. Open the configuration screen of the relevant component. 2. Check the "Timestamp error tolerance" setting. 3. Configure the setting on the publishing device side to send data that has a timestamp within the tolerance confirmed in step 2. (Action 3) 1. Open the configuration screen of the relevant component. 2. For the "Timestamp error tolerance" setting, enter a larger value, and save it. 3. Restart the relevant component.

15.2.7.6 TIMESTAMP_PARSE_ERROR

Item	Description
Message	Timestamp parse error
Content	The timestamp defined in the received data was not in the expected format.

Item	Description
Action	Configure the setting on the publishing device side to send data that has a timestamp in ISO 8601 format.
Remarks	An example of a timestamp in ISO 8601 format is: 2023-10-30T10:00:00.012345678Z

15.2.7.7 RECEIVED_PARAMETER_ERROR

Item	Description
Message	Received data error Note: This message includes the following information. ▪ Column name (or key name) ▪ Error cause
Content	There was a problem with a parameter defined in the received data. The types of error causes are as follows. ▪ data_type: Different in the type of the created column and the received data ▪ array_size: Different in the array size of the created column and the received data ▪ value: Value of the received data outside the range of the column type ▪ key_name: Keys not usable as column names when the "Dynamic column" setting is ON and the collector generates a column based on the received JSON data (e.g., incorrect character, exceeded maximum number of characters)
Action	(When the cause of the error is data_ type) 1. Confirm the column name included in the message. 2. Check the configuration on the publishing device side for the column data type confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Data type" setting of the column confirmed in step 1, specify the data type confirmed in step 2, and save it. 5. Restart the relevant component. (When the cause of the error is array_size) 1. Confirm the column name included in the message. 2. Check the configuration on the publishing device side for the array size of the column confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Number of data" setting of the column confirmed in step 1, specify the array size confirmed in step 2, and save it. 5. Restart the relevant component.

Item	Description
	(When the cause of the error is value) 1. Confirm the column name included in the message. 2. Configure the setting on the publishing device side so that the value of the data confirmed in step 1 falls within the range of the data type. (When the cause of the error is key_name) 1. Confirm the key name included in the message. 2. Open the configuration screen of the relevant component. 3. Check the names that can be entered for the "Data name" setting. 4. Configure the setting on the publishing device side to change the column name confirmed in step 1 to the name confirmed in step 3.

15.2.7.8 CA_FILE_ERROR

Item	Description
Message	Certificate file error
Content	The certificate uploaded to the "CA certificate" setting is invalid.
Action	1. Obtain the correct CA certificate in the MQTT Broker settings. 2. Open the configuration screen of the relevant component. 3. Upload the certificate gotten in step 1 to the "CA certificate" setting, and save it. 4. Restart the relevant component.

15.2.8 MQTT Emitter

15.2.8.1 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to MQTT Broker.
Action	(When turning OFF the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side for the correct user information (user name and password).

Item	Description
	2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to OFF. 4. For the "User name" and "Password" settings, enter the user name and password confirmed in step 1, and save them. 5. Restart the relevant component. (When turning ON the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side to make sure that anonymous access is allowed. 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to ON, and save it. 4. Restart the relevant component.

15.2.8.2 UNSUPPORTED_DATA_TYPE

Item	Description
Message	Unsupported data type
Content	Some of the data to be sent is of an unsupported data type.
Action	1. Right-click the IN Port of the relevant component. 2. Specify "Receive and set data definitions" for the "Setting option" setting, and check the data list. 3. In the data list, check the data type that the MQTT Emitter does not support. 4. Take one of the following actions on the component sending the data confirmed in step 3. ▪ Delete the flowlink ▪ Configure the flowlink filter function to prevent the data confirmed in step 3 from passing ▪ Stop or remove the component ▪ Configure the component not to send the data confirmed in step 3
Remarks	The MQTT Emitter supports the following data types. ・BOOLEAN ・INT8 ・INT16 ・INT32 ・INT64 ・UINT8 ・UINT16 ・UINT32 ・UINT64 ・FLOAT ・DOUBLE

Item	Description
	• STRING • BINARY • FILE

15.2.8.3 PUBLISH_ERROR

Item	Description
Message	Publish error
Content	Failed to send data.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the configuration on the MQTT Broker side for the host and port capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component. (Workaround 3) Consult your System Administrator and make sure that the configuration on the MQTT Broker side is correct.

15.2.9 File Emitter

15.2.9.1 UNSUPPORTED_DATA_TYPE

Item	Description
Message	Unsupported data type
Content	Some of the data to be sent is of an unsupported data type.
Action	1. Check the IN Port of the relevant component. 2. If it is connected with a flowlink to a component other than the following ones, delete the flowlink. ▪ CSV Serializer ▪ JSON Serializer

15.2.10 Cloud Emitter for AWS

15.2.10.1 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to AWS IoT Core.
Action	1. Check the AWS IoT Core configuration for the correct host port, and obtain the correct CA certificate, SSL certificate, and private key. 2. Open the configuration screen of the relevant component. 3. Enter the information confirmed in step 1 for the following settings, and upload and save the obtained file. ▪ Host ▪ Port ▪ CA certificate ▪ SSL certificate ▪ Private key 4. Restart the relevant component.

15.2.10.2 PUBLISH_ERROR

Item	Description
Message	Publish error
Content	Failed to send data to AWS IoT Core.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the AWS IoT Core configuration for the correct host port, and obtain the correct CA certificate, SSL certificate, and private key. 2. Open the configuration screen of the relevant component. 3. Enter the information confirmed in step 1 for the following settings, and upload and save the obtained file. ▪ Host ▪ Port ▪ CA certificate ▪ SSL certificate ▪ Private key

Item	Description
	4. Restart the relevant component. (Workaround 3) Consult your System Administrator and make sure that the configuration on the AWS Iot Core side is correct.

15.2.10.3 UNSUPPORTED_DATA_TYPE

Item	Description
Message	Unsupported data type
Content	Some of the data to be sent is of an unsupported data type.
Action	1. Check the IN Port of the relevant component. 2. If it is connected with a flowlink to a component other than the following ones, delete the flowlink. ▪ CSV Serializer ▪ JSON Serializer

15.2.11 Cloud Emitter for Azure

15.2.11.1 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to Azure IoT Hub.
Action	1. Check the configuration on the Azure IoT Hub side for the correct connection string. 2. Open the configuration screen of the relevant component. 3. For the "Connection string" setting, enter the connection string confirmed in step 1, and save it. 4. Restart the relevant component.

15.2.11.2 PUBLISH_ERROR

Item	Description
Message	Publish error
Content	Failed to send data.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the configuration on the Azure IoT Hub side for the correct connection string. 2. Open the configuration screen of the relevant component. 3. For the "Connection string" setting, enter the connection string confirmed in step 1, and save it. 4. Restart the relevant component. (Workaround 3) Consult your System Administrator and make sure that the configuration on the Azure IoT Hub side is correct.

15.2.11.3 UNSUPPORTED_DATA_TYPE

Item	Description
Message	Unsupported data type
Content	Some of the data to be sent is of an unsupported data type.
Action	1. Check the IN Port of the relevant component. 2. If it is connected with a flowlink to a component other than the following ones, delete the flowlink. ▪ CSV Serializer ▪ JSON Serializer

15.2.12 FTP/FTPS/SFTP Emitter

15.2.12.1 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to FTP/FTPS/SFTP Server.

Item	Description
Action	(When turning OFF the "Anonymous access" setting) 1. Check the configuration on the FTP/FTPS/SFTP Server side for the correct user information (user name and password). 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to OFF. 4. For the "User name" and "Password" settings, enter the user name and password confirmed in step 1, and save them. 5. Restart the relevant component. (When turning ON the "Anonymous access" setting) 1. Check the configuration on the FTP/FTPS/SFTP Server side to make sure that anonymous access is allowed. 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to ON, and save it. 4. Restart the relevant component.

15.2.12.2 SEND_ERROR

Item	Description
Message	Publish error
Content	Failed to send data.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the configuration on the FTP/FTPS/SFTP Server side for the hosts and ports capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component. (Workaround 3) Consult your System Administrator and make sure that the configuration on the FTP/FTPS/SFTP Server is correct.

15.2.12.3 PERMISSION_ERROR

Item	Description
Message	Directory operation error
Content	Failed to perform a directory operation on the FTP/FTPS/SFTP Server. (E.g., permission error)
Action	1. Open the configuration screen of the relevant component. 2. Check the "Output directory" setting for the directory entered. 3. Consult your System Administrator, and on the FTP/FTPS/SFTP Server, grant write permission to the directory confirmed in step 3.

15.2.13 JSON Serializer

15.2.13.1 MAX_SIZE_ERROR

Item	Description
Message	Max size error
Content	Unable to create the file because the file size exceeds the "Max size" setting.
Action	1. Open the configuration screen of the relevant component. 3. For the "Max size" setting, enter a larger value, and save it. 3. Restart the relevant component.

15.2.14 CSV Serializer

15.2.14.1 MAX_SIZE_ERROR

Item	Description
Message	Max size error
Content	Unable to create the file because the file size exceeds the "Max size" setting.
Action	1. Open the configuration screen of the relevant component. 3. For the "Max size" setting, enter a larger value, and save it. 3. Restart the relevant component.

15.2.15 JSON Deserializer

15.2.15.1 JSON_FORMAT_ERROR

Item	Description
Message	Incorrect formatting of JSON
Content	The received file is not in the correct JSON format.
Action	Connect the flowlink so that it can receive a file in the correct JSON format.

15.2.15.2 ARRAY_VALUES_NOT_SUPPORTED

Item	Description
Message	Array values are not supported Note: This message includes the following information. Pattern Data name
Content	The pattern matched an array values, which is not supported.
Action	1. Confirm the component and data included in the message. 2. Right-click the IN Port of the relevant component. 3. Enter a pattern that does not result in an array value and save it. 4. Restart the relevant component.

15.2.15.3 DATA_TYPE_MISMATCH

Item	Description
Message	Data type mismatch Note: This message includes the following information. Pattern Data name
Content	A data type different from the configuration on the result of matching the pattern has been specified for the "Data type" setting.

Item	Description
Action	1. Confirm the component and data included in the message. 2. Right-click the IN Port of the relevant component. 3. Select the data type that matches the match result and save. 4. Restart the relevant component.

15.2.15.4 STRING_TRUNCATED

Item	Description
Message	String truncated Note: This message includes the following information. Pattern Data name
Content	The string value matched by the pattern exceeds the maximum size, so the output string is truncated.
Action	No special action is required. However, some data is lost.
Remarks	The maximum string size is 511 bytes.

15.2.16 Event Trigger

15.2.16.1 IGNORE_INVALID_TIMESTAMP_DETECT

Item	Description
Message	Event detection ignored due to invalid timestamp data
Content	This error indicates that after the detection of a start or end event, another trigger occurred from data with an invalid timestamp, and the detection by that trigger was ignored. An invalid timestamp indicates data with the same timestamp as or earlier than the first encountered timestamp. The event trigger ignores detection from data with these timestamps.
Action	Revise the conditional expression that allows start and end events to be satisfied at the same time. This error does not occur if the start and end events occur at different times.

15.2.17 Shell Command

15.2.17.1 COMMAND_EXEC_ERROR

Item	Description
Message	Command execution error
Content	Failed to execute the command entered for the "Command" setting.
Action	1. Open the configuration screen of the relevant component. 2. For the "Command" setting, enter an executable command, and save it. 3. Restart the relevant component.

15.2.18 PLC Writing

15.2.18.1 CONNECTION_REFUSED

Item	Description
Message	Disconnected
Content	The PLC has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the configuration on the PLC side for the host and port capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component.

15.2.18.2 NETWORK_TIMEOUT

Item	Description
Message	Timeout Note: This message includes the following information. ▪ Address ▪ Port number

Item	Description
	▪ Setting timeout value ▪ Operation type
Content	(When the operation type is connect or close) Failed to connect the PLC within the time specified for the "Connect timeout" setting. (When the operation type is receive or send) Failed to receive data from the PLC within the time specified for the "Read timeout" setting.
Action	1. Confirm the operation type included in the message. 2. Open the configuration screen of the relevant component. 3. (When the operation type confirmed in step 1 is connect or close) Increase the "Connect timeout" setting. (When the operation type confirmed in step 1 is receive or send) Increase the "Read timeout" setting. 4. Restart the relevant component.

15.2.18.3 SERIAL_CONNECTION_REFUSED

Item	Description
Message	Disconnected (Serial Communication)
Content	The PLC has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the PLC settings for the correct "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit". 2. Open the configuration screen of the relevant component. 3. Enter and save the values confirmed in step 1 for the "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit" settings. 4. Restart the relevant component.

15.2.18.4 SERIAL_NETWORK_TIMEOUT

Item	Description
Message	Timeout (Serial Communication) Note: This message includes the following information.

Item	Description
	▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Failed to receive data from the PLC within the time specified for the "Read timeout" setting.
Action	1. Open the configuration screen of the relevant component. 2. Increase the "Read timeout" setting. 3. Restart the relevant component.

15.2.18.5 SERIAL_NETWORK_ACCESS_ERROR

Item	Description
Message	Communication error Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Error number ▪ Operation type
Content	(When the operation type is connect) Failed to connect to the PLC. (When the operation type is receive or send) Failed to send or receive data to/from the PLC.
Action	1. Confirm the operation type included in the message. (If the operation type confirmed in step 1 is connect) 2. Open the configuration screen of the relevant component. 3. Check the PLC settings for the correct "Device", "Baud rate", "Data bit", "Stop bit", and "Parity bit", and enter and save the confirmed settings. 4. Restart the relevant component. (If the operation type confirmed in step 1 is receive or send) 2. Check for any connection issues with the PLC.

15.2.18.6 SERIAL_LOCK_TIMEOUT

Item	Description
Message	Connection exclusive control timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Could not connect to the PLC within the time specified for the "Connect timeout" setting because another component using the same device is currently connected.
Action	1. Open the configuration screen of the relevant component. 2. Increase the value of the "Connect timeout" setting. 3. Restart the relevant component.

15.2.18.7 REGISTER_ADDRESS_ERROR

Item	Description
Message	Specified register is invalid Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address not supporting bit specification has been entered in the "Address" setting with bit specification.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address without bit specification, and save it. 4. Restart the relevant component.

15.2.18.8 REGISTER_ADDRESS_RANGE_ERROR

Item	Description
Message	Incorrect register address Note: This message includes the following information.

Item	Description
	▪ Register name ▪ Register address ▪ Operation type
Content	A register address that cannot be handled by the PLC protocol has been entered in the "Address" setting.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that can be handled by the PLC protocol, and save it. 4. Restart the relevant component. Note: Refer to the manual of the relevant device and others for usable addresses and their range.

15.2.18.9 REGISTER_ADDRESS_DATA_TYPE_ERROR

Item	Description
Message	Incorrect data type Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A data type different from the configuration on the PLC side has been specified for the "Data type" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the PLC configuration for the data type of the register confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Data type" setting of the register confirmed in step 1, specify the data type confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.18.10 REGISTER_ADDRESS_SIZE_ERROR

Item	Description
Message	Incorrect number of contiguous regions Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	The number of contiguous regions different from the configuration on the PLC side has been entered for the "Num array" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the PLC configuration for the number of contiguous regions confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Num array" setting of the register confirmed in step 1, enter the number of contiguous regions confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.18.11 REGISTER_WRITE_FIX_VALUE_ERROR

Item	Description
Message	Incorrect fixed value Note: This message includes the following information. ▪ Register name ▪ Register address
Content	The "Write value" setting is invalid. Note: This error occurs when the "Value type" setting is "Fix value."
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. Make sure that the "Write value" setting confirmed in step 1 satisfies the following conditions, and save it. ▪ Write value type matching the "Data type" setting ▪ Number of characters less than or equal to the "Num array" setting, if the write value is a string 5. Restart the relevant component.

15.2.18.12 REGISTER_WRITE_NO_VALUE_ERROR

Item	Description
Message	Register write value acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address
Content	Failed to get the data specified for the "Write value" setting. Note: This error occurs when the "Value type" setting is "Component."
Action	1. Confirm the register name and register address included in the message. 2. Click the IN Port of the relevant component, and open the data monitor. 3. Confirm the column in which at least one data item is registered. 4. Open the configuration screen of the relevant component. 5. For the "Write value" setting of the register confirmed in step 1, specify the column confirmed in step 3, and save it. 6. Restart the relevant component.

15.2.18.13 REGISTER_WRITE_ERROR

Item	Description
Message	Register write error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed in write access to the register.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.19 Modbus Writing

15.2.19.1 CONNECTION_REFUSED

Item	Description
Message	Disconnected
Content	The device has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the configuration on the device side for the host and port capable of communication. 2. Open the configuration screen of the relevant component. 3. For the "Host" and "Port" settings, enter the host and port confirmed in step 1, and save them. 4. Restart the relevant component.

15.2.19.2 NETWORK_TIMEOUT

Item	Description
Message	Timeout Note: This message includes the following information. ▪ Address ▪ Port number ▪ Setting timeout value ▪ Operation type
Content	(When the operation type is connect or close) Failed to connect to the device within the time specified for the "Connect timeout" setting. (When the operation type is receive or send) Failed to receive data from the device within the time specified for the "Read timeout" setting.
Action	1. Confirm the operation type included in the message. 2. Open the configuration screen of the relevant component. 3. (When the operation type confirmed in step 1 is connect or close) Increase the "Connect timeout" setting. (When the operation type confirmed in step 1 is receive or send) Increase the "Read timeout" setting. 4. Restart the relevant component.

15.2.19.3 SERIAL_CONNECTION_REFUSED

Item	Description
Message	Disconnected (Serial Communication)
Content	The device has been disconnected.
Action	(Action 1) Restart the component. (Action 2) 1. Check the device settings for the correct "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits". 2. Open the configuration screen of the relevant component. 3. Enter and save the values confirmed in step 1 for the "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits" settings. 4. Restart the relevant component.

15.2.19.4 SERIAL_NETWORK_TIMEOUT

Item	Description
Message	Timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Failed to receive data from the device within the time specified for the "Read timeout" setting.
Action	1. Open the configuration screen of the relevant component. 2. Increase the "Read timeout" setting. 3. Restart the relevant component.

15.2.19.5 SERIAL_NETWORK_ACCESS_ERROR

Item	Description
Message	Communication error Note: This message includes the following information. ▪ Device ▪ Baud rate

Item	Description
	▪ Error number ▪ Operation type
Content	(When the operation type is connect) Failed to connect to the device. (When the operation type is receive or send) Failed to send or receive data to/from the device.
Action	1. Confirm the operation type included in the message. (If the operation type confirmed in step 1 is connect) 2. Open the configuration screen of the relevant component. 3. Check the device settings for the correct "Device", "Baud rate", "Communication protocol", "Data bits", "Stop bits", and "Parity bits", and enter and save the confirmed settings. 4. Restart the relevant component. (If the operation type confirmed in step 1 is receive or send) 2. Check for any connection issues with the device.

15.2.19.6 SERIAL_LOCK_TIMEOUT

Item	Description
Message	Connection exclusive control timeout (Serial Communication) Note: This message includes the following information. ▪ Device ▪ Baud rate ▪ Configured timeout value ▪ Operation type
Content	Could not connect to the device within the time specified for the "Connect timeout" setting because another component using the same device is currently connected.
Action	1. Open the configuration screen of the relevant component. 2. Increase the value of the "Connect timeout" setting. 3. Restart the relevant component.

15.2.19.7 REGISTER_ADDRESS_ERROR

Item	Description
Message	Specified register is invalid Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address not supporting bit specification has been entered in the "Address" setting with bit specification.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address without bit specification, and save it. 4. Restart the relevant component.

15.2.19.8 REGISTER_ADDRESS_RANGE_ERROR

Item	Description
Message	Incorrect register address Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A register address that cannot be handled by the device protocol has been entered for the "Address" setting.
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. For the "Address" setting of the register confirmed in step 1, enter the register address that can be handled by the device protocol, and save it. 4. Restart the relevant component. Note: Refer to the manual of the relevant device and others for usable addresses and their range.

15.2.19.9 REGISTER_ADDRESS_DATA_TYPE_ERROR

Item	Description
Message	Incorrect data type Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	A data type different from the configuration on the device side has been specified for the "Data type" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the configuration on the device side for the data type of the register confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Data type" setting of the register confirmed in step 1, specify the data type confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.19.10 REGISTER_ADDRESS_SIZE_ERROR

Item	Description
Message	Incorrect number of contiguous regions Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Operation type
Content	The number of contiguous regions different from the configuration on the device side has been entered for the "Num array" setting.
Action	1. Confirm the register name and register address included in the message. 2. Check the configuration on the device side for the number of contiguous regions confirmed in step 1. 3. Open the configuration screen of the relevant component. 4. For the "Num array" setting of the register confirmed in step 1, enter the number of contiguous regions confirmed in step 2, and save it. 5. Restart the relevant component.

15.2.19.11 REGISTER_WRITE_FIX_VALUE_ERROR

Item	Description
Message	Incorrect fixed value Note: This message includes the following information. ▪ Register name ▪ Register address
Content	The "Write value" setting is invalid. Note: This error occurs when the "Value type" setting is "Fix value."
Action	1. Confirm the register name and register address included in the message. 2. Open the configuration screen of the relevant component. 3. Make sure that the "Write value" setting confirmed in step 1 satisfies the following conditions, and save it. ▪ Write value type matching the "Data type" setting ▪ Number of characters less than or equal to the "Num array" setting, if the write value is a string 5. Restart the relevant component.

15.2.19.12 REGISTER_WRITE_NO_VALUE_ERROR

Item	Description
Message	Register write value acquisition error Note: This message includes the following information. ▪ Register name ▪ Register address
Content	Failed to get the data specified for the "Write value" setting. Note: This error occurs when the "Value type" setting is "Component."
Action	1. Confirm the register name and register address included in the message. 2. Click the IN Port of the relevant component, and open the data monitor. 3. Confirm the column in which at least one data item is registered. 4. Open the configuration screen of the relevant component. 5. For the "Write value" setting of the register confirmed in step 1, specify the column confirmed in step 3, and save it. 6. Restart the relevant component.

15.2.19.13 REGISTER_WRITE_ERROR

Item	Description
Message	Register write error Note: This message includes the following information. ▪ Register name ▪ Register address ▪ Protocol error number
Content	Failed in write access to the register.
Action	1. Confirm the register name, register address, and protocol error number included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the protocol error number confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions) Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.20 EtherNet/IP Writing

15.2.20.1 WRITE_ERROR

Item	Description
Message	Write error Note: This message includes the following information. ▪ Error cause ▪ Variable name or class ID ▪ Instance ID ▪ Attribute ID
Content	Failed in writing.
Action	1. Confirm the error cause, variable name, class ID, instance ID, and attribute ID included in the message. 2. Check the communication specifications manual of the device to see if a workaround is available for the cause of the error confirmed in step 1. 3. (If a workaround is available) Implement the workaround. (If no workaround is available, or if you are not familiar with the workaround instructions)

Item	Description
	Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.20.2 UNSUPPORTED_FUNCTION_ERROR

Item	Description
Message	Unsupport function
Content	This function is not currently supported. Writing with CIPObject is not supported.
Action	1. Open the configuration screen of the relevant component. 2. Make sure that the "Setting option" setting of the variable satisfies the following condition, and save it. ▪ To be "Variable name" (Writing with "CIPObject" is not supported.) 3. Restart the relevant component.

15.2.20.3 FIX_DATA_TYPE_INCOLLECT

Item	Description
Message	Fixed value write data type mismatch Note: This message includes the following information. ▪ Data type ▪ Write value
Content	The fixed write value does not match the destination data type. Note: This error occurs when the "Value type" setting is "Fix value."
Action	1. Confirm the data type and write value included in the message. 2. Open the configuration screen of the relevant component. 3. Make sure that the "Write value" setting of the variable confirmed in step 1 satisfies the following condition, and then save it. ▪ Write value type matching the "Data type" setting 4. Restart the relevant component.

15.2.20.4 STRUCT_ERROR

Item	Description
Message	Specified structure does not exist Note: This message includes the following information. ▪ Variable name
Content	The specified structure does not exist.
Action	1. Confirm the variable name included in the message. 2. Open the configuration screen of the relevant component. 3. Make sure that the "Variable name" setting of the variable confirmed in step 1 satisfies the following conditions, and then save it. ▪ Specified variable name is correct ▪ Specified variable is a structure 4. Restart the relevant component.

15.2.21 Send mail

15.2.21.1 SMTP_AUTHENTICATION_ERROR

Item	Description
Message	SMTP authentication error Note: This message includes the following information. ▪ Username ▪ Password
Content	Failed in authentication to SMTP Server.
Action	1. Check the configuration on the SMTP Server side for the correct SMTP user information (user name and password). 2. Open the configuration screen of the relevant component. 3. For the "SMTP user" and "SMTP password" settings, enter the user name and password confirmed in step 1, and save them. 4. Restart the relevant component.

15.2.22 MQTT Client

15.2.22.1 MQTT_PUBLISH_ERROR

Item	Description
Message	MQTT publish error
Content	Failed to send data.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the MQTT Broker configuration for the correct host port, and obtain the correct CA certificate. 2. Open the configuration screen of the relevant component. 3. Enter the information confirmed in step 1 for the following settings, and upload and save the obtained file. ▪ Host ▪ Port ▪ CA certificate 4. Restart the relevant component. (Workaround 3) Consult your System Administrator and make sure that the configuration on the MQTT Broker side is correct.

15.2.22.2 MQTT_SUBSCRIBE_ERROR

Item	Description
Message	MQTT subscribe error
Content	Failed to receive data.
Action	(Workaround 1) Make sure that the machine running SpeeDBee Synapse is connected to the network. (Workaround 2) 1. Check the MQTT Broker configuration for the correct host port, and obtain the correct CA certificate. 2. Open the configuration screen of the relevant component. 3. Enter the information confirmed in step 1 for the following settings, and upload and save the obtained file. ▪ Host ▪ Port ▪ CA certificate 4. Restart the relevant component. (Workaround 3)

Item	Description
	Consult your System Administrator and make sure that the configuration on the MQTT Broker side is correct.

15.2.22.3 CA_FILE_ERROR

Item	Description
Message	Certificate file error
Content	The certificate uploaded to the "CA certificate" setting is invalid.
Action	1. Obtain the correct CA certificate in the MQTT Broker settings. 2. Open the configuration screen of the relevant component. 3. Upload the certificate obtained in step 1 to the "CA certificate" setting, and save it. 4. Restart the relevant component.

15.2.22.4 TLS_CERT_VERIFY_ERROR

Item	Description
Message	Certificate verification error
Content	Failed to verify the certificate uploaded to the “CA Certificate” setting item.
Action	1. Obtain the correct CA certificate for the MQTT broker. 2. Open the settings screen for the relevant component. 3. Upload the certificate obtained in step 1 to the “CA Certificate” setting field and save it. 4. Restart the relevant component.

15.2.22.5 TLS_HANDSHAKE_ERROR

Item	Description
Message	Encrypted communication error
Content	Failed to establish encrypted communication.

Item	Description
Action	(Action 1) 1. The TLS version of the relevant component may differ from the TLS versions supported by the MQTT broker. Consult your system administrator and verify the MQTT broker settings. (Action 2) 1. The encryption method (TLS one-way authentication) of the relevant component may differ from the encryption methods supported by the MQTT broker. Consult your system administrator and verify the MQTT broker settings. (Action 3) 1. Verify the port number the MQTT broker uses for encrypted communication. 2. Open the settings screen for the affected component. 3. Enter the port number verified in step 1 into the “Port” setting field and save. 4. Restart the affected component.

15.2.22.6 CLIENT_ID_REFUSED

Item	Description
Message	Client ID error
Content	Connection refused due to the “Client ID” setting.
Action	1. Verify the client ID rules for the MQTT broker. 2. Open the settings screen for the relevant component. 3. Enter the client ID in the “Client ID” setting field according to the rules from step 1. 4. Restart the relevant component.

15.2.22.7 AUTHENTICATION_FAILURE

Item	Description
Message	Authentication error
Content	Failed in authentication to MQTT Broker.
Action	(When turning OFF the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side for the correct user information (user name and password). 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to OFF.

Item	Description
	4. For the "User name" and "Password" settings, enter the user name and password confirmed in step 1, and save them. 5. Restart the relevant component. (When turning ON the "Anonymous access" setting) 1. Check the configuration on the MQTT Broker side to make sure that anonymous access is allowed. 2. Open the configuration screen of the relevant component. 3. Set the "Anonymous access" setting to ON, and save it. 4. Restart the relevant component.

15.2.22.8 INCORRECT_PROTOCOL_VERSION

Item	Description
Message	Protocol version error
Content	The connection was refused because the protocol version is not supported by the MQTT broker.
Action	1. The MQTT protocol version of the relevant component may differ from the protocol version supported by the MQTT broker. Consult your system administrator and verify the MQTT broker settings.

15.2.23 Core Internal

15.2.23.1 DELETE_STORAGE_GROUPS_ERROR

Item	Description
Message	Past data deletion error
Content	Failed to delete old data in the persistent column. A storage error is one of the possible causes.
Action	Consult your System Administrator and make confirmation on the storage status and the like. (E.g., possibility of failure)

15.2.23.2 DELETE_FILE_TYPE_OLD_DATA_ERROR

Item	Description
Message	File type column data deletion error
Content	Failed to delete old data in the file type column. A storage error is one of the possible causes. (The file type column refers to a file saved by the serializer.)
Action	Consult your System Administrator and make confirmation on the storage status and the like. (E.g., possibility of failure)

15.2.23.3 ACCESS_FILE_TYPE_DATA_DIR_ERROR

Item	Description
Message	File type column data deletion error Note: This message includes the following information. ▪ Number of files
Content	Failed to delete old data in the file type column because the directory was inaccessible.
Action	Consult your System Administrator and make confirmation on the storage status and the like. (E.g., possibility of failure)

15.2.23.4 SYNC_COLUMNS_TO_STORAGE_ERROR

Item	Description
Message	Persistent data write error
Content	Failed to write the persisted column to storage. A storage error is one of the possible causes.
Action	Consult your System Administrator and make confirmation on the storage status and the like. (E.g., possibility of failure)

15.2.23.5 CORE_INTERRUPT

Item	Description
Message	Detection of abnormal end of previous run Note: This message includes the following information. ▪ Core start time ▪ Final state ▪ Last state transition time
Content	Due to a cause such as a power failure of the machine running SpeeDBee Synapse, the core stopped without a graceful termination procedure.
Action	Make sure that the components registered in SpeeDBee Synapse operate normally.

15.2.23.6 SPEEDBEE_ERROR_ACCESS

Item	Description
Message	DB access error detected
Content	Failed to access the time series database file for SpeeDBee Synapse.
Action	(Workaround 1) If this occurs frequently, initialize the database as described in "Server Settings." (Workaround 2) If the error persists even after Workaround 1 is implemented, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.23.7 SPEEDBEE_ERROR_CORRUPT

Item	Description
Message	DB file corruption detected
Content	The time series database file for SpeeDBee Synapse is corrupt.
Action	(Workaround 1) If this occurs frequently, initialize the database as described in "Server Settings." (Workaround 2) If the error persists even after Workaround 1 is implemented, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.23.8 SPEEDBEE_ERROR_INTR

Item	Description
Message	DB power failure detection section
Content	Due to a cause such as a power failure of the machine running SpeeDBee Synapse, the time series database operation was interrupted.
Action	Make sure that the components registered in SpeeDBee Synapse operate normally.

15.2.23.9 SPEEDBEE_ERROR_NO_REPORTS

Item	Description
Message	DB missing log acquisition error
Content	Failed to obtain the missing detection log of the time series database file for SpeeDBee Synapse.
Action	(Workaround 1) If this occurs frequently, initialize the database as described in "Server Settings." (Workaround 2) If the error persists even after Workaround 1 is implemented, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.24 DB query service

15.2.24.1 PARAMETER_ERROR_IPADDRESS

Item	Description
Message	IP address setting error
Content	The "Address" setting is blank.
Action	1. Open the configuration screen of the relevant component. 2. For the "Address" setting, enter the IP address, and save it. 3. Restart the relevant component.

15.2.24.2 PARAMETER_ERROR_PORT_NUMBER

Item	Description
Message	Port number setting error
Content	The "Port" setting is blank.
Action	1. Open the configuration screen of the relevant component. 2. For the "Port" setting, enter the port number, and save it. 3. Restart the relevant component.

15.2.24.3 PARAMETER_ERROR_HTTPS_CERTIFICATION

Item	Description
Message	HTTPS certificate error
Content	No HTTPS certificate file has been uploaded to the "SSL/TLS certificate" setting in the "HTTP/HTTPS communication" Setting Menu.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.24.4 PARAMETER_ERROR_HTTPS_SECRETKEY

Item	Description
Message	HTTPS private key error
Content	No HTTPS key file has been uploaded to the "Private key" setting in the "HTTP/HTTPS communication" Setting Menu.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.24.5 PARAMETER_ERROR_JWS_SETTING

Item	Description
Message	Enable/disable JWS verification setting error
Content	The "Token authentication" setting is unspecified.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.24.6 PARAMETER_ERROR_JWS_VERIFICATION

Item	Description
Message	JWS validation file error
Content	The JWS validation file is incomplete.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.24.7 DB_QUERY_HTTP_SERVER_START_ERROR

Item	Description
Message	DB query service startup error
Content	Failed to start HTTP Server.
Action	1. Open the configuration screen of the relevant component. 2. Check the "Host" and "Port" settings. 3. Consult your System Administrator and confirm if the host and port checked in step 2 conflicts with other services.

15.2.25 Error manager

15.2.25.1 PARAMETER_ERROR_INTERVAL

Item	Description
Message	Processing interval setting error
Content	The "Processing interval" setting value is incorrect.
Action	1. Open the configuration screen of the relevant component. 2. For the "Processing interval" setting, enter a value in the range of 5 to 180, and save it. 3. Restart the relevant component.

15.2.25.2 PARAMETER_ERROR_CORE_PORT

Item	Description
Message	Core port number setting error
Content	The core port number is set to an incorrect value.
Action	This error should not normally occur. Consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.25.3 NETWORK_TIMEOUT_ERROR

Item	Description
Message	Communication timeout error
Content	Communication timed out.
Action	1. Open the configuration screen of the relevant component. 2. For the "Timeout" setting, enter a larger value, and save it. 3. Restart the relevant component.

15.2.26 Event data manager

15.2.26.1 RESTORE_DB

Item	Description
Message	Restoring data from backup due to database corruption
Content	This is a warning to let you know that during access to the internal database of i-Trigger, corruption was detected and data was restored from a backed-up database.
Action	No special action is required. However, some data is lost.

15.2.27 VIDEO Collector (i-Trigger Extension package)

15.2.27.1 READ_FRAME_ERROR

Item	Description
Message	Failed to load frame
Content	Failed to load streaming data received from a network camera.
Action	Usually, no special action is required. However, if this error persists, check the following points. 1. Check to see if the connected camera supports i-Trigger. (Reference) 2. Revise the camera settings. Try a lower resolution or lower frame rate.

15.2.27.2 MAKE_CONTAINER_ERROR

Item	Description
Message	Error occurred while creating mp4
Content	An internal error occurred while creating the mp4 file.
Action	This error should not normally occur. If the symptom persists, consult your System Administrator, and contact manufacturer following the steps described in Contact manufacturer.

15.2.28 Event Recorder (i-Trigger Extension package)

All errors generated by this component are "Component default" errors.

15.3 Custom component Python-API reference

This section provides the API specifications for custom component development in the Python language. See below for the custom component development procedures.

Python Custom Component Creation

15.3.1 speedbeesynapse.component.base

Hive Componentbased Module

This is a module to import when implementing SpeeDBee Synapse components in Python.

15.3.2 HiveComponentBase Objects

```
class HiveComponentBase()
```

Base Class for Components

All components must inherit this class in their definitions.

Attributes:

- `in_portX` *HiveInPort* - Instance variable representing the IN Port no. X. Practically, where `X` is a number such as in_port1 and in_port2.
- `out_portX` *HiveOutPort* - Instance variable representing the OUT Port no. X. Practically, where `X` is a number such as out_port1 and out_port2.

15.3.2.1 premain

```
def premain(param: str | dict) -> typing.Optional[ErrorInfoBase]
```

Function Called Immediately before Component Main Processing Called when the component starts, immediately before main processing. Read parameters here to perform processing such as column creation or custom initialization.

Note, however, that the processing with this function is under exclusive control in all components, and so if the processing takes a long time, premain of other components cannot operate during that time. Instead, creating the column here is faster than creating it in the main process.

If the processing such as initialization ends normally, this method should exit without returning anything. The `main` method is then called.

If the component cannot continue to run for some reason, return an instance of the class that inherits ErrorInfoBase. The content of the error then appears on the screen and the component stops (without calling the `main` or `postmain` method).

The definition of this method is not required.

Arguments:

- `param` *str | dict* - Parameters to the component

Returns:

- `None` - Normal end
- `ErrorInfoBase` - Error object inheriting ErrorInfoBase returned if the component cannot start for some reason

15.3.2.2 main

```
def main(param: str | dict)
```

Component Main Processing Function

This is a method to implement the main processing of the component.

For classes that inherit HiveComponentBase, this method must be overridden and implemented.

The end of this function means that the component has finished processing. In normal end, no return is required.

If terminated with an error, it should return an instance of a class that inherits ErrorInfoBase. The returned instance information is displayed as the stop reason on the screen.

If an exception is raised outside this method, the component stops with an unknown error. If that behavior is undesirable, catch the exception and continue the processing, or return an instance of a class that inherits ErrorInfoBase and exit this method.

Arguments:

- `param` *str | dict* - Parameters to the component

Returns:

- `None` - Normal end
- `ErrorInfoBase` - Error object inheriting ErrorInfoBase returned if the component cannot start for some reason

15.3.2.3 postmain

```
def postmain(param: str | dict)
```

Function Called Immediately after Component Main Processing Called immediately after the main processing function finishes, such as when the component stops.
This is always called even when the main processing ends due to an error or the like. Therefore, it is assumed that, for example, used resources are released.
The definition of this method is not required.

Arguments:

- `param` *str | dict* - Parameters to the component

15.3.2.4 is_runnable

```
def is_runnable()
```

Component Sustainability Determination

This determines whether the main processing of this component is to continue. The component must call this function periodically (at least within 5 seconds) to confirm that it can continue with the main processing. When this function returns True, the main processing is continuable; when it returns False, the main process must end as soon as possible.

Returns:

- True if continuable, False if a request to stop the component issued

15.3.2.5 get_timestamp

```
def get_timestamp() -> int
```

Current Time Acquisition

This gets the current time expressed in Unix time in nanoseconds.

Returns:

- Data timestamp.

15.3.2.6 notify_stop

```
def notify_stop()
```

Component Stop Request

Called when this component receives a stop request from the system.
For classes that inherit HiveComponentBase, this method must be overridden according to need.
When the method is called, the main processing of the component must be stopped immediately.

If `is_runnable()` is periodically called in the main process to check whether it can continue, the implementation of this function is not required.

15.3.2.7 register_status

```
def register_status(status: RunningStatus)
```

Status Information Registration

Registers the status with error information added

Arguments:

- `status` - Status object with error information registered

15.3.2.8 Status

```
def Status()
```

Status Information Generation

This generates a status object to register error information.

15.3.2.9 interval_iteration

```
def interval_iteration(interval, base_time=0)
```

Iterator for Periodic Processing

This is an iterator to perform some processing on the component at fixed time intervals.
It yields values from the iterator starting at the specified base time and at the specified periodic interval. However, the actual timing of the start of processing does not exactly coincide with the scheduled time and is affected by the OS, device load, and clock accuracy.

In addition, if user-implemented periodic processing takes longer time than the periodic interval, the next periodic processing may be skipped. There, the number of times the periodic processing is skipped is taken as the second element of yield.

On receipt of a component stop request, this function ends the iteration.

Arguments:

- `interval` - Periodic processing interval (in nanoseconds)
- `base_time` - Unix time representing the iteration start reference time (in nanoseconds)

Yields:

- `[0]` - Unix time representing the time for the scheduled iteration execution at that time (in nanoseconds)
- `[1]` - Number of previous iterations skipped

Examples:

This is usable as an iterator in a for statement as follows.

```
for [ts, skip] in self.interval_iteration(1000000000):
    #
    # Processing to be executed periodically
    #
```

In this example, the portion of "processing to be executed periodically" is executed every one second.

15.3.2.10 save_running_info

```
def save_running_info(key: str, data: bytes)
```

Saving Arbitrary Information

This records an arbitrary sequence of bytes into the system.
The saved data is referable with load_running_info() on the same component even after the system is restarted. The data is saved as different data depending on the specified key. Therefore, to save multiple data sets, use different keys.

Arguments:

- `key` *str* - Key to identify the information to be saved
- `data` *bytes* - Information of bytes type to be saved

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.2.11 load_running_info

```
def load_running_info(key: str) -> typing.Optional[bytes]
```

Loading Arbitrary Information

Arguments:

- `key` *str* - Key to identify the information to be loaded

Returns:

- `data` - Data of byte type loaded

- `None` - No data

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.2.12 get_common_variable

```
def get_common_variable(varname: str) -> typing.Optional[str]
```

Getting Component Default Variable

Arguments:

- `varname` *str* - Name of the common variable to get

Returns:

- Value of the specified common variable
- `None` - Specified variable not found

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.2.13 create_lock

```
def create_lock(key: typing.Union[str, bytes],
                interbase: bool = False,
                timeout: typing.Optional[float] = None,
                auto_unlock_at_stop: bool = True,
                unlock_by_any_threads: bool = False) -> SynapseLock
```

Lock acquiring method for mutual exclusion inter-component.

This method acquires a mutex lock for a resource identifed by `key`. Only one thread can get a lock of a `key.

This method acquires a lock for exclusive control of the resource identified by the string or byte string specified by the `key` argument. A lock with the same `key` can only be acquired by one thread. The acquired lock can be released with the `unlock()` method of the returned lock object. Alternatively, if this method is used in a `with` statement, the lock will be released when the scope is exited. The lock will also be automatically released if the component instance signel while the lock is still acquired.

If the argument `interbase` is true, mutual exclusion is possible between different component bases by using the same `key`. In this case, mutual exclusion with custom components using the C-API `hive_lock()` is also available. If it is false, mutual exclusion is possible only within the same component base.

If an external component instance has already acquired the lock when this method is called, execution will be blocked within the method. The maximum blocking time can be specified in seconds using the `timeout` argument. If the lock cannot be acquired after the timeout period has elapsed, a `HiveTimeoutError` exception will be thrown. If a value less than or equal to `0` is specified for `timeout`, execution will not be blocked and an immediate timeout will occur if the lock cannot be acquired. If None is specified for `timeout`, the method will continue to wait until the lock is acquired without timing out.

If a running component instance receives a stop request while waiting to acquire a lock, it will throw a HiveLockCanceled exception regardless of the timeout period.

Even if it is the same thread, it cannot lock the same `key` twice. In that case, it will continue to block until it times out or receives a stop request.

When using from within the main function, you should generally specify the default values for `auto_unlock_at_stop` and `unlock_by_any_threads`. When using from a component method, the component may not be running, so you must set `auto_unlock_at_stop` to `False`.

Arguments:

- `key` *str|bytes* - String or byte sequence for identifying lock context.
- `interbase` *bool* - Set `True` if the mutual exclusion affects over different component-base.
- `timeout` *float|None* - timeout for waiting other component or thread to lock.
- `auto_unlock_at_stop` *bool* - Set `True` to automatically release the lock when the component stops.
- `unlock_by_any_threads` *bool* - Set `True` to allow lock release from different threads within the same instance.

Returns:

SynapseLock object.

Raises:

- `HiveTimeoutError` - Could not take lock in a specified `timeout`.
- `HiveLockCanceled` - Stop request received before to take a lock.
- `HiveApiError` - Hive errors.

Examples:

Use `with` statement like following.

```
while self.is_runnable();
    try:
        with self.create_lock('lock-serial-device', False, 10.0):
            #
            # Procedure which is need mutual exclusion.
            #
    except HiveTimeoutError as exc:
        self.log.debug('wait lock')
        continue
    except HiveLockCanceled:
        return
```

15.3.3 DataType Objects

```
class DataType(IntEnum)
```

Data type Definition Enum Class

This is an Enum class that indicates the data type. Use this attribute as the data type to be specified at the time of column creation.

Attributes:

- `NONE(0)` - Undefined type not matching any type
- `BOOLEAN(1)` - Boolean type
- `INT8(2)` - 8-bit signed integer type
- `INT16(3)` - 16-bit signed integer type
- `INT32(4)` - 32-bit signed integer type
- `INT64(5)` - 64-bit signed integer type
- `UINT8(6)` - 8-bit unsigned integer type
- `UINT16(7)` - 16-bit unsigned integer type
- `UINT32(8)` - 32-bit unsigned integer type
- `UINT64(9)` - 64-bit unsigned integer type
- `FLOAT(10)` - 32-bit floating-point number type
- `DOUBLE(11)` - 64-bit floating-point number type
- `STRING(13)` - String
- `BINARY(14)` - Binary
- `FILE(16)` - File

15.3.4 ColumnOption Objects

```
class ColumnOption()
```

Column Options

Optional configuration for OUT Port columns

Attributes:

- `column_type` *ColumnType* - Column type (LOW, MIDDLE, HIGH)
- `samplingrate` *float* - Sampling rate
- `compression_unit` *int* - Compression unit
- `fixedsize_binary` *int* - Column data size
- `max_binarysize` *int* - Maximum length of LO variable-length type data

15.3.5 FileTypeMetaInfo Objects

```
class FileTypeMetaInfo(typing.TypedDict)
```

File Type Data Meta Information

This class contains the meta information of the file to be registered in a file type column. All members are optional.

Attributes:

- `media_type` *ColumnType* - Media type (e.g., application/json and text/csv are settable)
- `filename` *float* - File name hint (this file name is used if connected to a file emitter)
- `begin_timestamp` *int* - Leading time in the time range of the data contained in this file (Unix time in nanoseconds)
- `end_timestamp` *int* - Last time in the time range of the data contained in this file (Unix time in nanoseconds)

15.3.6 OutColumn Objects

```
class OutColumn()
```

OUT Port Column

For example, data registration for an OUT Port column is done through an instance of this class.

Attributes:

- `name` *str* - Column name specified at creation time
  Note: For information about characters that cannot be used, please refer to "About Column Names That Cannot Be Used" below
- `data_type` *DataType* - Data type of the column specified at creation time

- `data_array` *int* - Number of elements in the column specified at creation time: 0 for the scalar type
- `options` *ColumnOption* - Column option specified at creation time

About Column Names That Cannot Be Used: The following characters cannot be used in column names

- `\` - backslash
- `/` - slash
- `*` - asterisk
- `?` - question mark
- `"` - double quote
- `< >` - angle brackets
- `|` - pipe
- `'` - single quote
- ` ` - space

15.3.6.1 insert

```
def insert(*args) -> None
```

Value Registration in Column

This registers the timestamp `ts` value specified for this column.
If the timestamp is omitted, the current time is assumed.
For data, pass the data that matches the data type at the time of column creation. If mismatching, the passed data will be converted when it is implicitly convertible, but otherwise an error is raised.

Arguments:

- `data` - Value to register
- `ts` - Timestamp to register this value

Raises:

- `TypeError` - Incorrect data not matching the data type of this column passed
- `InvalidTimestamp` - Timestamp is too far from the current time or is in the past of a previously registered timestamp

15.3.6.2 open_file

```
@contextmanager
def open_file()
```

New File Creation

This opens a new file to be registered with a column created as a file type and returns its file object.

Returns:

- `FileObject` - Opened file object

Raises:

- `TypeError` - This method called for a non-FILE type column
- `HiveApiError` - Various errors in the Hive framework
- `OSError` - Various errors related to file open

Examples:

This is usable in a with clause as follows.

```
file_column = self.out_port1.Column("clmfile", DataType.FILE)
with file_column.open_file() as fo:
    fo.write("filedata - line 1\n".encode('utf-8))
    fo.write("filedata - line 2\n".encode('utf-8))
    fo.write("filedata - line 3\n".encode('utf-8))

    file_column.insert_file(fo, ts)
```

15.3.6.3 insert_file

```
def insert_file(fo: typing.IO[typing.AnyStr],
                ts: int = 0,
                meta: typing.Optional[FileTypeMetaInfo] = None) -> None
```

File DB Registration

This registers a file in the file type column.

Arguments:

- `file` - Opened file object
- `ts` - Timestamp to register this value
- `meta` - Meta information of the file to be registered

Raises:

- `ValueError` - File targeted for registration already closed
- `HiveApiError` - Various errors in the Hive framework
- `OSError` - Various errors related to file operations

15.3.6.4 insert_file_move

```
def insert_file_move(pathlike,
                     ts: int = 0,
                     meta: typing.Optional[FileTypeMetaInfo] = None) -> None
```

Existing File DB Registration (Move)

This registers a file already created in the file type column. The specified file is moved to the DB management directory. Note that no file remains in the path specified as an argument. This cannot register a directory.

Arguments:

- `pathlike` - Object representing the path of the file to be registered
- `ts` - Timestamp to register this value
- `meta` - Meta information of the file to be registered

Raises:

- `ValueError` - Specified path
- `HiveApiError` - Various errors in the Hive framework
- `OSError` - Various errors related to file operations

15.3.6.5 insert_file_copy

```
def insert_file_copy(pathlike,
                     ts: int = 0,
                     meta: typing.Optional[FileTypeMetaInfo] = None) -> None
```

Existing File DB Registration (Copy)

This registers a file already created in the file type column. The specified file is copied to the DB's managed directory, and the copy destination file is registered in the column. No further reference is made to the original file path specified as an argument. This cannot register a directory.

Arguments:

- `pathlike` - Object representing the path of the file to be registered
- `ts` - Timestamp to register this value
- `meta` - Meta information of the file to be registered

Raises:

- `ValueError` - Specified path
- `HiveApiError` - Various errors in the Hive framework
- `OSError` - Various errors related to file operations

15.3.6.6 insert_file_ref

```
def insert_file_ref(pathlike,
                    ts: int = 0,
                    meta: typing.Optional[FileTypeMetaInfo] = None) -> None
```

Existing File DB Registration (Reference)

This registers a file already created in the file type column. The specified file is registered with its path in the column. Therefore, if the file is deleted, the information in this column will no longer be referable from other components. This cannot register a directory.

Arguments:

- `pathlike` - Object representing the path of the file to be registered
- `ts` - Timestamp to register this value
- `meta` - Meta information of the file to be registered

Raises:

- `ValueError` - Specified path
- `HiveApiError` - Various errors in the Hive framework
- `OSError` - Various errors related to file operations

15.3.7 AggregationType Objects

```
class AggregationType(IntEnum)
```

Aggregation Type Definition Enum Class

Represents the aggregation type of the column data retrieved from the IN Port

Attributes:

- `NONE(0)` - None
- `OLDER(1)` - Older
- `NEWER(2)` - Newer
- `COUNT(3)` - Count
- `SUM(4)` - Sum
- `SUMSQ(5)` - Sum of squares
- `SUMSQD(6)` - Sum of squared deviation
- `MIN(7)` - Minimum
- `MAX(8)` - Maximum

- `RANGE(9)` - Range
- `MEAN(10)` - Arithmetic mean value
- `VAR(11)` - Variance
- `STDEV(12)` - Standard deviation
- `UVAR(13)` - Unbiased variance
- `USTDEV(14)` - Sample unbiased standard deviation
- `STDER(15)` - Standard error
- `CV(16)` - Coefficient of variation
- `MEDIAN(17)` - Median
- `QUARTMIN(18)` - Quartile
- `QUART1(19)` - Quartile
- `QUART2(20)` - Quartile
- `QUART3(21)` - Quartile
- `QUARTMAX(22)` - Quartile
- `MODE(23)` - Mode value
- `MODECOUNT(24)` - Mode value count
- `MDEV(25)` - Mean deviation
- `HMEAN(26)` - Harmonic mean
- `GMEAN(27)` - Geometric mean
- `KURT(28)` - Kurtosis
- `SKEW(29)` - Skewness

15.3.8 AggregationTypeSet Objects

```
class AggregationTypeSet()
```

A set of flags indicating which statistics information is enabled when getting data from IN Port columns.

Pass the `AggregationType` attribute to the `test()` method to confirm that this statistic information is enabled. It is also possible to get a list of enabled statistics information by using an iterator.

Examples:

Pass the `AggregationType` attribute to the `test()` method to confirm that this statistic information is enabled.

```
if a_type_set.test(AggregationType.MAX):
  pring(max: ON")
else:
  pring(max: OFF")
```

It is also possible to get a list of enabled statistics information by using an iterator.

```
for a_type in aggregation_type_set:
    print(a_type.name)
```

15.3.9 InColumn Objects

```
class InColumn()
```

IN Port Column

This class supports only information acquisition. Operations such as data registration with columns are not supported.

Attributes:

- `source_name` *str* - Name of the component that generated the column
- `data_name` *str* - Name of the column
- `data_type` *DataType* - Column type information
- `array_size` *int* - Number of array elements in the column (0 for the scalar type)
- `raw_data` *bool* - Boolean indicating whether to include raw data when getting data from the column
- `stat_type_set` *AggregationTypeSet* - Set of flags indicating which statistics information is enabled when getting data from the column
- `output_name` *str* - Output name of the column. None by default

15.3.9.1 get_latest_value

```
def get_latest_value(use_storage: bool = False) -> (int, typing.Any)
```

Latest Column Value Acquisition

This reads the latest data registered in the column and returns its timestamp and a tuple of values.

Arguments:

- `use_storage` *bool* - Note: Not supported at this time
  Specify True to perform the reading from data persisted to storage.
  If False, it reads data from memory only, and may return No data available even if persisted data is present.
  Even if True, it does not access storage if the latest data is present in memory.

Returns:

- `(0, None)` : No value readable from the column
- `(>0, !=None)` : Read timestamp and values

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.10 InPortReader Objects

```
class InPortReader()
```

IN Port Reader

This class reads data from the IN Port column.
Do not directly instantiate this class; instead, use the `self.in_portX.ContinuousReader()` or `self.in_portX.TimeRangeReader()` function.

15.3.10.1 read

```
def read() -> typing.Optional[HiveWindowData]
```

Reading Column Data

This reads column data from the IN Port.

Returns:

- `HiveWindowData` - Read window data returned.
- `None` - None returned if data is not readable at the time.

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.11 HiveOutPort Objects

```
class HiveOutPort()
```

Class Representing the OUT Port

For example, column creation for the OUT Port is done through an instance of this class.
It is not possible to instantiate this class directly. Use the attribute `self.out_port1`, which is accessible from the beginning when the component starts.

15.3.11.1 Column

```
def Column(name: str,
           data_type: DataType,
           data_array: int = 0,
           opt: typing.Optional[ColumnOption] = None) -> OutColumn
```

Column Creation

This creates a column with the specified name in this OUT Port.

Arguments:

- `name` - Created column name
- `data_type` - Data type of the column to create
- `data_array` - Number of elements in the column to create (0 for the scalar type)
- `opt` - Column options

About Column Names That Cannot Be Used: The following characters cannot be used in column names

- `\` - backslash
- `/` - slash
- `*` - asterisk
- `?` - question mark
- `"` - double quote
- `< >` - angle brackets
- `|` - pipe
- `'` - single quote
- ` ` - space

Returns:

- OutColumn

Raises:

- `HiveApiError` - Various errors in the Hive framework

15.3.11.2 register_confirmed_time

```
def register_confirmed_time(ts: int = 0) -> None
```

Confirmed Time Registration with OUT Port

This notifies that data registration with the column of this OUT Port has been confirmed up to the specified time `ts`. This registration allows another component connected through a flowlink to know that data before the specified time `ts` can be read.

- `Note` - Calling this API is not required.

Arguments:

- `ts` - Time at which data registration with the column of the OUT Port has been confirmed Current time by default

15.3.12 HiveInPort Objects

```
class HiveInPort()
```

Class Representing IN Port

Processing such as data acquisition from the IN Port is done through an instance of this class.

15.3.12.1 get_columns

```
def get_columns() -> list[InColumn]
```

Column List Acquisition

Gets information in all columns connected to this IN Port

Returns:

- list[InColumn]

15.3.12.2 ContinuousReader

```
def ContinuousReader(start: int) -> InPortReader
```

Continuous Column Reader Generation

Generates a ContinuousReader that continuously reads data from the columns connected to this IN Port

Arguments:

- `start` - Time to start reading from the IN Port (Unix time in nanoseconds)

Returns:

- ContinuousReader

Examples:

Get and use a reader in a with clause as follows.

```
with self.in_port1.ContinuousReader(start=self.get_timestamp()) as reader:
    while self.is_runnable():
        window_data = reader.read()
        if not window_data:
            continue
            :
```

15.3.12.3 TimeRangeReader

```
def TimeRangeReader(start: int, end: int) -> InPortReader
```

Time Range Data Reader Generation

Generates a TimeRangeReader that reads data in the specified time range from the specified column or from all columns connected to the IN Port

Arguments:

- `start` - Time to start reading from the IN Port (Unix time in nanoseconds)
- `end` - Time to end reading from the IN Port (Unix time in nanoseconds)
- `column` - Column from which to get data (or all columns of the IN Port by default)

Returns:

- data

Examples:

Get and use a reader in a with clause as follows.

```
with self.in_port1.ContinuousReader(start=self.get_timestamp()) as reader:
    for window_data in reader:
        :
```

15.3.13 RunningStatus Objects

```
class RunningStatus()
```

15.3.13.1 add_error

```
def add_error(err: ErrorInfoBase)
```

Error Information Registration

This registers error information with a status object. Simply registering the information with this object does not result in saving it. Finally, call HiveComponentBase.register_status to register it with the system.

Arguments:

- `err` - Error information

15.3.13.2 ErrorType

```
def ErrorType(error_id: str, *param_names: str)
```

Error Information Class Generation

This generate a class to manage error information. For the actual error information, generate an instance from the class generated by this function.

Arguments:

- `error_id` - String identifying error information
- `param_names` - Names of error information parameters

15.3.14 ErrorInfoBase Objects

```
class ErrorInfoBase(Exception)
```

Error Information for Registration in Status

This is a class to hold the types and details of component errors.
Do not directly instantiate or inherit from this class; instead, instantiate a class generated with the `ErrorType` function.

Attributes:

- `errorTypeIndex` *int* - Type of error parameters
- `params` *list[ErrorParam]* - Error parameters

15.3.15 HiveLogLevel Objects

```
class HiveLogLevel(IntEnum)
```

Log Level Type Definition Enum Class

This is an Enum class that indicates the log level type.

Attributes:

- `ERROR(0)` - Error log
- `WARNING(1)` - Warning log
- `INFO(2)` - Normal information log
- `DEBUG(3)` - Debugging log
- `TRACE(4)` - Trace log

15.3.16 HiveLog Objects

```
class HiveLog()
```

Log output class

15.3.16.1 get_current_level

```
def get_current_level()
```

Log Level Acquisition

Returns:

- `HiveLogLevel` - Current log level Enum

15.3.16.2 error

```
def error(*msg)
```

Error Log Output

Arguments:

- `msg` - String for log output

15.3.16.3 warning

```
def warning(*msg)
```

Warning Log Output

Arguments:

- `msg` - String for log output

15.3.16.4 info

```
def info(*msg)
```

Normal Information Log Output

Arguments:

- `msg` - String for log output

15.3.16.5 debug

```
def debug(*msg)
```

Debugging Log Output

Arguments:

- `msg` - String for log output

15.3.16.6 trace

```
def trace(*msg)
```

Trace Log Output

Arguments:

- `msg` - String for log output

15.3.16.7 exception

```
def exception(exc: Exception)
```

Exception Dedicated Error Log Output

Arguments:

- `exc` - Exception for log output

15.3.17 HiveWindowData Objects

```
class HiveWindowData()
```

Window Data Class

This is a collection of data grouped into specific time units when getting column data from the IN Port. The time unit varies depending on the IN Port configuration.

Attributes:

- `time_range` *tuple[int:2]* - Two-element tuple representing the time range for this window. The first element represents the start time and the second element represents the end time. Both are in nanoseconds.
- `window_id` *int* - Numerical value identifying the window. Note that a different value is acquired for each window, but the number is not always incremented by one.
- `event_id` *int* - Numerical value identifying the event range. Fixed to 0 when the event has not been defined or lies outside the event range. A value of 1 or greater is inserted during the event.
- `records` *list[HiveRecord]* - List of records in this window
- `columns` *list[InColumn]* - List of columns connected to the source IN Port. Same as that acquired with HiveInPort.get_columns().
- `event` *HiveInPortEventSetting* - Event information set for the IN Port

15.3.18 HiveRecordType Objects

```
class HiveRecordType(IntEnum)
```

Record Type Enum Class

This is an Enum class that indicates the record type.

Attributes:

- `RAW_DATA(0)` - Raw data
- `STATISTICS(1)` - Aggregate data

15.3.19 HiveRecord Objects

```
class HiveRecord()
```

Record Class

This is a collection of data from the same time of day grouped when getting column data from the IN Port.
If there are data pieces in different columns with the same time, all of them will be collected into one instance. Two types of records, which are raw data and aggregate data, are identifiable with `record_type`.

Attributes:

- `record_type` *HiveRecordType* - List of records in this window
- `timestamp` *int* - Timestamp of the data contained in this record. Unix time (in nanoseconds)
- `data` *list[HiveRecordData]* - List of data for this record

15.3.20 HiveRecordData Objects

```
class HiveRecordData()
```

Record Data Class

Data of each column in a record.

Attributes:

- `value` *any* - Actual data. A format that depends on the type of column, such as a numerical value, string, or array.
- `column` *InColumn* - Object indicating the column of interest
- `stat_type` *AggregationType* - Aggregation type for this data

15.3.21 HiveCompError Objects

```
class HiveCompError()
```

Component Error Information Class

Component running or error information at the time of stop due to an error

Attributes:

- `error_type` *str* - String indicating the type of error in the component
- `parameters` *list[any]* - Additional error information

15.3.22 HiveCompStatus Objects

```
class HiveCompStatus()
```

Component Status Class

Component status and error information

Attributes:

- `status` *str* - String representing the execution status of the component
- `errors` *list[HiveCompError]* - Component running or error information at the time of stop due to an error

15.3.23 HiveInPortEventSetting Objects

```
class HiveInPortEventSetting()
```

IN Port Event Configuration Information Class

Class to hold the event information set for the IN Port Changing the attribute values of this class does not affect the actual Inport setting

Attributes:

- `source_name` *str* - Component name that is subject to event determination
- `data_name` *str* - Column name that is subject to event determination
- `pre_time` *int* - Extended time range ahead of the event interval
- `post_time` *int* - Extended time range behind the event interval

15.3.24 SynapseLock Objects

```
class SynapseLock()
```

Lock object for mutual exclusion for inter-components.

The object is returned from `HiveComponentBase.create_lock()` method. Normally, no need to use this object. Like usage example of the `create_lock()` method, the lock is automatically released when the runtime exit the `with` statement scope. Use `acquire()` and `release()` methods of this class, in the case of not to use `with` statement.

15.3.24.1 acquire

```
def acquire() -> None
```

acquire a lock.

This method returns nothing in the case of lock acquiring succeeded.

Raises:

- `HiveTimeoutError` - Failed to acquire the lock within the specified timeout period.
- `HiveLockCanceled` - Received a stop request for the component before acquiring the lock.
- `HiveApiError` - Various errors from the Hive framework.

15.3.24.2 release

```
def release() -> None
```

unlock.

Release lock context in the object, then make another component or thread can take lock.

15.3.25 HiveTimeoutError Objects

```
class HiveTimeoutError(Exception)
```

Exception class of timeout in the case of using blocking API.

15.3.26 HiveLockCanceled Objects

```
class HiveLockCanceled(Exception)
```

Exception class of cancel of lock acquiring when the component received stop request.

15.4 Custom component C-API reference

This section provides the API specifications for custom component development in the C language. See below for the custom component development procedures.

Custom Component Creation in C Language

15.4.1 OUT Port Functions

15.4.1.1 hive_outport_create_column() … Create new column

Function to get the handle of the column used to register the data collected by the component with SpeeDBee Synapse

Definition

```
HIVE_OUTCOLUMN hive_outport_create_column(HIVE_COMPONENT component, int out_port_no, const char
*name, HIVE_DATA_TYPE type, HIVE_COLUMN_OPTIONS opt)
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`int out_port_no`	Port number to create the column Specify a macro constant, such as OUTPORT1 or OUTPORT2. Use of only OUTPORT1 is usually sufficient.
3	`const char *name`	Column name Specify the name of the column to be created Note: For information about characters that cannot be used, please refer to "About Column Names That Cannot Be Used" below
4	`HIVE_DATA_TYPE type`	Data type Specify the type of the column to be created
5	`HIVE_COLUMN_OPTIONS opt`	Column options Specify the options of the column to be created. See HIVE_COLUMN_OPTIONS for more information.

About Column Names That Cannot Be Used

The following characters cannot be used in column names

- `\` - backslash
- `/` - slash
- `*` - asterisk
- `?` - question mark
- `"` - double quote
- `< >` - angle brackets
- `|` - pipe
- `'` - single quote
- ` ` - space

Return value

- `HIVE_OUTCOLUMN` column handle: Handle to the column. Used later to register data in the column or to change the column settings
- NULL: Failed to create the column

Description

This function gets the handle of the column used to register the data collected by the component with SpeeDBee Synapse.
To register data in a column, the column handle returned by this function is used.

The column name cannot contain "/" (slash). Other characters are allowable, but non-ASCII characters must be UTF-8 encoded. The column name must not exceed 127 bytes.

If one component gets columns with the identical column names, only the handle of the last one is enabled. Therefore, it is possible to change the settings of a column by getting the column handle of the same column name with this function. However, this method should not be used because it disables any previously acquired column handles.

See HIVE_COLUMN_OPTIONS for more information about specifying column options.

15.4.1.2 hive_outport_register_confirmed_time() … Register column data confirmed time in status information

Function to register column data confirmed time with the OUT Port

Definition

```
void hive_outport_register_confirmed_time(HIVE_COMPONENT component, int out_port_no, hive_time_t
ts);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`int out_port_no`	OUT Port number to be registered
3	`hive_time_t ts`	Time to register

Return value

None

Description

This function registers column data confirmed time with the OUT Port.

By registering this information, this component can tell the framework that all data before the time of registration has been registered.
This optimizes the processing of the component that receives data so that it can handle the data with less latency.

The component that registers data should register the time after registering data of some timestamp, when it is confirmed that data of a timestamp earlier than that time is not to be registered in the future.
Various components can operate without this API, but some delay occurs in data reception on the component that receives data.

Conversely, when registering a timestamp with this API, it is necessary to exercise caution not to register data with a timestamp earlier than that timestamp. If registered, the component that receives the data may drop the data.

15.4.2 Output Column Functions

15.4.2.1 hive_outcolumn_insert() … Register data

Function to register data in a column

Definition

```
bool hive_outcolumn_insert(HIVE_OUTCOLUMN column, const void *data);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle acquired with hive_outport_create_column()
2	`void *data`	Pointer to the data to register

Return value

- true: Successfully registered
- false: Failed to register

Description

This function registers a single piece of data into the column specified in the first argument. The second argument `data` must be a memory address that matches the data type of the column to be registered.

Examples

- Top address of a 2-byte memory area for a scalar of type uint16
- Top address of a 20-byte memory area for a 5 element array of float type

The data timestamp registered with this function will be the current time when this function is called. If it is necessary to specify the time, use hive_outcolumn_insert_ts().

15.4.2.2 hive_outcolumn_insert_ts() … Register timestamp specified data

Function to register timestamp specified data in a column

Definition

```
bool hive_outcolumn_insert_ts(HIVE_OUTCOLUMN column, const void *data, hive_time_t ts);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle acquired with hive_outport_create_column()
2	`const void *data`	Pointer to the data to register

No	Item	Description
3	`hive_time_t ts`	Data timestamp to register

Return value

- true: Successfully registered
- false: Failed to register

Description

The first and second arguments are similar to those of hive_outcolumn_insert().
For the third argument `ts`, specify a timestamp. The current timestamp is acquirable with hive_timestamp().

This function is intended for use in situations where data is registered with the same timestamp for multiple columns.

Usage example: Registering data in multiple columns with the same timestamp

```
  :
hive_time_t ts0 = hive_timestamp();
hive_outcolumn_insert_ts(clmA, dataA, ts0);
hive_outcolumn_insert_ts(clmB, dataB, ts0);
hive_outcolumn_insert_ts(clmC, dataC, ts0);
  :
```

Using hive_outcolumn_insert() will result in different timestamps for clmA, clmB, and clmC.

In this system, the data timestamps to be registered in the same column must be registered in chronological order. Therefore, it is not possible to register data at time `T0` in a column and then register data with a timestamp earlier than `T0` in the same column.

15.4.2.3 hive_outcolumn_insert_binary() … Register binary data

Function to register variable-length binary data in a BINARY column

Definition

```
bool hive_outcolumn_insert_binary(HIVE_OUTCOLUMN column, const void *data, size_t n);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle acquired with hive_outport_create_column()

No	Item	Description
2	`const void *data`	Pointer to the data to register
3	`size_t n`	Number of bytes of data to be registered

Return value

- true: Successfully registered
- false: Failed to register

Description

This function registers arbitrary length data for a column created as type BINARY.
The first and second arguments are similar to those of hive_outcolumn_insert(). For the third argument `n`, specify the number of bytes of data to be registered.

The data timestamp registered with this function will be the current time when this function is called. If it is necessary to specify the time, use hive_outcolumn_insert_binary_ts().

15.4.2.4 hive_outcolumn_insert_binary_ts() ... Register timestamp specified binary type data

Function to register variable-length binary data with a timestamp specification in a BINARY column

Definition

```
bool hive_outcolumn_insert_binary_ts(HIVE_OUTCOLUMN column, const void *data, size_t n,
hive_time_t ts);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle acquired with hive_outport_create_column()
2	`const void *data`	Pointer to the data to register
3	`size_t n`	Number of bytes of data to be registered
4	`hive_time_t ts`	Data timestamp to register

Return value

- true: Successfully registered
- false: Failed to register

Description

The first, second, and third arguments are similar to those of hive_outcolumn_insert_binary().
For the third argument `ts`, specify a timestamp. See hive_outcolumn_insert_ts() for notes on timestamps.

15.4.2.5 hive_outcolumn_insert_multisamples() … Register multi-sample data

Function to register multiple timestamp data sets in a single column

Definition

```
bool hive_outcolumn_insert_multisamples(HIVE_OUTCOLUMN column, const void *data, hive_time_t ts,
int sample_count);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle returned by hive_outport_create_column()
2	`const void *data`	Pointer to the data to register
3	`hive_time_t ts`	Data timestamp to register
4	`int sample_count`	Number of samples of data to be registered

Return value

- true: Successfully registered
- false: Failed to register

Description

This function is usable only when a sampling rate of column type MIDDLE is specified in the column option.
The first, second, and third arguments are similar to those of hive_outcolumn_insert_ts().
For the fourth argument `sample_count`, specify the number of samples of the data to be registered. Multiple pieces of data to be registered must be located consecutively in the memory pointed to by the second argument `data`.

Usage example: Registering consecutive samples of data

```
  HIVE_COLUMN_OPTIONS opt = {
    .type = column_type_middle,
    .sampling_rate = 10;
  };
  HIVE_OUTCOLUMN clmA = hive_outport_create_column(compo, "clmA",
HIVE_DATA_SCALAR(HIVE_TYPE_UINT16), opt)
  uint16_t data[10] = {.}
   :
  get_data(data, 10);
  hive_time_t ts0 = hive_timestamp();
  hive_outcolumn_insert_multisamples(clmA, data, ts0, 10);
   :
```

In the above example, data [0] is registered with the specified timestamp `ts0`, followed by `data [1]` as `ts0 + 0.1 seconds` and data [2] as `ts0 + 0.2` seconds.
The interval between each timestamp is derived from `opt.sampling_rate`.

15.4.2.6 hive_outcolumn_open_file() ... Create file

Function to create a file for a file-type column

Definition

```
HIVE_FILE *hive_outcolumn_open_file(HIVE_OUTCOLUMN column);
```

Arguments

No	Item	Description
1	`HIVE_OUTCOLUMN column`	Column handle Specify a column handle returned by hive_outport_create_column()

Return value

- HIVE_FILE pointer `HIVE_FILE`: Pointer to HIVE_FILE with a created file pointer.
- NULL: Failed to create the file

Description

This function creates a file for registering data to a file-type column and returns the `HIVE_FILE` pointer with the file pointer.
Use the file pointer in the returned `HIVE_FILE` to write to the file. Subsequently, use hive_outcolumn_insert_file() to register the file with the column.

Usage example: Creating a file and writing data

```
  HIVE_OUTCOLUMN clmfile  = hive_outport_create_column(comp, OUTPORT1, "clmfile",
HIVE_DATA_SCALAR(HIVE_TYPE_FILE),   HIVE_COLUMN_OPTION_NONE);
   :
  HIVE_FILE *hf = hive_outcolumn_open_file(clmfile);
  if (hf == NULL) {
    HIVE_API_ERROR error = hive_get_api_error();
    HIVE_LOG_ERROR("hive_outcolumn_open_file error(0x%x,%d,%d)", error.code, error.system_errno,
error.db_errno);
  }
   :
  for (int i = 0; i < 10; i++) {
    fprintf(hf->fp, "data: %d\n");
  }
   :
  HIVE_FILE_METAINFO meta = {
   .filename = "a",
   .media_type = "text/plain",
   .begin_timestamp = 1,
   .end_timestamp = 2,
  };

  if (!hive_outcolumn_insert_file(hf, 0, &meta)) {
    HIVE_API_ERROR error = hive_get_api_error();
    HIVE_LOG_ERROR("hive_outcolumn_insert_file error(0x%x,%d,%d)", error.code,
error.system_errno, error.db_errno);
  }
```

Ensure that the `HIVE_FILE` pointers returned by this function release the resources with hive_outcolumn_insert_file() or hive_outcolumn_dispose_file(). Otherwise, the resources will be kept unreleased, resulting in memory leaks.

15.4.2.7 hive_outcolumn_insert_file() … Register file type data

Function to register data in a file type column

Definition

```
bool hive_outcolumn_insert_file(HIVE_FILE *hf, hive_time_t ts, const HIVE_FILE_METAINFO *meta);
```

Arguments

No	Item	Description
1	`HIVE_FILE *hf`	Pointer to HIVE_FILE Specify the HIVE_FILE pointer returned by hive_outcolumn_open_file()
2	`hive_time_t ts`	Data timestamp to register
3	`HIVE_FILE_METAINFO *meta`	Meta information of the file to be registered

Return value

- true: Successfully registered
- false: Failed to register

Description

This function registers data in a file type column.
For the first argument `hf`, specify the `HIVE_FILE` pointer created by hive_outcolumn_open_file(). For the second argument `ts`, specify a timestamp. If `0` is specified, the current time is registered. For the third argument `meta`, insert and specify the meta information of the file to be registered in `HIVE_FILE_METAINFO`. The meta information is referenced by the receiver of the data in this file type column, but can be NULL if not required.

See hive_outcolumn_open_file() for a usage example.

15.4.2.8 hive_outcolumn_dispose_file() ... Discard files

Function to discard files

Definition

```
void hive_outcolumn_dispose_file(HIVE_FILE *hf);
```

Arguments

No	Item	Description
1	`HIVE_FILE *hf`	Pointer to HIVE_FILE Specify the HIVE_FILE pointer returned by hive_outcolumn_open_file()

Return value

None

Description

This function discards the file for a file-type column created with hive_outcolumn_open_file() and HIVE_FILE.
When a file is opened with hive_outcolumn_open_file() but is no longer needed, call this function to discard the file.

15.4.3 Input Column Functions

15.4.3.1 hive_inport_get_column_list() ... Get input column information

Get IN Port column information

Definition

```
const HIVE_INCOLUMN_LIST *hive_inport_get_column_list(HIVE_COMPONENT component, int inPortNo);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`int in_port_no`	IN Port setting

Return value

- HIVE_INCOLUMN_LIST: Column information list
- NULL: Column information not acquirable due to a cause such as insufficient memory

Description

This function gets the column information (e.g., component names, column names, types) of another component connected to the specified IN Port.
The memory of structure received as a return value is managed internally by the system and should not be modified, released, etc. Note also that calling another `hive_inport_xxxx` API may cause the contents to be changed. If it is necessary to save this information, copy it to a separate memory area.

15.4.3.2 hive_inport_continuous_reader() … Get Reader for continuous retrieval for column data

Get a Reader to continuously get data from the IN Port columns

Definition

```
HIVE_CONTINUOUS_READER *hive_inport_continuous_reader(HIVE_COMPONENT component, int in_port_no,
hive_time_t search_begin);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`int in_port_no`	IN Port setting

No	Item	Description
3	`hive_time_t search_begin`	Search range starting time

Return value

- HIVE_CONTINUOUS_READER: Reader for continuous column data retrieval

Description

This function gets a Reader used to continuously retrieve the column data of other components connected to the specified IN Port.
This Reader can then be used in hive_continuous_reader_read() to search for and retrieve column data. The memory of the structure received as a return value is managed internally by the system and should not be modified, released, etc.

Usage example:

```
HIVE_CONTINUOUS_READER *creader;
creader = hive_inport_continuous_reader(comp, INPORT1, hive_timestamp());

while (hive_component_runnable(comp)) {
  const HIVE_COLUMN_READ_RESULT *read_result = hive_continuous_reader_read(creader);
  if (!read_result) { continue; }

  HIVE_RECORD_ITERATOR iter = HIVE_GET_RECORD_ITERATOR(read_result);
  const HIVE_RECORD *record;
  while ((record = HIVE_RECORD_ITERATOR_GET_NEXT(&iter)) != NULL)
  {
    for (int i=0; i < record->data_count; i++) {
      const HIVE_RECORD_DATA *rd = HIVE_RECORD_GET_DATA(record, i);
      if (!rd) { continue; }

      const char *sourcename   = read_result->incolumn_list->incolumns[i].source_component;
      const char *dataname     = read_result->incolumn_list->incolumns[i].data_name;
      HIVE_DATA_TYPE type      = read_result->incolumn_list->incolumns[i].data_type;
      hive_aggregation_stat_type_t stat = read_result->incolumn_list->incolumns[i].stat_type;
      const char *dptr         = rd->data;
      size_t size              = rd->data_size;

      process_record_data(sourcename, dataanme, type, stat_type, dptr, size);
    }
  }
}
hive_continuous_reader_release(creader);
```

15.4.3.3 hive_continuous_reader_read() … Continuously retrieve column data

Continuously search for and retrieve data from the columns connected to the IN Port specified for the Reader

Definition

```
const HIVE_COLUMN_READ_RESULT *hive_continuous_reader_read(HIVE_CONTINUOUS_READER *contReader);
```

Arguments

No	Item	Description
1	`HIVE_CONTINUOUS_READER *contReader`	Reader acquired with hive_inport_continuous_reader()

Return value

- HIVE_COLUMN_READ_RESULT: Column data retrieval results

Description

This function gets column data from the Reader acquired with hive_inport_continuous_reader().
Each call updates the search time, allowing column data to be continuously acquired.
The memory of structure received as a return value is managed internally by the system and should not be modified, released, etc.

15.4.3.4 hive_continuous_reader_release() … Release Reader for continuous retrieval of column data

Release the Reader acquired with hive_inport_continuous_reader()

Definition

```
const void hive_continuous_reader_release(HIVE_CONTINUOUS_READER *contReader);
```

Arguments

No	Item	Description
1	`HIVE_CONTINUOUS_READER *contReader`	Reader acquired with hive_inport_continuous_reader()

Return value

None

Description

This function releases the Reader acquired with hive_inport_continuous_reader().
Call this function when the Reader is no longer needed during component processing.

15.4.3.5 hive_inport_timerange_reader() … Get Reader for time range retrieval of column data

Get a Reader to get data from the IN Port columns specifying a time range

Definition

```
HIVE_TIMERANGE_READER *hive_inport_timerange_reader(HIVE_COMPONENT component, const HIVE_INCOLUMN
*incolumn, hive_time_t searchBegin, hive_time_t searchEnd);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`HIVE_INCOLUMN *incolumn`	Column to get data from
3	`hive_time_t search_begin`	Search range starting time
3	`hive_time_t search_end`	Search range ending time

Return value

- HIVE_TIMERANGE_READER: Reader for time range retrieval of column data

Description

This function gets a Reader used to retrieve data in the specified column within a specified time range.
If no column is specified, all columns of other components connected to the IN Port will be subject to search. This Reader can then be used in hive_timerange_reader_read() to search for and retrieve column data.
The memory of structure received as a return value is managed internally by the system and should not be modified, released, etc.

Usage example:

```
hive_time_t search_bigin_ts = hive_timestamp() - 2*HIVE_TIME_1S;

while (hive_component_runnable(comp)) {
  HIVE_TIMERANGE_READER *reader = hive_inport_timerange_reader(comp, incolumn, search_bigin_ts,
search_bigin_ts + 1*HIVE_TIME_1S);

  const HIVE_COLUMN_READ_RESULT *read_result;
  while ((read_result = hive_timerange_reader_read(reader)) != NULL) {
    if (!hive_component_runnable(comp)) { break; }

    HIVE_LOG_INFO("time range: %lu ~ %lu", begin, end);

    HIVE_RECORD_ITERATOR iter = HIVE_GET_RECORD_ITERATOR(read_result);
    const HIVE_RECORD *record;
    while ((record = HIVE_RECORD_ITERATOR_GET_NEXT(&iter)) != NULL)
    {
      Logbuff logb;
      logbuff_reset(&logb);
      logbuff_write(&logb, " %c %19lu", (record->record_type==0)?'D':'S', record->timestamp);

      for (int i=0; i < record->data_count; i++) {
        const HIVE_RECORD_DATA *rd = HIVE_RECORD_GET_DATA(record, i);
        if (!rd) {
          logbuff_write(&logb, "%10s", "(no data)");
          continue;
        }
        const char *sourcename   = read_result->incolumn_list->incolumns[i].source_component;
        const char *dataname     = read_result->incolumn_list->incolumns[i].data_name;
        HIVE_DATA_TYPE type      = read_result->incolumn_list->incolumns[i].data_type;
        hive_aggregation_stat_type_t stat = read_result->incolumn_list->incolumns[i].stat_type;
        const char *dptr         = rd->data;
        size_t size              = rd->data_size;

        process_record_data(sourcename, dataanme, type, stat_type, dptr, size);
      }
    }
  }
  hive_timerange_reader_release(reader);
  usleep(1*1000*1000);
  search_next_ts += 1*HIVE_TIME_1S;
}
```

15.4.3.6 hive_timerange_reader_read() … Search for column data in a time range

Search for and retrieve data in the column specified for the Reader or in all columns connected to the IN Port over a specified time range

Definition

```
const HIVE_COLUMN_READ_RESULT *hive_timerange_reader_read(HIVE_TIMERANGE_READER *rangeReader);
```

Arguments

No	Item	Description
1	`HIVE_TIMERANGE_READER *rangeReader`	Reader acquired with hive_inport_timerange_reader()

Return value

- HIVE_COLUMN_READ_RESULT: Column data retrieval results

Description

This function gets column data from the Reader acquired with hive_inport_timerange_reader().
It can get column data in the time range specified at the time of Reader acquisition.
The memory of structure received as a return value is managed internally by the system and should not be modified, released, etc.

15.4.3.7 hive_timerange_reader_release() ... Release Reader for time range retrieval of column data

Release the Reader acquired with hive_inport_timerange_reader()

Definition

```
const void hive_timerange_reader_release(HIVE_TIMERANGE_READER *reader);
```

Arguments

No	Item	Description
1	`HIVE_TIMERANGE_READER *reader`	Reader acquired with hive_inport_timerange_reader()

Return value

None

Description

This function releases the Reader acquired with hive_inport_timerange_reader().
Call this function when the Reader is no longer needed during component processing.

15.4.3.8 HIVE_GET_RECORD_ITERATOR() ... Get record iterator

Get the record iterator from window data

Definition

```
HIVE_RECORD_ITERATOR HIVE_GET_RECORD_ITERATOR(const HIVE_COLUMN_READ_RESULT *data);
```

Arguments

No	Item	Description
1	`const HIVE_COLUMN_READ_RESULT *data`	Window data acquired with hive_continuous_reader_read()

Return value

- HIVE_RECORD_ITERATOR: Record iterator

Description

This function acquires the record iterator from the window data acquired with hive_continuous_reader_read(). Call HIVE_RECORD_ITERATOR_GET_NEXT() for the acquired iterator to sequentially get records contained in the window.

Usage example:

```
while (hive_component_runnable(component)) {
  const HIVE_RECORD *record;
  const HIVE_COLUMN_READ_RESULT *read_result;

  read_result = hive_continuous_reader_read(creader);
  if (!read_result) { continue; }

  HIVE_RECORD_ITERATOR iter = HIVE_GET_RECORD_ITERATOR(read_result);
  while ((record = HIVE_RECORD_ITERATOR_GET_NEXT(&iter)) != NULL)
  {
    for (int i=0; i < record->data_count; i++) {
      const HIVE_RECORD_DATA *rd = HIVE_RECORD_GET_DATA(record, i);
    }
  }
}
```

15.4.3.9 HIVE_RECORD_ITERATOR_GET_NEXT() … Get record iterator

Get records

Definition

```
const HIVE_RECORD *HIVE_RECORD_ITERATOR_GET_NEXT(HIVE_RECORD_ITERATOR *iter);
```

Arguments

No	Item	Description
1	`HIVE_RECORD_ITERATOR *iter`	Pointer to the iterator acquired with HIVE_GET_RECORD_ITERATOR()

Return value

- `HIVE_RECORD*` : Record pointer
- `NULL` : All records acquired

Description

This function is used in combination with HIVE_GET_RECORD_ITERATOR() to sequentially get records contained in window data.
See HIVE_GET_RECORD_ITERATOR() for a usage example.

15.4.3.10 HIVE_RECORD_GET_DATA() … Get record iterator

Get record data

Definition

```
const HIVE_RECORD_DATA *HIVE_RECORD_GET_DATA(const HIVE_RECORD *record, size_t index);
```

Arguments

No	Item	Description
1	`HIVE_RECORD *record`	Pointer to the record acquired with HIVE_RECORD_ITERATOR_GET_NEXT()
2	`size_t index`	Index of the record data to be acquired. To be a value less than `record->data_count`

Return value

- `HIVE_RECORD_DATA*` : Record data pointer

Description

This function gets individual data pieces contained in a record.
See HIVE_GET_RECORD_ITERATOR() for a usage example.

15.4.3.11 hive_incolumn_get_latest_value() … Get latest value of column data

Get the latest data from the IN Port column

Definition

```
bool hive_incolumn_get_latest_value(const HIVE_INCOLUMN *incolumn, hive_time_t *rts, char
**dataPointer, size_t *dataSize);
```

Arguments

No	Item	Description
1	`HIVE_INCOLUMN *incolumn`	Column to get data from
2	`hive_time_t *rst`	Pointer to put in the acquired data timestamp
3	`char **dataPointer`	Pointer to store the top address of the acquired data
4	`size_t *dataSize`	Pointer to store the size of the acquired data

Return value

- true: Successfully acquired the latest value
- false: Failed to acquire the latest value

Description

This function gets the latest data from a specified column.
If acquisition fails, the return value is false. If acquisition is unavailable due to a reason such as no latest data, the acquired data timestamp is 0 and the data size is 0. The loaded data area is kept in `HIVE_INCOLUMN`. Therefore, if multiple threads are to call this function for the same `HIVE_INCOLUMN`, exclusive control is required.

15.4.4 Component Status Functions

15.4.4.1 hive_status_clear() … Clear status information

Clear status information and make sure there are no errors

Definition

```
void hive_status_clear(HIVE_STATUS status);
```

Arguments

No	Item	Description
1	`HIVE_STATUS status`	Pointer to the status structure

Return value

None

Description

This function erases the error and other information registered in the status structure and returns it to the normal state without any error.
When a status structure in this state is registered by hive_component_register_status(), the component is determined to be operating normally.

15.4.4.2 hive_status_add_error() … Register an error in status information

Register an error in status information

Definition

```
void hive_status_add_error(HIVE_STATUS status, HIVE_ERROR_INFO error);
```

Arguments

No	Item	Description
1	`HIVE_STATUS status`	Pointer to the status structure
2	`HIVE_ERROR_INFO error`	Number of the error to be registered Allowed to specify a predefined error number

Return value

None

Description

This function registers an error in the status structure. For `error`, specify a value generated by the HIVE_ERROR() macro function.

Usage example:

```
const static HIVE_ERROR_TYPE error_types[] = {
  { .name="CONNECTION REFUSED",       .parameter_names={"b","c"}},
  { .name="REGISTER READ ERROR",      .parameter_names={"f","g"}},
  { .name=NULL },
};
const int ERROR_CONNECTION_REFUSED       = 0;
const int ERROR_REGISTER_READ_ERROR      = 1;

int proc(COMPONENT *comp, HIVE_STATUS status) {
         :
  hvie_status_clear(status);
  if (detect_error1())
    hive_status_add_error(status, HIVE_ERROR(ERROR_CONNECTION_REFUSED, STR("hostname:port")));

  if (detect_error2())
    hive_status_add_error(status, HIVE_ERROR(ERROR_REGISTER_READ_ERROR, STR("REGDATA_A"),
STR("DT0001")));

         :
  hive_component_register_status(comp, status);
}
```

15.4.4.3 hive_status_set_total_xxx_xxx() … Register run-time auxiliary information in status information

Register run-time auxiliary information in status information

Definition

```
void hive_status_set_total_emission_count(HIVE_STATUS status, uint64_t value);
void hive_status_set_total_emission_bytes(HIVE_STATUS status, uint64_t value);
void hive_status_set_total_received_count(HIVE_STATUS status, uint64_t value);
void hive_status_set_total_received_bytes(HIVE_STATUS status, uint64_t value);
```

Arguments

No	Item	Description
1	`HIVE_STATUS status`	Pointer to the status structure
2	`uint64_t value`	Registered value

Return value

None

Description

This function registers component run-time auxiliary information in the status structure. The auxiliary information currently defined is as follows.

- Cumulative number sent (`hive_status_set_total_emission_count`)
  Cumulative number of outbound data pieces sent by the component
- Cumulative number of bytes sent (`hive_status_set_total_emission_bytes`)
  Cumulative number of bytes of outbound data sent by the component
- Cumulative number received (`hive_status_set_total_received_count`)
  Cumulative number of inbound data pieces received by the component
- Cumulative number of bytes received (`hive_status_set_total_received_bytes`)
  Cumulative number of bytes of inbound data received by the component

The UI uses this information for supplementary display purposes. Therefore, not calling the API will not affect the behavior of the component.

15.4.4.4 hive_component_register_status() … Register status with component

Register status information with the component

Definition

```
void hive_component_register_status(HIVE_COMPONENT component, const HIVE_STATUS status);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`HIVE_STATUS status`	Pointer to the status structure

Return value

None

Description

This function registers status information with the component.
If no errors are registered in the status information, the system assumes that the component is operating normally. If an error is registered in the status information, the system receives a report that the component has an error.

The system determines the status of the component from the status information that was last registered. If the error is resolved, the component must use this function to re-register the status information that the error has been resolved.

15.4.4.5 HIVE_ERROR() … Generate error information

Macro function to generate an error information structure

Definition

```
HIVE_ERROR_INFO HIVE_ERROR(int error_no, ...);
```

Arguments

No	Item	Description
1	`int error_no`	Specify the index of the error type information defined for the component.
2	...	Specify the additional information of the error that occurred. This can be an integer, a floating point number, or a string, but its usage is special as described below.

Return value

- Error information structure `HIVE_ERROR_INFO`

Description

This function generates a structure with information about the error that occurred. (It does not need to be explicitly released because it allocates stack space, not heap space.)

For the first argument, specify the index of the error type information defined for the component. The system determines the error type from this information. To inform the system of the details of the error that occurred, arguments are useful for specifying additional information. In that event, use a macro as shown below to specify arguments according to the type of additional information.

```
HIVE_ERROR_INFO error = HIVE_ERROR(ERROR_CONNECTION_REFUSED, STR(hostname), I64(portno));
```

The following macros are specifiable for the additional information, depending on the type.

- `I64(x)` : 64-bit signed integer
- `U64(x)` : 64-bit unsigned integer
- `DBL(x)` : Double-precision floating-point number
- `STR(x)` : String pointer

When using a string for additional information, make sure that the memory is not released until the generated HIVE_ERROR_INFO is registered by hive_component_register_status().

15.4.4.6 hive_status_alloc() … Allocate status information structure

Allocate a status information structure

Definition

```
HIVE_STATUS hive_status_alloc(HIVE_COMPONENT component);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.

Return value

- Pointer to the status information structure `HIVE_STATUS`
- NULL: Low memory

Description

This function allocates a structure to record status information.
Normal component development does not require the use of this function. Status information `HIVE_STATUS` is automatically allocated and passed as an argument to `mainloop` before the component starts.

15.4.4.7 hive_status_free() … Release status information structure

Release a status information structure

Definition

```
void hive_status_free(HIVE_STATUS status);
```

Arguments

No	Item	Description
1	`HIVE_STATUS status`	Pointer to the status information structure to be released

Return value

None

Description

This function releases the status information structure allocated by hive_status_alloc().
Normal component development does not require the use of this function. The status information `HIVE_STATUS` that was automatically allocated before the component started is automatically released when the component stops.

15.4.5 Log Related Functions

15.4.5.1 HIVE_LOG_XXXX() ... Log output macros

Log output macro function for components

Definition

```
void HIVE_LOG_ERROR(const char *format, ...);
void HIVE_LOG_WARNING(const char *format, ...);
void HIVE_LOG_INFO(const char *format, ...);
void HIVE_LOG_DEBUG(const char *format, ...);
void HIVE_LOG_TRACE(const char *format, ...);
```

Arguments

No	Arguments	Description
1	`const char *format`	Log output format string Specify the format string to be sent to the log output. This is the same format as printf in the standard C library.
2	-	Parameters converted based on format string conversion specifiers

Return value

None

Description

This is a macro function to send character strings to the SpeeDBee Synapse main log output.
The `format` argument and the subsequent variadic arguments can be used as with printf() in the standard C library.
Available macros are of five types, each corresponding to a log level.

15.4.5.2 hive_current_loglevel() ... Get log level

Get the current log level of the system

Definition

```
HIVE_FW_API HIVE_LOG_LEVEL hive_get_loglevel();
```

Arguments None

Return value

- HIVE_LOG_LEVEL_ERROR(0): Status in which only error logs are sent to the log output
- HIVE_LOG_LEVEL_WARNING(1): Status in which error and warning logs are sent to the log output
- HIVE_LOG_LEVEL_INFO(2): Status in which logs at the normal level are subject to output
- HIVE_LOG_LEVEL_DEBUG(3): Status in which debugging logs are subject to output
- HIVE_LOG_LEVEL_TRACE(4): Status in which detailed trace logs are subject to output

Description

This function gets the current log level of the system. ERROR is the lowest value, TRACE is the highest, and a log with a value less than the return value is sent to the log file.

15.4.6 Other Functions

15.4.6.1 hive_component_interval_call() … Periodic function

Function to continuously run a specified function at regular intervals

Definition

```
bool hive_component_interval_call(HIVE_COMPONENT component, hive_interval_proc_t proc, void
*data, hive_time_t base_time, uint64_t interval_usec);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`hive_interval_proc_t proc`	Periodic function Specify a pointer to the function that defines the processing to be run periodically.
3	`void *data`	User data Any data pointer is specifiable. Passed as an argument when proc is called.

No	Item	Description
4	`hive_time_t base_time`	Time to start running the periodic function Base time for starting to run the proc function Current time if 0 is specified
5	`uint64_t interval_usec`	Periodic function execution interval Specify the interval of calling the proc function in usec units.

Return value

- true: The component stopped normally without any error in the proc function
- false: An error occurred in the proc function

Description

This function is useful for executing a certain process at regular intervals.

15.4.6.2 hive_component_save_running_info() ... Save run-time information

Function to save component run-time information

Definition

```
bool hive_component_save_running_info(HIVE_COMPONENT component, const char *key, const void
*data, size_t size);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`const char *key`	Information to identify the data to be saved Specify different `keys` to hold different information.
3	`const void *data`	Data to be saved
4	`size_t size`	Size of data to be saved

Return value

- true: Successfully saved
- false: Failed to save

Description

This function saves arbitrary information during component execution.
It enables data to be held so that the processing can resume after stopping, for example, at an emitter.

15.4.6.3 hive_component_load_running_info() … Load saved run-time information

Function to load data saved with hive_component_save_running_info()

Definition

```
const void *hive_component_load_running_info(HIVE_COMPONENT component, const char *key, size_t
*size);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`const char *key`	Information to identify the data saved
3	`size_t *size`	Returns the size of the data loaded Output argument

Return value

The top address of the data loaded. If no data exists, null is returned.
This memory is managed by the framework and does not require release by the user. However, if the same API is called again, the previous memory area will be overwritten. According to need, the user should copy the memory area.

Description

This function loads data saved with hive_component_save_running_info(). To restart the processing of data after stopping the running component, the component can resume the processing by loading the saved data.

15.4.6.4 hive_component_runnable() … Component sustainability

Determine whether the component is continuable

Definition

```
bool hive_component_runnable(HIVE_COMPONENT component);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.

Return value

- true: Main processing of component continuable
- false: Requested to stop the component

Description

This function is useful for determining whether the main processing of the component is continuable.
The main processing of the component should continue as long as this function returns true, and should terminate promptly if it returns false. Therefore, in the main processing, ensure that this function is called periodically (at most every 5 seconds) and the return value is checked.

If this function returns false but the component does not stop for a given amount of time, the system may force to stop the component's thread.

15.4.6.5 hive_timestamp() … Get current time

Get the current system time

Definition

```
hive_time_t hive_timestamp();
```

Arguments

None

Return value

- Current system time

Description

This function gets the current system time.
See hive_time_t for timestamp units.

15.4.6.6 hive_get_api_error() … Get occurring error information

Function to get information about errors that occur during component execution

Definition

```
HIVE_API_ERROR hive_get_api_error();
```

Arguments None

Return value

- HIVE_API_ERROR: Error information

Description

This function gets information about errors that occur during component execution. See the `HIVE_API_ERROR` structure for the content of the error information.

Usage example: Getting error information

```
   :
  HIVE_OUTCOLUMN clm  = hive_outport_create_column(comp, OUTPORT1, "clm",
HIVE_DATA_SCALAR(HIVE_TYPE_INT8),   HIVE_COLUMN_OPTION_NONE);
  if (clm == nullptr) {
    HIVE_API_ERROR error = hive_get_api_error();
    HIVE_LOG_ERROR("hive_outport_create_column error(0x%x,%d,%d)", error.code,
error.system_errno, error.db_errno);
  }
   :
```

15.4.6.7 HIVE_DATA_TYPE_NAME() … Get data type name (string)

Function to get the name of the data type of SpeeDBee Synapse (string)

Definition

```
const char *HIVE_DATA_TYPE_NAME(hive_primitive_type_t primitive_type);
```

Arguments

No	Item	Description
1	`hive_primitive_type_t primitive_type`	Data type

Return value

- Returns the name of the data type passed in as a string

Description

This function gets the name of the SpeeDBee Synapse data type (hive_primitive_type_t) as a character string.

Usage example: Getting the name (string) of a data type

```
  HIVE_LOG_INFO("HIVE_DATA_TYPE_NAME: %s", HIVE_DATA_TYPE_NAME(HIVE_TYPE_INT16));
  :
```

15.4.6.8 HIVE_AGGREGATION_STAT_TYPE_NAME() … Get statistics type name (string)

Function to get the name of the statistic type of SpeeDBee Synapse (string)

Definition

```
const char *HIVE_AGGREGATION_STAT_TYPE_NAME(hive_aggregation_stat_type_t stat_type);
```

Arguments

No	Item	Description
1	`hive_aggregation_stat_type_t stat_type`	Statistic type

Return value

- Returns the name of the statistic type passed in as a string

Description

This function gets the name of the statistic type of SpeeDBee Synapse.

Usage example: Getting the name (string) of a statistic type

```
  HIVE_LOG_INFO("HIVE_AGGREGATION_STAT_TYPE_NAME: %s",
HIVE_AGGREGATION_STAT_TYPE_NAME(HIVE_AGGREGATION_STAT_TYPE_BASICSUM));
  :
```

15.4.6.9 HIVE_DATA_TYPE_FROM_NAME() … Get data type and array size from data name (string)

Function to get the data type and array size of SpeeDBee Synapse from the data name (string)

Definition

```
HIVE_DATA_TYPE HIVE_DATA_TYPE_FROM_NAME(const char *typeName);
```

Arguments

No	Item	Description
1	`char *typeName`	Data name

Return value

- HIVE_DATA_TYPE: SpeeDBee Synapse data type and array size information

Description

This function gets the data type and array size of SpeeDBee Synapse from the data name (string).

Usage example: Getting data type and array size

```
HIVE_DATA_TYPE data_type = HIVE_DATA_TYPE_FROM_NAME("int16[3]");
HIVE_LOG_INFO("HIVE_DATA_TYPE: type: %d array: %d", data_type.type, data_type.array);
:
```

15.4.6.10 hive_get_common_variable() … Get component default variable

Function to get a component default variable

Definition

```
const char *hive_get_common_variable(HIVE_COMPONENT component, const char *varname);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
1	`char *varname`	Variable name of the common variable to be acquired

Return value

The top address of the value (string) set for the specified variable name. If no data exists, null is returned.
This memory is managed by the framework and does not require release by the user. However, if the same API is called again, the previous memory area will be overwritten. According to need, the user should copy the memory area.

Description

This function acquires the value of the set common variable by specifying the variable name.

Usage example: Getting data type and array size

```
const char *project_dir = hive_get_common_variable(component, "PROJECT_DIR");
HIVE_LOG_INFO("project_dir: %s", project_dir);
:
```

15.4.6.11 hive_lock() … Exclusive control between components

Function to acquire a lock for exclusive control between components

Definition

```
HIVE_LOCK_RESULT hive_lock(HIVE_COMPONENT component, const char *key, bool interbase, uint64_t
timeout);
HIVE_LOCK_RESULT synapse_lock(HIVE_COMPONENT component, const char *key, bool interbase, uint64_t
timeout);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`const char *key`	Arbitrary string to identify the exclusive control target
3	`bool interbase`	Set to `true` when performing exclusive control across different component bases
4	`uint64_t timeout`	Timeout duration while waiting for the lock to be released

Return value

- HIVE_LOCK_OK: Lock acquired successfully
- HIVE_LOCK_TIMEOUT: Failed to acquire the lock within the specified timeout
- HIVE_LOCK_CANCELED: Stop request was received before the lock was acquired

Description

This function acquires a lock for the resource identified by the string specified in `key` to ensure exclusive control. Only one thread can acquire the lock for the same `key`. The acquired lock can be released using the corresponding API function hive_unlock(). Alternatively, if the component terminates holding the lock, it will be automatically released.

Usage example: Exclusive control logic using a device file

```
HIVE_LOCK_RESULT result = hive_lock(component, "/dev/serial", false, 10000000000);
if (result == HIVE_LOCK_TIMEOUT) {
  HIVE_LOG_ERROR("could not take lock.");
  return false;
} else if (result == HIVE_LOCK_CANCELED) {
  return true;
}

  // This section can only be executed by a single component instance

hive_unlock(component, "/dev/serial", false);
```

You may specify any arbitrary string for the `key`.

If the `interbase` argument is set to true, exclusive control can be achieved across different component bases by using the same `key`. In this case, exclusive control can also be used with custom components using the Python API's `HiveComponentBase::lock()`. If `interbase` is set to false, exclusive control is limited to components within the same component base only.

If another component instance has already acquired the lock when this function is called, execution within the function will be blocked. The maximum blocking time can be specified in nanoseconds via the `timeout` argument. (The actual granularity depends on the clock precision of the execution environment.)
Specifying `0` will return immediately if the lock cannot be acquired. To wait indefinitely until the lock is acquired, use `HIVE_LOCK_NO_TIMEOUT` as the value for `timeout`.

A thread cannot acquire the same `key` more than once simultaneously. The thread enters a lock-wait state and remains blocked until a timeout occurs or a stop request is received.

15.4.6.12 hive_unlock() … Exclusive control between components

Release a lock used for exclusive control between components

Definition

```
HIVE_LOCK_RESULT hive_unlock(HIVE_COMPONENT component, const char *key, bool interbase);
HIVE_LOCK_RESULT synapse_unlock(HIVE_COMPONENT component, const char *key, bool interbase);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance Pass the component instance exactly as it was passed to the premain/main/postmain function argument.
2	`const char *key`	Arbitrary string to identify the exclusive control target
3	`bool interbase`	Set to `true` when performing exclusive control across different component bases

Return value

- HIVE_LOCK_OK: Unlock successful
- HIVE_LOCK_ERROR: An error occurred

Description

This function releases a lock acquired using `hive_lock`. Use the same `key` and `interbase` values as used during lock acquisition.

Usage example: Exclusive control logic using a device file

```
HIVE_LOCK_RESULT result = hive_lock(component, "lock-serialdevice", false, 10000000000);
if (result == HIVE_LOCK_TIMEOUT) {
  HIVE_LOG_ERROR("could not take lock.");
  return false;
} else if (result == HIVE_LOCK_CANCELED) {
  return true;
}

  // This section can only be executed by a single component instance

hive_unlock(component, "lock-serialdevice", false);
```

15.4.7 Types, Structures, and Functions

15.4.7.1 hive_time_t … HIVE time type

Integer representing the system time

Definition

```
typedef uint64_t hive_time_t;
```

Description

This is an integer value representing the time.

It is an unsigned 64-bit integer value, expressed in elapsed nanoseconds assuming that the Unix epoch (1970-1-1 00:00:00 [UTC]) is 0.

Time	Value
1970-1-1 00:00:00. 0 (UTC)	0
1970-1-1 09:00:00. 0 (JST)	0
1970-1-1 09:00:01. 0 (JST)	1000000000
1970-1-1 09:01:00. 0 (JST)	60000000000
1970-1-1 10:00:00. 0 (JST)	3600000000000
2022-1-31 13:00:00. 0 (JST)	1643688000000000000
2022-1-31 13:00:01. 0 (JST)	1643688001000000000
2022-1-31 13:01:00. 0 (JST)	1643688060000000000
2022-1-31 14:00:00. 0 (JST)	1643691600000000000

15.4.7.2 hive_primitive_type_t … HIVE column primitive type

Enum definition of the primitive types of columns

Definition

```
typedef enum {
  HIVE_TYPE_NONE = 0,
  HIVE_TYPE_BOOLEAN,
  HIVE_TYPE_INT8,
  HIVE_TYPE_INT16,
  HIVE_TYPE_INT32,
  HIVE_TYPE_INT64,
  HIVE_TYPE_UINT8,
  HIVE_TYPE_UINT16,
  HIVE_TYPE_UINT32,
  HIVE_TYPE_UINT64,
  HIVE_TYPE_FLOAT,
  HIVE_TYPE_DOUBLE,
  HIVE_TYPE_TIMESTAMP,         // unsupported
  HIVE_TYPE_STRING,
  HIVE_TYPE_BINARY,
  HIVE_TYPE_COMPLEX,           // unsupported
  HIVE_TYPE_FILE,
  HIVE_TYPE_JSON,              // unsupported
  HIVE_TYPE_BSON,              // unsupported
  HIVE_TYPE_MESSAGEPACK,       // unsupported
  HIVE_TYPE_COMPONENT_STATUS,  // unsupported

  HIVE_TYPE_USERDEFINED = 100,// unsupported
} hive_primitive_type_t;
```

Members

No	Item	Value	Description
1	`HIVE_TYPE_NONE`	0	Incorrect type
2	`HIVE_TYPE_BOOLEAN`	1	Boolean type
3	`HIVE_TYPE_INT8`	2	8-bit signed integer
4	`HIVE_TYPE_INT16`	3	16-bit signed integer
5	`HIVE_TYPE_INT32`	4	32-bit signed integer
6	`HIVE_TYPE_INT64`	5	64-bit signed integer
7	`HIVE_TYPE_UINT8`	6	8-bit unsigned integer
8	`HIVE_TYPE_UINT16`	7	16-bit unsigned integer
9	`HIVE_TYPE_UINT32`	8	32-bit unsigned integer

No	Item	Value	Description
10	`HIVE_TYPE_UINT64`	9	64-bit unsigned integer
11	`HIVE_TYPE_FLOAT`	10	32-bit floating-point number
12	`HIVE_TYPE_DOUBLE`	11	64-bit floating-point number
13	`HIVE_TYPE_STRING`	13	String
14	`HIVE_TYPE_BINARY`	14	Binary data
15	`HIVE_TYPE_FILE`	16	File data

Description

This represents the base type of the column when creating a column for the OUT Port or retrieving column information from the IN Port.

15.4.7.3 HIVE_DATA_TYPE ... HIVE column data type

Structure to identify the data type of the column

Definition

```
typedef struct _tag_HIVE_DATA_TYPE {
  uint8_t type;
  uint8_t array;
} HIVE_DATA_TYPE;
```

Members

No	Item	Description
1	`uint8_t type`	Primitive type
2	`uint8_t array`	Number of array elements

Description

This structure indicates the data type of the column when creating a column for the OUT Port or retrieving column information from the IN Port.
The column data type can not only be the scalar type but also the array type, and so if the array type is specified, `array` is set to the number of elements. However, `array = 0` indicates that the data type is scalar.

The following primitive types supports scalar only. Always set `array` to 0 because it cannot be an array.

- `HIVE_TYPE_TIMESTAMP`
- `HIVE_TYPE_STRING`
- `HIVE_TYPE_BINARY`
- `HIVE_TYPE_FILE`

15.4.7.4 HIVE_COMPONENT … Component instance

Component instance information

Definition

```
typedef struct {
  void *user_data;
} *HIVE_COMPONENT;
```

Members

No	Item	Description
1	`void *user_data`	Dedicated member that can be freely referenced and edited on the component side

Description

This is a pointer to a structure that holds component execution information.
In addition to the above definition, other members are defined for the system side that controls components to reference and identify individual components.

It is required when calling various APIs, and so use it as an argument.

The user can view and edit `user_data` arbitrarily. Use this if you have data that should persist beyond the component's callback function.

15.4.7.5 hive_interval_proc_t … Periodic function signature

Function interface typedef specified when the hive_component_interval_call API is used.

Definition

```
typedef bool (*hive_interval_proc_t)(HIVE_COMPONENT component, void *data, hive_time_t now, int
skipped, HIVE_STATUS status);
```

Arguments

No	Item	Description
1	`HIVE_COMPONENT component`	Component instance `component` passed at the time of calling `hive_component_interval_call`.
2	`void *data`	User data Arbitrary pointer specified by `hive_component_interval_call`.
3	`hive_time_t now`	Timestamp of the time this function is called during each periodic execution
4	`int skipped`	Number of skips of the periodic task that is not processed in time and is skipped
5	`HIVE_STATUS status`	`status` used to register error information or the like

Return value

- true: To continue the periodic processing
- false: To stop the periodic processing

Description

Function interface typedef specified when the hive_component_interval_call API is used.
Define periodic functions so that they match this declaration.

The return value of this function determines whether further periodic processing continues or stops. Ensure that true is returned unless an error prevents further processing from continuing.

15.4.7.6 HIVE_COLUMN_READ_RESULT ... Result of reading column information (window data)

Structure to store the result of reading column values from the IN Port

Definition

```
typedef struct hive_HIVE_COLUMN_READ_RESULT {
  hive_time_t timerange_begin;
  hive_time_t timerange_end;
  const HIVE_INCOLUMN_LIST *incolumn_list;
  uint64_t window_id;
  uint64_t event_id;
  const struct hive_HIVE_RECORD *statistics_record;
  bool is_continued;
  const char *buffer;
  size_t buffer_size;
} HIVE_COLUMN_READ_RESULT;
```

Members

No	Item	Description
1	`hive_time_t timerange_begin`	Leading time in the time range of the data contained in this structure
2	`hive_time_t timerange_end`	Last time in the time range of the data contained in this structure
3	`const HIVE_INCOLUMN_LIST *incolumn_list`	List of columns connected to the IN Port
6	`uint64_t window_id`	Window ID of this data
6	`uint64_t event_id`	Event ID of this data
6	`HIVE_RECORD *statistics_record`	Aggregation record
6	`bool is_continued`	Whether the window continues to the next READ_RESULT
4	`const char *buffer`	Space used internally by the system. Not to access
5	`size_t buffer_size`	Space used internally by the system. Not to access

Description

This is a data type received as a return value when the hive_continuous_reader_read() API is used.
This single structure holds data of one window. The window range varies depending on the IN Port settings.

To retrieve data from this structure, use HIVE_GET_RECORD_ITERATOR() or HIVE_RECORD_ITERATOR_GET_NEXT().

15.4.7.7 HIVE_RECORD … Column data record

Structure that consolidates data of various columns with the same timestamp contained in window data

Definition

```
typedef struct hive_HIVE_RECORD {
  uint8_t record_type;
  uint8_t unused;
  uint16_t data_count;
  uint32_t data_size;
  uint64_t timestamp;
  uint32_t offset[0];
} HIVE_RECORD;
```

Members

No	Item	Description
1	`uint8_t record_type`	Record type
2	`uint8_t unused`	Unused space
3	`uint16_t data_count`	Number of data in the record
4	`uint32_t data_size`	Number of bytes of all data in the record
5	`uint64_t timestamp`	Record timestamp
6	`uint32_t offset[0]`	Space used internally by the system. Not to access

Description

This is a data set with the same timestamp in the window data acquired from the IN Port. It is acquirable with HIVE_RECORD_ITERATOR_GET_NEXT().
The number of data contained in this record, acquirable with `data_count`, is the same as the number of columns in incolumn_list of the window data.
Each piece of data in the record is accessible with `HIVE_RECORD_GET_DATA(record, i)`.

15.4.7.8 HIVE_RECORD_DATA … Record data

Structure containing individual data contained in a record

Definition

```
typedef struct hive_HIVE_RECORD_DATA {
  uint16_t index;
  uint16_t stat_type;
  uint32_t data_size;
  uint64_t timestamp;
  char data[0];
} HIVE_RECORD_DATA;
```

Members

No	Item	Description
1	`uint16_t index`	Unused space
2	`uint16_t stat_type`	Aggregation type of this data
3	`uint32_t data_size`	Data size
4	`uint64_t timestamp`	Timestamp
5	`char data[0]`	Data

Description

This is a structure that holds individual data in a record.
The `data` member is defined as a zero element char array, but actually has enough memory for `data_size`.
However, a `data_size` of 0 indicates that no data exists for this timestamp. Note that the `data` pointer will not be `NULL` even in this event.

15.4.7.9 HIVE_COLUMN_OPTIONS … Column options

Structure to store optional information for column creation

Definition

```
typedef struct _tag_HIVE_COLUMN_OPTIONS {
  hive_column_type_t type;
  double sampling_rate;
  int compression_unit;
  int fixed_size_binary;
  int max_binary_size;
} HIVE_COLUMN_OPTIONS;

#define HIVE_COLUMN_OPTION_NONE ((HIVE_COLUMN_OPTIONS){column_type_low,0,0,0,0})
#define HIVE_COLUMN_OPTION_MIDDLE(r) ((HIVE_COLUMN_OPTIONS){column_type_middle,r,0,0,0})
```

Members

No	Item	Description
1	`hive_column_type_t type`	Column type (`column_type_low` or `column_type_middle`)
2	`double sampling_rate`	Sampling rate
3	`int compression_unit`	Compression unit
4	`int fixed_size_binary`	Binary fixed size
5	`int max_binary_size`	Binary maximum size

Description

This is optional information supplied as the fifth argument when a column is created with hive_outport_create_column(). This is not required.

Macro

The macro `HIVE_COLUMN_OPTION_NONE` is the default option definition. Normally use this macro.
The macro `HIVE_COLUMN_OPTION_MIDDLE(r)` is optional information for defining the middle type column. For middle columns, sampling rate specification is required.

15.4.7.10 HIVE_API_ERROR ... API error

API error information structure

Definition

```
typedef struct hive_HIVE_API_ERROR {
  uint32_t code;
  int system_errno;
  int db_errno;
} HIVE_API_ERROR;
```

Members

No	Item	Description
1	`uint32_t code`	Numeric value representing the error type
2	`int system_errno`	System error number
3	`int db_errno`	Database error number

Description

This structure holds the error type and detailed information when an API error occurs.
If an error occurs, the `hive_get_api_error()` function is useful to get this information.

`code` contains a numerical value that identifies the error that occurred.

If the API error is a system error, the error number is stored in `system_errno`. If it is not a system error, `system_errno` is 0.

If it is attributable to a database system error, that error number is stored in `db_errno`. If it is not attributable to a database system error, `db_errno` is 0.

15.5 Custom UI JS-API reference

This section provides the JavaScript API specification for creating custom UIs in HTML. See below for instructions on creating a custom UI in HTML.

Create Custom UI: Create Definition File Using HTML

15.5.1 Callback Functions

When the parameter settings dialog is displayed, and when you save and close the settings, you need callback functions to exchange parameter information with the main UI screen of SpeeDBee Synapse.

Define the two functions described below, `onLoad` and `onSave`, in JavaScript within a custom UI (HTML). They are called from the main UI screen of SpeeDBee Synapse.

15.5.1.1 onLoad ... Custom UI load hook

Hook function called when the custom component settings dialog is displayed to the user

Definition

```
onLoad?: (editable: boolean, parameter: {}, newComponent: boolean, componentId?: string, locale:
string) => Promise<void>;
```

Arguments

- editable: True to allow editing of the currently displayed settings dialog, or false otherwise
- parameter: Saved parameter object
- newComponent: True when the component is newly created, or false when it is updated
- componentId: ID of the target component instance
- locale: Use ja for Japanese, en for English, etc., depending on the currently selected language settings.

Return value

None

Description

This property registers a function to be called when the custom component settings dialog is displayed to the user. This is required to retrieve the saved parameters, so be sure to register it as follows.

```
window.customUiLib.onLoad = function (editable, parameter, newComponent, componentId, locale) {
  :
};
```

The previously saved parameter object can be received as an argument of this function. Set the initial values in the input boxes in the HTML based on this object.

In Run Mode, `editable` is `false`. In this case, take measures, such as making the input items inactive, so that the user cannot change the settings.

Otherwise, the user can change the values on the screen. However, the parameters still cannot be saved.

The fourth argument, `componentId`, can be used as an argument when using the API in the `window.customUiLib.api` object. However, when creating a new component, `componentId` will be `undefined` because the component has not yet been created.

The fifth argument, `locale`, passes the code for the currently used language settings (for example, `ja` for Japanese, `en` for English, e.g.). Every time the custom component's settings dialog appears, it fetches updated information. This enables dynamic multilingual UI support. If multilingual support is not required, this argument can be ignored.

15.5.1.2 onSave ... Parameter save hook

Hook function that is automatically called when the user presses the Save button to save parameters in the custom component settings dialog

Definition

```
onSave?: () => Promise<{} | null>;
```

Arguments

None

Return value

- null: Returns null to cancel saving parameters.
- {}: Returns the parameters to be saved.

Description

This function is automatically executed when the user presses the Save button. Implement the processing to be performed at the time of saving as one function and register it to `onSave` as follows.

```
window.customUiLib.onSave = function () {
  :
  return ..;
};
```

When this function is called, the parameter information to be saved must be returned as an object. The object must be data that can be converted to JSON. The returned object is saved as parameters of the relevant component and the settings dialog is closed.

If null is returned, saving the parameters is canceled. In this case, the dialog is not closed. Saving can be cancelled in this way if, for example, a problem is found with the input after the check of the format of the data entered by the user on the custom UI screen.

However, since the cancellation is not automatically notified to the user, implement error display, etc. in the HTML of the custom UI to show the cause of the cancellation of saving.

15.5.2 SpeeDBee Synapse Settings Acquisition Functions

The `window.customUiLib.api` object defines utility functions that can retrieve various information set on the SpeeDBee Synapse screen. Although you are not required to use these functions, you can improve user convenience and operability for input assistance and other purposes by using them appropriately.

15.5.2.1 getComponentbases ... Get componentbase list

Function to get a list of componentbase information

Definition

```
window.customUiLib.api.getComponentbases: () => Promise<ComponentbaseResponse[]>;
```

Arguments

None

Return value

- ComponentbaseResponse[]: List of componentbase information

Description

This function gets componentbase information in a list.

15.5.2.2 getPanels ... Get panel list

Function to get a panel list

Definition

```
window.customUiLib.api.getPanels: () => Promise<PanelResponse[]>;
```

Arguments

None

Return value

- PanelResponse[]: List of panel information

Description

This function gets information of all panels in a list.

15.5.2.3 getPanel ... Get panel information

Function to get panel information

Definition

```
window.customUiLib.api.getPanel: (id: number) => Promise<PanelResponse>;
```

Arguments

No	Item	Description
1	`id` *number*	Panel ID

Return value

- PanelResponse: Panel information

Description

This function gets the panel information for the specified panel ID.

15.5.2.4 getComponents ... Get component list

Function to get a component list

Definition

```
window.customUiLib.api.getComponents: (panelId?: number) => Promise<ComponentResponse[]>;
```

Arguments

No	Item	Description
1	`panelId` *number	undefined*

Return value

- ComponentResponse[]: Component list

Description

This function gets component information in a list.
You can specify a panel ID to get only the components placed on that panel.

15.5.2.5 getComponent ... Get component information

Function to get component information

Definition

```
window.customUiLib.api.getComponent: (id: string) => Promise<ComponentResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- ComponentResponse: Component information

Description

This function gets component information for the specified component ID.

15.5.2.6 getInPort ... Get inport setting

Function to get inport setting

Definition

```
window.customUiLib.api.getInPort: (id: string) => Promise<InportResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- InportResponse: Inport setting of the component

Description

This function gets the inport setting of the component.
The inport setting includes settings for incoming data, aggregation, delay time, and events.

15.5.2.7 getInPortData ... Get IN Port information

Function to get information about the data received at the IN Port

Definition

```
window.customUiLib.api.getInPortData: (id: string) => Promise<PortDataResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataResponse: Information of the data received at the IN Port of the component

Description

This function gets information about the data received at the IN Port of the component.
The IN Port information includes information such as the type of the incoming data, whether it is an array, etc.

15.5.2.8 getInPortDataValue ... Get IN Port data

Function to get the data received at the IN Port

Definition

```
window.customUiLib.api.getInPortDataValue: (id: string) => Promise<PortDataValueResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataValueResponse: Data received at the IN Port of the component

Description

This function gets the data and timestamp received at the IN Port of the component.
The value that can be acquired is the most recent one for each data.

15.5.2.9 getSystemPort ... Get system port setting

Function to get system port setting

Definition

```
window.customUiLib.api.getSystemPort: (id: string) => Promise<OutportResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- OutportResponse: System port setting of the component

Description

This function gets the system port setting of the component.
The system port setting includes settings for the storage, conversion, conditions of the data to be emitted, and added comments.

15.5.2.10 getSystemPortData ... Get SYSTEM Port information

Function to get information about the data emitted from the SYSTEM Port

Definition

```
window.customUiLib.api.getSystemPortData: (id: string) => Promise<PortDataResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataResponse: Information of the data emitted from the SYSTEM Port of the component

Description

This function gets information about the data emitted from the SYSTEM Port of the component.
The SYSTEM Port information includes information such as the type of data to be emitted, whether it is an array, etc.

15.5.2.11 getSystemPortDataValue ... Get SYSTEM Port data

Function to get the latest data emitted from the SYSTEM Port

Definition

```
window.customUiLib.api.getSystemPortDataValue: (id: string) => Promise<PortDataValueResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataValueResponse: Data emitted from the SYSTEM Port of the component

Description

This function gets the data and timestamp emitted from the SYSTEM Port of the component. The value to be acquired is the most recent one for each data.

15.5.2.12 getOutPort ... Get outport setting

Function to get outport setting

Definition

```
window.customUiLib.api.getOutPort: (id: string) => Promise<OutportResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- OutportResponse: Outport setting of the component

Description

This function gets the outport setting of the component.
The outport setting includes settings for the storage, conversion, conditions of the data to be emitted, and added comments.

15.5.2.13 getOutPortData ... Get OUT Port information

Function to get information about the data emitted from the OUT Port

Definition

```
window.customUiLib.api.getOutPortData: (id: string) => Promise<PortDataResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataResponse: Information of the data emitted from the OUT Port of the component

Description

This function gets information about the data emitted from the OUT Port of the component.
The OUT Port information includes information such as the type of data to be emitted, whether it is an array, etc.

15.5.2.14 getOutPortDataValue ... Get OUT Port data

Function to get the latest data emitted at the OUT Port

Definition

```
window.customUiLib.api.getOutPortDataValue: (id: string) => Promise<PortDataValueResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to get

Return value

- PortDataValueResponse: Data emitted from the OUT Port of the component

Description

This function gets the data and timestamp emitted from the OUT Port of the component. The value to be acquired is the most recent one for each data.

15.5.2.15 runMethod ... Run component instance method

Function to execute the component instance method.

> **Caution**
>
> You must be a Manager or higher to execute a method.

Definition

```
runMethod: (id: string, methodName: string, parameter: string | object | null) =>
Promise<ComponentMethodResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the component to execute
2	`methodName` *string*	Name of the method to get
3	`parameter` *string	object

Return value

- ComponentMethodResponse: Result and return value of the executed instance method

Description

This function executes the instance method implemented in the component.
For information about instance methods, see .

15.5.2.16 getFlowlinks ... Get flowlink list

Function to get a flowlink list

Definition

```
window.customUiLib.api.getFlowlinks: (componentId?: string) => Promise<FlowlinkResponse[]>;
```

Arguments

No	Item	Description
1	`componentId` *string*	ID of the component to get (all flowlinks if not specified)

Return value

- FlowlinkResponse[]: Flowlink list

Description

This function gets flowlink information.
If a component ID is specified, the flowlinks connected to that component are acquired, otherwise all flowlinks are acquired.

15.5.2.17 getFlowlink ... Get flowlink information

Function to get flowlink information

Definition

```
window.customUiLib.api.getFlowlink: (id: string) => Promise<FlowlinkResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the flowlink to get

Return value

- FlowlinkResponse: Flowlink information

Description

This function gets flowlink information by ID search.

15.5.2.18 getFlowlinkFilter ... Get flowlink filter information

Function to get the filter information set for a flowlink

Definition

```
window.customUiLib.api.getFlowlinkFilter: (id: string) => Promise<FlowlinkFiltersResponse>;
```

Arguments

No	Item	Description
1	`id` *string*	ID of the flowlink to get

Return value

- FlowlinkFiltersResponse: Flowlink information

Description

This function gets the filter information set for the flowlink by ID search.

15.5.3 Types

15.5.3.1 ComponentbaseResponse

Componentbase information registered in SpeeDBee Synapse

Definition

```
type ComponentbaseResponse = {
    id: string,
    name: string,
    tag: string,
    inports: number,
    outports: number,
    required_point: number,
    error_types: {
        id: string,
        parameter_names: string[],
    }[],
}
```

Configuration

No	Item	Description
1	`id`	Componentbase ID
2	`name`	Componentbase name
3	`tag`	Tag

No	Item	Description
4	`inports`	Number of IN Ports
5	`outports`	Number of OUT Ports
6	`required_point`	Number of points required at startup
7	`error_types`	List of error type objects

> *error_types*
>
> | No | Item | Description |
> |---|---|---|
> | 1 | `id` | Error ID |
> | 2 | `parameter_names` | Parameter name |

Description

This is a data type for the base information of each component.
A list can be acquired as a return value when getComponentbases is used.

15.5.3.2 PanelResponse

Panel information

Definition

```
type PanelResponse = {
    id: number,
    order: number,
    name: string,
    type: string,
    options: any,
    components: any[],
};
```

Configuration

No	Item	Description
1	`id`	Panel ID

No	Item	Description
2	`order`	Display order
3	`name`	Name
4	`type`	Kind
5	`options`	Options
6	`components`	Component information

Description

This is a data type for each registered panel.
It can be acquired as a return value when getPanels or getPanel is used.

15.5.3.3 ComponentResponse

Component information

Definition

```
type ComponentResponse = {
    id: string,
    baseid: string,
    name: string,
    enable: boolean,
    parameter: any | undefined,
    options: any | undefined,
};
```

Configuration

No	Item	Description
1	`id`	Component ID
2	`baseid`	Componentbase ID
3	`name`	Name
4	`enable`	Enable/Disable

No	Item	Description
5	`parameter`	Parameters
6	`options`	Options

Description

This is a data type for getting each component set by the user.
It can be acquired as a return value when getComponents or getComponent is used.

15.5.3.4 InportResponse

Inport setting

Definition

```
type InportResponse = {
    setting_option: 'reception' | 'regexp',
    column_data: {
        target: {
            source_name: string,
            data_name: string,
            data_types?: string[],
        }[],
        raw_data: boolean,
        typical_value?: string,
        basic_statistics: string[],
    }[],
    window: {
        window_type: 'tumbling' | 'value_change',
        time_span?: number,
        columns?: {
            source_name: string,
            data_name: string,
            data_types: string[]
        },
        max_duration?: number
    },
    delaytime: number,
    event?: {
        source_name: string,
        data_name: string,
        pre_event_time: number,
        post_event_time: number
    },
    column_order_output?: {
        specify_columns: {
            source_name: string,
            data_name: string,
            outputname?: string,
        }[],
        order_type: string
    }
};
```

Configuration

No	Item	Description
1	`setting_option`	Whether to filter by regular expression or by data
2	`column_data`	Input data information
3	`window`	Window settings
4	`delaytime`	Delay time (nanoseconds)

No	Item	Description
5	`event`	Settings for turning data to the IN Port ON/OFF in response to an external event
6	`column_order_output`	Settings for sorting data coming into the IN Port and aliases

column_data

No	Item	Description
1	`target`	Pattern of source and data to be aggregated
2	`raw_data`	Use of raw data
3	`typical_value`	Setting of representative values. Newest value (newer) or oldest value (older)
4	`basic_statistics`	Basic stats data

target

No	Item	Description
1	`source_name`	Regular expression indicating the pattern of the target component name
2	`data_name`	Regular expression indicating the pattern of the target data name
3	`data_types`	List of the target data types (omission/null matches any type)

window

No	Item	Description
1	`window_type`	Window type
2	`time_span`	Search time width (nanoseconds)
3	`columns`	List of the target data information
4	`max_duration`	Maximum window time width (nanoseconds)

columns

No	Item	Description
1	`source_name`	Regular expression indicating the pattern of the target component name
2	`data_name`	Regular expression indicating the pattern of the target data name
3	`data_types`	List of the target data types (omission/null matches any type)

event

No	Item	Description
1	`source_name`	Name of the component that generates the column for the external event
2	`data_name`	Column name of the external event (only bool type is valid)
3	`pre_event_time`	Pre-event time (nanoseconds)
4	`post_event_time`	Post-event time (nanoseconds)

column_order_output

No	Item	Description
1	`specify_columns`	Sort order and output name
2	`order_type`	Sort order of data that is not specified

specify_columns

No	Item	Description
1	`source_name`	Component name
2	`data_name`	Data name
3	`outputname`	Output name assigned to the column

Description

This is a data type for the inport setting of the component.
It can be acquired as a return value when getInPort is used.

15.5.3.5 OutportResponse

System port, outport setting

Definition

```
type OutportResponse = {
    setting_option: 'reception' | 'regexp';
    pattern_settings: {
        target: {
            source_name?: string,
            data_name: string,
            data_types?: string[],
        }[],
        storage_type: string,
        file_index: boolean,
        conversion?: {
            data_type?: string,
            expression?: string,
        },
        value_filter?: {
            expression: string,
        },
        comment?: string,
    }[];
};
```

Configuration

No	Item	Description
1	`target`	Pattern definition for specifying target data
2	`storage_type`	Storage
3	`file_index`	Flag to create a search index
4	`conversion`	Conversion
5	`value_filter`	Conditions
6	`comment`	Comment

target

No	Item	Description
1	`source_name`	Regular expression indicating the pattern of the target component name
2	`data_name`	Regular expression indicating the pattern of the target data name
3	`data_types`	List of the target data types (omission/null matches any type)

conversion

No	Item	Description
1	`data_type`	Data type after conversion
2	`expression`	Conversion expression

value_filter

No	Item	Description
1	`expression`	Conditional expression

This is a data type for the outport setting of the component.
It can be acquired as a return value when getSystemPort or getOutPort is used.

15.5.3.6 PortDataResponse

Information about the data received and emitted by the IN/OUT Ports

Definition

```
type PortDataResponse = {
    id: string;
    name: string;
    data?: {
        name: string;
        source?: string;
        column_type: string,
        sampling_rate?: number,
        data_type: string;
        data_array?: number;
    }[]
};
```

Configuration

No	Item	Description
1	`id`	Component ID
2	`name`	Name
3	`data`	Data information received at the IN Port

> *data*
>
> | No | Item | Description |
> |---|---|---|
> | 1 | `name` | Data name |
> | 2 | `source` | Component name |
> | 3 | `column_type` | Column type |
> | 4 | `sampling_rate` | Sampling rate |
> | 5 | `data_type` | Data type |
> | 6 | `data_array` | Array size |

Description

This is a data type for the data received and emitted by the IN/OUT Ports of the component.
It can be acquired as a return value when getInPortData, getSystemPortData, or getOutPortData is used.

15.5.3.7 PortDataValueResponse

Data received and emitted by the IN/OUT Ports

Definition

```
type PortDataValueResponse = {
    id: string;
    name: string;
    data?: {
        value: any | any[];
        timestamp?: number;
        name: string;
        source?: string;
        data_type: string;
        data_array?: number;
    }[]
};
```

Configuration

No	Item	Description
1	`id`	Component ID
2	`name`	Name
3	`data`	Data received at the IN Port

> *data*
>
> | No | Item | Description |
> | --- | --- | --- |
> | 3 | `value` | Value |
> | 4 | `timestamp` | Timestamp (nanoseconds) |
> | 1 | `name` | Data name |
> | 2 | `source` | Component name |
> | 5 | `data_type` | Data type |
> | 6 | `data_array` | Array size |

Description

This is a data type for the data received and emitted by the IN/OUT Ports of the component.
It can be acquired as a return value when getInPortDataValue, getSystemPortDataValue, or getOutPortDataValue is used.

15.5.3.8 ComponentMethodResponse

Result after executing the instance method of component

Definition

```
type ComponentMethodResponse = {
    method_name: string,
    result: string | object | null
};
```

Configuration

No	Item	Description
1	`method_name`	Name of the executed function
2	`result`	Return value of the executed function

Description

This is a data type for the data returned after executing the instance method of the component.
The content of `result` depends on the implementation of the custom component, so implement the return value of the custom component according to the processing that is to be performed later.

15.5.3.9 FlowlinkResponse

Flowlink information

Definition

```
type FlowlinkResponse = {
    id: string,
    source?: {
        id: string,
        port: number,
    },
    destination?: {
        id: string,
        port: number,
    }
};
```

Configuration

No	Item	Description
1	`id`	Flowlink ID
2	`source`	Connection source information
3	`destination`	Connection destination information

> *source, destination*
>
> | No | Item | Description |
> |---|---|---|
> | 1 | `id` | Port ID of the connection destination |
> | 2 | `port` | Port number |

Description

This is a data type for the flowlink to which the component is connected.
It can be acquired as a return value when getFlowlinks or getFlowlink is used.

15.5.3.10 FlowlinkFiltersResponse

Flowlink filter information

Definition

```
type FlowlinkFiltersResponse = {
    setting_option: 'reception' | 'regexp',
    pattern_settings: {
        target: {
            data_name: string,
            data_types?: string[]
        },
        pass: boolean,
    }[],
};
```

Configuration

No	Item	Description
1	`setting_option`	Whether to filter by regular expression or by data
2	`pattern_settings`	List of filter setting objects

pattern_settins

No	Item	Description
1	`target`	Pattern definition for specifying target data
2	`pass`	Flag to determine whether to pass the value

target

No	Item	Description
1	`data_name`	Regular expression indicating the pattern of the target data name
2	`data_types`	List of the target data types (omission/null matches any type)

Description

This is a data type for the filter set for the flowlink to which the component is connected. It can be acquired as a return value when getFlowlinkFilter is used.

15.6 Open source used

SpeeDBee Synapse uses open-source. See below for open-source software and licensing information.

15.6.1 Frontend

OSS and Licenses Used in SpeeDBee Synapse WEBUI Frontend

- @babel/helper-string-parser | MIT
- @babel/helper-validator-identifier | MIT
- @babel/parser | MIT
- @babel/types | MIT
- @jridgewell/sourcemap-codec | MIT
- @kurkle/color | MIT
- @quasar/extras | MIT
- @vue/compiler-core | MIT
- @vue/compiler-dom | MIT
- @vue/compiler-sfc | MIT
- @vue/compiler-ssr | MIT
- @vue/reactivity | MIT
- @vue/runtime-core | MIT
- @vue/runtime-dom | MIT
- @vue/server-renderer | MIT
- @vue/shared | MIT
- chart.js | MIT
- cross-env | MIT
- cross-spawn | MIT
- csstype | MIT
- deep-is | MIT
- drawflow | MIT
- emittery | MIT
- entities | BSD-2-Clause
- escodegen | BSD-2-Clause
- esprima | BSD

- esprima | BSD-2-Clause
- estraverse | BSD-2-Clause
- estree-walker | MIT
- esutils | BSD-2-Clause
- fast-levenshtein | MIT
- hls.js | Apache-2.0
- isexe | ISC
- jsonpath | MIT
- levn | MIT
- magic-string | MIT
- maska | MIT
- nanoid | MIT
- optionator | MIT
- path-key | MIT
- picocolors | ISC
- postcss | MIT
- prelude-ls | MIT
- quasar | MIT
- shebang-command | MIT
- shebang-regex | MIT
- sortablejs | MIT
- source-map-js | BSD-3-Clause
- source-map | BSD-3-Clause
- static-eval | MIT
- type-check | MIT
- typescript | Apache-2.0
- underscore | MIT
- vue-chartjs | MIT
- vue | MIT
- when-json-met-bigint | MIT
- which | ISC
- word-wrap | MIT

15.6.2 Backend

OSS and Licenses Used in SpeeDBee Synapse WEBUI Backend

- PyYAML | MIT License
- aiofiles | Apache Software License
- aiohappyeyeballs | Python Software Foundation License
- aiohttp | Apache-2.0 AND MIT
- aiohttp-sse-client | Apache Software License
- aiorwlock | Apache Software License
- aiosignal | Apache Software License
- annotated-types | MIT License
- anyio | MIT
- async-timeout | Apache Software License
- attrs | MIT
- bcrypt | Apache Software License
- certifi | Mozilla Public License 2.0 (MPL 2.0)
- cffi | MIT
- click | BSD-3-Clause
- colorama | BSD License
- cryptography | Apache-2.0 OR BSD-3-Clause
- ecdsa | MIT
- exceptiongroup | MIT License
- execnet | MIT
- fastapi | MIT License
- frozenlist | Apache-2.0
- grafana-api-sdk | Apache-2.0 License
- h11 | MIT License
- httpcore | BSD License
- httptools | MIT
- httpx | BSD-3-Clause
- idna | BSD-3-Clause
- iniconfig | MIT
- itsdangerous | BSD License
- jsonschema-specifications | MIT

- mslex | Apache Software License
- multidict | Apache License 2.0
- mypy_extensions | MIT
- packaging | Apache Software License; BSD License
- passlib | BSD
- pluggy | MIT License
- propcache | Apache Software License
- psutil | BSD License
- pyasn1 | BSD-2-Clause
- pycodestyle | MIT
- pycparser | BSD-3-Clause
- pycryptodome | BSD License; Public Domain
- pycryptodomex | BSD License; Public Domain
- pydantic | MIT
- pydantic_core | MIT
- python-dotenv | BSD-3-Clause
- python-jose | MIT License
- python-multipart | Apache-2.0
- pyzipper | MIT License
- referencing | MIT
- rpds-py | MIT
- rsa | Apache Software License
- six | MIT License
- sniffio | Apache Software License; MIT License
- sse-starlette | BSD License
- starlette | BSD-3-Clause
- stream-zip | MIT License
- taskipy | MIT License
- tomli | MIT
- types-pyasn1 | Apache-2.0
- typing-inspection | MIT
- typing_extensions | PSF-2.0
- uvicorn | BSD-3-Clause

- uvloop | Apache Software License; MIT License
- watchfiles | MIT License
- websockets | BSD-3-Clause
- yarl | Apache Software License

15.6.3 Core

OSS and Licenses Used in SpeeDBee Synapse Core

- cpp-httplib | MIT
- cppcodec | MIT
- ghc | MIT
- martinmoene-optional-lite | Boost Software License (BSL) Version 1.0
- martinmoene-variant-lite | Boost Software License (BSL) Version 1.0
- nlohmann-json | MIT
- tiny-AES-c | The Unlicense
- PicoSHA2 | MIT
- takamin/win-c | MIT
- tronkko/dirent | MIT
- libuuid | BSD-3-Clause
- libpython | Python Software Foundation License Version 2
- openSSL | Apache-2.0

15.6.4 Components

OSS and Licenses Used in SpeeDBee Synapse Standard Components

- pycomm3 | MIT
- paramiko | GNU Lesser General Public License Version 2.1
- wheel | MIT
- Eclipse Paho MQTT Python Client | Eclipse Public License 2.0 (EPL-2.0) OR Eclipse Distribution License 1.0 (EDL-1.0)
- Eclipse Paho MQTT C Client | Eclipse Public License 2.0 (EPL-2.0) OR Eclipse Distribution License 1.0 (EDL-1.0)
- OpenSSL | Apache-2.0
- json-c | MIT
- nlohmann-json | MIT

- tronkko/dirent | MIT
- ghc | MIT
- martinmoene-optional-lite | Boost Software License (BSL) Version 1.0
- JSONCONS | Boost Software License (BSL) Version 1.0

15.6.5 Viewer

OSS and Licenses Used in i-Trigger Components

- FFmpeg | This software uses code of FFmpeg licensed under the LGPLv2.1
- its source can be downloaded here
- Project License Info:FFmpeg License and Legal Considerations

OSS and Licenses Used in i-Trigger Viewer Frontend

- @babel/helper-string-parser | MIT
- @babel/helper-validator-identifier | MIT
- @babel/parser | MIT
- @babel/types | MIT
- @jridgewell/sourcemap-codec | MIT
- @kurkle/color | MIT
- @quasar/extras | MIT
- @vue/compiler-core | MIT
- @vue/compiler-dom | MIT
- @vue/compiler-sfc | MIT
- @vue/compiler-ssr | MIT
- @vue/reactivity | MIT
- @vue/runtime-core | MIT
- @vue/runtime-dom | MIT
- @vue/server-renderer | MIT
- @vue/shared | MIT
- chart.js | MIT
- chartjs-adapter-moment | MIT
- chartjs-plugin-annotation | MIT
- cross-env | MIT
- cross-spawn | MIT

- csstype | MIT
- drawflow | MIT
- emittery | MIT
- entities | BSD-2-Clause
- estree-walker | MIT
- hls.js | Apache-2.0
- isexe | ISC
- magic-string | MIT
- maska | MIT
- moment | MIT
- nanoid | MIT
- path-key | MIT
- picocolors | ISC
- postcss | MIT
- quasar | MIT
- shebang-command | MIT
- shebang-regex | MIT
- sortablejs | MIT
- source-map-js | BSD-3-Clause
- typescript | Apache-2.0
- vue-chartjs | MIT
- vue | MIT
- when-json-met-bigint | MIT
- which | ISC

15.7 Index

15.7.1 A

Access token

Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Publish access token
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Common Items
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Acquire the execution status of all components
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Acquire the execution status of individual component
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Start batch execution of components
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Stop batch execution of components

Action

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Introduction Flow/Running the Collection Service/Test
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Email Sending
Appendix/Error code list/Component default
Appendix/Error code list/Resource Collector
Appendix/Error code list/PLC Collector
Appendix/Error code list/Modbus Collector
Appendix/Error code list/EtherNet/IP Collector
Appendix/Error code list/Compute Collector
Appendix/Error code list/MQTT Collector
Appendix/Error code list/MQTT Emitter
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/FTP/FTPS/SFTP Emitter
Appendix/Error code list/JSON Serializer
Appendix/Error code list/CSV Serializer
Appendix/Error code list/Event Trigger
Appendix/Error code list/Shell Command
Appendix/Error code list/PLC Writing
Appendix/Error code list/Modbus Writing
Appendix/Error code list/EtherNet/IP Writing
Appendix/Error code list/Send mail
Appendix/Error code list/Core Internal
Appendix/Error code list/DB query service
Appendix/Error code list/Error manager
Appendix/Error code list/Event data manager
Appendix/Error code list/VIDEO Collector (i-Trigger Extension package)

Aggregation

Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Component Default Settings/IN Port/Aggregation Configuration
Appendix/Custom component Python-API reference/OutColumn Objects/insert_file_ref
Appendix/Custom component Python-API reference/AggregationType Objects
Appendix/Custom component Python-API reference/AggregationTypeSet Objects
Appendix/Custom component Python-API reference/InColumn Objects
Appendix/Custom component Python-API reference/HiveRecordData Objects
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_COLUMN_READ_RESULT ...
Result of reading column information (window data)
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_RECORD_DATA ... Record data

15.7.2 B

basic stats

Explanation of Functions/Logic (Event)/Event Trigger/Setting Method
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Basic Stats/Basic Information
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Basic Stats/Setting Method

15.7.3 C

Cloud Emitter for AWS

Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/Cloud Emitter for AWS
Security Guide/Using Encrypted Communications
Appendix/Error code list/Cloud Emitter for AWS

Cloud Emitter for Azure

Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/Cloud Emitter for Azure
Security Guide/Using Encrypted Communications
Appendix/Error code list/Cloud Emitter for Azure

Collection

Overview of SpeeDBee Synapse/Features of SpeeDBee Synapse
Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Running the Collection Service/Start/Stop Collection Service
Introduction Flow/Running the Collection Service/Configuration Tuning
Introduction Flow/Running the Collection Service/Stop Collection Service
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Component Default Settings/IN Port/Set Delay Time


Collector

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Setting of Components/Configuring the Resource Collector
Introduction Flow/Setting of Components/Configuring the CSV Serializer
Introduction Flow/Running the Collection Service/Test
Introduction Flow/Running the Collection Service/Configuration Tuning
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Data directory setting
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/Resource Collector
Explanation of Functions/Collector/PLC Collector
Explanation of Functions/Collector/Modbus Collector
Explanation of Functions/Collector/EtherNet/IP Collector
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Serializer/JSON Serializer
Explanation of Functions/Serializer/CSV Serializer
Explanation of Functions/Development Components/Dummy Collector
Explanation of Functions/Component Default Settings/OUT Port/Conversion Configuration
Explanation of Functions/Component Default Settings/OUT Port/Set Registration Conditions
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Security Guide/Using Encrypted Communications
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
Execution License/Execution license operation method/Quick Tutorial/Setting up of the execution license operating environment
i-Trigger (Expansion Custom Package)/Installation/Uninstallation
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Video Record Play
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Event Record Play/Play
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Live Camera
i-Trigger (Expansion Custom Package)/Components/VIDEO Collector
i-Trigger (Expansion Custom Package)/Components/Event Recorder
Appendix/Error code list/Component default
Appendix/Error code list/Resource Collector
Appendix/Error code list/PLC Collector
Appendix/Error code list/Modbus Collector
Appendix/Error code list/EtherNet/IP Collector
Appendix/Error code list/Compute Collector
Appendix/Error code list/MQTT Collector
Appendix/Error code list/VIDEO Collector (i-Trigger Extension package)


Component

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Running the Collection Service/Start/Stop Collection Service
Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/General Settings
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Component default
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Custom (Python)
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Custom (C)
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/Panel/Component Search
Explanation of Functions/Flow Editor/Screen Operation/Component
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Logic (Event)/Event Data/Setting Method
Explanation of Functions/Logic (Event)/Event Data/Setting Example
Explanation of Functions/Logic (Event)/Event Trigger/Setting Method
Explanation of Functions/Logic (Event)/Event Trigger/Setting Example
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Basic Stats/Setting Method
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/FFT/Setting Method
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Moving Average/Setting Method
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion/Setting Method (Common)
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion/Setting Method (Array Split)
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion/Setting Method (Bit Conversion)
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion/Setting Method (Change Data Name)
Explanation of Functions/Action/PLC Writing
Explanation of Functions/Action/Modbus Writing
Explanation of Functions/Action/EtherNet/IP Writing
Explanation of Functions/System Components/DB Query Service/Configuration Items
Explanation of Functions/System Components/Core Internal/Configuration Items
Explanation of Functions/System Components/Error Manager/Configuration Items
Explanation of Functions/System Components/Event Data Manager/Configuration Items
Explanation of Functions/Component Default Settings/IN Port/Aggregation Configuration
Explanation of Functions/Status Timeline/Status list
Explanation of Functions/Various Logs/Component Individual Log
Explanation of Functions/Various Logs/Download
Explanation of Functions/Managing Configuration Information (Project)/Project Overview

Advanced Use/Creating and Using Custom Components/Overview/Custom Component
Advanced Use/Creating and Using Custom Components/Overview/Component Bases and Component Instances
Advanced Use/Creating and Using Custom Components/Overview/Lifecycle of Component Instance
Advanced Use/Creating and Using Custom Components/Overview/Input and Output Ports of Components
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Sample Custom Components
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Custom Component Implementation Details
Advanced Use/Creating and Using Custom Components/Creating Python custom components/API for Custom Components
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Sample Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Custom Component Implementation Details
Advanced Use/Creating and Using Custom Components/Creating C language custom components/API for Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Creating Custom UI
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/JavaScript File Coding
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Function Executed When the Parameters Dialog Appears
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Controlling Edit Restrictions
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to Python Custom Component
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to C Custom Component
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Overview of SCCDE
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Initializing the Custom Component Administration Directory
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Adding Custom Components
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Testing Custom Components
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Register/Update
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Download
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Delete
Advanced Use/Creating and Using Custom Components/Custom component development environment

(SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Protect
Advanced Use/Creating and Using Custom Components/Custom component development environment
(SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Unprotect
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Common Items
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Acquire the execution status of all components
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Acquire the execution status of individual component
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Start batch execution of components
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Stop batch execution of components
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Email Sending
Backward Compatibility/Backwards Compatible Target
Execution License/Operating specifications of the execution license usage environment
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
i-Trigger (Expansion Custom Package)/Components/VIDEO Collector
i-Trigger (Expansion Custom Package)/Components/Event Recorder
Appendix/Error code list/Component default
Appendix/Error code list/Compute Collector
Appendix/Error code list/PLC Writing
Appendix/Error code list/Modbus Writing
Appendix/Error code list/Event Recorder (i-Trigger Extension package)
Appendix/Custom component Python-API reference/HiveComponentBase Objects
Appendix/Custom component Python-API reference/DataType Objects
Appendix/Custom component Python-API reference/RunningStatus Objects/add_error
Appendix/Custom component Python-API reference/HiveCompError Objects
Appendix/Custom component Python-API reference/HiveCompStatus Objects
Appendix/Custom component Python-API reference/HiveInPortEventSetting Objects
Appendix/Custom component Python-API reference/SynapseLock Objects
Appendix/Custom component C-API reference/OUT Port Functions/hive_outport_create_column() … Create new column
Appendix/Custom component C-API reference/OUT Port Functions/hive_outport_register_confirmed_time() … Register column data confirmed time in status information
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_get_column_list() … Get input column information
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_continuous_reader() … Get Reader for continuous retrieval for column data
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_timerange_reader() … Get Reader for time range retrieval of column data

Appendix/Custom component C-API reference/Component Status Functions
Appendix/Custom component C-API reference/Other Functions/hive_component_interval_call() … Periodic function
Appendix/Custom component C-API reference/Other Functions/hive_component_save_running_info() … Save run-time information
Appendix/Custom component C-API reference/Other Functions/hive_component_load_running_info() … Load saved run-time information
Appendix/Custom component C-API reference/Other Functions/hive_component_runnable() … Component sustainability
Appendix/Custom component C-API reference/Other Functions/hive_get_common_variable() … Get component default variable
Appendix/Custom component C-API reference/Other Functions/hive_lock() … Exclusive control between components
Appendix/Custom component C-API reference/Other Functions/hive_unlock() … Exclusive control between components
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_COMPONENT … Component instance
Appendix/Custom component C-API reference/Types, Structures, and Functions/hive_interval_proc_t … Periodic function signature
Appendix/Custom UI JS-API reference/Callback Functions/onLoad ... Custom UI load hook
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getComponentbases ... Get componentbase list
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getComponents ... Get component list
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getComponent ... Get component information
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/runMethod ... Run component instance method
Appendix/Custom UI JS-API reference/Types/ComponentbaseResponse
Appendix/Custom UI JS-API reference/Types/PanelResponse
Appendix/Custom UI JS-API reference/Types/ComponentResponse
Appendix/Custom UI JS-API reference/Types/InportResponse
Appendix/Custom UI JS-API reference/Types/PortDataResponse
Appendix/Custom UI JS-API reference/Types/PortDataValueResponse
Appendix/Custom UI JS-API reference/Types/ComponentMethodResponse

Component Base

Advanced Use/Creating and Using Custom Components/Overview/Component Bases and Component Instances

Compute Collector

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Appendix/Error code list/Component default
Appendix/Error code list/Compute Collector

conditional expression

Explanation of Functions/Logic (Event)/Event Data/Setting Method
Explanation of Functions/Logic (Event)/Event Trigger/Setting Method
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SELECT
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Explanation of Functions/Arithmetic and Conditional Expressions/Expression and conditional expression (common)
Explanation of Functions/Arithmetic and Conditional Expressions/Conditional expression
Advanced Use/Global Variables/Overview of Global Variables
Advanced Use/Global Variables/Syntax of Global Variable
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Appendix/Error code list/Component default
Appendix/Error code list/Event Trigger

Conversion

Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion
Explanation of Functions/Component Default Settings/OUT Port/Conversion Configuration
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Appendix/Custom UI JS-API reference/Types/OutportResponse

Core

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Overview of engine functions/Operating Specifications/Normal Stop
Explanation of Functions/Overview of engine functions/Operating Specifications/Abnormal Stop (e.g., power interruption)
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Server Settings
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/Cloud Emitter for AWS
Explanation of Functions/System Components/Core Internal
Explanation of Functions/Various Logs/Log Types
Explanation of Functions/Various Logs/Download
Explanation of Functions/Database Query (SQL)/Cautions in using the DB query service
Advanced Use/Linking with Grafana/Creating a Graph/Start DB Query Service
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Core Internal
Appendix/Error code list/Error manager

Core Internal

Explanation of Functions/System Components/Core Internal
Appendix/Error code list/Core Internal

CSV Serializer

Introduction Flow/Setting of Components/Configuring the CSV Serializer
Introduction Flow/Setting of Components/File Emitter Settings
Explanation of Functions/Serializer/CSV Serializer
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/CSV Serializer


Custom Component

Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Custom (Python)
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Custom (C)
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Advanced Use/Creating and Using Custom Components/Overview/Custom Component
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Sample Custom Components
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Custom Component Implementation Details
Advanced Use/Creating and Using Custom Components/Creating Python custom components/API for Custom Components
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Sample Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Custom Component Implementation Details
Advanced Use/Creating and Using Custom Components/Creating C language custom components/API for Custom Components
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Creating Custom UI
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to Python Custom Component
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to C Custom Component
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Initializing the Custom Component Administration Directory
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Adding Custom Components
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Testing Custom Components
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Register/Update
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Download
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Delete
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Protect
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Unprotect
Execution License/Operating specifications of the execution license usage environment

15.7.4 D

data collection flow

Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Flow Editor/About the mode
Explanation of Functions/Flow Editor/Screen Operation/Component/Start/Stop
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Controlling Edit Restrictions

Data Conversion

Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Data Conversion

DB query service

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Publish access token
Explanation of Functions/System Components/DB Query Service
Explanation of Functions/Database Query (SQL)/DB Query Overview
Explanation of Functions/Database Query (SQL)/DB Query Detail/Web-API Specification
Explanation of Functions/Database Query (SQL)/Cautions in using the DB query service
Advanced Use/Linking with Grafana/Creating a Graph/Registering Data
Advanced Use/Linking with Grafana/Manually Setting a Grafana Data Source
Execution License/Operating specifications of the execution license usage environment
Execution License/Notes on the execution license operating environment
Appendix/Error code list/DB query service

15.7.5 E

Edit Mode

Explanation of Functions/Flow Editor/About the mode
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu
Explanation of Functions/Flow Editor/Screen Operation/Panel/Add
Explanation of Functions/Flow Editor/Screen Operation/Panel/Changing a name
Explanation of Functions/Flow Editor/Screen Operation/Panel/Rearranging
Explanation of Functions/Flow Editor/Screen Operation/Panel/Delete
Explanation of Functions/Flow Editor/Screen Operation/Panel/Replicate
Explanation of Functions/Flow Editor/Screen Operation/Component/Add
Explanation of Functions/Flow Editor/Screen Operation/Component/Edit
Explanation of Functions/Flow Editor/Screen Operation/Component/Delete
Explanation of Functions/Flow Editor/Screen Operation/Component/Copy & Paste
Explanation of Functions/Flow Editor/Screen Operation/Component/Multiple Select
Explanation of Functions/Flow Editor/Screen Operation/Flowlink
Explanation of Functions/Various Logs/Component Individual Log/Individual Log Level Setting
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Controlling Edit Restrictions
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Testing Custom Components
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Email Sending
Execution License/Operating specifications of the execution license usage environment
Execution License/Execution license operation method/Quick Tutorial/Setting up of the execution license operating environment

Emitter

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Setting of Components/File Emitter Settings
Introduction Flow/Running the Collection Service/Test
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Component default
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Data directory setting
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/About Points
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/MQTT Emitter
Explanation of Functions/Emitter/File Emitter
Explanation of Functions/Emitter/Cloud Emitter for AWS
Explanation of Functions/Emitter/Cloud Emitter for Azure
Explanation of Functions/Emitter/FTP/FTPS/SFTP Emitter
Explanation of Functions/Development Components/Dummy Emitter
Security Guide/Using Encrypted Communications
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
Appendix/Error code list/MQTT Emitter
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/FTP/FTPS/SFTP Emitter

Error manager

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/System Components/Error Manager/Basic Information
Appendix/Error code list/Error manager

EtherNet/IP Collector

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/EtherNet/IP Collector
Appendix/Error code list/EtherNet/IP Collector

EtherNet/IP Writing

Introduction/Safety Precautions
Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Action/EtherNet/IP Writing
Appendix/Error code list/EtherNet/IP Writing

Event Data

Explanation of Functions/Logic (Event)/Event Data
Explanation of Functions/System Components/Event Data Manager
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
i-Trigger (Expansion Custom Package)/Components/Event data manager

Event Recorder

Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Data directory setting
Explanation of Functions/System Components/Event Data Manager/Basic Information
Explanation of Functions/System Components/Event Data Manager/Data Management Operational Specifications
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Event Record Play
i-Trigger (Expansion Custom Package)/Components/Event Recorder
Appendix/Error code list/Event Recorder (i-Trigger Extension package)

Event Trigger

Explanation of Functions/Logic (Event)/Event Trigger
Explanation of Functions/Component Default Settings/IN Port/Configure Events
i-Trigger (Expansion Custom Package)/Components/Event Trigger
Appendix/Error code list/Event Trigger

Expression

Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Action/PLC Writing
Explanation of Functions/Action/Modbus Writing
Explanation of Functions/Action/EtherNet/IP Writing
Explanation of Functions/Component Default Settings/OUT Port/Storage Configuration
Explanation of Functions/Component Default Settings/OUT Port/Conversion Configuration
Explanation of Functions/Component Default Settings/OUT Port/Set Registration Conditions
Explanation of Functions/Component Default Settings/Filter/Setting Option
Explanation of Functions/Component Default Settings/IN Port/Aggregation Configuration
Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SELECT
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Explanation of Functions/Arithmetic and Conditional Expressions/Expression and conditional expression (common)
Explanation of Functions/Arithmetic and Conditional Expressions/Expression
Appendix/Error code list/Compute Collector

15.7.6 F

FFT

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Logic (Event)/Event Trigger/Setting Method
Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/FFT

File Emitter

Introduction Flow/Setting of Components/File Emitter Settings
Introduction Flow/Running the Collection Service/Test
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Component default
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Data directory setting
Explanation of Functions/Emitter/File Emitter
Appendix/Error code list/File Emitter

Filter

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Screen Operation/Flowlink/Filter
Explanation of Functions/Component Default Settings/Filter
Explanation of Functions/Component Default Settings/Flowlink Icon
Execution License/Operating specifications of the execution license usage environment
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getFlowlinkFilter ... Get flowlink filter information
Appendix/Custom UI JS-API reference/Types/FlowlinkFiltersResponse

Flow Editor

Explanation of Functions/Flow Editor/About the mode
Explanation of Functions/Flow Editor/Items on the screen
Explanation of Functions/Flow Editor/Screen Operation
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License

Flowlink

Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Flowlink
Explanation of Functions/Component Default Settings/Flowlink Icon
Explanation of Functions/Component Default Settings/IN Port/Configure Events
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getFlowlinks ... Get flowlink list
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getFlowlink ... Get flowlink information
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getFlowlinkFilter ... Get flowlink filter information
Appendix/Custom UI JS-API reference/Types/FlowlinkResponse
Appendix/Custom UI JS-API reference/Types/FlowlinkFiltersResponse

FTP/FTPS/SFTP Emitter

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Emitter/FTP/FTPS/SFTP Emitter
Security Guide/Using Encrypted Communications
Appendix/Error code list/FTP/FTPS/SFTP Emitter

15.7.7 G

Global Variables

Advanced Use/Global Variables/Overview of Global Variables

Grafana

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Grafana Setup
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Publish access token
Explanation of Functions/Component Default Settings/OUT Port/Storage Configuration
Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Advanced Use/Linking with Grafana/Grafana Integration Overview
Advanced Use/Linking with Grafana/Installing Grafana
Advanced Use/Linking with Grafana/Grafana Setup
Advanced Use/Linking with Grafana/Creating a Graph
Advanced Use/Linking with Grafana/Viewing a Graph
Advanced Use/Linking with Grafana/Manually Setting a Grafana Data Source
Execution License/Operating specifications of the execution license usage environment

Graph

Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Advanced Use/Linking with Grafana/Creating a Graph
Advanced Use/Linking with Grafana/Viewing a Graph
Advanced Use/Linking with Grafana/Changing the data count limit for a graph
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Event Record Play/Play

H

Header

Introduction Flow/Initial Setting/Changing the Administrator Password
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Operation File Format
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Custom Component Implementation Details/Include Header

HTTP/HTTPS communication

Explanation of Functions/Flow Editor/Screen Operation/Setting menu/HTTP/HTTPS communication
Appendix/Error code list/DB query service

15.7.8 I


IN Port

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Component/Add
Explanation of Functions/Flow Editor/Screen Operation/Flowlink/Add
Explanation of Functions/Collector/Compute Collector
Explanation of Functions/Emitter/Cloud Emitter for AWS
Explanation of Functions/Emitter/Cloud Emitter for Azure
Explanation of Functions/Emitter/FTP/FTPS/SFTP Emitter
Explanation of Functions/Development Components/Dummy Emitter
Explanation of Functions/Component Default Settings/IN Port
i-Trigger (Expansion Custom Package)/Components/Event Recorder
Appendix/Error code list/Component default
Appendix/Error code list/MQTT Emitter
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/PLC Writing
Appendix/Error code list/Modbus Writing
Appendix/Custom component Python-API reference/HiveComponentBase Objects
Appendix/Custom component Python-API reference/AggregationType Objects
Appendix/Custom component Python-API reference/AggregationTypeSet Objects
Appendix/Custom component Python-API reference/InColumn Objects
Appendix/Custom component Python-API reference/InPortReader Objects
Appendix/Custom component Python-API reference/HiveInPort Objects
Appendix/Custom component Python-API reference/HiveWindowData Objects
Appendix/Custom component Python-API reference/HiveRecord Objects
Appendix/Custom component Python-API reference/HiveInPortEventSetting Objects
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_get_column_list() … Get input column information
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_continuous_reader() … Get Reader for continuous retrieval for column data
Appendix/Custom component C-API reference/Input Column Functions/hive_continuous_reader_read() … Continuously retrieve column data
Appendix/Custom component C-API reference/Input Column Functions/hive_inport_timerange_reader() … Get Reader for time range retrieval of column data
Appendix/Custom component C-API reference/Input Column Functions/hive_timerange_reader_read() … Search for column data in a time range
Appendix/Custom component C-API reference/Input Column Functions/hive_incolumn_get_latest_value() … Get latest value of column data
Appendix/Custom component C-API reference/Types, Structures, and Functions/hive_primitive_type_t … HIVE column primitive type
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_DATA_TYPE … HIVE column data type
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_COLUMN_READ_RESULT … Result of reading column information (window data)

Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_RECORD … Column data record
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getInPortData ... Get IN Port information
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getInPortDataValue ... Get IN Port data
Appendix/Custom UI JS-API reference/Types/ComponentbaseResponse
Appendix/Custom UI JS-API reference/Types/InportResponse
Appendix/Custom UI JS-API reference/Types/PortDataResponse
Appendix/Custom UI JS-API reference/Types/PortDataValueResponse

i-Trigger

Introduction Flow/Install (Windows)/Installing and Updating
Explanation of Functions/System Components/Event Data Manager
i-Trigger (Expansion Custom Package)/Installation/Installation and Updates
i-Trigger (Expansion Custom Package)/Installation/Uninstallation
Appendix/Error code list/Event data manager
Appendix/Error code list/VIDEO Collector (i-Trigger Extension package)
Appendix/Error code list/Event Recorder (i-Trigger Extension package)

15.7.9 J

JSON Serializer

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Serializer/JSON Serializer
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/JSON Serializer

JSON Deserializer

Explanation of Functions/Deserializer/JSON Deserializer

15.7.10 L

License

Introduction/Agreements
Introduction Flow/Install (Windows)/Installing and Updating
Introduction Flow/Initial Setting/Changing the Administrator Password
Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Software Information
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/License Information
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Execution License/License Types
Execution License/Operating specifications of the execution license usage environment
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License

Log Level

Explanation of Functions/Various Logs/Component Individual Log
Appendix/Custom component Python-API reference/HiveLogLevel Objects
Appendix/Custom component Python-API reference/HiveLog Objects/get_current_level

Log Output

Advanced Use/Creating and Using Custom Components/Creating Python custom components/Custom Component Implementation Details/Log Output
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Custom Component Implementation Details/Log Output
Appendix/Custom component Python-API reference/HiveLog Objects/error
Appendix/Custom component Python-API reference/HiveLog Objects/warning
Appendix/Custom component Python-API reference/HiveLog Objects/info
Appendix/Custom component Python-API reference/HiveLog Objects/debug
Appendix/Custom component Python-API reference/HiveLog Objects/trace
Appendix/Custom component Python-API reference/HiveLog Objects/exception

Logic

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Logic (Event)/Event Trigger/Setting Method
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SELECT
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data

Login

Introduction Flow/Initial Setting/Changing the Administrator Password
Introduction Flow/Initial Setting/Add User
Explanation of Functions/Flow Editor/Screen Operation/User Management/Add
Explanation of Functions/Flow Editor/Screen Operation/User Management/Edit
Explanation of Functions/Flow Editor/Screen Operation/User Management/Profile
Explanation of Functions/Flow Editor/Screen Operation/User Management/Login

Logout

Introduction Flow/Initial Setting/Add User
Explanation of Functions/Flow Editor/Screen Operation/User Management/Logout

15.7.11 M

Modbus Collector

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/Modbus Collector
Appendix/Error code list/Modbus Collector

Modbus Writing

Introduction/Safety Precautions
Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Action/Modbus Writing
Appendix/Error code list/Modbus Writing


Mode

Introduction Flow/Initial Setting/Add User
Introduction Flow/Running the Collection Service/Test
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Flow Editor/About the mode
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu
Explanation of Functions/Flow Editor/Screen Operation/User Management/Add
Explanation of Functions/Flow Editor/Screen Operation/User Management/Edit
Explanation of Functions/Flow Editor/Screen Operation/Panel/Add
Explanation of Functions/Flow Editor/Screen Operation/Panel/Changing a name
Explanation of Functions/Flow Editor/Screen Operation/Panel/Rearranging
Explanation of Functions/Flow Editor/Screen Operation/Panel/Delete
Explanation of Functions/Flow Editor/Screen Operation/Panel/Replicate
Explanation of Functions/Flow Editor/Screen Operation/Panel/Overall Panel
Explanation of Functions/Flow Editor/Screen Operation/Panel/Panel Status
Explanation of Functions/Flow Editor/Screen Operation/Component/Add
Explanation of Functions/Flow Editor/Screen Operation/Component/Edit
Explanation of Functions/Flow Editor/Screen Operation/Component/Delete
Explanation of Functions/Flow Editor/Screen Operation/Component/Copy & Paste
Explanation of Functions/Flow Editor/Screen Operation/Component/Multiple Select
Explanation of Functions/Flow Editor/Screen Operation/Component/Confirming configurations
Explanation of Functions/Flow Editor/Screen Operation/Flowlink
Explanation of Functions/Flow Editor/Screen Operation/System resources
Explanation of Functions/Flow Editor/Screen Operation/How to operate on a table device
Explanation of Functions/Various Logs/Component Individual Log/Individual Log Level Setting
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Function Executed When the Parameters Dialog Appears
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Controlling Edit Restrictions
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Initializing the Custom Component Administration Directory
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Testing Custom Components
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Using SCCDE and Packaging/Packaging
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to

Power Outages/Configuring Email Sending
Execution License/Operating specifications of the execution license usage environment
Execution License/Execution license operation method/Quick Tutorial/Setting up of the execution license operating environment
Appendix/Custom component Python-API reference/AggregationType Objects
Appendix/Custom UI JS-API reference/Callback Functions/onLoad ... Custom UI load hook

Moving Average

Explanation of Functions/Logic (Real Time Analysis and Data Manipulation)/Moving Average

MQTT Collector

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/MQTT Collector
Security Guide/Using Encrypted Communications
Appendix/Error code list/MQTT Collector

MQTT Emitter

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/About Points
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/MQTT Emitter
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
Appendix/Error code list/MQTT Emitter

MQTT protocol

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/MQTT Emitter

Network

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Collector/Resource Collector
Appendix/Error code list/Component default
Appendix/Error code list/Resource Collector

15.7.13 O

Operation Log

Explanation of Functions/Various Logs/Log Types
Explanation of Functions/Various Logs/Operation Log

OUT Port

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Flowlink/Add
Explanation of Functions/Component Default Settings/OUT Port
Explanation of Functions/Arithmetic and Conditional Expressions/Overview of expressions and conditional expressions
Appendix/Custom component Python-API reference/HiveComponentBase Objects
Appendix/Custom component Python-API reference/ColumnOption Objects
Appendix/Custom component Python-API reference/OutColumn Objects
Appendix/Custom component Python-API reference/HiveOutPort Objects
Appendix/Custom component C-API reference/OUT Port Functions
Appendix/Custom component C-API reference/Types, Structures, and Functions/hive_primitive_type_t … HIVE column primitive type
Appendix/Custom component C-API reference/Types, Structures, and Functions/HIVE_DATA_TYPE … HIVE column data type
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getOutPortData ... Get OUT Port information
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getOutPortDataValue ... Get OUT Port data
Appendix/Custom UI JS-API reference/Types/ComponentbaseResponse
Appendix/Custom UI JS-API reference/Types/PortDataResponse
Appendix/Custom UI JS-API reference/Types/PortDataValueResponse

Overall Panel

Explanation of Functions/Flow Editor/Screen Operation/Panel/Overall Panel

15.7.14 P

Panel

Introduction Flow/Running the Collection Service/Start/Stop Collection Service/Start/Stop All Components on a Panel
Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/User Management/Add
Explanation of Functions/Flow Editor/Screen Operation/User Management/Edit
Explanation of Functions/Flow Editor/Screen Operation/Panel
Explanation of Functions/Flow Editor/Screen Operation/Component/Start/Stop
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Common Items
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Acquire the execution status of individual component
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Start batch execution of components
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification/Stop batch execution of components
Execution License/Operating specifications of the execution license usage environment
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getPanels ... Get panel list
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getPanel ... Get panel information
Appendix/Custom UI JS-API reference/Types/PanelResponse

persisted data

Appendix/Custom component Python-API reference/InColumn Objects/get_latest_value

PLC Collector

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Simulation mode/Outline of simulation mode
Explanation of Functions/Collector/PLC Collector
Execution License/Execution license operation method/Quick Tutorial/Building a System Using a Development License
Execution License/Execution license operation method/Quick Tutorial/Setting up of the execution license operating environment
Appendix/Error code list/PLC Collector

PLC Writing

Introduction/Safety Precautions
Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Action/PLC Writing
Appendix/Error code list/PLC Writing

Point

Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Software Information
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/License Information
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/About Points
Explanation of Functions/Flow Editor/Screen Operation/Data monitor/Distribution
i-Trigger (Expansion Custom Package)/Components/VIDEO Collector/Basic Information
i-Trigger (Expansion Custom Package)/Components/Event Recorder/Basic Information
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert() … Register data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert_ts() … Register timestamp specified data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert_binary() … Register binary data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert_binary_ts() … Register timestamp specified binary type data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert_multisamples() … Register multi-sample data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_open_file() … Create file
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_insert_file() … Register file type data
Appendix/Custom component C-API reference/Output Column Functions/hive_outcolumn_dispose_file() … Discard files
Appendix/Custom component C-API reference/Input Column Functions/HIVE_RECORD_ITERATOR_GET_NEXT() … Get record iterator
Appendix/Custom component C-API reference/Input Column Functions/HIVE_RECORD_GET_DATA() … Get record iterator
Appendix/Custom component C-API reference/Input Column Functions/hive_incolumn_get_latest_value() … Get latest value of column data
Appendix/Custom component C-API reference/Component Status Functions/hive_status_clear() … Clear status information
Appendix/Custom component C-API reference/Component Status Functions/hive_status_add_error() … Register an error in status information
Appendix/Custom component C-API reference/Component Status Functions/hive_status_set_total_xxx_xxx() … Register run-time auxiliary information in status information
Appendix/Custom component C-API reference/Component Status Functions/hive_component_register_status() … Register status with component
Appendix/Custom component C-API reference/Component Status Functions/hive_status_alloc() … Allocate status information structure
Appendix/Custom component C-API reference/Component Status Functions/hive_status_free() … Release status information structure

Profile

Introduction Flow/Initial Setting/Add User
Explanation of Functions/Flow Editor/Screen Operation/User Management/Profile

15.7.15 R

Real time analysis

Explanation of Functions/Flow Editor/Items on the screen/Edit Mode

Registration Conditions

Explanation of Functions/Component Default Settings/OUT Port/Set Registration Conditions

Resource Collector

Introduction Flow/Setting of Components/Configuring the Resource Collector
Introduction Flow/Setting of Components/Configuring the CSV Serializer
Introduction Flow/Running the Collection Service/Test
Introduction Flow/Running the Collection Service/Configuration Tuning
Explanation of Functions/Collector/Resource Collector
Explanation of Functions/Serializer/JSON Serializer
Explanation of Functions/Serializer/CSV Serializer
Explanation of Functions/Component Default Settings/OUT Port/Conversion Configuration
Explanation of Functions/Component Default Settings/OUT Port/Set Registration Conditions
Appendix/Error code list/Resource Collector

Run Mode

Introduction Flow/Initial Setting/Add User
Introduction Flow/Running the Collection Service/Test
Explanation of Functions/Flow Editor/About the mode
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Server Settings
Explanation of Functions/Flow Editor/Screen Operation/User Management/Add
Explanation of Functions/Flow Editor/Screen Operation/User Management/Edit
Explanation of Functions/Flow Editor/Screen Operation/Panel/Overall Panel
Explanation of Functions/Flow Editor/Screen Operation/Panel/Panel Status
Explanation of Functions/Flow Editor/Screen Operation/Component/Confirming configurations
Explanation of Functions/Flow Editor/Screen Operation/System resources
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Function Executed When the Parameters Dialog Appears
Advanced Use/Creating and Using Custom Components/Creating custom UI/Creating and Applying HTML for Synapse Custom UI/Controlling Edit Restrictions
Execution License/Operating specifications of the execution license usage environment
Appendix/Custom UI JS-API reference/Callback Functions/onLoad ... Custom UI load hook

15.7.16 S

Send Mail

Explanation of Functions/Action/Send Mail
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Email Sending
Security Guide/Using Encrypted Communications

Serializer

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Setting of Components/Configuring the CSV Serializer
Introduction Flow/Setting of Components/File Emitter Settings
Explanation of Functions/Overview of engine functions/Feature List/Component
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Server Settings
Explanation of Functions/Serializer/JSON Serializer
Explanation of Functions/Serializer/CSV Serializer
Appendix/Error code list/File Emitter
Appendix/Error code list/Cloud Emitter for AWS
Appendix/Error code list/Cloud Emitter for Azure
Appendix/Error code list/JSON Serializer
Appendix/Error code list/CSV Serializer


Setting Menu

Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu
Explanation of Functions/Managing Configuration Information (Project)/Create project
Explanation of Functions/Managing Configuration Information (Project)/Copy project
Explanation of Functions/Managing Configuration Information (Project)/Import project
Explanation of Functions/Managing Configuration Information (Project)/Export project
Explanation of Functions/Managing Configuration Information (Project)/Switch project
Explanation of Functions/Managing Configuration Information (Project)/Information Update
Explanation of Functions/Managing Configuration Information (Project)/Delete project
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Add
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Delete
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Display
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Download
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Protect
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Unprotect
Advanced Use/Creating and Using Custom Components/Creating Python custom components/Using Custom Components/Deleting External Libraries
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components/Add
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components/Delete
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components/Download
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components/Protect
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Using Custom Components/Unprotect
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to Python Custom Component/Register
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to Python Custom Component/Delete
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to C Custom Component/Register
Advanced Use/Creating and Using Custom Components/Creating custom UI/Applying Custom UI (JSON) to C Custom Component/Delete
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Register/Update
Advanced Use/Creating and Using Custom Components/Custom component development environment (SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Download
Advanced Use/Creating and Using Custom Components/Custom component development environment

(SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Delete
Advanced Use/Creating and Using Custom Components/Custom component development environment
(SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Protect
Advanced Use/Creating and Using Custom Components/Custom component development environment
(SCCDE)/Registering SCCPKG File in and Deleting SCCPKG File from Synapse/Unprotect
Advanced Use/Linking with Grafana/Grafana Setup
Advanced Use/Linking with Grafana/Creating a Graph/Registering Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect When a Problem Occurs at a Specific Component/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Execution License/Execution license operation method/Quick Tutorial/Output of execution license settings
Execution License/Execution license operation method/Quick Tutorial/Setting up of the execution license operating environment
Appendix/Error code list/DB query service

Shell Command

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Action/Shell Command
Appendix/Error code list/Shell Command

Simulation mode

Explanation of Functions/Flow Editor/Screen Operation/Simulation mode

SQL

Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Feature List/Database
Explanation of Functions/System Components/DB Query Service
Explanation of Functions/Component Default Settings/OUT Port/Storage Configuration
Explanation of Functions/Component Default Settings/IN Port/Aggregation Configuration
Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Explanation of Functions/Database Query (SQL)/DB Query Overview
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SELECT
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SET
Advanced Use/Global Variables/Overview of Global Variables

Status Timeline

Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Overview of engine functions/Operating Specifications/Abnormal Stop (e.g., power interruption)
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Status Timeline/Status timeline/View/All components
Explanation of Functions/Status Timeline/Status timeline/View/Individual component
Troubleshooting/Common procedure when a trouble occurs
Execution License/Operating specifications of the execution license usage environment

Storage

Explanation of Functions/Overview of engine functions/Feature List/Database
Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications Related to Data Collection
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Server Settings
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Publish access token
Explanation of Functions/Component Default Settings/OUT Port/Storage Configuration
Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Appendix/Custom UI JS-API reference/Types/OutportResponse

system component

Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode

System panel

Advanced Use/Linking with Grafana/Creating a Graph/Start DB Query Service
Troubleshooting/How to Detect When a Problem Occurs/How to detect when any one of components in Synapse fails/Configuring Email Sending
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Event Data
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages/Configuring Email Sending

System port

Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Appendix/Custom UI JS-API reference/SpeeDBee Synapse Settings Acquisition Functions/getSystemPort ... Get system port setting
Appendix/Custom UI JS-API reference/Types/OutportResponse

15.7.17 T

time series database

Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Feature List/Database
Explanation of Functions/Component Default Settings/OUT Port/Storage Configuration
Explanation of Functions/Component Default Settings/SYSTEM Port/Storage Configuration
Explanation of Functions/Database Query (SQL)/DB Query Overview
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SELECT
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/SHOW
Explanation of Functions/Database Query (SQL)/DB Query Detail/Syntax Specifications/INSERT
Troubleshooting/How to Detect When a Problem Occurs/How to Detect Data Loss in a Time Series Database due to Power Outages
Appendix/Error code list/Core Internal

15.7.18 U

User

Overview of SpeeDBee Synapse/Flow up to operation
Introduction Flow/Initial Setting/Changing the Administrator Password
Introduction Flow/Initial Setting/Add User
Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Proxy Settings
Explanation of Functions/Flow Editor/Screen Operation/User Management
Explanation of Functions/Flow Editor/Screen Operation/Panel
Explanation of Functions/Flow Editor/Screen Operation/Component/Start/Stop
Explanation of Functions/Collector/MQTT Collector
Explanation of Functions/Emitter/MQTT Emitter
Explanation of Functions/Emitter/FTP/FTPS/SFTP Emitter
Explanation of Functions/Action/Send Mail
Explanation of Functions/Various Logs/Operation Log/View
Explanation of Functions/Various Logs/Operation Log/Narrowing Down
Explanation of Functions/Managing Configuration Information (Project)/Project Overview
Advanced Use/Linking with Grafana/Grafana Setup
Security Guide/User Management
Execution License/Operating specifications of the execution license usage environment
Appendix/Error code list/MQTT Collector
Appendix/Error code list/MQTT Emitter
Appendix/Error code list/FTP/FTPS/SFTP Emitter
Appendix/Error code list/Send mail
Appendix/Custom component C-API reference/Other Functions/hive_component_interval_call() … Periodic function
Appendix/Custom component C-API reference/Types, Structures, and Functions/hive_interval_proc_t … Periodic function signature

15.7.19 V

VIDEO Collector

Explanation of Functions/Overview of engine functions/Operating Specifications/Operational Specifications for Network Errors and Recovery
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Data directory setting
i-Trigger (Expansion Custom Package)/Installation/Uninstallation
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Video Record Play
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Event Record Play/Play
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Live Camera
i-Trigger (Expansion Custom Package)/Components/VIDEO Collector
i-Trigger (Expansion Custom Package)/Components/Event Recorder
Appendix/Error code list/VIDEO Collector (i-Trigger Extension package)

Viewer

Introduction Flow/Initial Setting/Add User
Explanation of Functions/Flow Editor/Items on the screen/Edit Mode
Explanation of Functions/Flow Editor/Items on the screen/Run Mode
Explanation of Functions/Flow Editor/Screen Operation/User Management/Add
Explanation of Functions/Flow Editor/Screen Operation/User Management/Edit
Explanation of Functions/System Components/Event Data Manager/Data Management Operational Specifications
i-Trigger (Expansion Custom Package)/Video Viewer for Video Playback/Video Viewer Access
i-Trigger (Expansion Custom Package)/Components/VIDEO Collector
i-Trigger (Expansion Custom Package)/Components/Event Recorder

15.7.20 W

WebAPI

Advanced Use/Linking with Control Programs (WebAPI)/Introduction to WebAPI
Advanced Use/Linking with Control Programs (WebAPI)/How to run the WebAPI
Advanced Use/Linking with Control Programs (WebAPI)/WebAPI Specification

WEBUI

Overview of SpeeDBee Synapse/Configuration of SpeeDBee Synapse
Explanation of Functions/Overview of engine functions/Feature List/Core
Explanation of Functions/Overview of engine functions/Feature List/WEBUI
Explanation of Functions/Overview of engine functions/Operating Specifications/Basic Operation
Explanation of Functions/Flow Editor/Screen Operation/Setting menu/Server Settings
Explanation of Functions/Various Logs/Log Types
Explanation of Functions/Various Logs/Download
Advanced Use/Creating and Using Custom Components/Creating C language custom components/Sample Custom Components/Execution

15.8 DB query service Web-API specification

This section provides the DB query service Web-API specification.

Core Interface (0.1.0)

Download OpenAPI specification: Download

Core DB Query-Specific API
If the above download is not available, please download openapi.json from the following link.
openapi.json

DB Query Execution

Executes a DB query and returns the result.
If the query results in an error, a response of `result-type:error` is returned with the status code `200`.
If the query itself cannot be executed, a status code other than `200` is returned.

HEADER PARAMETERS

- `X-hive-api-key` — string
  The token string to specify when token authentication is enabled.

REQUEST BODY SCHEMA: application/json
required

DB query

- `query` (required) — string
  Query string
- `timeout` — int [1 .. 180]
  Default: `30`
  Query processing timeout (seconds), a timeout occurs after specified time
- `timestamp_unit` — string
  Default: `"nanosecond"`
  Enum: `"nanosecond"` `"microsecond"` `"millisecond"` `"second"`
  Unit of return result timestamp
- `omit_option` — string
  Default: `0`
  Enum: `0` `1` `2`
  Specify whether to omit data absence or null value data (0= Null and data absence are omitted, 1 or 2= Not omitted)

Responses

200 DB query result

RESPONSE SCHEMA: application/json

- `query` (required) — string
  Query string
- `timestamp_unit` (required) — string
  Timestamp unit specification
- `success` (required) — boolean
  Indicates successful query execution
- `cutoff` — boolean
  Indicates whether the query result is cut off
- `error` — object (DbQueryError)
  Object containing query execution error information
  - `code` — string
    Error identifier
  - `message` — string
    Error message
- `result` — object (DbQueryResult)
  Object that is set when the query has a result
  - `columns` — Array of strings (HiveColumnListResult) [items]
    Object that is set only when column list acquisition is requested
  - `record_columns` — Array of objects (DbQueryResultColumnMeta)
    List of column meta information

    Array [
    - `name` (required) — string
    - `data_type` (required) — string (DbQueryDataType)
      Enum: `"bool"` `"int8"` `"int16"` `"int32"` `"int64"` `"uint8"` `"uint16"` `"uint32"` `"uint64"` `"float"`

"double" "timestamp" "string" "binary"

data_array
required
integer (DbQueryDataArray) >= 1

]

records
Array of objects (DbQueryResultRecord)

Array [

timestamp
required
int
Record timestamp

record_type
required
string (DbQueryResultRecordType)
Enum: "D" "S"

data
Array of objects

Array [

name
required
string

value
required
any

data_type
required
string (DbQueryDataType)
Enum: "bool" "int8" "int16" "int32" "int64" "uint8" "uint16" "uint32" "uint64" "float" "double" "timestamp" "string" "binary"

data_array
integer (DbQueryDataArray) >= 1

]

]

500 This function is currently disabled

RESPONSE SCHEMA: application/json

id
required
string

code required	integer	
codeHex required	string `0x[0-9a-f]{8}`	
message required	string	
parameters	Array of any	
extendedInfo	any	

POST /dbquery

Request samples

Payload

Content type
application/json

Copy

```
{
    "query": "SELECT ¥"comp1:data1¥", ¥"comp2:dataA¥" FROM SPDB;",
    "timeout": 15,
    "timestamp_unit": "nanosecond"
}
```

Response samples

200 500

Content type
application/json

Example
DB query result

Copy Expand all Collapse all

```
{
    "query": "SELECT ¥"comp1:data1¥", ¥"comp2:dataA¥" FROM SPDB;",
    "timestamp_unit": "nanosecond",
    "success": true,
```

```
  - "result": {
      + "record_columns": [ ··· ],
      + "records": [ ··· ]
    }
}
```

DB Query Execution (Stream Format)

Executes a DB query and returns the result in stream format.
This API only supports queries that do not change the state of the server, such as SELECT.
Not supported at this time. (May be added in the future)

REQUEST BODY SCHEMA: application/json
required

DB query

- `query` (required)
  string
  Query string
- `timeout`
  int [1 .. 180]
  Default: `30`
  Query processing timeout (seconds), a timeout occurs after specified time
- `timestamp_unit`
  string
  Default: `"nanosecond"`
  Enum: `"nanosecond"` `"microsecond"` `"millisecond"` `"second"`
  Unit of return result timestamp
- `omit_option`
  string
  Default: `0`
  Enum: `0` `1` `2`
  Specify whether to omit data absence or null value data (0= Null and data absence are omitted, 1 or 2= Not omitted)

Responses

200 DB query result

RESPONSE SCHEMA: application/jsonlines

```
Array [
```

- timestamp
  required
  int
  Record timestamp
- record_type
  required
  string (DbQueryResultRecordType)
  Enum: "D" "S"
- data
  Array of objects

Array [

- name
  required
  string
- value
  required
  any
- data_type
  required
  string (DbQueryDataType)
  Enum: "bool" "int8" "int16" "int32" "int64" "uint8" "uint16" "uint32" "uint64" "float" "double" "timestamp" "string" "binary"
- data_array
  integer (DbQueryDataArray) >= 1

]

]

500 This function is currently disabled

RESPONSE SCHEMA: application/json

- id
  required
  string
- code
  required
  integer
- codeHex
  required
  string 0x[0-9a-f]{8}
- message
  required
  string
- parameters
  Array of any
- extendedInfo
  any

GET /dbquery

Request samples

Payload

Content type
application/json

Copy

```
{
    "query": "SELECT ¥"comp1:data1¥", ¥"comp2:dataA¥" FROM SPDB;",
    "timeout": 15,
    "timestamp_unit": "nanosecond"
}
```

Response samples

200 500

Content type
application/jsonlines

Copy Expand all Collapse all

```
[
  - {
        "timestamp": null,
        "record_type": "D",
      + "data": [ … ]
    }
]
```

OMRON AUTOMATION AMERICAS HEADQUARTERS • Chicago, IL USA • 847.843.7900 • 800.556.6766 • automation.omron.com

OMRON CANADA, INC. • HEAD OFFICE
Montreal, Quebec, Canada • 416.286.6465 • 866.986.6766 • automation.omron.com

OMRON MÉXICO • HEAD OFFICE
Mexico City, Mexico • 55 5901 4300 - mela@omron.com

OMRON MÉXICO • GUADALAJARA
Guadalajara, Jal, Mexico• 55.5901.4300

OMRON ELETRÔNICA DO BRASIL LTDA • HEAD OFFICE
São Paulo, SP, Brasil • 55 11 5171-8920 • automation.omron.com

OTHER OMRON LATIN AMERICA SALES
+54.11.4521.8630 • +54.11.4523.8483 • mela@omron.com

Controllers & I/O
• Machine Automation Controllers (MAC) • Motion Controllers
• Programmable Logic Controllers (PLC) • Temperature Controllers • Remote I/O

Operator Interfaces
• Human Machine Interface (HMI)

Motion & Drives
• Machine Automation Controllers (MAC) • Motion Controllers • Servo Systems
• Frequency Inverters

Identification and Vision
• Vision Sensors & Systems • Measurement Sensors • Auto Identification Systems

Sensing
• Photoelectric Sensors • Fiber-Optic Sensors • Proximity Sensors
• Rotary Encoders • Ultrasonic Sensors

Safety
• Safety Light Curtains • Safety Laser Scanners • Programmable Safety Systems
• Safety Mats and Edges • Safety Door Switches • Emergency Stop Devices
• Safety Switches & Operator Controls • Safety Monitoring/Force-guided Relays

Control Components
• Power Supplies • Timers • Counters • Programmable Relays
• Digital Panel Meters • Monitoring Products

Switches & Relays
• Limit Switches • Pushbutton Switches • Electromechanical Relays
• Solid State Relays

Software
• Programming & Configuration • Runtime

Note: Specifications are subject to change.

© 2026 OMRON. All Rights Reserved.