Bivrost IoT Gateway

彼络物联网关

User Manual

Version V1.19.7.60

Bivrost Technologies Company Limited

深圳市彼络科技有限公司

Scan for support on WeChat

Contents

1. Introduction 4

2. Networking 8

3. Using the Gateway 10

3.1. Signing In 12

3.2. Home 14

3.3. Machines 20

3.4. Groups 44

3.5. Tasks 53

3.6. Communication 69

3.7. Network 82

3.8. API Test 86

3.9. DNC (Program Transfer) 95

3.10. Analysis 103

3.11. Monitor 111

3.12. Settings 114

3.13. Other 128

4. Protocol Conventions 129

4.1. Identifiers 129

4.2. Data Description 131

4.3. Variable Definitions 153

5. HTTP Communication 157

5.1–5.2. Basics and Error Handling 157

5.3–5.4. Authentication 161

5.5. Data Read/Write APIs 165

5.5.1. Direct Read/Write 165

5.5.1.14–5.5.1.22. Direct Read/Write: Tool Offsets, Coordinates, Program & PLC 181

5.5.1.23–5.5.1.26. Direct Read/Write: Tool Life 208

5.5.2. Cached Read 227

5.6. File Management Interface 236

5.7. Data Analysis APIs 273

5.7.1. Machine Data Analysis 273

5.7.2. Machine Group Data Analysis 283

5.8. History Data Interface 297

5.9. Gateway Configuration APIs 305

5.9.1. Global Configuration 305

5.9.2. User Configuration API 312

5.9.3. Machine Configuration Interface 319

5.9.4. Group Configuration API 341

5.9.5. Task Configuration Interface 353

5.9.6. Communication Configuration Interface 371

5.10. Gateway Function APIs 393

5.10.1. Core Service Functions 393

5.10.2. Gateway Service Function Interfaces 406

6. MODBUS Communication 433

7. MQTT Communication 443

7.1. Data Upload Message Format 443

7.2. RPC Interface 456

8. Database Communication 468

9. MCP Service 471

10. Mock Machine Testing 480

11. Appendices 481

11.1. Glossary 482

11.2. Command Format 484

11.3. Cluster Failover (Optional) 500

11.4. Supported Devices 505

11.5. Interface Support 509

12. FAQ 548

13. Known Issues 556

Changelog 558

1. Introduction

The Bivrost IoT Gateway is a data-acquisition appliance installed on the factory floor. It collects data from, and transfers machining programs to, CNC machine tools, laser welders, robots, PLCs and similar equipment. One gateway can connect to and manage up to 255 devices of different brands and models, and everything is configured from a web management page in a browser, so it works out of the box.

1.1. Key Features

- **Broad controller support** Fanuc, Siemens, Mitsubishi, Heidenhain, Mazak, Okuma, Brother, HNC, GSK, Syntec, KND and more than 20 CNC controls in total, plus laser welders, industrial robots and mainstream PLCs; see 11.4. Supported Devices.
- **Parallel acquisition from one unit** One gateway polls many machines at the same time, at intervals down to the millisecond; machines can be organised into groups for aggregated part counts and OEE.
- **Reliable** Automatic reconnection after a dropped link and resumable program transfers; the gateway starts on power-up, with optional cluster failover.
- **Multiple protocols** Read and write machine data on demand over HTTP; MODBUS, MQTT and databases receive automatic data collection task results; an MCP server lets AI clients query machines directly.
- **Program transfer** Upload, send and back up machining programs from the web page, with a built-in toolpath viewer; see 3.9. DNC (Program Transfer).
- **Analysis and monitoring** Alarm, count, cycle, OEE and overall analysis, plus a live monitor of machine and group status; see 3.10. Analysis and 3.11. Monitor.

1.2. How This Manual Is Organised

The manual has two parts. Chapters 1–3 and 11–13 cover networking, the web management page and each feature, for the people who install and operate the gateway on site. Chapters 4–10 are the communication protocol reference for developers integrating MES and other applications with the gateway. For which interfaces each brand, system and model supports, see 11.4. Supported Devices and 11.5. Interface Support.

On-site use

Networking

What each port is for, networking steps and network requirements.

Gateway management page

Signing in, and machine, group, task, communication and network settings.

Program transfer

Uploading, sending and backing up machining programs.

Analysis and monitor

Alarm, count, cycle and OEE analysis, and a live dashboard.

Integration

Before integrating, read 4. Protocol Conventions for the identifiers such as machineID, the data classes and the variable definitions.

HTTP Communication

Read and write machine data, transfer files, analyse data and configure the gateway on demand.

MODBUS Communication

The gateway acts as a MODBUS server (port 502) and serves collection results by address.

MQTT Communication

Collection results are published as JSON messages to an MQTT server; an RPC interface is also available.

Database Communication

Collection results are written to InfluxDB, MySQL, SQL Server, PostgreSQL and others.

MCP Service

Machine data, analysis and gateway status exposed to AI clients as read-only tools.

Mock Machine Testing

Test every protocol with a mock machine when no real machine tool is available.

> **Note**
>
> MODBUS, MQTT and database communication all output the results of **automatic data collection tasks**: first enable automatic collection for the target machine or group in 3.3.1.2. Task Settings or 3.4.1.2. Group Task Settings, enable the corresponding communication method in 3.6. Communication, then restart the service on the home page.

1.3. Quick Start

1. Wire the gateway as described in 2.1. Gateway Network: LAN1 to the machine or the acquisition network switch, LAN2 or Wi-Fi to the external network.
2. Open `http://iotgw.local` in a browser and sign in with the initial account admin (password admin); see 3.1. Signing In.
3. Add machines in 3.3. Machines, enable automatic collection and restart the service on the home page; with no real device at hand, start with a mock machine.
4. Confirm that machine data comes through in 3.8. API Test.
5. Enable the communication methods you need in 3.6. Communication and integrate your server application using the corresponding protocol chapter.

1.4. Reference and Help

Glossary

Definitions and calculation of machine OEE, group OEE and group part count.

Command Format

PLC data tasks, preconditions, post-processing and other command formats.

Supported Devices

The device types, systems and models the gateway can connect to.

FAQ

Common problems when using the gateway or integrating over each protocol, and how to resolve them.

Known Issues

Confirmed issues, their impact and how to work around them.

Changelog

Gateway and manual changes in each release.

When reading online, the button in the top-right corner downloads the whole manual as a PDF, and **Ask AI** in the bottom-right corner answers questions about its contents. For anything still unresolved, contact support.

2. Networking

2.1. Gateway Network

The Bivrost IoT Gateway has two wired Ethernet ports, LAN1 and LAN2, plus a built-in wireless adapter.

LAN1 is dedicated to the data-acquisition network. Its default IP address is 192.168.100.1 and its default subnet mask is 255.255.0.0. If several gateways share the same internal network, change the gateway network settings so that no two static IP addresses collide (see 3.7.1. Wired Network). To collect machine data, connect LAN1 directly to the Ethernet port of a CNC machine with a network cable for one-to-one acquisition, or connect LAN1 and the CNC machines through a switch or similar device so that they share one acquisition network for parallel acquisition.

LAN2 is the external communication port. If you need to upload data to the cloud or to receive remote assistance, contact Bivrost first, then connect LAN2 to the external network.

The **wireless adapter** can join a WiFi network (see 3.7.2. WiFi Settings) and serves the same purpose as LAN2. It is especially useful when testing in a plant that has no network in place: connect the gateway to a phone hotspot and remote assistance becomes much easier to arrange.

2.2. Networking Steps

In a typical customer network architecture, machines of all kinds connect to the gateway over Ethernet, serial over Ethernet, WiFi, a 4G/5G network or other links. Servers running applications such as MES connect to the gateway and read data and issue commands over HTTP, MODBUS, MQTT, a database or other channels.

1. Connect the gateway to the CNC machines, switches, servers and other equipment (see 2.1. Gateway Network).
2. Configure the network settings on the CNC machines.
3. Start the gateway, sign in, configure it, add the CNC machines and restart the gateway service (see 3.1. Signing In through 3.7. Network).
4. Check that the gateway is receiving machine data (see 3.8. API Test).
5. Following the protocol chapters of this manual (4. Protocol Conventions and the chapters after it), debug your server-side program and confirm that it receives data from the gateway. If no physical machine is available, choose the **Mock** machine type when adding

a machine so you can test communication between the gateway and the server (see 3.3.1. Adding a Machine).

6. If you need data storage, analysis or dashboard displays, configure them on the Bivrost cloud platform.

To ensure acquisition quality, we recommend shielded Cat 5e or Cat 6 cabling and gigabit network equipment. If a CNC machine has to connect over a wireless network, fit it with a wireless module or a 4G/5G module.

3. Using the Gateway

The gateway is configured to start automatically when power is applied. After connecting the power supply, wait about 1 minute for start-up to finish; the gateway is then ready to use.

You connect to the gateway from a computer and sign in to the gateway management page in a browser.

The first time you connect to the gateway, use an Ethernet cable between the computer’s LAN port and the gateway’s LAN port.

We recommend the latest version of Chrome on the computer used to sign in. Browsers such as 360 or IE may not render the page completely.

To switch the gateway off, use the **Shutdown/Restart Gateway** function button in the top right of the gateway management Home page, and unplug the power cable only after the gateway’s power LED has gone out.

> **Caution**
>
> **Frequent loss of power can make the gateway behave abnormally.** For example, if power is reapplied within 15 seconds of a momentary trip, the gateway may fail to restart automatically, in which case you have to press its power button manually. To avoid frequent power loss, install the gateway where the supply is stable, and fit a UPS if possible.

The navigation bar at the top of the gateway management page contains four menus: **Configurations** (Machines / Group / Task / Communication), **Diagnosis** (API Test / Network Tools / Logger), **App** (DNC / Analysis / Monitor) and **System** (Network / Settings). This chapter covers each function of the gateway management page in turn:

- 3.1. Signing In
- 3.2. Home
- 3.3. Machines
- 3.4. Groups
- 3.5. Tasks
- 3.6. Communication
- 3.7. Network
- 3.8. API Test
- 3.9. DNC (Program Transfer)

- 3.10. Analysis
- 3.11. Monitor
- 3.12. Settings
- 3.13. Other

3.1. Signing In

You can connect a computer to the gateway in any of the following ways:

1. Connect the computer’s LAN port directly to the gateway’s LAN1 port, or to LAN1 through a switch, then set the computer’s IP address to an address on the same subnet as the gateway’s LAN1 port, with the same subnet mask.
2. Connect the computer’s LAN port directly to the gateway’s LAN2 port, or to LAN2 through a switch, then set the computer to obtain its IP address automatically.
3. Connect the computer and the gateway to the same wireless network. You must first use method (1) or (2) to connect, sign in and configure the gateway’s WiFi connection.

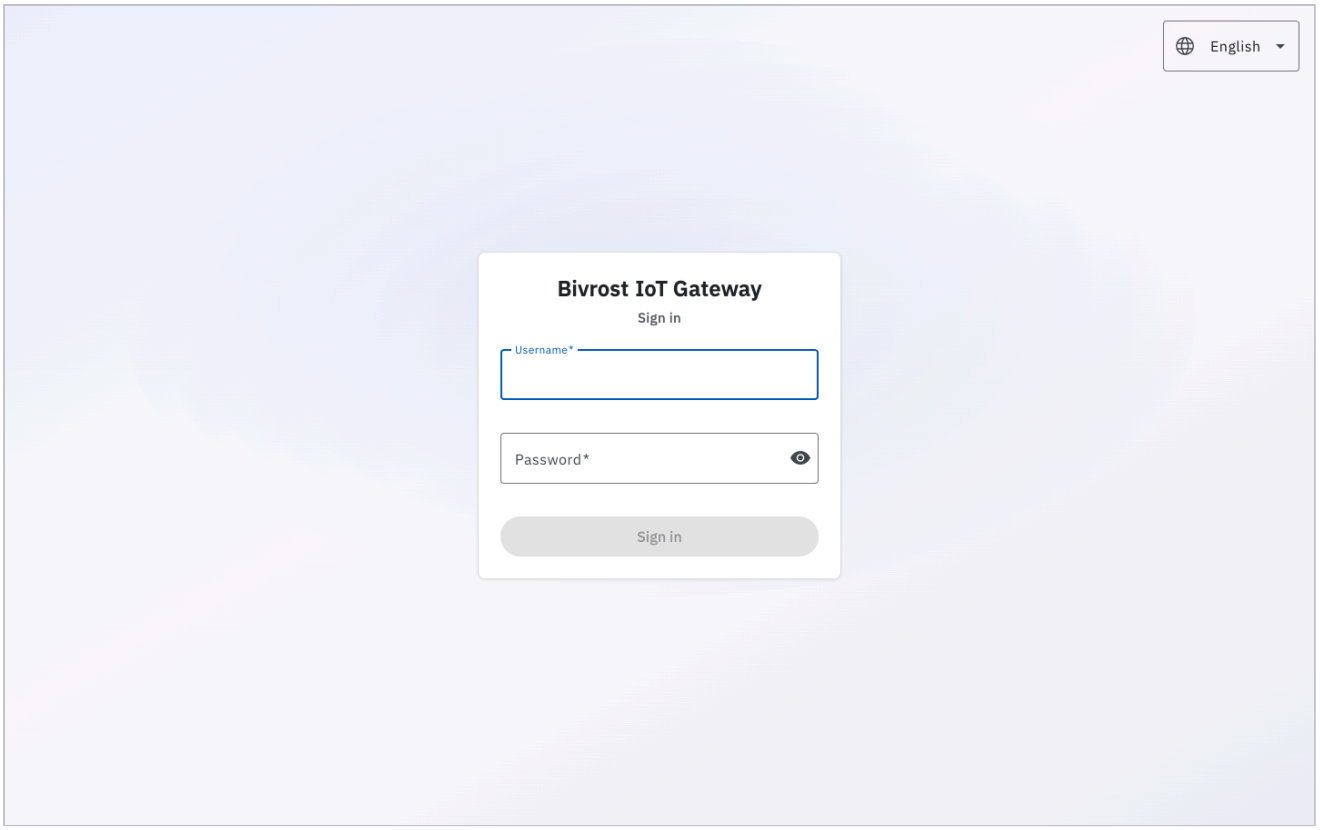
Once connected, you can sign in to the gateway management page in either of the following two ways:

Method 1: Signing In by Domain Name

In the address bar of the computer’s browser (Chrome recommended), enter the following address:

```
http://iotgw.local
```

Press Enter to open the gateway management login page. The initial account and password are both admin. Click **Login**. You can switch the interface language in the top-right corner.



If sign-in fails, disconnect any other devices so that the computer is connected to only one gateway, check the network cabling, and try again.

On some early gateway models, if `iotgw` does not work, replace `iotgw` with BIV-[first part of the gateway UID]. The gateway UID is printed on the label on the bottom of the gateway. For example, if the gateway UID is “152KUCG-1TOSJH4-1WXYSON-JY531Q”, the first part of the UID is “152KUCG”, so the address to enter in the browser is:

```
http://BIV-152KUCG.local
```

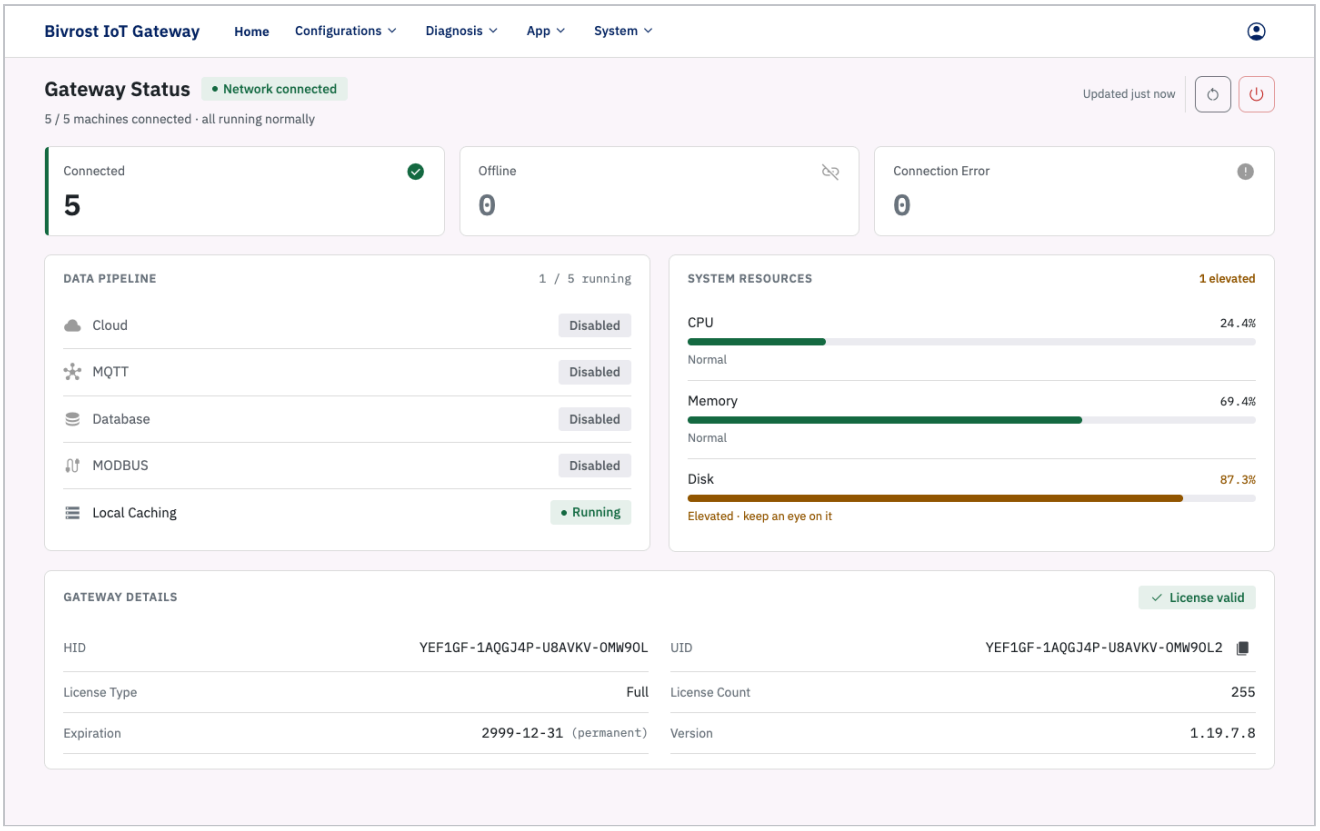
Method 2: Signing In by IP Address

Alternatively, in the address bar of the computer’s browser (Chrome recommended), enter the gateway’s IP address (the initial LAN1 IP is 192.168.100.1) and press Enter to open the login page. The initial account and password are both admin. Click **Login**. You can switch the interface language in the top-right corner.

> **Note**
>
> After changing the LAN1 IP address (see 3.7.1. Wired Network), sign in using the new IP address. If you are connected through the LAN2 port or over WiFi, use the corresponding IP address.

3.2. Home

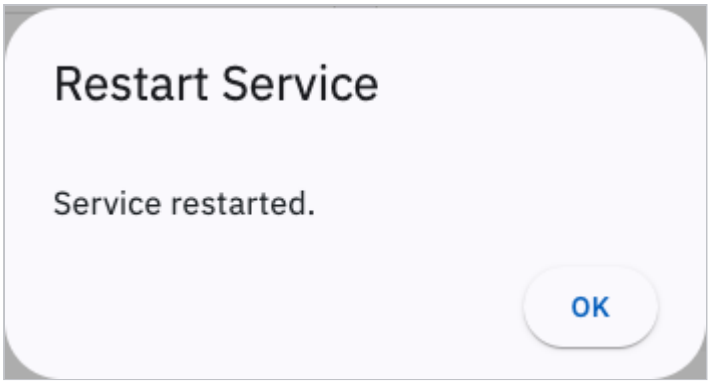
After you sign in, the gateway opens the **Home** page. The navigation bar at the top contains a **Home** link and four menus: **Configurations** (Machines/Group/Task/Communication), **Diagnosis** (API Test/Network Tools/Logger), **App** (DNC/Analysis/Monitor) and **System** (Network/Settings). The **Account** menu is in the top right corner. The page content is shown below the bar.



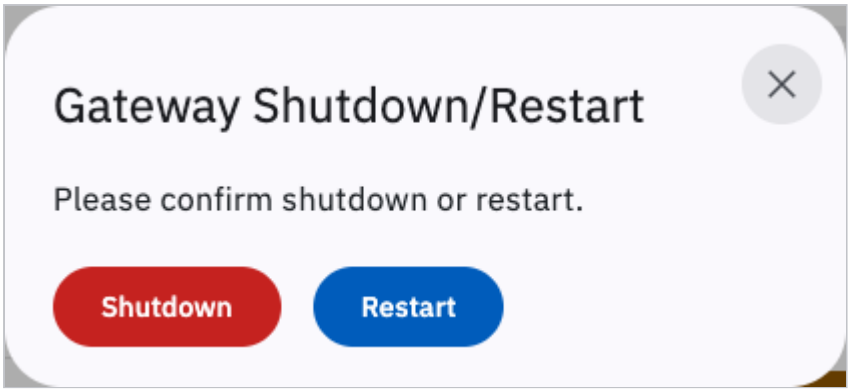
3.2.1. Function Buttons

The function buttons are in the top right corner of the "Gateway Status" panel on the Home page: **Restart Service** and **Gateway Shutdown/Restart**. After you change any configuration, the navigation bar also shows the prompt "Configuration changed. Restart the service for changes to take effect." — click it to restart the service.

The **Restart Service** button restarts the gateway service. After you change configuration parameters — on the **Machines**, **Groups**, **Communication** or **Tasks** pages, for example — you must click **Restart Service** for the changes to take effect. Clicking **Restart Service** opens the restart dialog; after about 5~10 seconds it reports that the service has restarted, and you can click **Confirm** to close it.



The **Gateway Shutdown/Restart** button powers the gateway hardware down or restarts it.



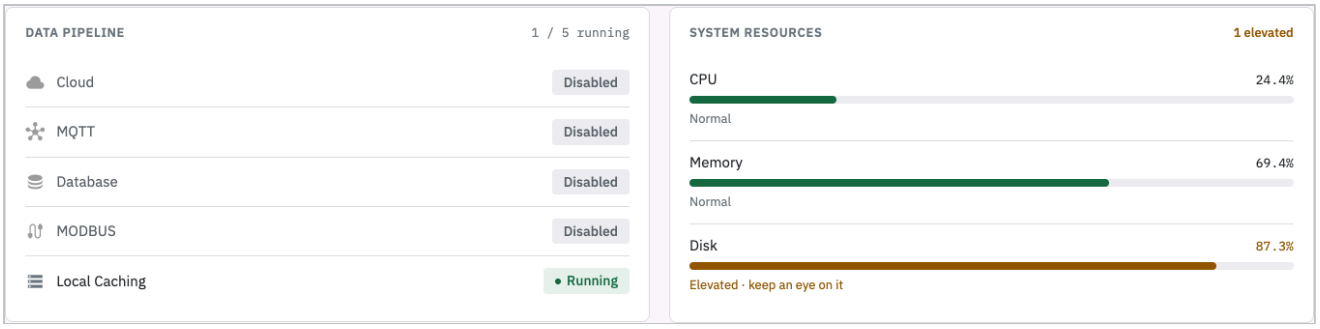
Click **Shutdown** to power the gateway hardware down safely. Wait until the gateway' s power LED goes out before unplugging the power cable.

Click **Restart** to restart the gateway hardware. The restart takes about a minute.

3.2.2. Gateway Status

"Gateway Status" shows the gateway' s main operating status. At the top is a summary of machine connections (the number of machines that are connected, offline or in connection error):

The "Data Pipeline" card shows the status of the Cloud, MQTT, Database, MODBUS and Local Caching services; the "System Resources" card shows CPU, Memory and Disk usage. All statuses refresh automatically every 5 seconds.



The individual entries under Gateway Status are described below:

Name	Description	Status
Internet	Whether the gateway can currently reach the internet — **unrelated to machine connections**	Internet reachable Internet unreachable: the gateway probes several internet targets in parallel every 15s and only reports this after 3 consecutive all-failed rounds Checking internet: the first round has not produced a verdict yet Internet status unknown: the browser cannot reach the gateway right now, so the state shown may be out of date
Cloud	Connection between the gateway and the cloud platform	Running: the Cloud service is on (see 3.6.1. Cloud Settings), correctly configured, and connected to the server Off: the Cloud service is off Delayed: an upload did not complete in time, for example because of network latency Other statuses: the Cloud service has failed
MQTT	Connection between the gateway and the MQTT server	Running: the MQTT service is on (see 3.6.3. MQTT Settings), correctly configured, and connected to the server Off: the MQTT service is off Delayed: an upload did not complete in time, for example because of network latency Other statuses: the MQTT service has failed

Name	Description	Status
Database	Connection between the gateway and the database	Running: the Database service is on (see 3.6.4. Database Settings), correctly configured, and connected to the server Off: the Database service is off Delayed: an upload did not complete in time, for example because of network latency Other statuses: the Database service has failed
MODBUS	Status of the gateway’s local MODBUS service	Running: the MODBUS service is on (see 3.6.2. MODBUS Settings), correctly configured, and the local server is running normally Off: the MODBUS service is off Delayed: an upload did not complete in time, for example because of network latency Other statuses: the MODBUS service has failed
Local Caching	Whether local caching is on or off	Running: local caching is on Off: local caching is off
Machine Connections - Connected	Number of machines connected successfully	0~255
Machine Connections - Offline	Number of machines offline	0~255
Machine Connections - Error	Number of machines with a connection error	0~255

For Cloud, MQTT, Database and MODBUS, the other statuses include “Initiated”, “Waiting for Connection” and “Error”. “Initiated” means the gateway has initialised the service but cannot proceed any further, because of a configuration error or a break in the network. “Waiting for Connection” means no reply has been received from the target server, which may indicate a network fault or a server fault; the local cache (database) also stays in “Waiting for Connection” until it has connected to InfluxDB, and while it waits the acquired data queues up and the backlog keeps growing. “Error” means a fault occurred at some stage.

3.2.2.1. Queue Backlog and Discarding It

When a service’s destination (the cloud platform, the MQTT server, the database, the Hub and so on) is temporarily unreachable, the data waiting to be sent is queued on the gateway and sent

once the connection comes back. Anything beyond the in-memory capacity is buffered on disk, so the backlog survives a gateway restart and is still sent afterwards.

Once a backlog reaches 100 items or more, the service’s row in the “Data Pipeline” card shows the current backlog size. If the backlog is shrinking, the status reads “Recovering”, meaning the connection is back and the data is being sent — no action is needed.

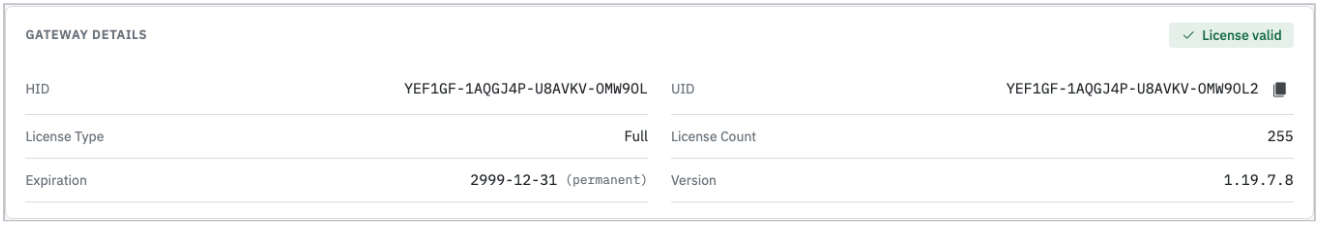
If the target server has been down for a long time (days, say), the backlog may no longer be worth sending, and sending it still costs disk space and bandwidth. In that case click the **Discard queued data** button on that row: after you confirm, everything that service has waiting to be sent, in memory and on disk, is **permanently discarded** and cannot be recovered, so use it with care. Discarding only affects the pending queue; it does not affect data acquisition from the machines or data already uploaded.

> **Caution**
>
> Discarded data cannot be recovered. If it still needs to be uploaded, restore the connection to the target server first and let the gateway send the backlog.

3.2.3. Gateway Details

The **UID** under “Gateway Details” is the device’s unique identifier. You need to give it to support when you request help or upgrade your license.

- **Expiration** is the date on which the collection license expires.
- **License Count** is the number of collection licenses configured on the gateway, that is, how many machines the gateway may collect from at the same time.
- **License Type** is the type of collection license. Contact support to find out which features each type provides.
- **License Status** shows whether the current license is valid. “Invalid” means the license has not been assigned, or has expired.
- **Version** is the software version currently running on the gateway.



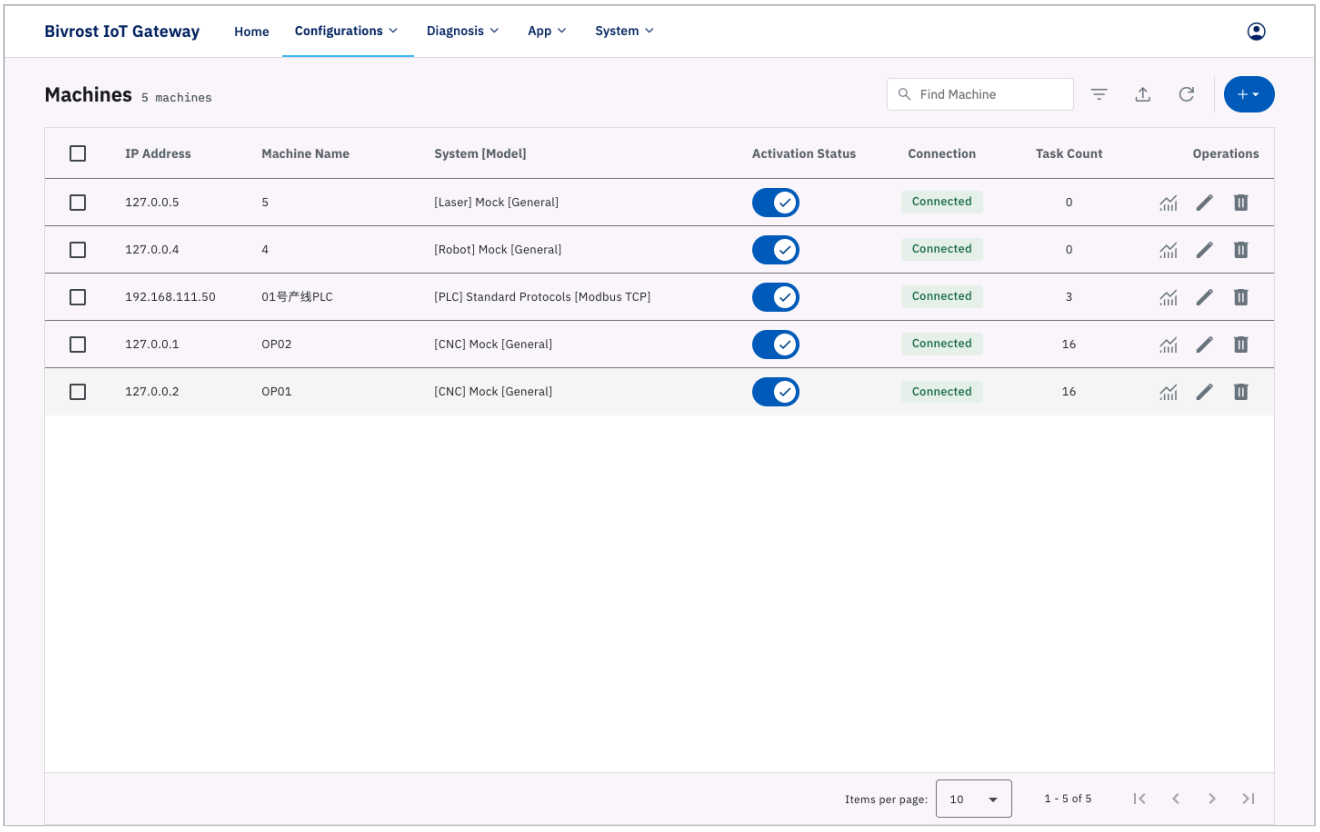
In the screenshot above, the gateway’s license type is “Full”. The available interfaces include Basic Data (Status, Count, Alarm, current program name, Current Program Block and so on),

Axis Data (Position, Feed and Spindle, Load), Cycle Data, OEE Monitoring, PLC Data, tool offset read/write and DNC. The gateway can collect from up to 255 machines at the same time, and the license expires on 31 December 2999. The current license status is valid. The gateway software version in the screenshot is 1.19.4.19.

3.3. Machines

The **Machines** page is where you configure the machines you want to collect data from.

After changing anything on the **Machines** page, go back to Home and click **Restart Service** to apply the changes.



The table on this page lists the IP address, machine name, system model, connection status and task count of every configured machine. A green check in the **Connection** column means communication is working; any other icon indicates a problem. Hover over the icon to see an explanation, or refer to the table of common connection icons below. **Task Count** is the number of automatic collection tasks enabled for that machine.

IP Address	Machine Name	System [Model]	Activation Status	Connection	Task Count	Operations
127.0.0.5	5	[Laser] Mock [General]		Connected	0	
127.0.0.4	4	[Robot] Mock [General]		Connected	0	
192.168.111.50	01号产线PLC	[PLC] Standard Protocols [Modbus TCP]		Connected	3	
127.0.0.1	OP02	[CNC] Mock [General]		Connected	16	
127.0.0.2	OP01	[CNC] Mock [General]		Connected	16	

Items per page: 10 1 - 5 of 5

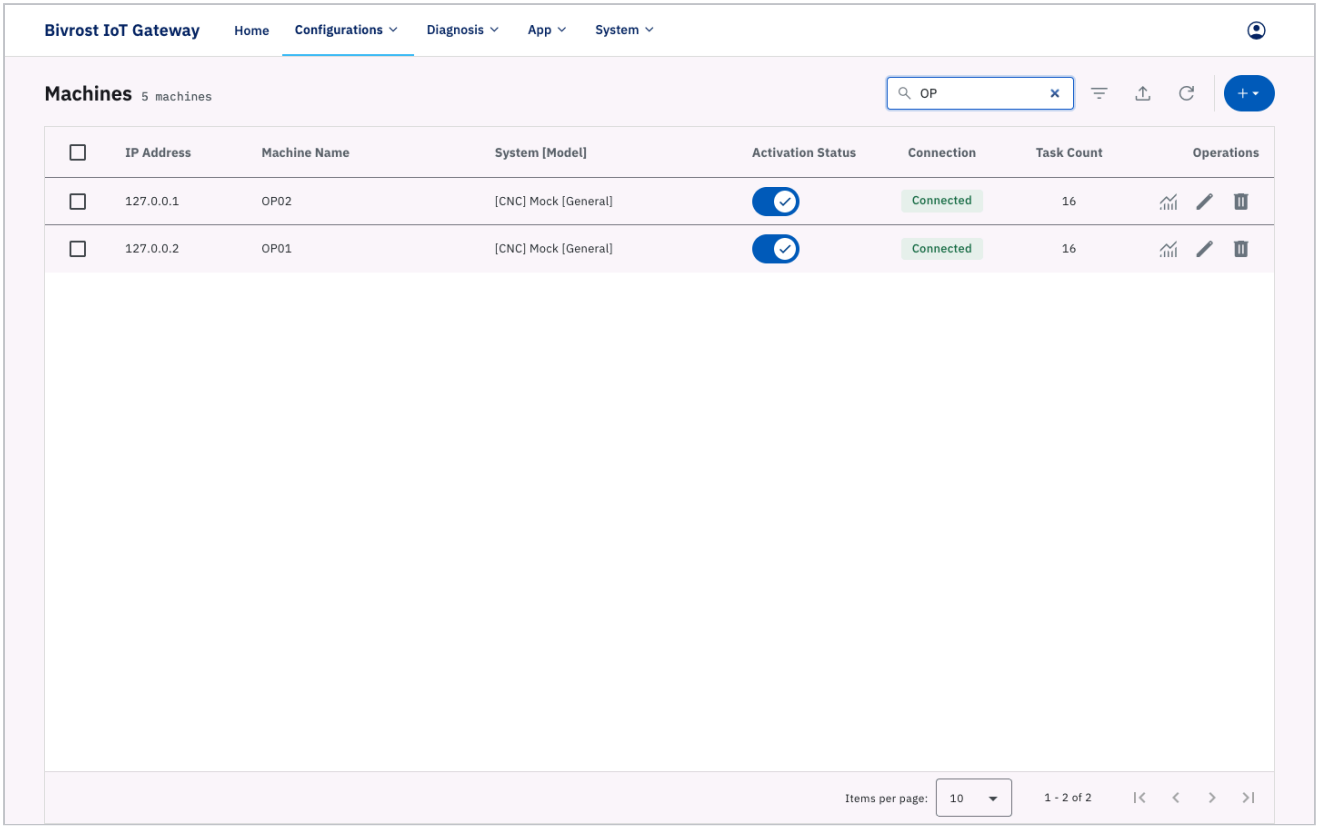
Common connection icons:

Icon	Meaning
✅ Green check	Communication successful.
🚫 Red no-entry	Unactivated. If activation fails in the create/edit machine dialog, the likely causes are: 1. the license type does not match the machine type; 2. the number of activated machines has reached the licensed maximum.
🔄 Yellow circle	Machine not loaded. Likely causes: 1. the gateway is waiting for the result of the first communication attempt after a service restart, which normally takes up to 10 seconds; 2. the machine was activated but the service has not been restarted.
🔗 Broken link	Machine offline. Likely causes: 1. the machine is powered off; 2. the network is not connected; 3. the machine configuration is wrong; 4. the network settings on the machine are incomplete.
❌ Red cross	Task not ready. Likely causes: 1. the machine configuration is wrong (for example the wrong system model); 2. the network settings on the machine are incomplete (wrong port setting, firewall not opened, and so on); 3. the machine’s IP address conflicts with another device.

> **Note**
>
> The **Connection** column refreshes automatically every 10 seconds. After changing a machine’s configuration, click **Restart Service** on Home. Following a **Restart Service** the gateway needs about 10 seconds to initialise its connections to the machines, so a “Not Loaded” status during that period is normal — wait about 10 seconds and the new connection status will appear on its own.

At the top right of the page are, from left to right, the **Find Machine** box and the **Filter**, **Batch Import**, **Refresh** and **Add Machine** buttons.

Type a keyword into the 🔍 **Find Machine** box and the matching machines appear below. The search covers IP address, machine name and system model. Clear the box to show all machines again.

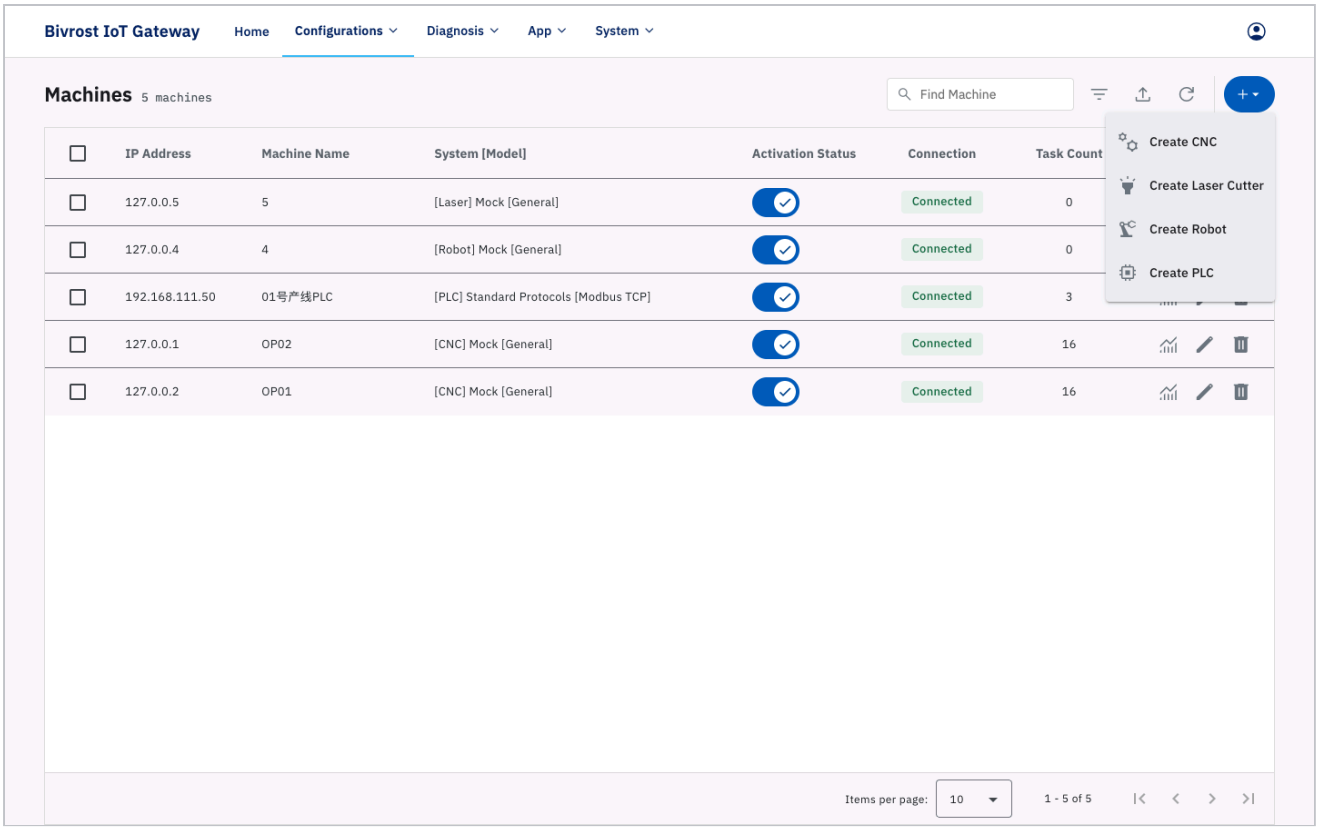


Click ↻ **Refresh** to reload the whole machine list.

Click + **Add Machine** to add a machine — see 3.3.1. Adding a Machine.

3.3.1. Adding a Machine

Click + **Add Machine** at the top right. The drop-down menu contains **Create CNC**, **Create Laser Cutter**, **Create Robot** and **Create PLC**.

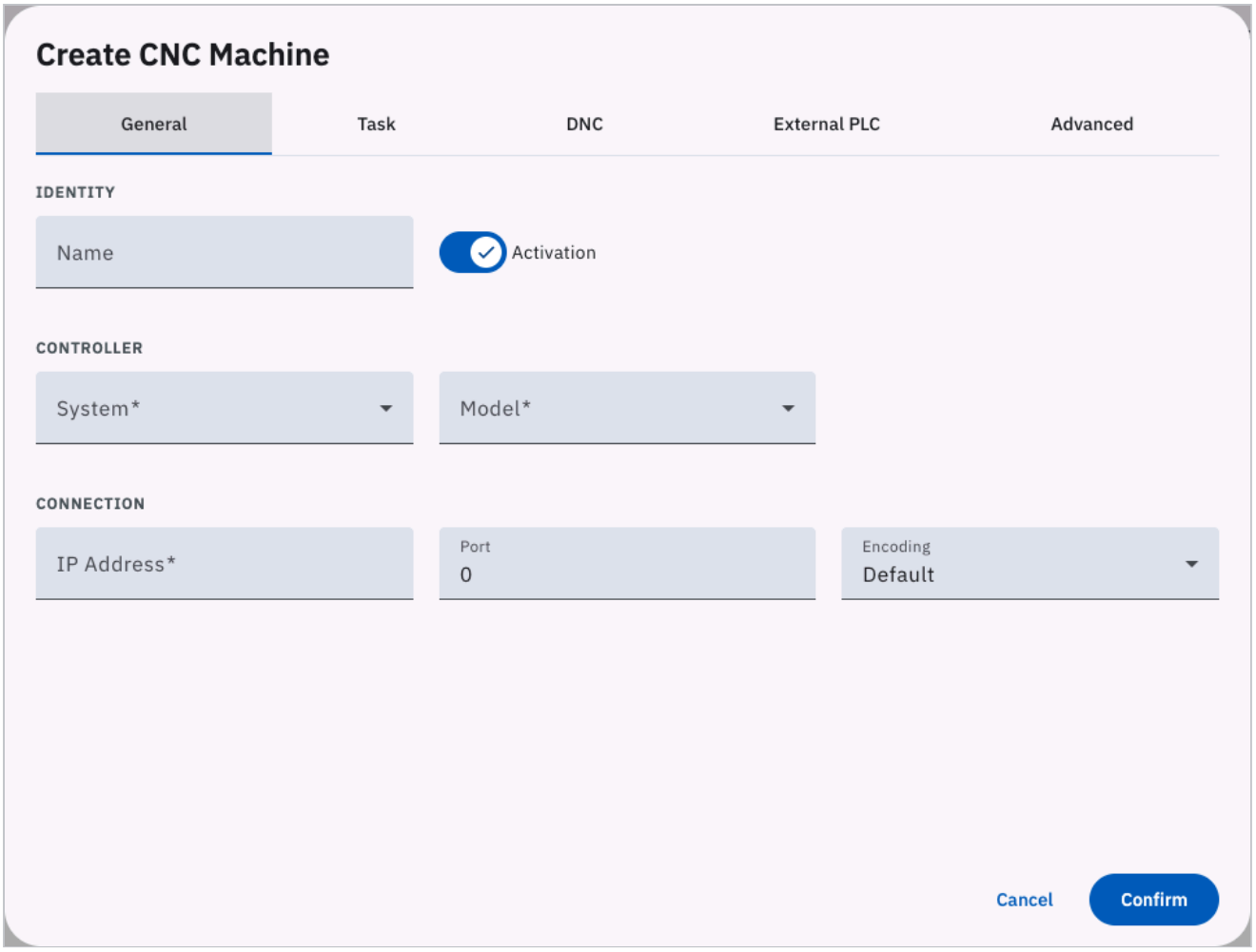


The dialogs for the different machine types are similar. The main differences are:

1. the tasks available on the Task tab differ by machine type;
2. robot and PLC machines have no DNC tab;
3. some machines have special settings (see the machine setup documentation, or contact support).

The example below adds a CNC machine:

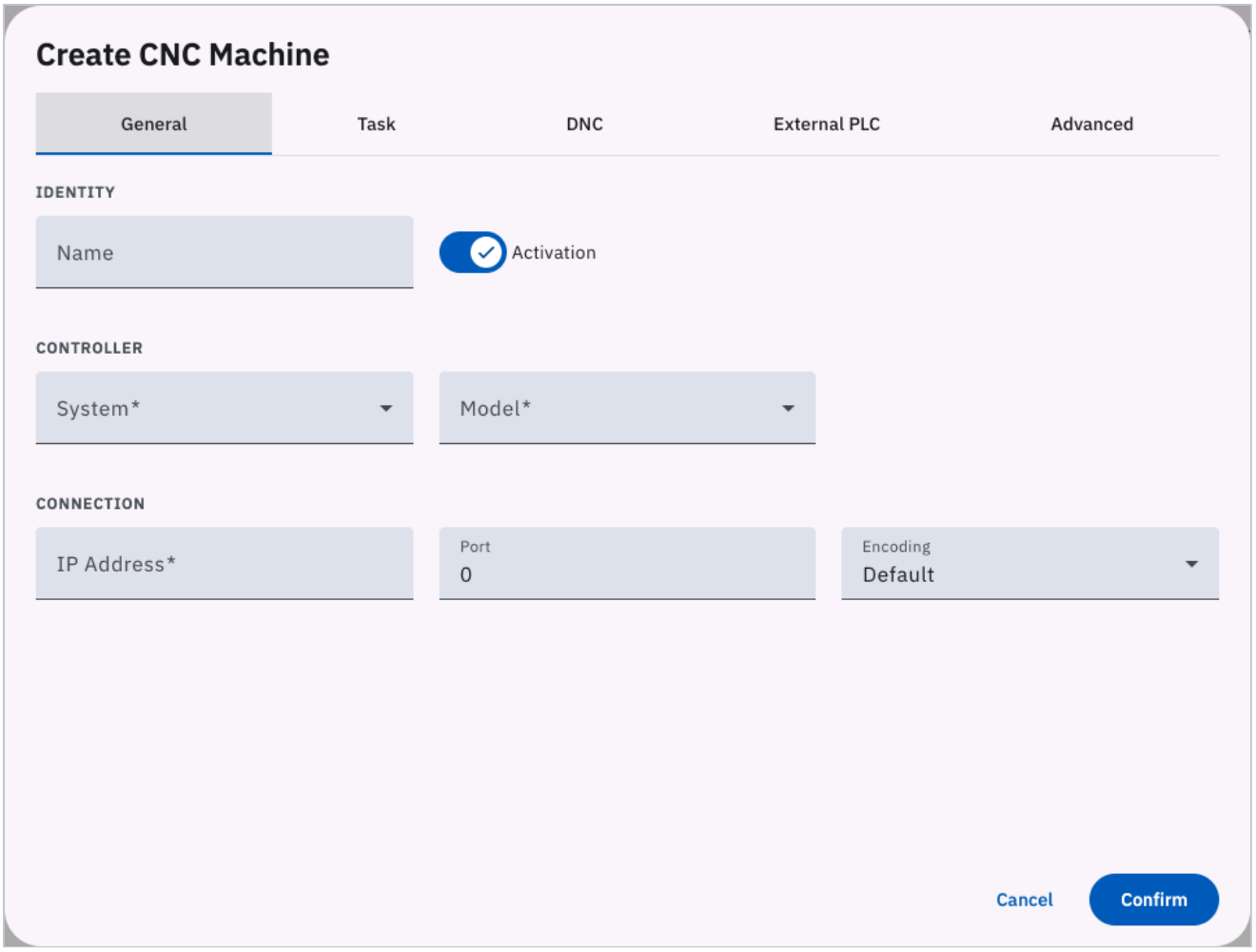
Click **Create CNC**. The “Create CNC Machine” dialog opens with the tabs “General”, “Task”, “DNC”, “External PLC” and “Advanced”. Fill in all of the tabs and click **Confirm** to add the machine. Anything you need to change afterwards can be edited in the “Edit CNC Machine” dialog (see 3.3.2. Editing a Machine).



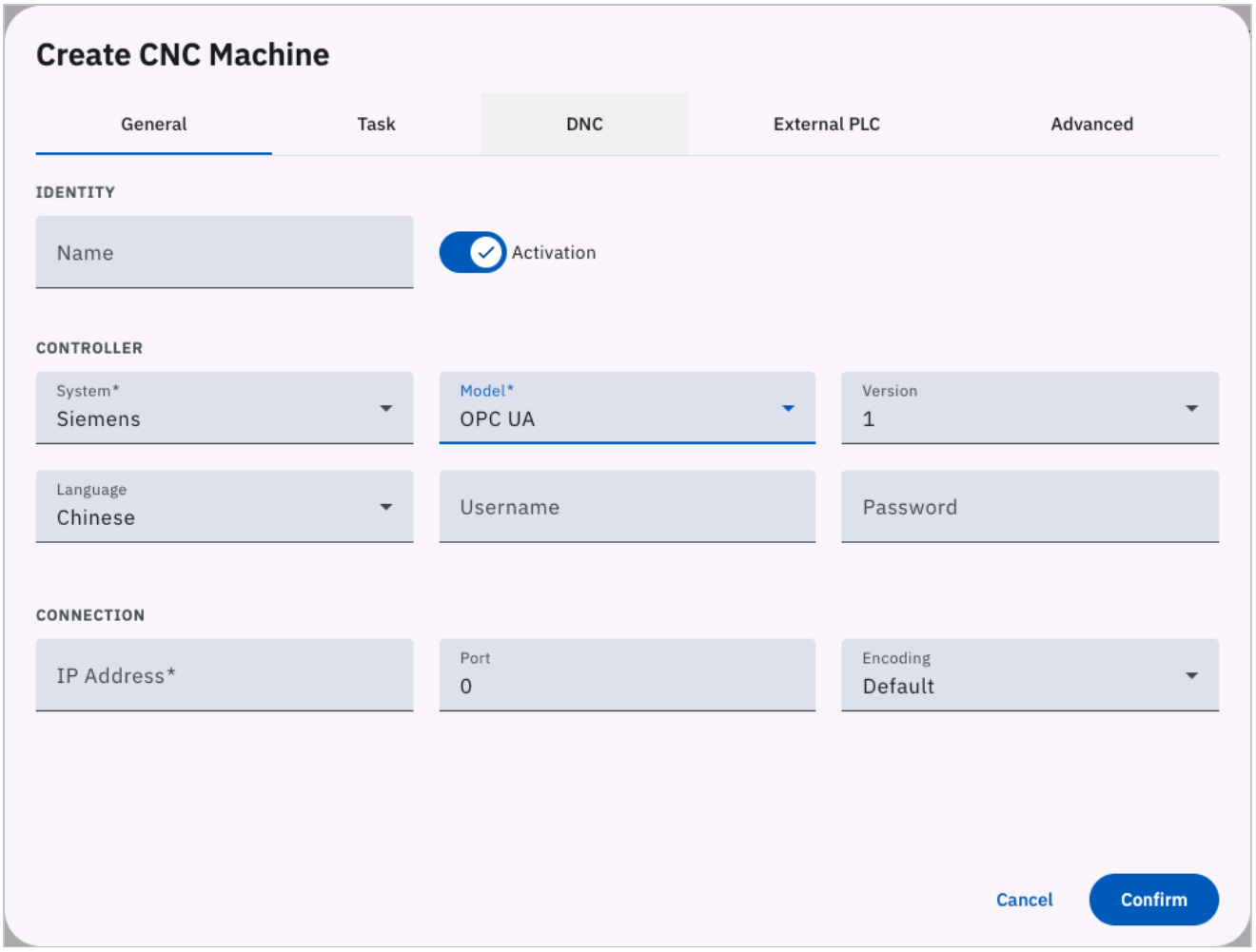
3.3.1.1. General Settings

The "General" tab holds the machine's connection settings: choose "System" and "Model", enter the machine's IP address and machine name, and select an encoding. If you choose Mock as the system, the IP address must be 127.0.0.X, where X is 1–255. Note that when "Activate" is selected in the dialog the machine is license-activated: it uses one collection license and collection is running. Clearing the checkbox withdraws the collection license used by that machine, pauses collection, and makes the withdrawn license available to another machine.

If the communication port on the machine is set to something other than the default, set "Port" to the matching value; otherwise leave it at 0 to use the default port. The encoding option is used when parsing text (alarm text, String values in PLC data, file names, the contents of received files, and so on); the wrong encoding produces garbled text. The default is normally correct.

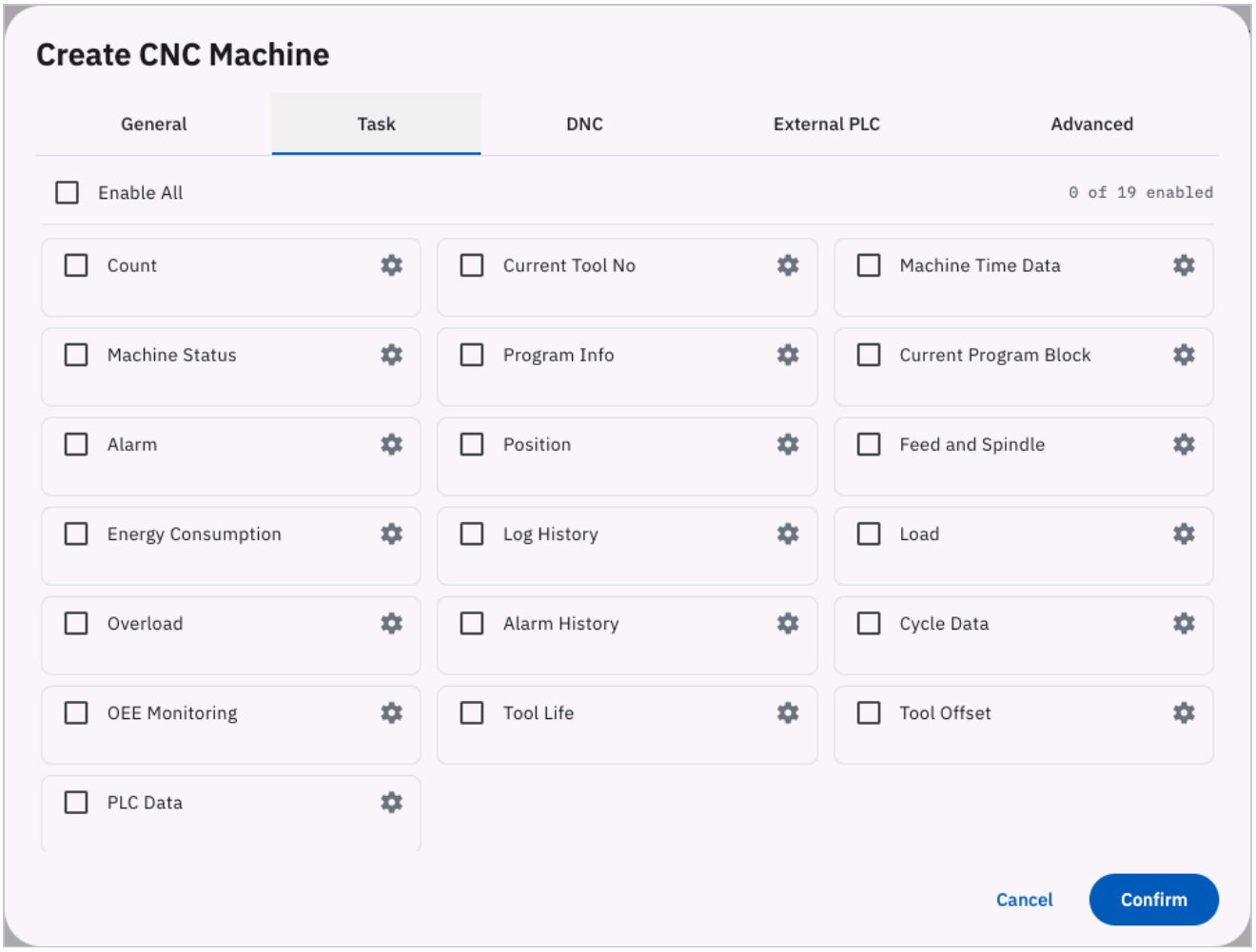


Some models and connection methods — such as Siemens machines that expose an OPC UA service — require the OPC username and password configured on the machine. The fields are hidden for models and connection methods that need no credentials.



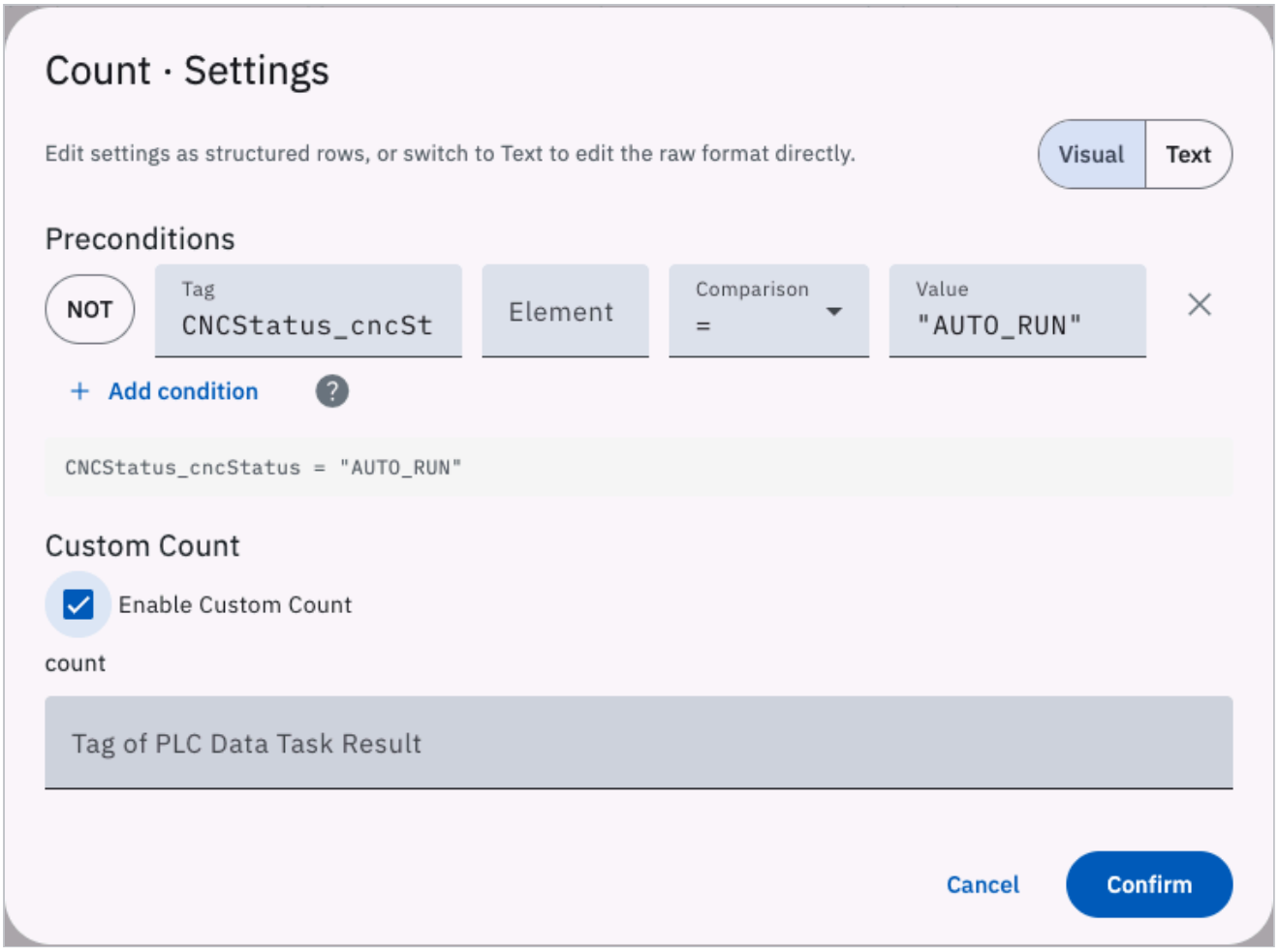
3.3.1.2. Task Settings

The "Task" tab configures the machine's automatic collection tasks. Select an available interface to turn on automatic collection for the corresponding task. Interfaces not covered by your license are shown as "Unlicensed" and cannot be selected. The number of enabled tasks appears in the "Task Count" column of the **Machines** page.



This tab turns the machine' s automatic collection tasks on and off. Task results can be published through the cloud platform, MODBUS, MQTT or a database. To receive task results you must enable and configure at least one of the cloud, MODBUS, MQTT or database transports in 3.6. Communication. Collection intervals and detailed task settings are configured in 3.5. Tasks. For the format in which task results are published over MODBUS, MQTT and databases, see 6. MODBUS Communication, 7. MQTT Communication and 8. Database Communication. Collection through the HTTP interfaces does not require any machine task to be enabled.

The ⚙ button to the right of a task opens its advanced settings — Precondition, Target Channels, Target Tools, Custom Functions, Special Settings and so on.



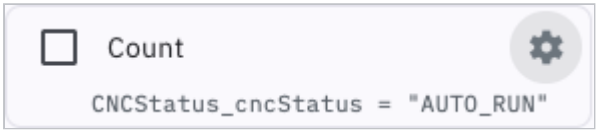
3.3.1.2.1. Precondition

When a precondition is set, the gateway evaluates it before each run of the task and only performs the collection if the condition is met. Left blank, the task has no precondition and always runs. For the precondition syntax, see 11.2.2. Precondition.

For example:

```
CNCStatus_cncStatus = "AUTO_RUN" and Load_spindleLoad[0] > 50
```

Tasks that have a precondition are marked with a blue branch icon to the right of the task name.



3.3.1.2.2. Target Channels

Some machines expose data for several channels — for example channel 1 for the main spindle and channel 2 for the sub-spindle. The default target channel is 0, which collects data from the

main channel only. To collect data from several channels, list them under target channels. The tasks that support this setting include Current Tool No, Machine Status, Program Info, Current Program Block, Position, Feed and Spindle, Load, Overload and Cutter comp.

For example: `0;1-3`

3.3.1.2.3. Target Tools

The Tool Life and Cutter comp tasks require target tools to be entered in the dialog behind the ⚙ button next to the task. How tools are specified depends on the machine’s control system; by default no tools are specified. Follow the example in the hint text, or see 5.5.1.14. readOffsetData (Read Offset Data) and 5.5.1.23. readToolLife (Read Tool Life).

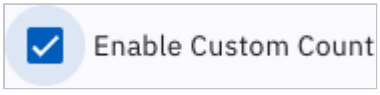
On a Siemens control, for instance, a tool is defined by a tool number (Tool) and an offset number (Offset):

```
1.2;3-5.1;7-9.1-3
```

- `1.2` means tool 1, offset 2;
- `3-5.1` means tool 3 offset 1, tool 4 offset 1 and tool 5 offset 1;
- `7-9.1-3` means tools 7, 8 and 9, offsets 1, 2 and 3 on each of them.

3.3.1.2.4. Custom Functions

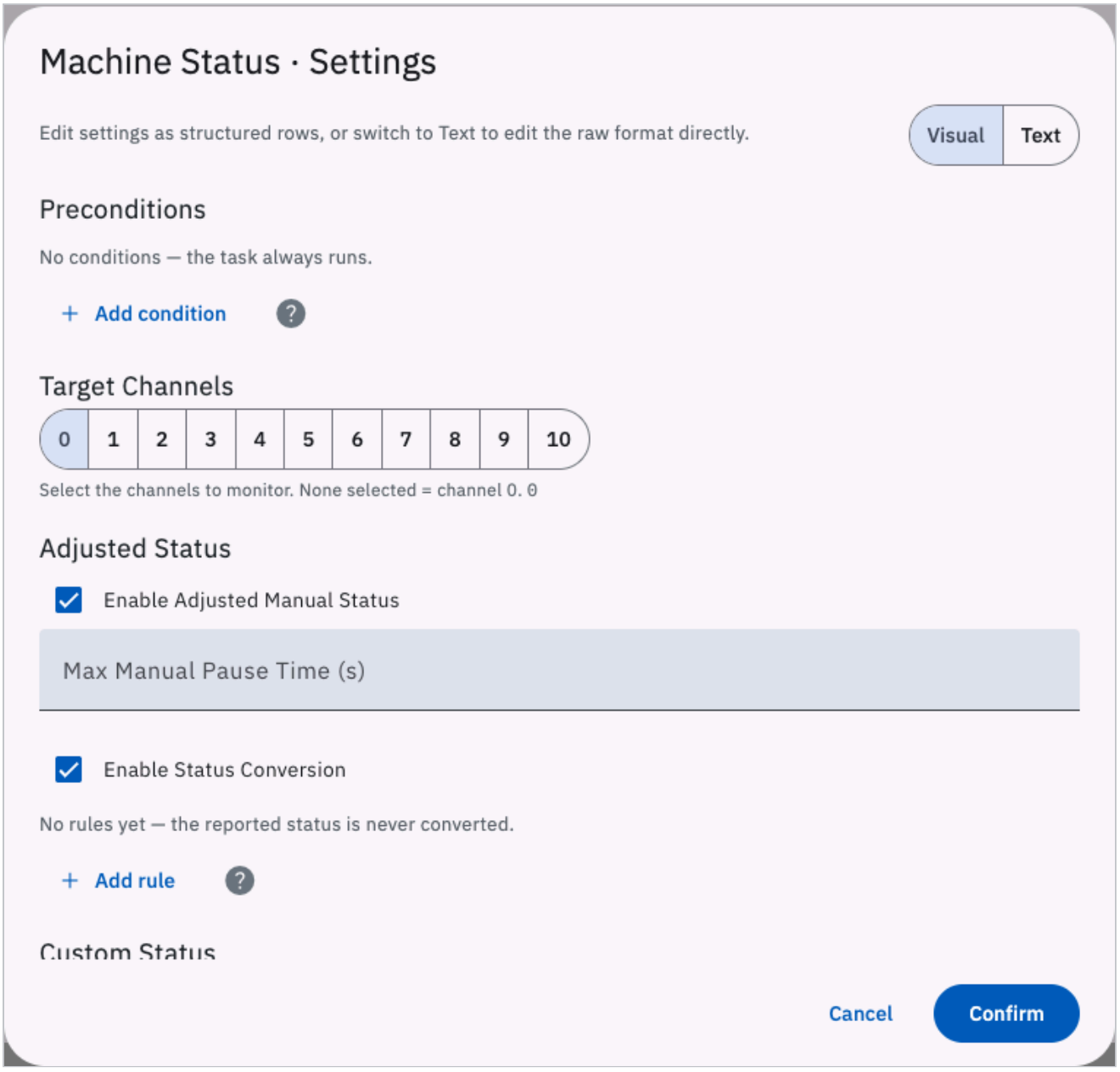
The Count, Machine Status and Alarm tasks support custom functions, which substitute the result of a PLC task for the data returned by the native interface. Custom functions are mainly used on controls where a customised module replaces the native one, or where the control does not support the required communication and the manufacturer supplies an address list so the data can be read from PLC addresses instead.



> **Caution**
>
> The data type of the PLC data task result must be compatible with the data type of the native interface. The `count` field of the Count task, for example, is Int32, so the data behind the tag you enter must be convertible to Int32. If the raw count in the PLC is a string, use the post-processing feature of the PLC data task to convert it to Int32 and enter the tag of the processed value. For an example of the PLC task settings behind a custom status, see example 4 in 11.2.4.2. $POST$ Post-processing.

3.3.1.2.5. Special Settings

The Machine Status task supports adjusted status settings: you can enable manual-status adjustment and status conversion, both off by default. These are the same settings as the ones with the same names in 3.5.8. Machine Status Monitoring Settings; those are global, and enabling them here overrides the global settings for this machine.



With **Enable Adjusted Manual Status** or **Enable Status Conversion** turned on, the status is adjusted according to the corresponding rules and published as `adjustedStatus` (4.2.4. CNCStatus: Machine Status). Calculations that involve machine status — OEE, cumulative status time and so on — use the adjusted status in preference to the raw one. With both options off, `adjustedStatus` is identical to the real-time status reported by the machine.

The rules for manual-status adjustment are:

1. the machine is currently in the Manual state;
2. the time since the machine went from a non-Manual state to Manual exceeds the configured Max Manual Pause Time;
3. the machine’s position data has stayed unchanged for longer than the configured Max Manual Pause Time.

Normally `adjustedStatus` matches the machine’s real-time status; when all of the conditions above are met at once, `adjustedStatus` is no longer Manual but is adjusted to Wait.

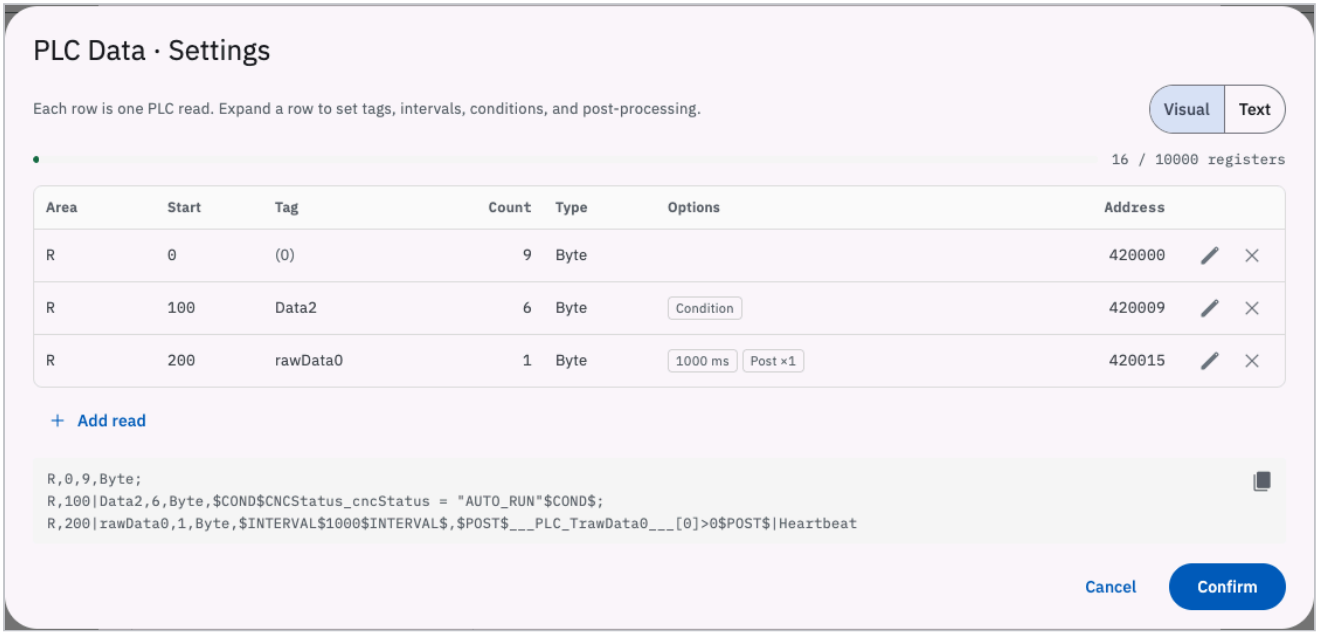
With status conversion enabled, `adjustedStatus` is set according to the commands you enter, for example:

```
CNCStatus_cncStatus = "MANUAL_MDI_RUN" | AUTO_RUN
```

`CNCStatus_cncStatus = "MANUAL_MDI_RUN"` is the precondition (see 11.2.2. Precondition): if the `cncStatus` value (the running status, of type String) returned by this machine’s Machine Status task is “MANUAL_MDI_RUN”, then `adjustedStatus` is set to AUTO_RUN. Separate multiple conversion commands with “;”.

When both options are enabled, manual-status adjustment runs first and status conversion second.

The PLC Data task requires the commands for the target PLC data to be entered in the dialog behind the ⚙ button next to the task — see 11.2.1. PLC Data Task.



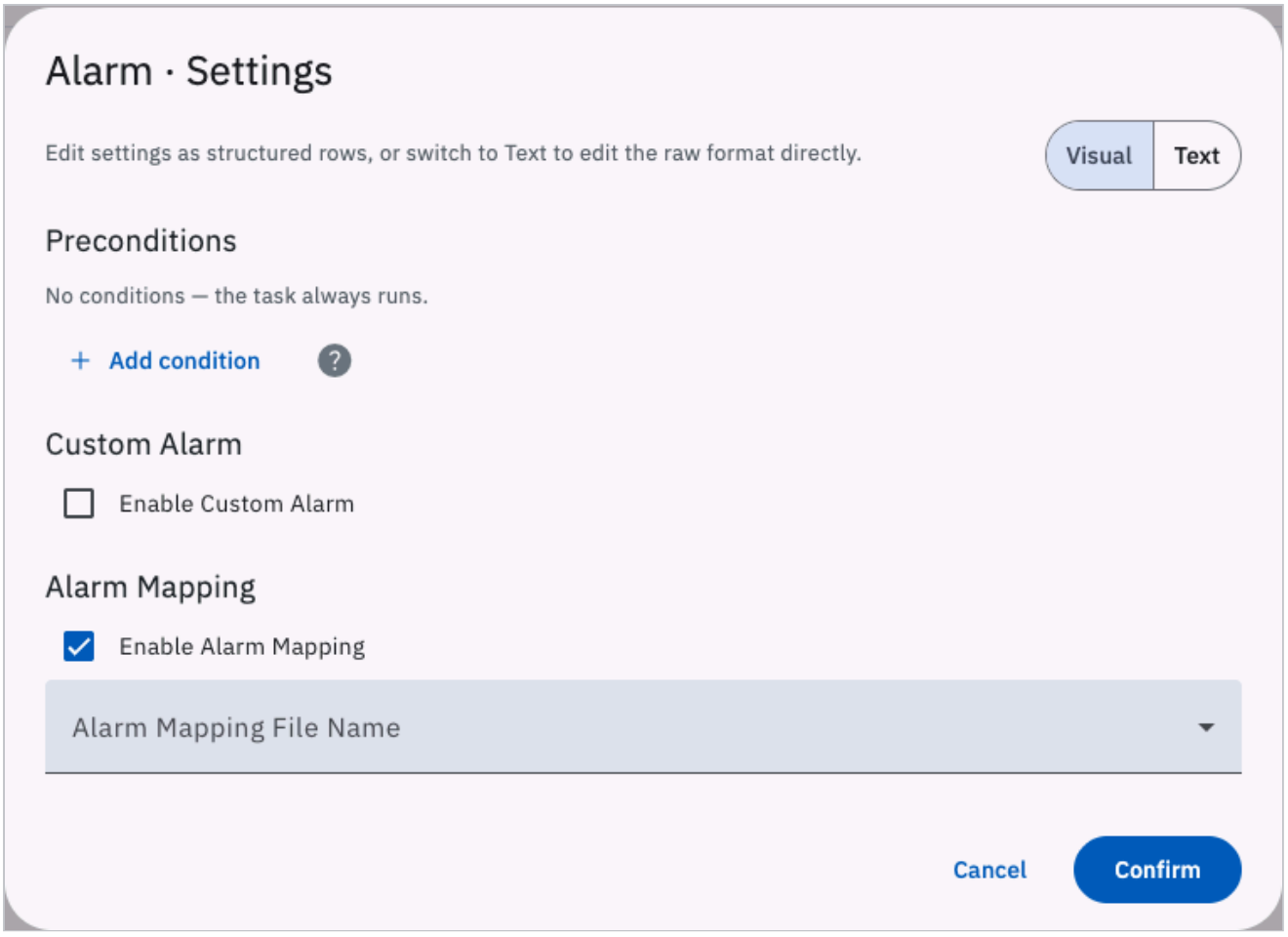
If MODBUS communication is enabled (3.6.2. MODBUS Settings), the reserved MODBUS address space limits a single machine’s PLC data tasks (including external PLC data) to 10000 address

registers after conversion — see 6.1. Machine-related Data.

3.3.1.2.6. Alarm Mapping

The Alarm task supports alarm mapping, which replaces the alarm numbers reported by the machine with your own alarm text. It suits machines that report only an alarm number and no alarm text, or installations where alarm descriptions need to be standardised.

In the dialog behind the ⚙ button next to the Alarm task, select "Enable Alarm Mapping" and then pick an uploaded mapping file from the "Alarm Mapping File Name" drop-down.



Mapping files are uploaded and managed in 3.5.7.1. Alarm Mapping Files; a file must be uploaded there before it can be selected here. After enabling it, click **Confirm** to save, then go to Home and click **Restart Service** to apply the change.

3.3.1.2.7. Additional Notes

Machine OEE data covers Off Time, Wait Time, Emergency Time, Autorun Time, Manual Time and machine Availability — see 11.1.1. Machine OEE Data. OEE data depends on local caching, so the "Local Caching" switch under Options in 3.12. Settings must be turned on.

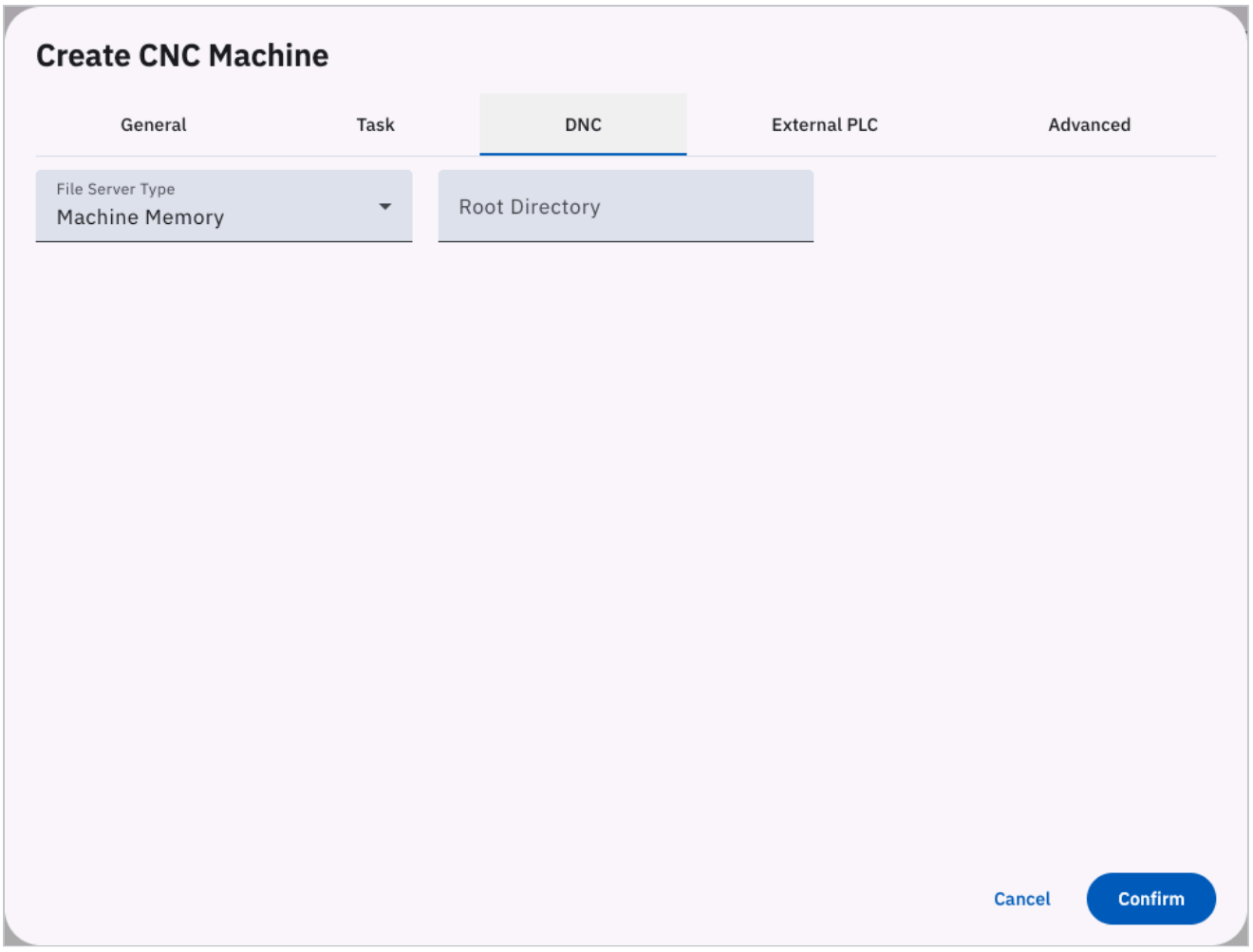
Write Offset Data and DNC are not available as machine tasks; they must be called through the HTTP interfaces or the MQTT RPC interfaces — see 5. HTTP Communication and 7.2. RPC Interface.

Cumulative Status Time is the total time spent in each status since the gateway started collecting from the machine, in seconds. The value is stored on the gateway, bound to the machine ID (MachineID), and is never reset.

Cycle Data gives the duration of the last cycle; the gateway derives it from Machine Status, Count and Program Info.

3.3.1.3. DNC Settings

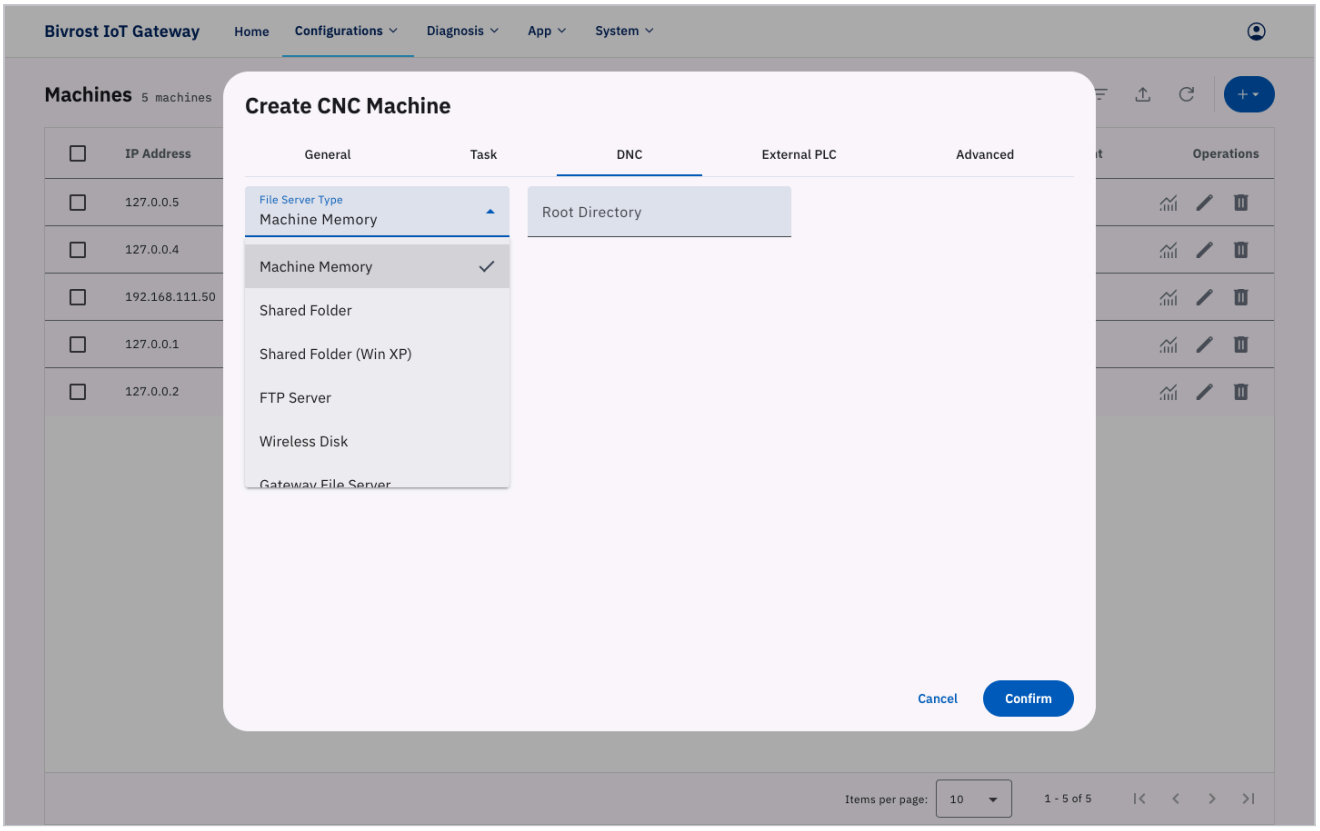
The “DNC” tab sets the file transfer method and target location.



The “File Server Type” options are:

- **Machine Memory** (default), the machine’s built-in storage, including a CF card plugged into the machine (supported on some machines; the root directory must point at the CF card path);

- **Shared Folder** and **Shared Folder (Win XP)**, which transfer files through a shared folder;
- **FTP Server**, which transfers files over FTP;
- **Wireless Disk**, an external storage device attached to the machine that connects to the gateway over WiFi and to the machine over USB or RS232;
- **Gateway File Server**, a file server hosted on the gateway itself that the machine can reach over FTP or a shared folder — see 3.6.5. Gateway File Server Settings.

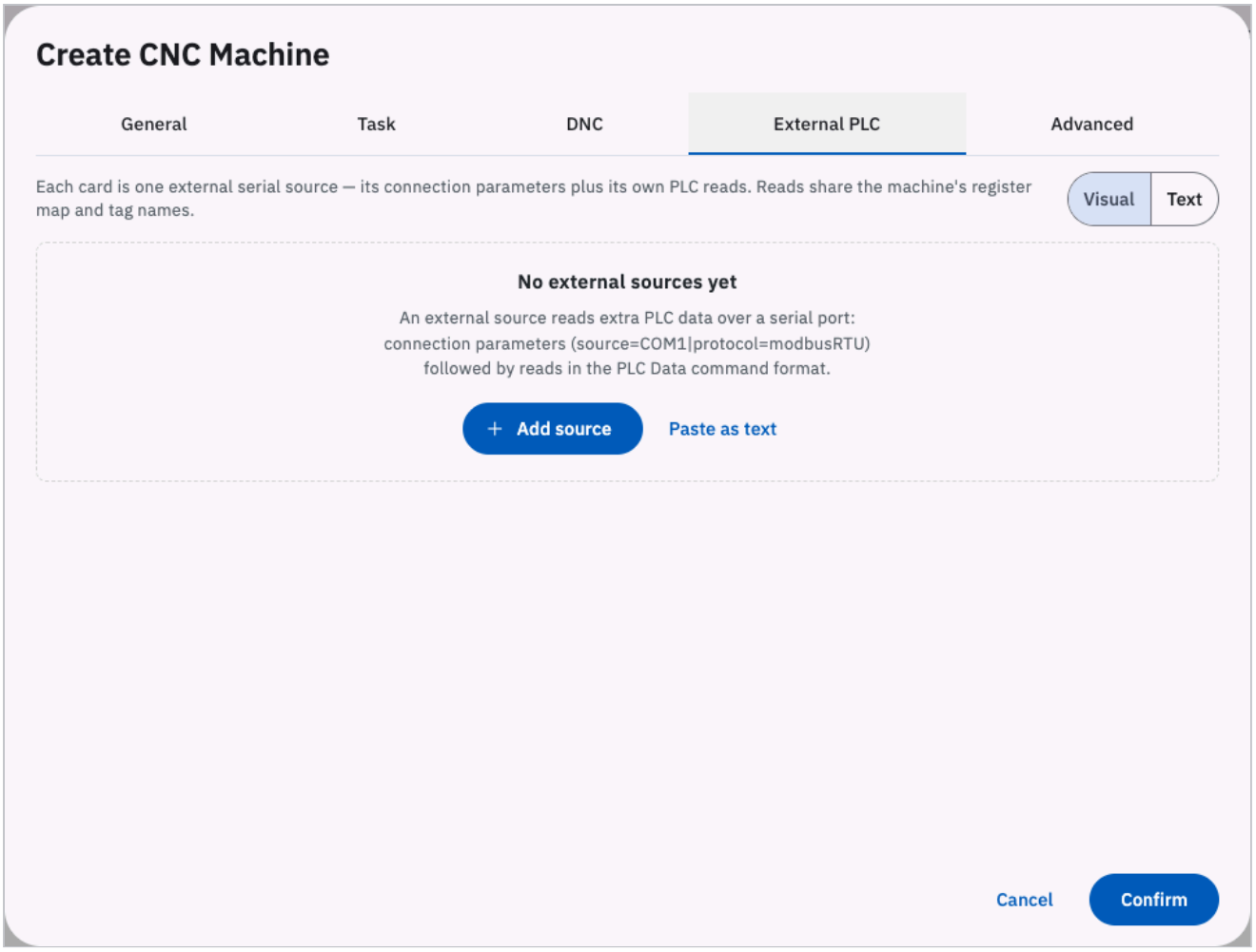


If the machine's own storage is too small, or the machine does not support transferring programs directly to its internal storage, use the "Gateway File Server" instead. You can also attach file server hardware to the machine and transfer programs through "Shared Folder", "FTP Server" or "Wireless Disk".

If the root directory is left blank, the machine's default path is used. The default path differs by machine model — see 5.6. File Management Interfaces.

3.3.1.4. External PLC Settings

The "External PLC" tab lets you add commands that collect data from an external PLC attached to equipment around the machine — see 11.2.5. External PLC Data Task. Once a valid external PLC data task has been added, collection starts automatically after a service restart. The collection interval for external PLC data tasks, and the way their results are retrieved, are the same as for PLC data tasks.

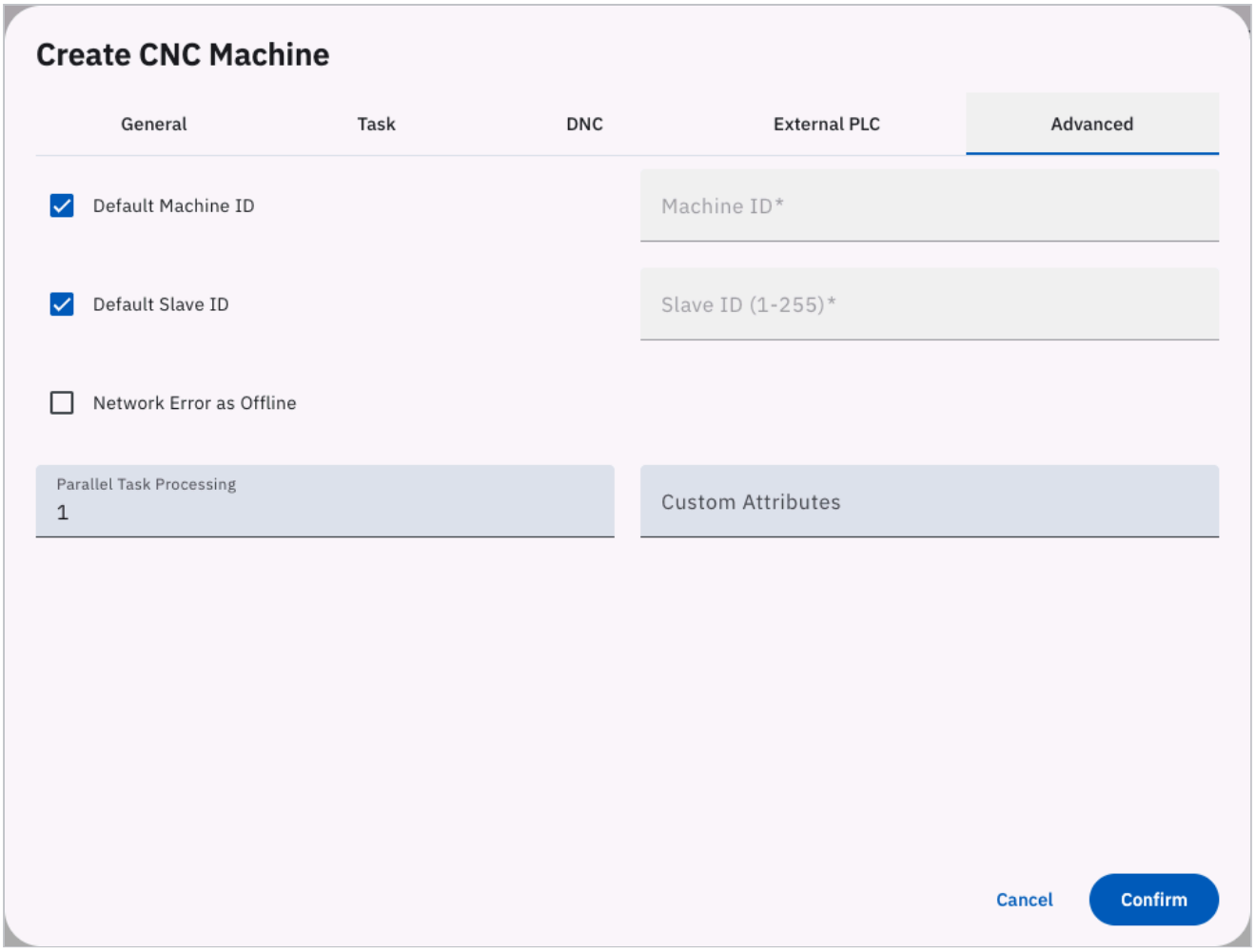


For example:

```
source=COM1|protocol=modbusRTU;4x,0,1,Int16;4x,10,2,Int32
```

3.3.1.5. Advanced Settings

The “Advanced” tab lets you customise the machine ID (MachineID), the slave ID (SlaveID), Network Error as Offline, Parallel Task Processing, and custom attributes.



Machine ID identifies the machine’s data on every transport except Modbus. The default machine ID is formed by joining the third and fourth octets of the machine’s IP address and stripping the leading zeros. If the third octet is not 0, the fourth octet is padded to three digits. For example, 192.168.0.1 gives a default machine ID of 1, and 192.168.1.12 gives 1012.

Slave ID identifies the machine’s data on Modbus. The default slave ID is the fourth octet of the machine’s IP address; 192.168.1.12 gives a default slave ID of 12.

Network Error as Offline is off by default. When it is on and the machine fails to connect for network reasons on the first attempt (previously reported as “Network Error”), the gateway treats the machine as offline instead of reporting a connection error. This suits machines that are powered down or lose network connectivity from time to time, where all such unreachable states should be treated uniformly as offline.

Parallel Task Processing is the number of task managers — that is, threads — allocated to the machine. The default is 1. Raising it consumes more system resources, so only raise it when the machine has a large number of tasks with very short intervals and the default setting cannot keep collection up to date. Some control systems do not support multi-threaded connections and must be left at 1. Note that the sum of the Parallel Task Processing values of all activated

machines should be less than the maximum number of task managers (see 3.5.9. Task Manager Settings).

Custom Attributes add user-defined attribute tags to the task data returned by the gateway. The format is:

```
{"attribute":value,…}
```

Example:

```
{"Custom ID":1, "Custom String": "abc123 string"}
```

In the task data, custom attributes are added after the `<entityID>` attribute (MachineID, in the case of a machine).

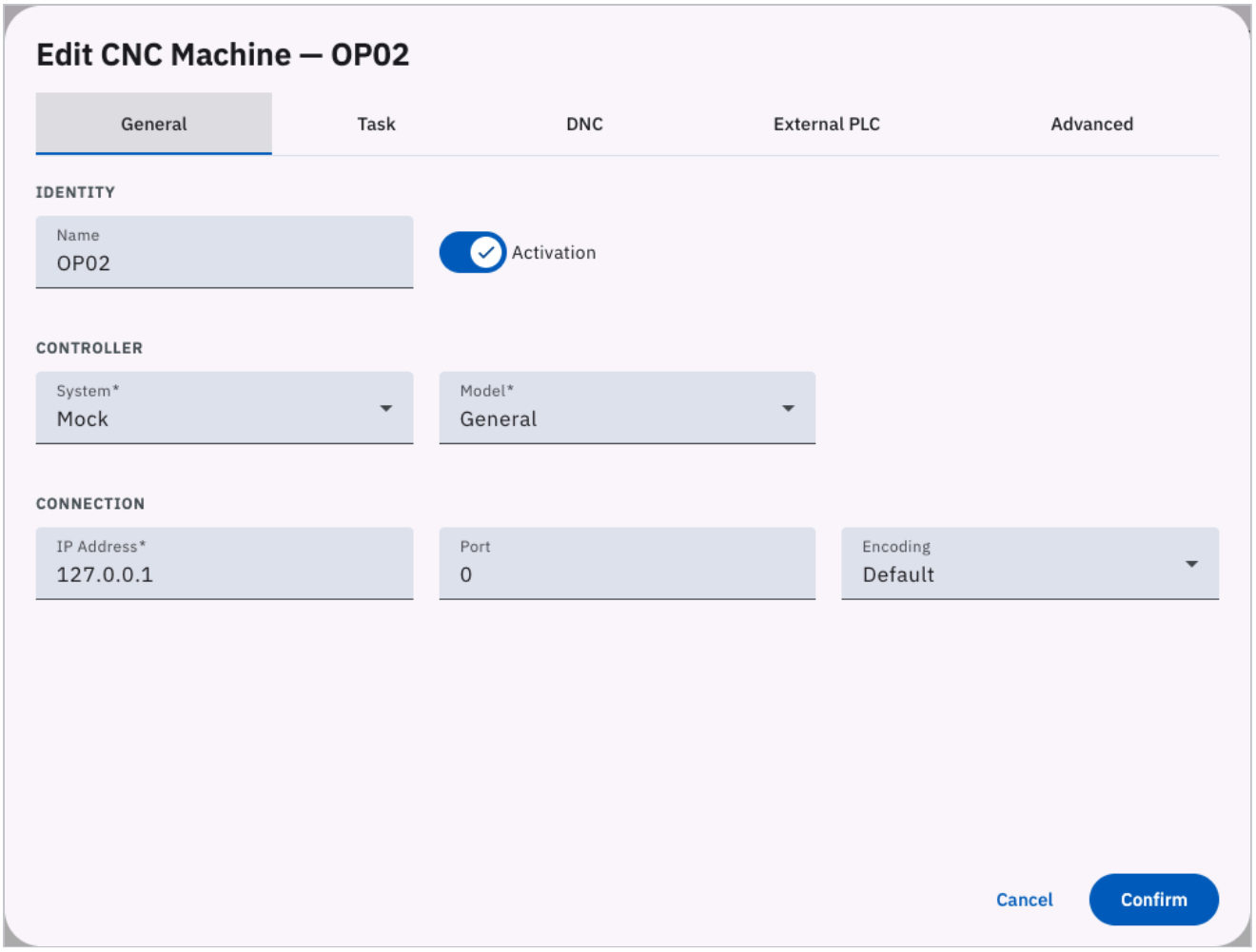
3.3.1.6. Connection Status

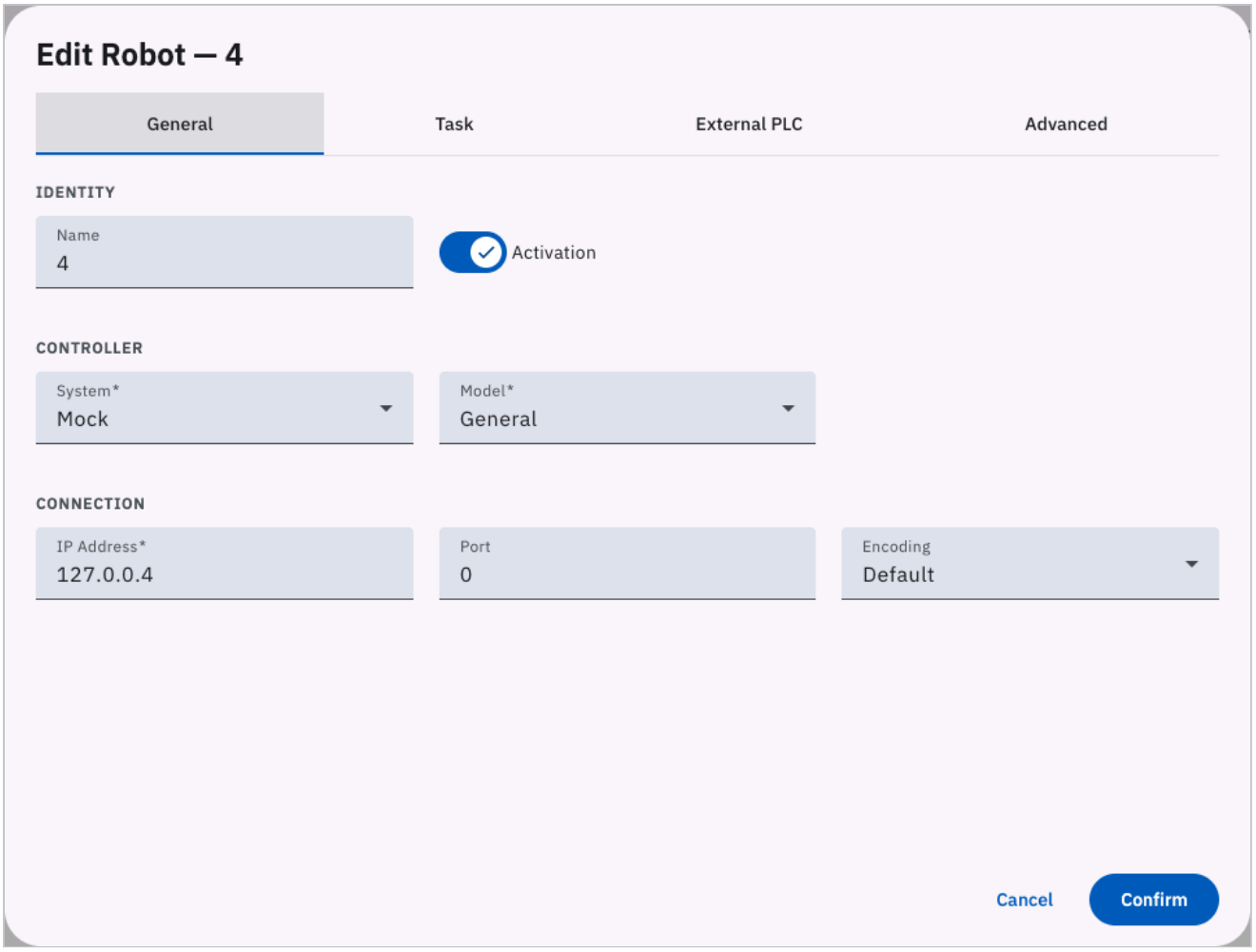
Once everything in the create machine dialog is set, click **Confirm** to save. The new machine appears on the **Machines** page. Note that its "Connection" icon is a yellow circle at this point, meaning the machine is not loaded; go to Home and click **Restart Service** to load the newly configured machine.

After the **Restart Service**, the Connection icon turns into a green check, meaning communication is working. Any other icon means the configuration is wrong or there is a fault in the machine, the gateway or the network — hover over the icon for an explanation, or refer to the table of common connection icons in 3.3. Machines.

3.3.2. Editing a Machine

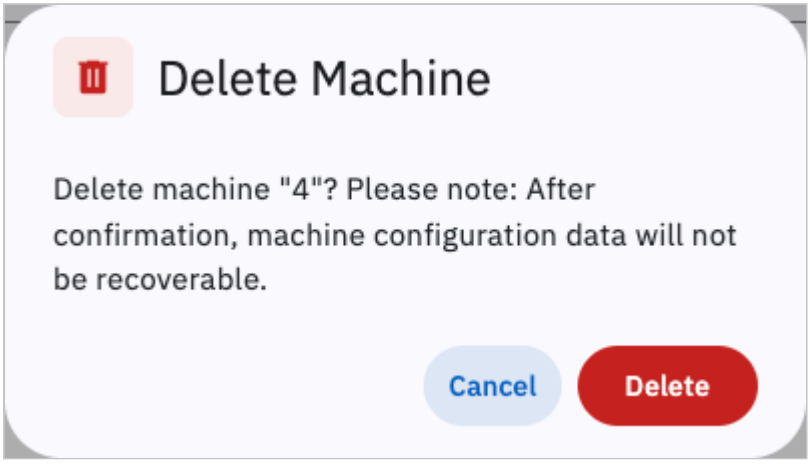
Click the edit button ✏️ in the "Operations" column at the right of a row in the machine list. The "Edit CNC Machine" / "Edit Laser Cutter" / "Edit Robot" / "Edit PLC Machine" dialog opens; it is similar to the corresponding "Create CNC Machine" / "Create Laser Cutter" / "Create Robot" / "Create PLC Machine" dialog. For a description of each tab, see 3.3.1. Adding a Machine. The machine type — CNC, laser cutter, robot or PLC — cannot be changed in the edit dialog.


Edit CNC Machine — OP02
General
Task
DNC
External PLC
Advanced
IDENTITY
Name
OP02
Activation
CONTROLLER
System*
Mock
Model*
General
CONNECTION
IP Address*
127.0.0.1
Port
0
Encoding
Default
Cancel
Confirm



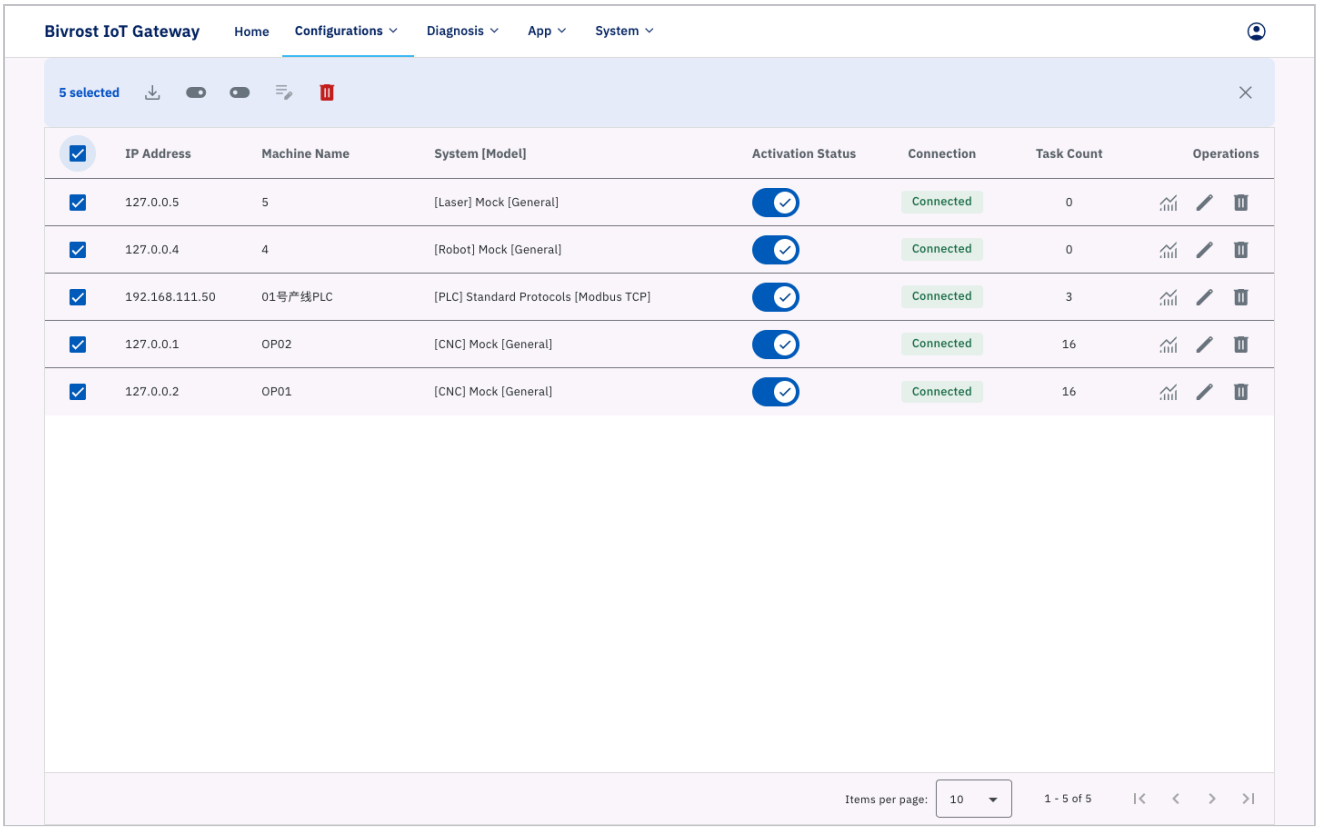
3.3.3. Deleting a Machine

Click the delete button 🗑️ in the "Operations" column at the right of a row in the machine list. The "Delete Machine" dialog appears; click **Confirm** to delete the machine permanently.



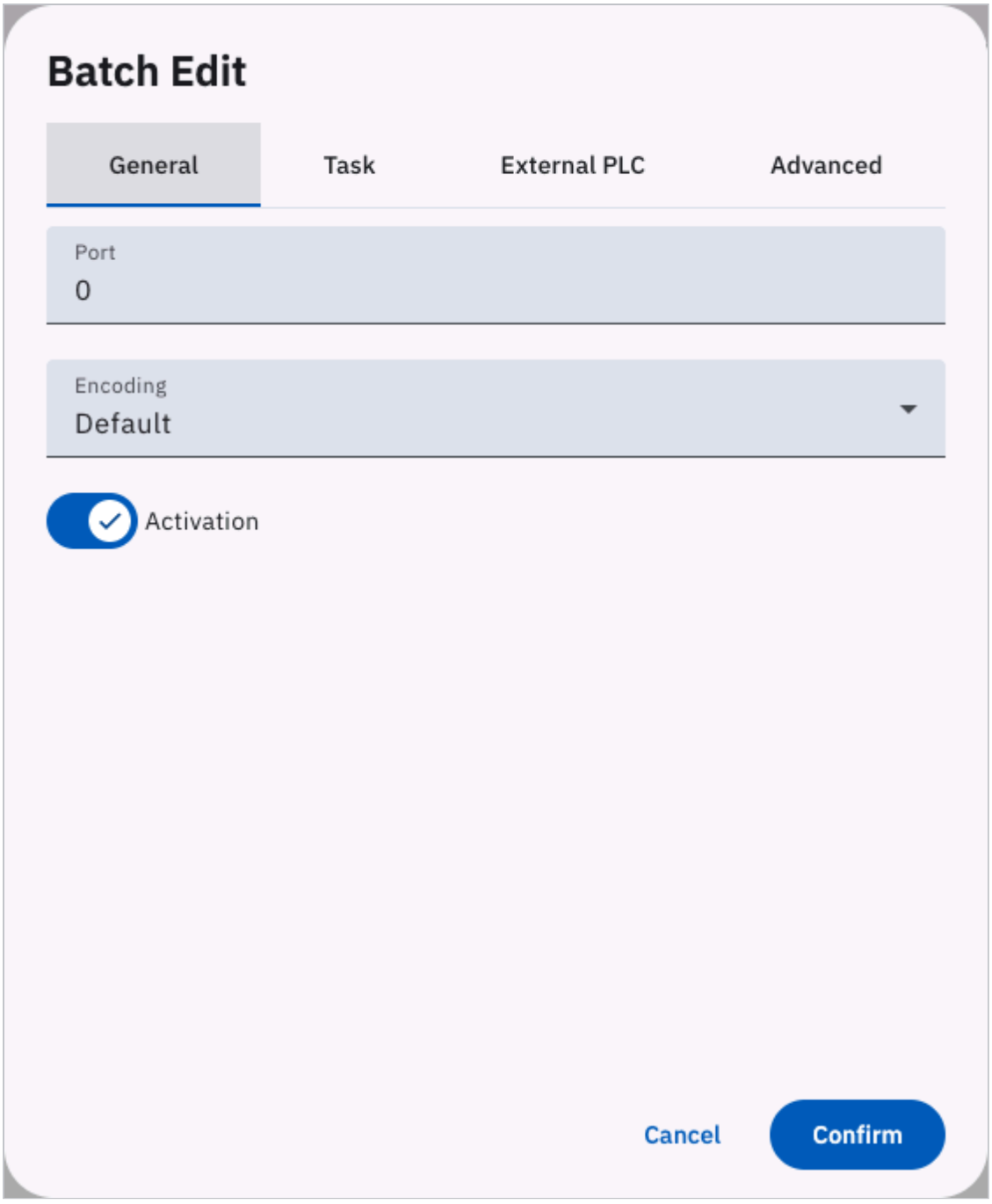
3.3.4. Batch Activation

Select several machines using the checkboxes at the far left of the list, or the select-all button at the top left of the list.



The batch buttons — Batch Activation, Batch Edit and Batch Delete — then appear at the top right.

Click Batch Activation, choose whether to activate, and click Confirm to set the selected machines to activated or unactivated.

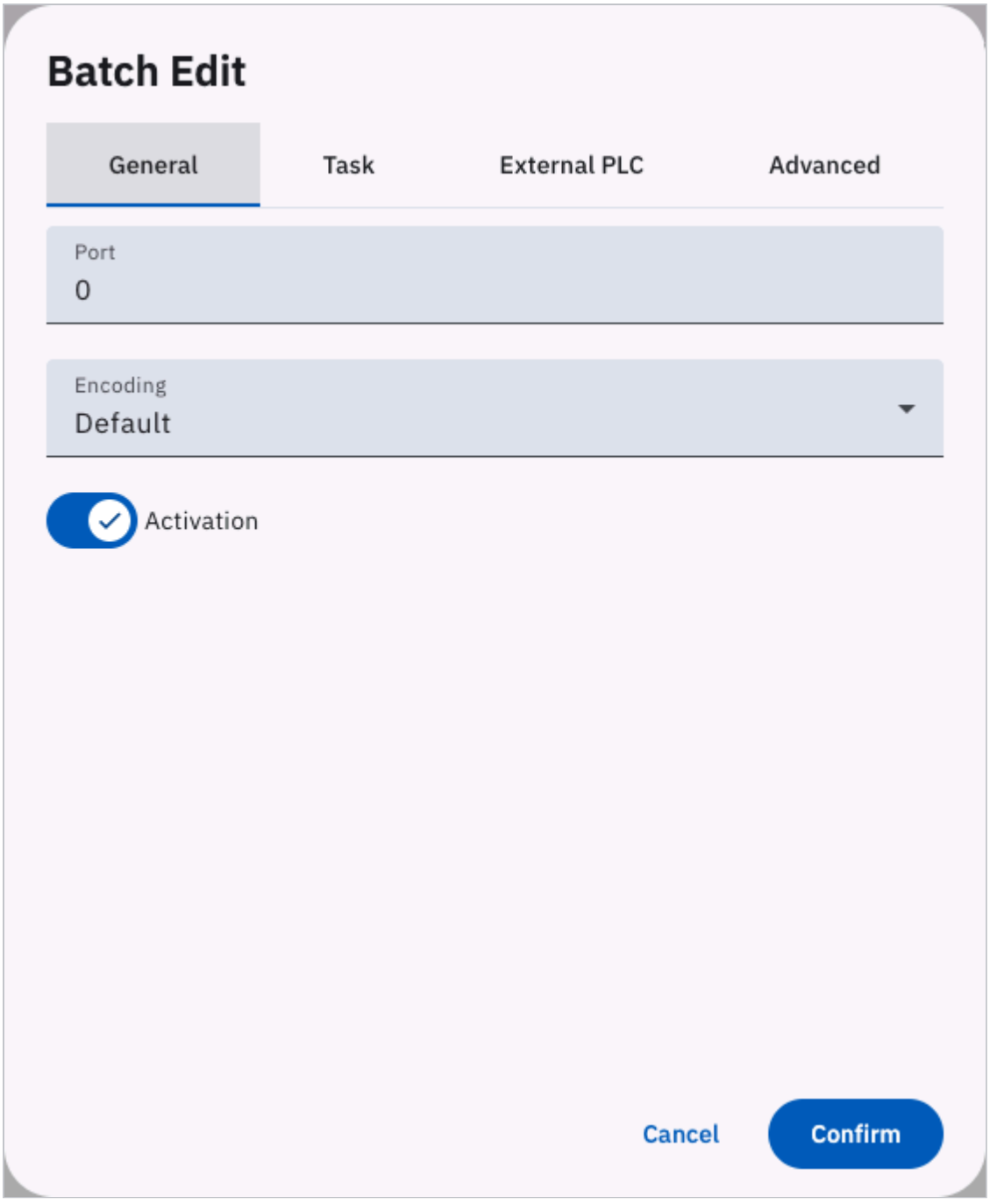


3.3.5. Batch Edit

Select several machines using the checkboxes at the far left of the list, or the select-all button at the top left of the list. Note that they must all have the same system and model.

Click Batch Edit at the top right, set the shared options in the Batch Edit dialog, and click Confirm to save.

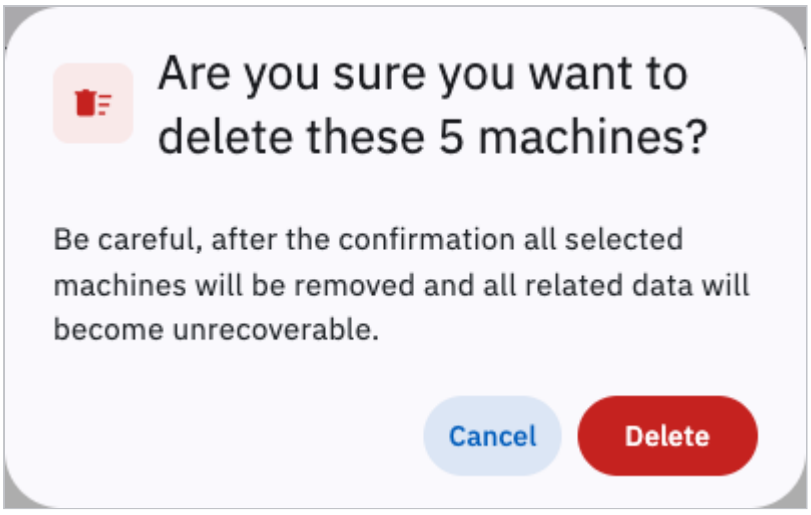
> **Note**
>
> Batch Edit covers shared options only. Options that are specific to a single machine — machine ID, slave ID, machine name, DNC directory, custom attributes and so on — must be edited machine by machine. On every tab of the Batch Edit dialog, options you do not set are initialised to their default values rather than keeping each machine’s individual settings.



3.3.6. Batch Delete

Select several machines using the checkboxes at the far left of the list, or the select-all button at the top left of the list.

Click Batch Delete at the top right and click Confirm in the confirmation dialog to delete the selected machines.

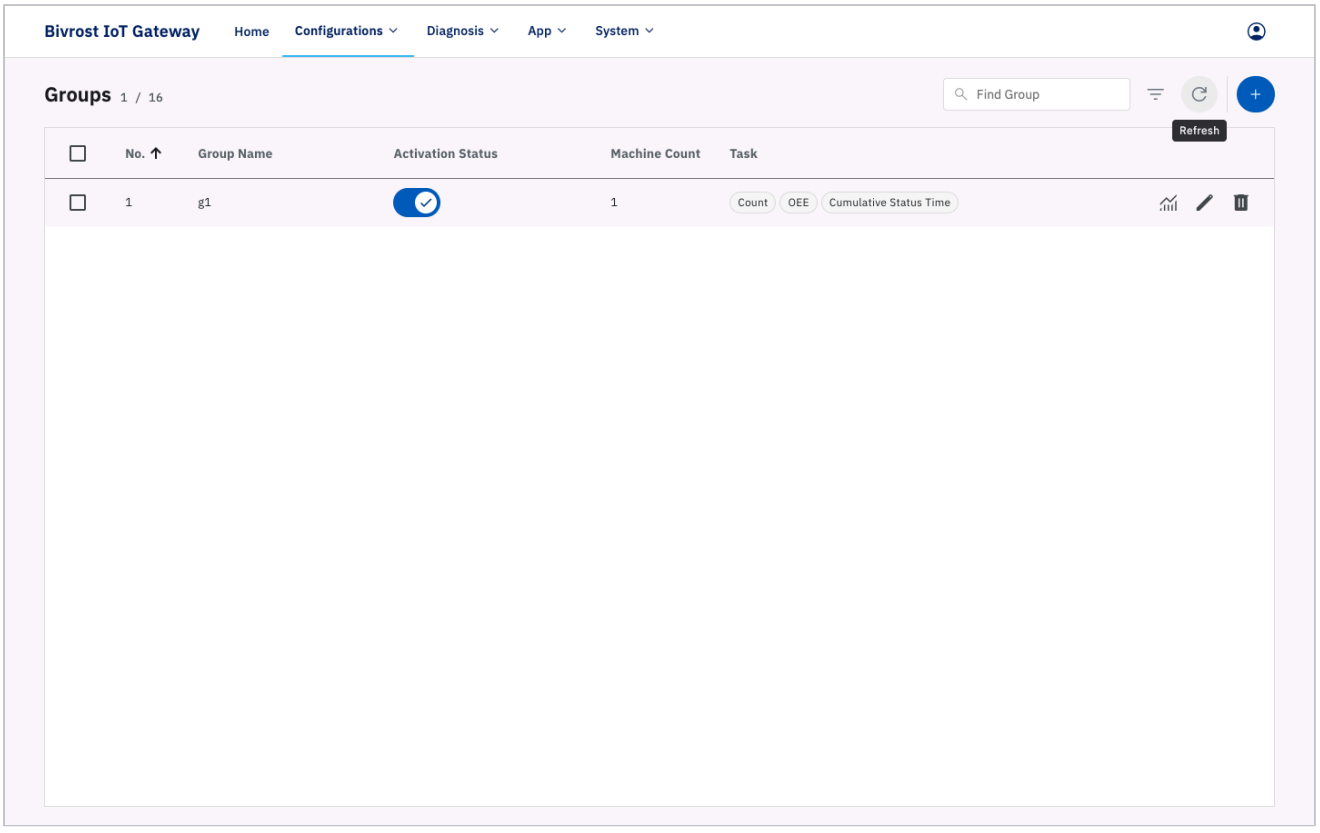


3.4. Groups

On the **Groups** page you can combine the machines defined on the **Machines** page into groups. The gateway then uses the data collected from each machine to calculate group-level statistics and uploads them as a package.

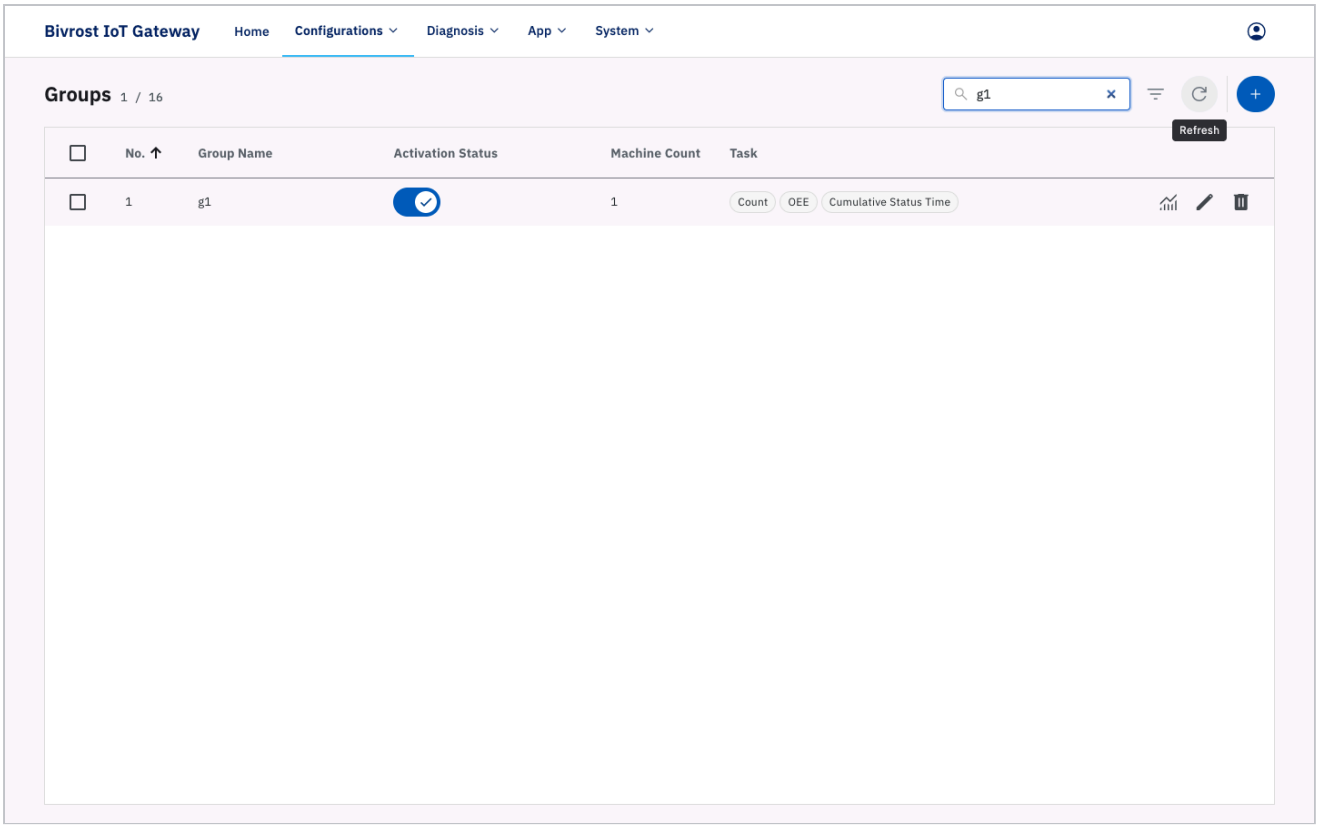
After changing anything on the **Groups** page, return to the home page and click **Restart Service** for the changes to take effect.

> **Note**
>
> Unlike machine activation, activating a group does not consume a license. An activated group is enabled, and its data is uploaded through the cloud platform (see 3.6.1. Cloud Settings for how to set this up), MODBUS, MQTT and a database (see 6. MODBUS Communication, 7. MQTT Communication and 8. Database Communication). A deactivated group stops collecting statistics for that group and stops uploading them. If a group contains a machine that is not activated (see 3.3. Machines), no data is collected or included in the statistics for that machine.



There are three buttons in the top-right corner of the page: Find Group, Refresh and Create Group.

Click the 🔍 Find Group button and type a keyword into the input field to its left; the matching results appear below. The search covers the group number and the group name. Click the close button on the right of the search field to show all groups again.

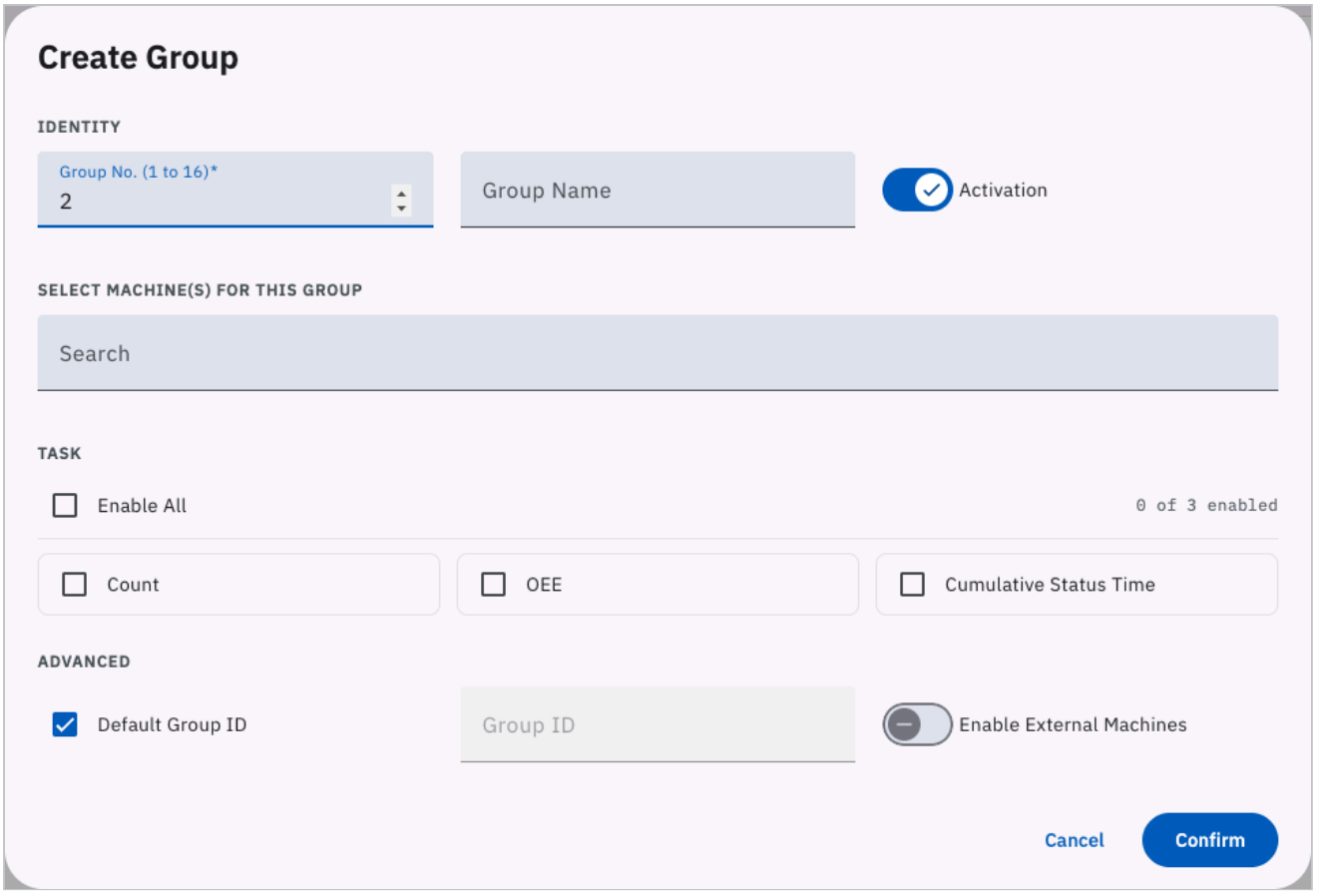


Click the ↻ Refresh button to refresh the whole group list.

Click the + Create Group button to add a group, as described in 3.4.1. Adding a Group.

3.4.1. Adding a Group

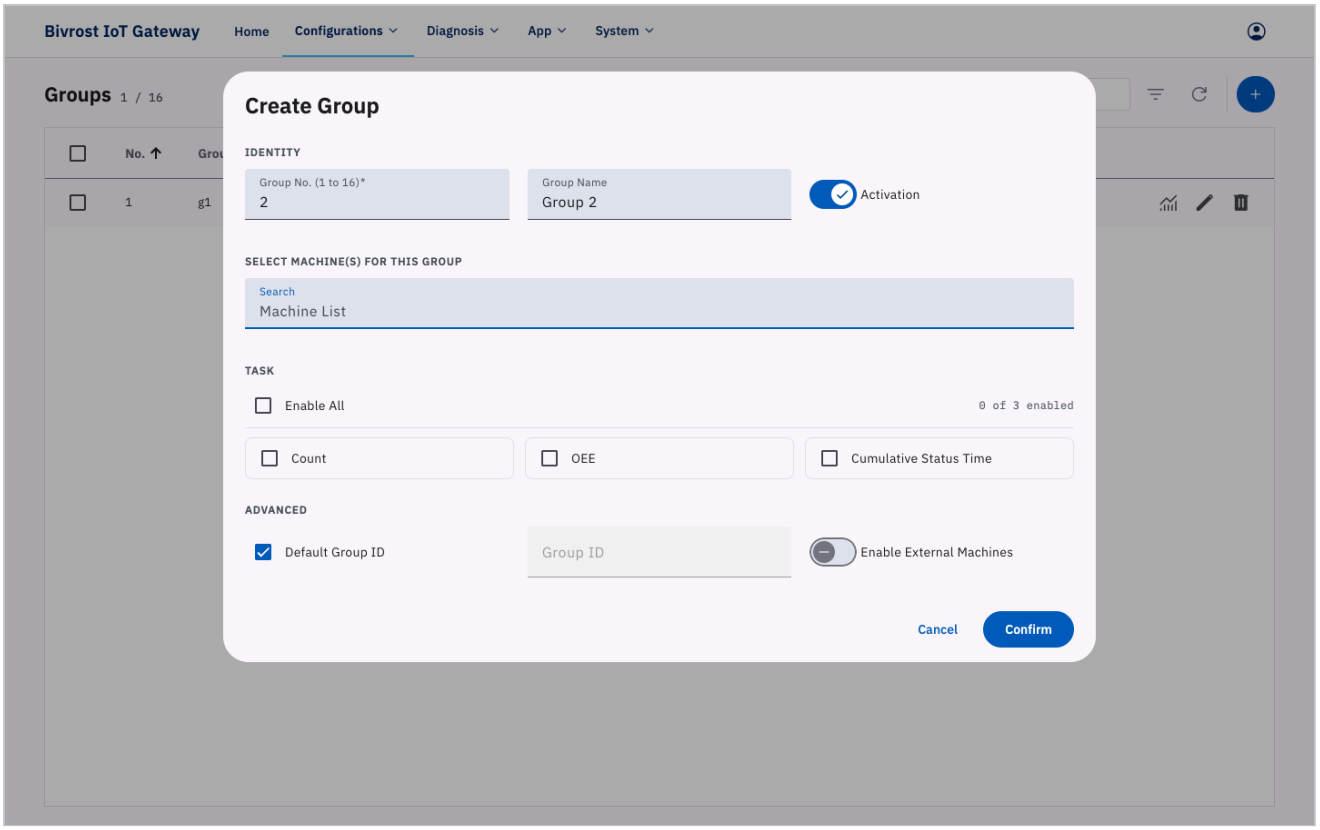
Click the + Create Group button in the top-right corner. The "Create Group" dialog opens, with four sections from top to bottom: "Identity", "Select Machine(s) for This Group", "Task" and "Advanced".



3.4.1.1. Group General Settings

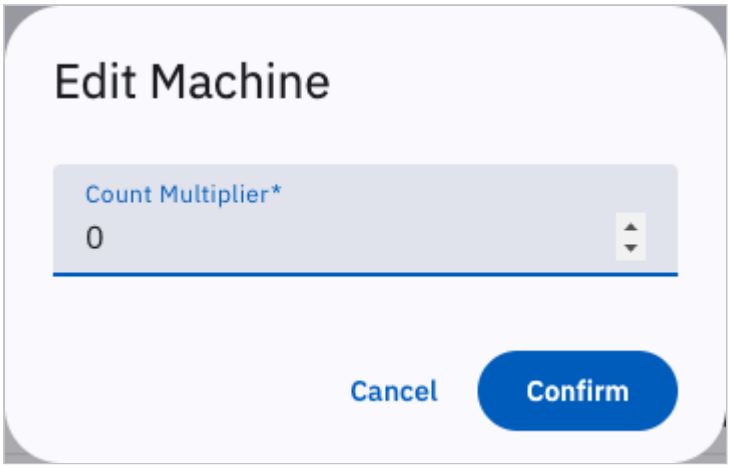
In the "Identity" section, enter the group number and the group name. The group number must currently be an integer between 1 and 16 inclusive, and must not duplicate an existing group number. You can also activate or deactivate the group: statistics are uploaded only while the group is activated. Deactivating it disables the group, and its statistics are no longer uploaded. If the group contains a machine that is not activated (for machine activation, see 3.3. Machines), that machine is not added to the collection list and is neither polled nor included in the statistics.

Click the "Search" box under "Select Machine(s) for This Group" and pick the machines for the group from the machine list. The selected machines are shown as tags, and the number of machines in the group appears in the "Machine Count" column on the **Groups** page.



Each selected machine tag shows the machine IP, the machine name and the count multiplier (for example ×0). Click the × on the right of a tag to remove that machine.

Adjusting the count multipliers of the machines in a group lets you define how the group’s count is calculated. Click a selected machine tag to edit its count multiplier.



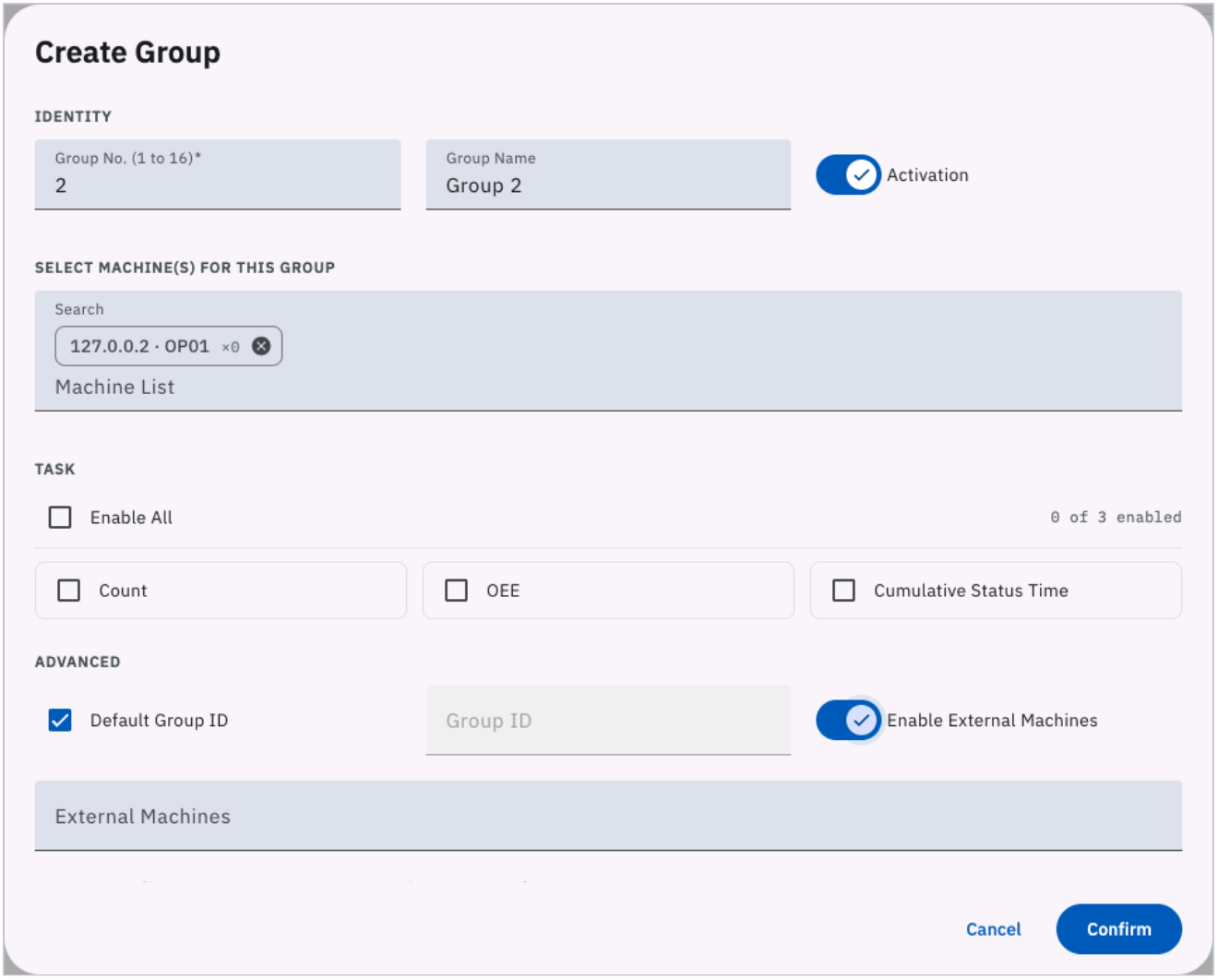
For example, if machine “OP01” and machine “OP02” form a production line and handle two consecutive operations, the output of the line should be taken from the last operation, “OP02”. Set the count multiplier of “OP01” to 0 and that of “OP02” to 1, giving:

```
Group Count = "OP01" Count * 0 + "OP02" Count * 1
```

If the last operation is handled by several machines, set the count multiplier of all of those machines to 1.

If a machine produces several parts per cycle, set its count multiplier to the corresponding number.

By enabling external machines you can add to the group machines that belong to the gateways linked in Cloud Settings or Hub Settings. This requires a command that identifies the gateway and the machine; see 11.2.6. External Machines.



3.4.1.2. Group Task Settings

In the "Task" section, configure the group's automatic collection tasks. Tick an available interface to turn on automatic statistics collection and upload for the corresponding group task.

The number of enabled tasks appears in the “Task Count” column of the table on the “Groups” page.

Group tasks work in much the same way as machine tasks: their results can be uploaded automatically through the cloud platform, MODBUS, MQTT or a database. To receive group task results, at least one of the cloud platform, MODBUS, MQTT or database communication methods must be enabled and configured in 3.6. Communication. The collection interval and the detailed settings for group tasks are configured in 3.5. Tasks. For the format in which group task results are published over MODBUS, MQTT and a database, see 6. MODBUS Communication, 7. MQTT Communication and 8. Database Communication.

> **Note**
>
> If the group contains a machine that is not activated (for machine activation, see 3.3. Machines), that machine is not on the collection list and is not included in the statistics.

Group OEE data covers all activated machines in the group: total Off time, total Wait time, total Emergency time, total Autorun time, total Manual time, the current number of machines that are Off, Wait, Emergency, Autorun and Manual, and the group availability. See 11.1.2. Group OEE Data.

For the group part count, see 11.1.3. Group Part Count.

Both group OEE and the group part count require local caching, so the “Local Caching” switch under Options in 3.12. Settings must be turned on before you enable them.

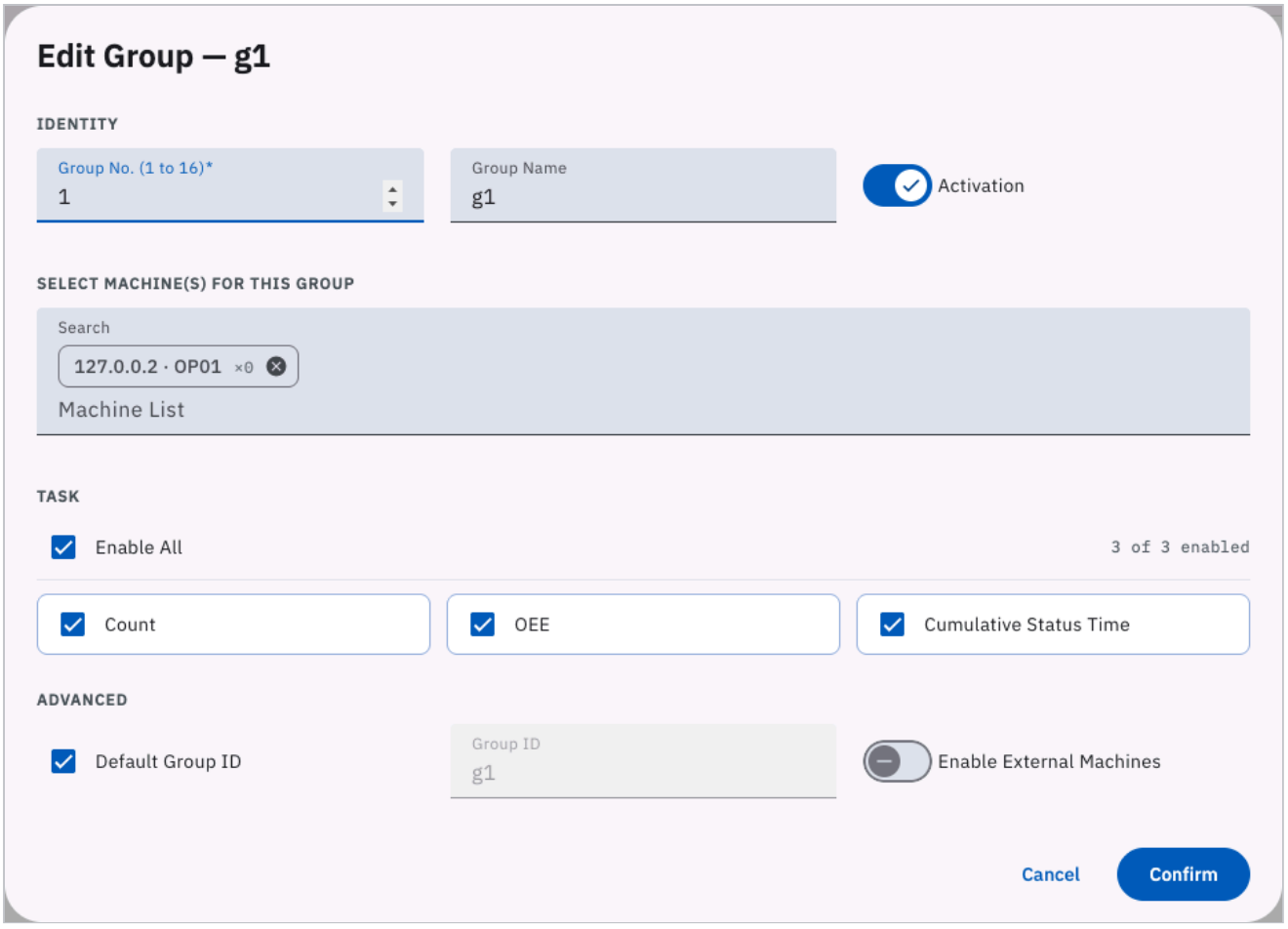
Group Cumulative Status Time is the total time spent in each status, summed over all machines in the group, since the gateway started collecting from those machines, in seconds.

3.4.1.3. Advanced Settings

In the “Advanced” section you can define a custom group ID (GroupID). The group ID identifies the group’s data for every communication method except MODBUS. The default group ID is the letter g followed by the group number, for example “g1”.

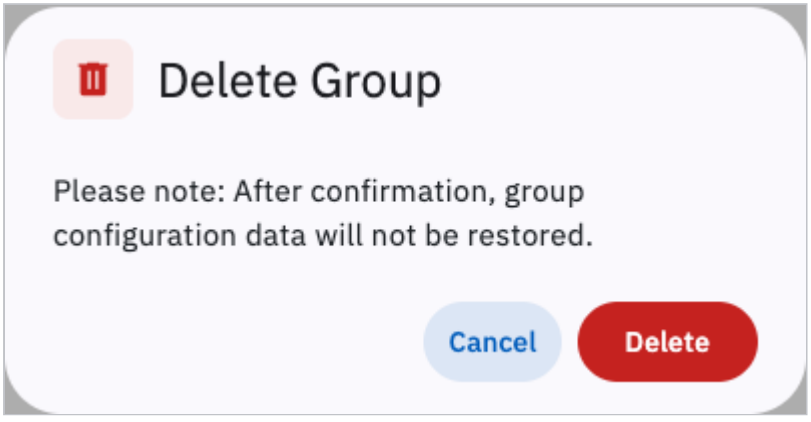
3.4.2. Editing a Group

Click the **Edit** button in the “Operations” column on the right of an existing group row to open the “Edit Group” dialog, which is similar to “Create Group”. Here you can change the group number, group name and activation state, select the machines in the group, adjust their count multipliers, configure group tasks, change the group ID, and so on.



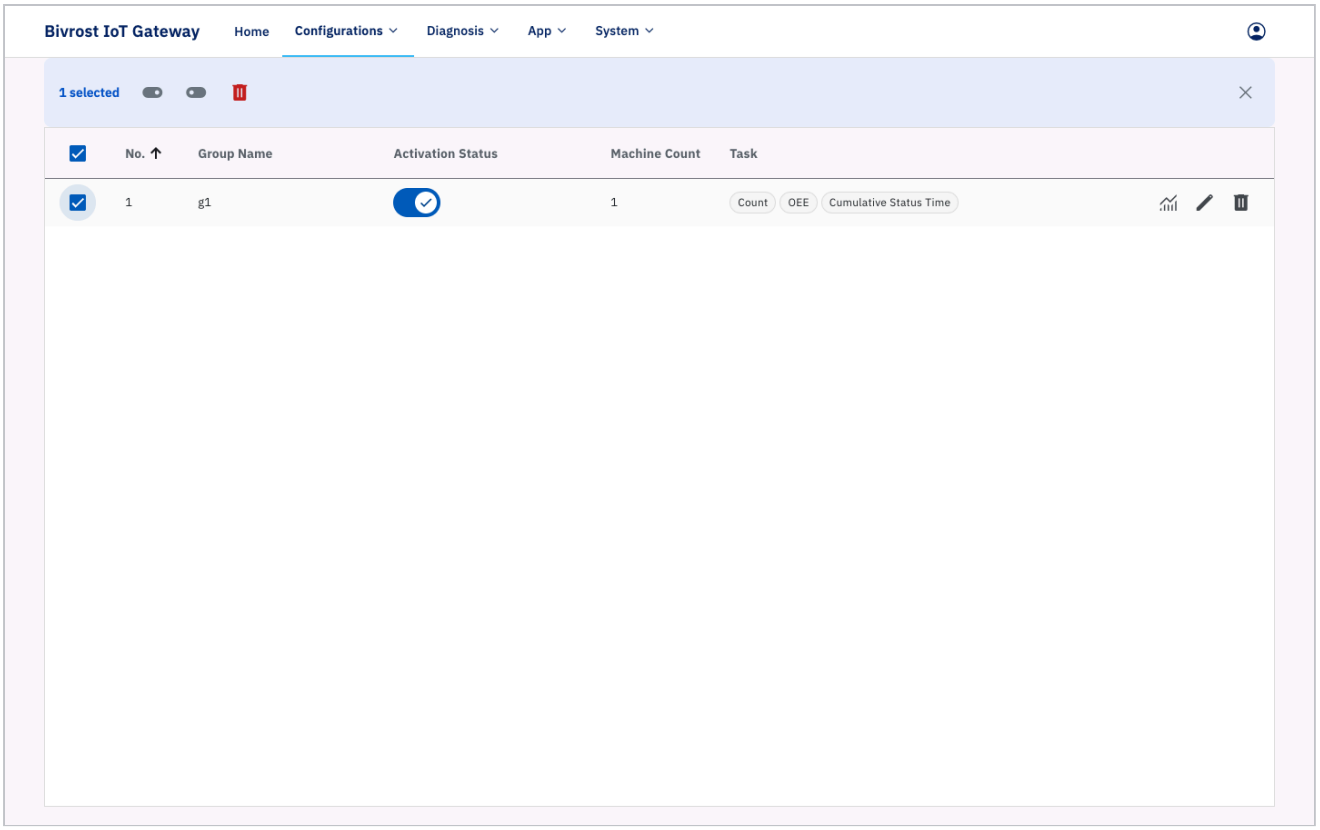
3.4.3. Deleting a Group

Click the **Delete** button in the "Operations" column on the right of an existing group row. The "Delete Group" dialog appears; click **Confirm** to delete the group.



3.4.4. Batch Activation

Select several groups using the checkboxes at the far left of the list, or the select-all button at the top left of the list.



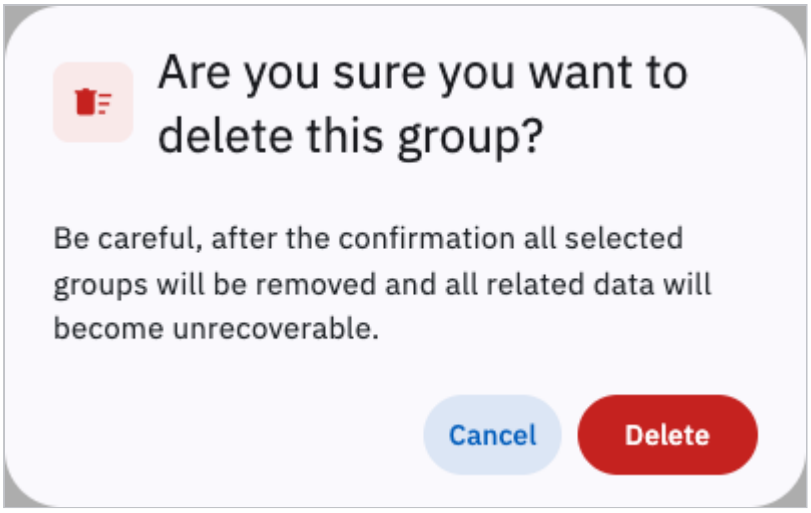
A batch action bar then appears at the top of the page with the **Activate**, **Deactivate**, **Batch Delete** and **Clear Selection** buttons.

Click **Activate** or **Deactivate** to activate or deactivate all the selected groups at once.

3.4.5. Batch Delete

Select several groups using the checkboxes at the far left of the list, or the select-all button at the top left of the list.

Click **Batch Delete** on the batch action bar, then click Confirm in the dialog that appears to delete the selected groups.

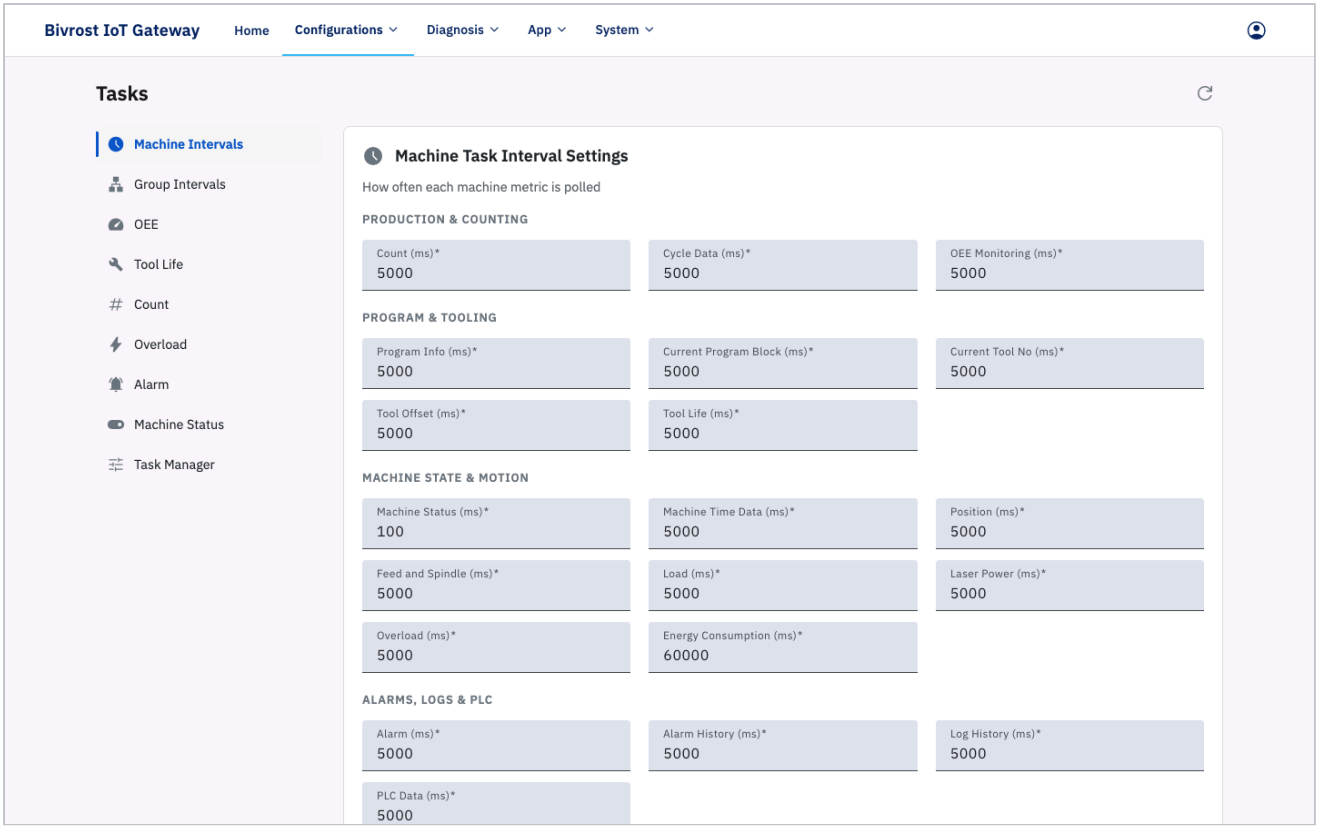

Are you sure you want to delete this group?
Be careful, after the confirmation all selected groups will be removed and all related data will become unrecoverable.
Cancel
Delete

3.5. Tasks

Tasks includes Machine Task Interval Settings, Group Task Interval Settings, OEE Monitoring Settings, Tool Life Monitoring Settings, Count Monitoring Settings, Overload Monitoring Settings, Alarm Monitoring Settings, Machine Status Monitoring Settings and Task Manager Settings.

On the **Tasks** page, click a tab to switch between the setting groups. When you are done, click **Save** below the settings area, then go back to Home and click **Restart Service** to apply the changes.

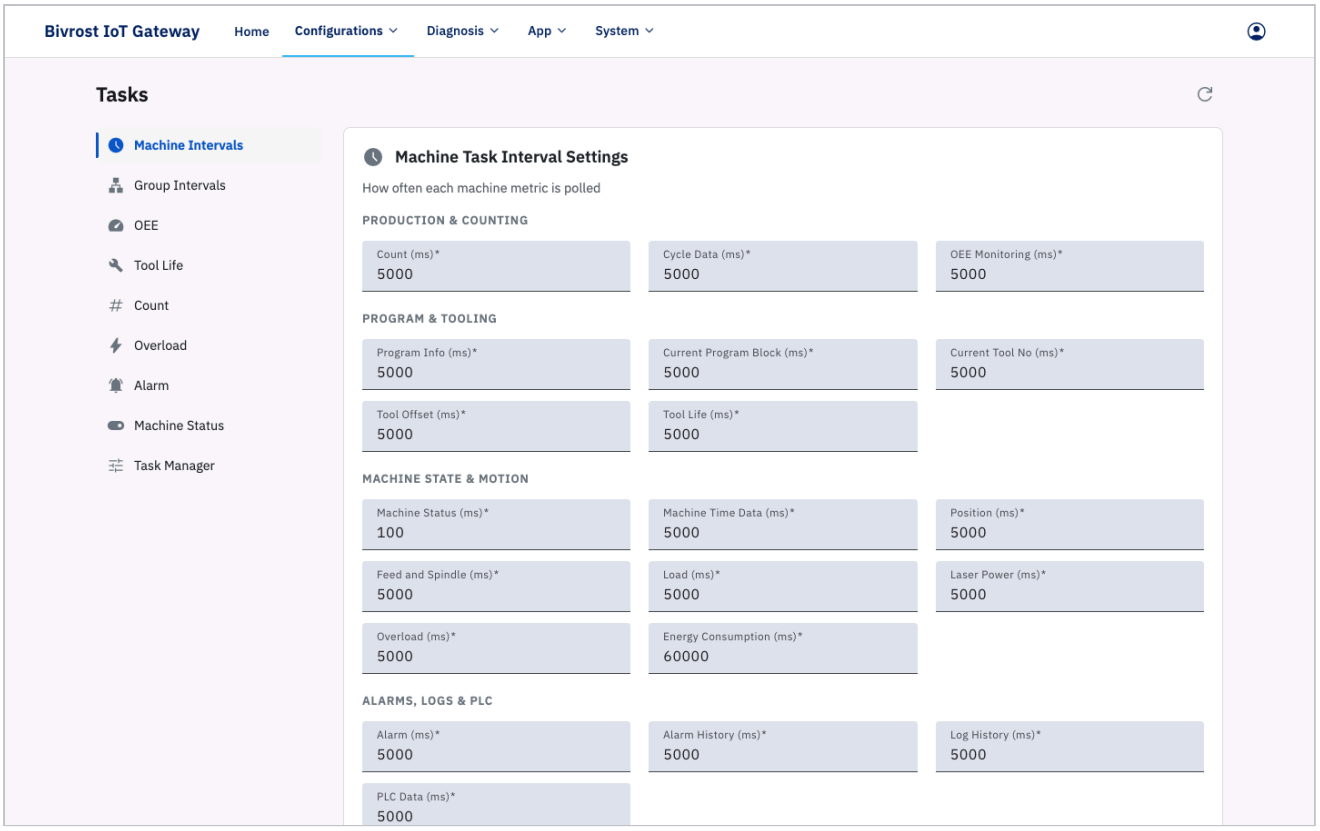
If you are not using automatic data collection tasks, you do not need to configure **Tasks**.



3.5.1. Machine Task Interval Settings

"Machine Task Interval Settings" sets the automatic collection interval for each of the machine tasks described in 3.3.1.2. Task Settings. Intervals are in milliseconds and all default to 5000 ms.

> **Note**
>
> An interval of 5000 ms or longer is recommended. Very short intervals can cause unpredictable faults on machines with limited processing power.



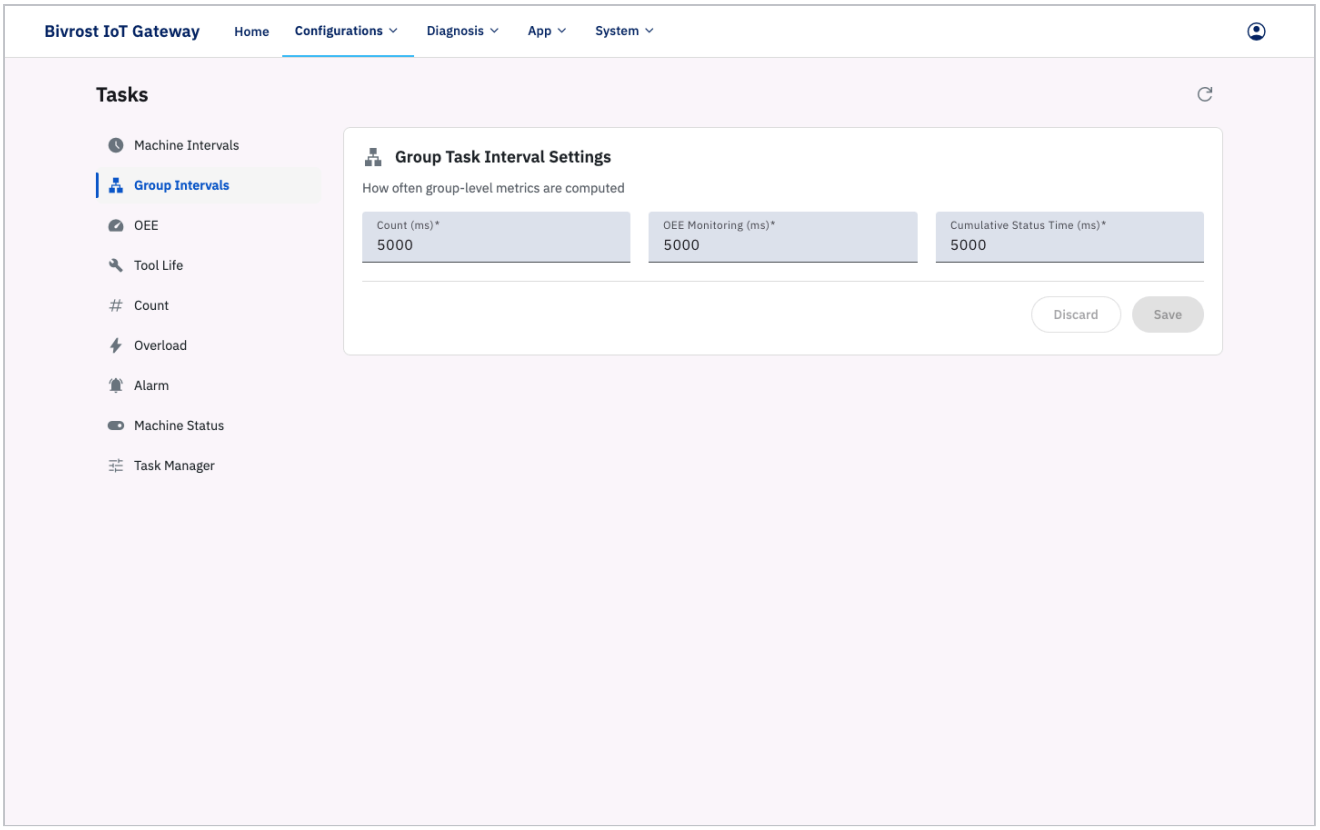
"Overload (ms)" in the screenshot above defaults to 100 ms. Overload events are very short-lived, so a short monitoring interval is needed to capture the data accurately.

"OEE Monitoring (ms)" in the screenshot above is the automatic collection interval for machine OEE data; see 11.1.1. Machine OEE Data.

When you are done, click the **Save** button at the bottom of the page, then click **Restart Service** on Home to apply the changes.

3.5.2. Group Task Interval Settings

"Group Task Interval Settings" sets the automatic collection interval for each of the group tasks described in 3.4.1.2. Group Task Settings. Intervals are in milliseconds and all default to 5000 ms.

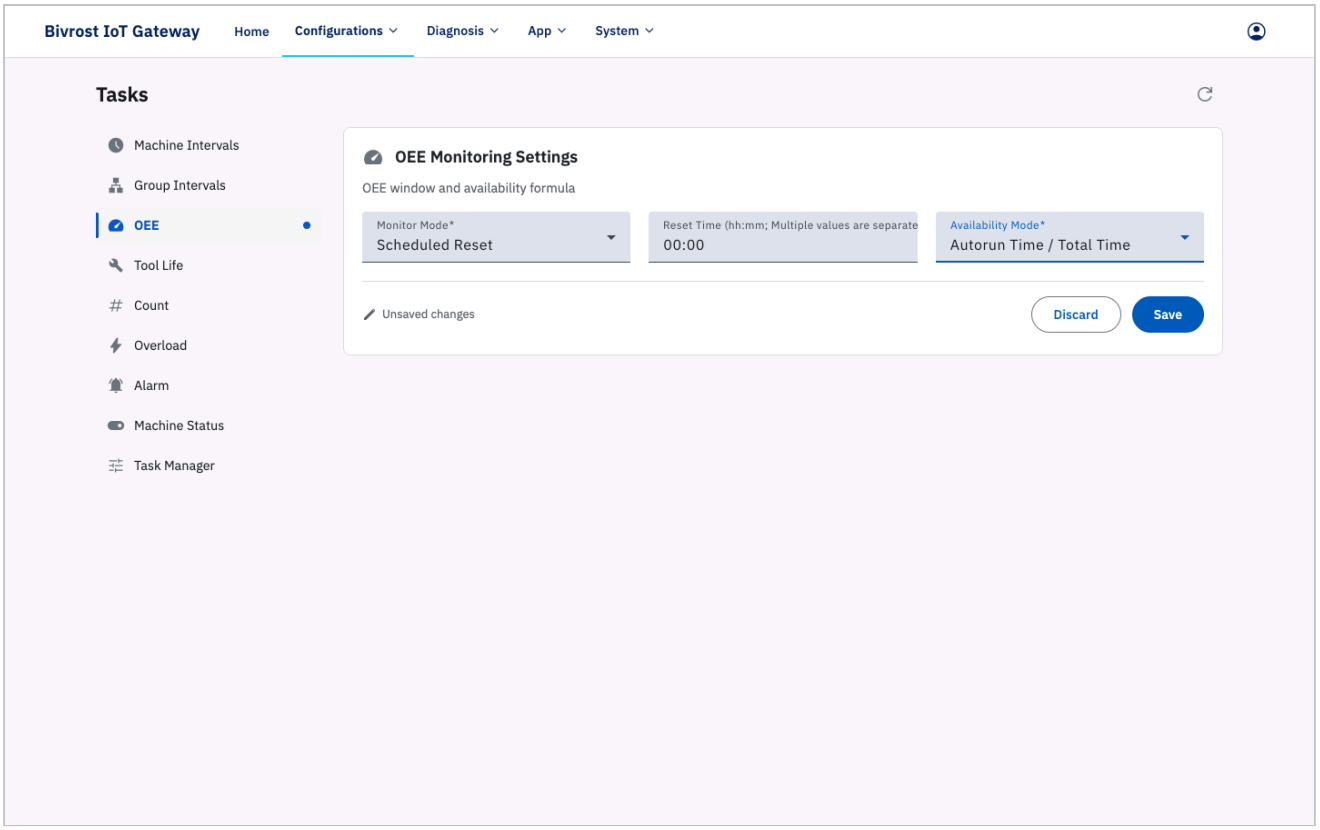


"Count (ms)" in the screenshot above is the automatic collection interval for the group part count; see 11.1.3. Group Part Count. "OEE Monitoring (ms)" is the automatic collection interval for group OEE data; see 11.1.2. Group OEE Data.

When you are done, click the **Save** button at the bottom of the page, then click **Restart Service** on Home to apply the changes.

3.5.3. OEE Monitoring Settings

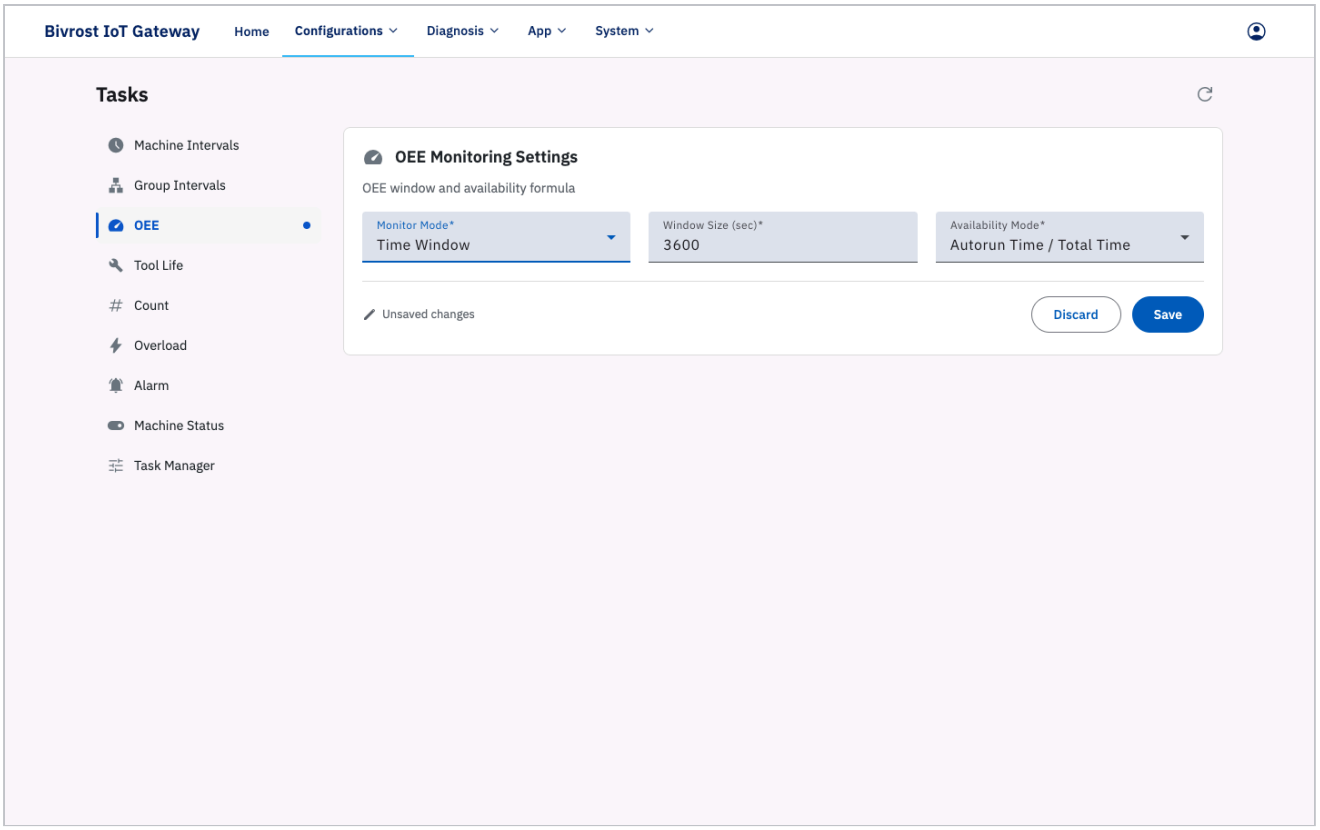
"OEE Monitoring Settings" defines the monitoring period and the availability formula used for machine OEE (see 11.1.1. Machine OEE Data) and group OEE (see 11.1.2. Group OEE Data).



"Monitor Mode" determines how the monitoring period is defined. It offers two options: "Scheduled Reset" and "Time Window".

Scheduled Reset: OEE statistics are calculated over the period that runs from the most recent reset time up to the present moment. "Reset Time" is the time of day at which the daily OEE statistics are reset. You can enter several reset times separated by ";", for example `08:00;12:00;16:00;20:00;`. If no reset time is entered, the default is 00:00, so OEE statistics are reset every day at 00:00.

Time Window: the window size defaults to 3600 seconds, so OEE statistics are calculated over the period that starts 3600 seconds before the present moment and ends at the present moment — that is, the last hour. A window size longer than 86400 seconds (the last day) is not recommended. If this is the first collection and the elapsed monitoring time is shorter than the window size, any time with no machine status data is treated as machine off.



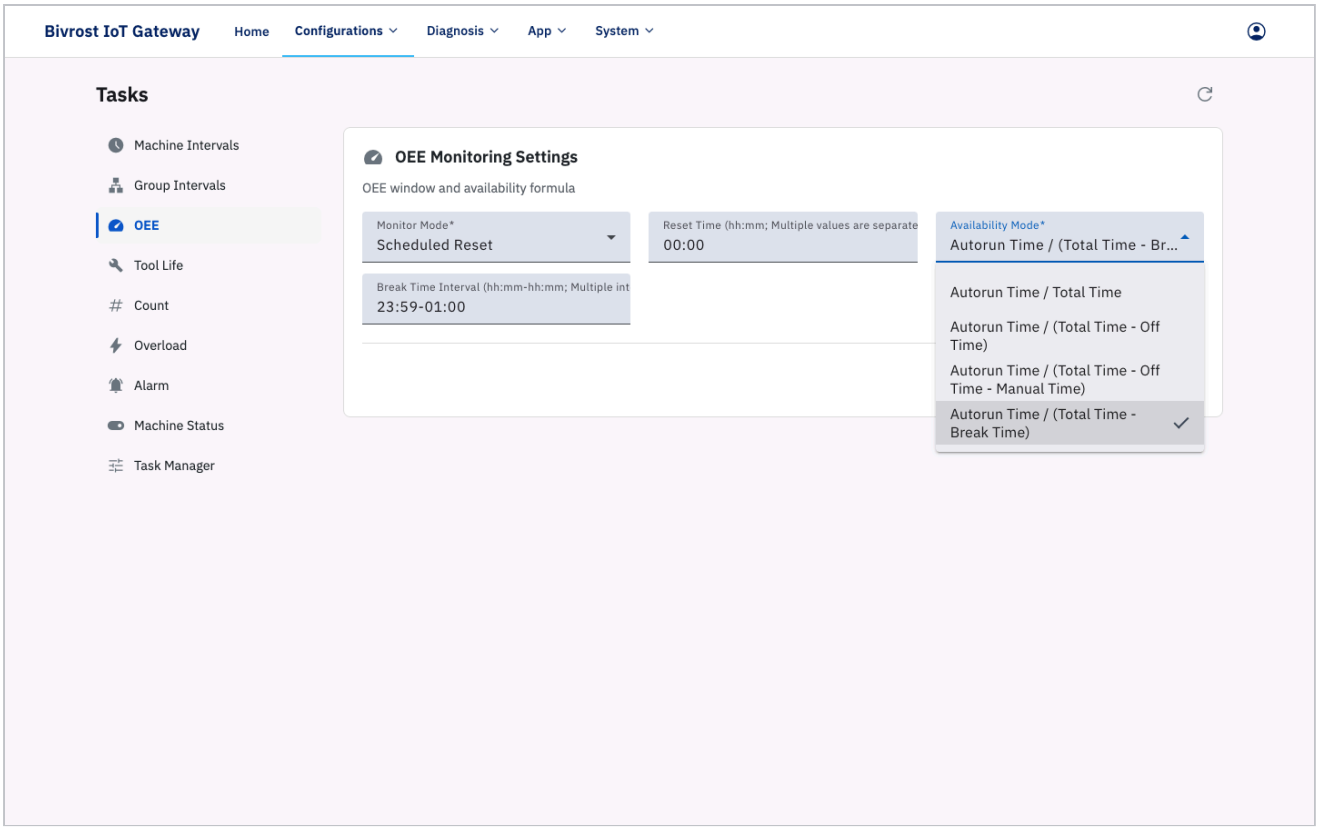
The “Availability Mode” option defaults to “Autorun Time / Total Time”, in which case:

```
Machine Availability = Autorun Time / Total Time
```

Here, autorun time is the machine’s automatic running time within the monitoring period, and total time is the monitoring period itself, as defined by “Monitor Mode”.

```
Group Availability = Autorun Time / Total Time
```

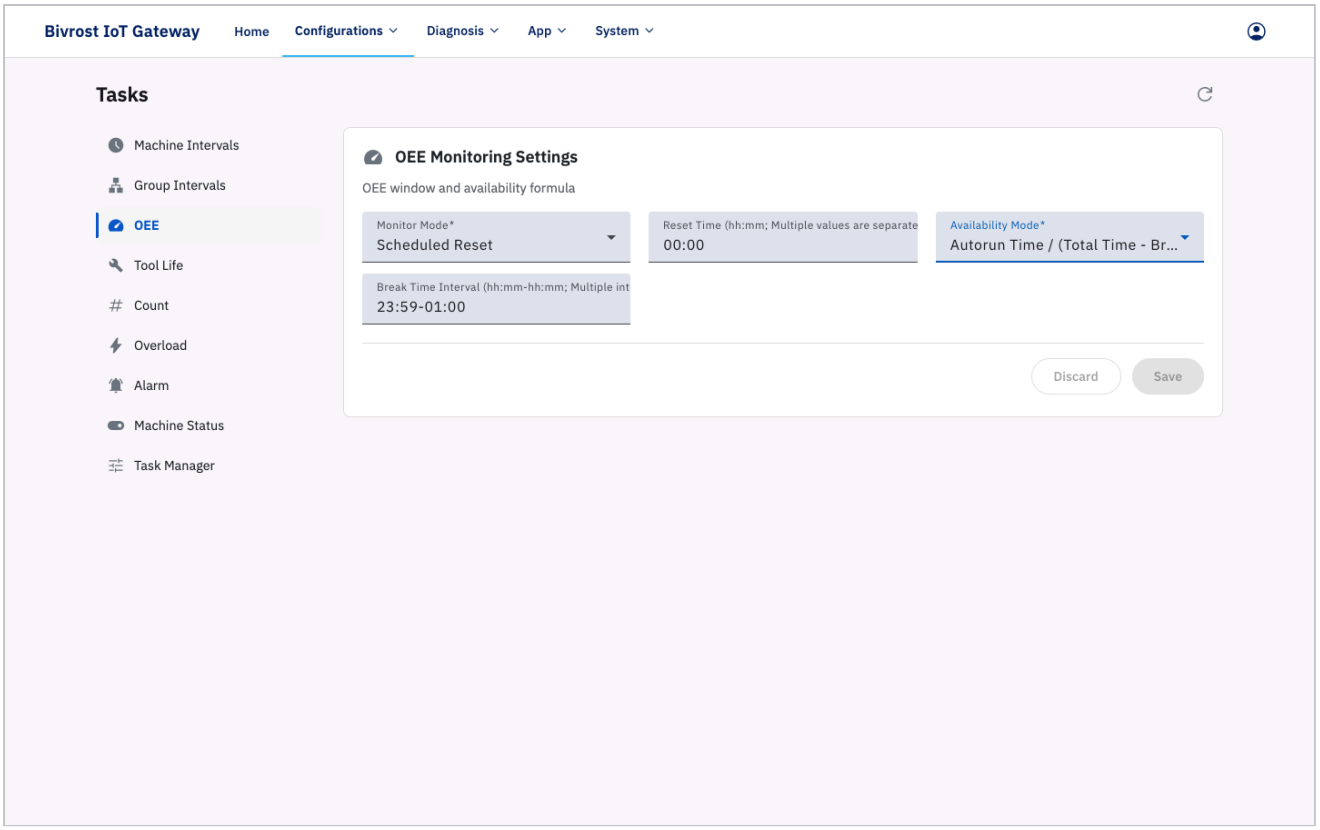
Here, autorun time is the sum of the automatic running time of every active machine in the group within the monitoring period, and total time is the total monitoring time — the monitoring period multiplied by the number of active machines in the group. The monitoring period is defined by “Monitor Mode”.



The “Availability Mode” options are:

- Autorun Time / Total Time
- Autorun Time / (Total Time - Off Time)
- Autorun Time / (Total Time - Off Time - Manual Time)
- Autorun Time / (Total Time - Break Time)

Choose whichever mode suits your needs to exclude off time, manual time or break time from the total time. Off time and manual time are derived from the machine status interface data. When you select “Autorun Time / (Total Time - Break Time)”, you can define the break time intervals, as shown below:

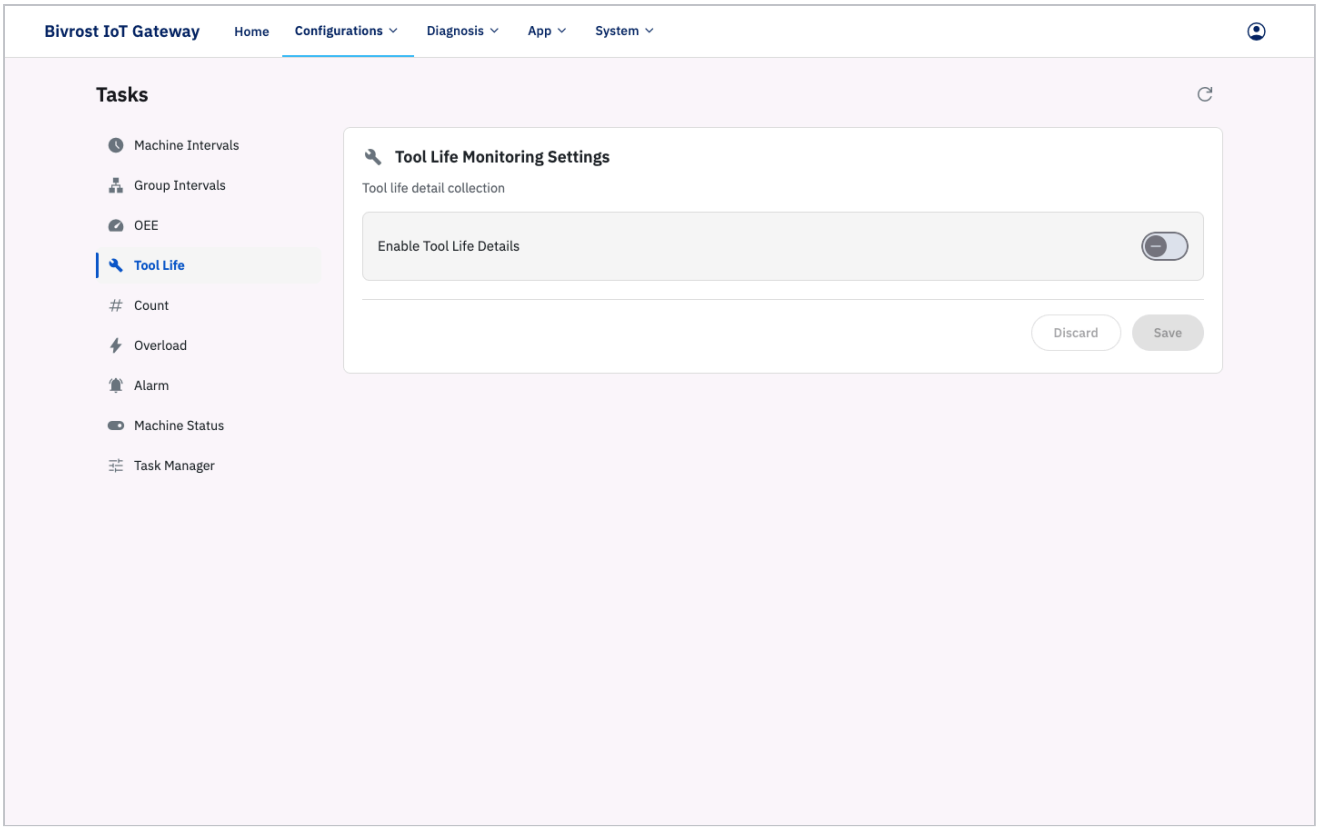


Suppose the break time intervals are set to `07:30-08:00;19:30-20:00;` and the current time is 19:45. With the settings above:

- Total time = 3600 seconds; the monitoring period runs from 18:45 to 19:45 and overlaps part of a break interval.
- Break time = 19:45 - 19:30 = 15 minutes = 900 seconds.

3.5.4. Tool Life Monitoring Settings

"Tool Life Monitoring Settings" configures the tool life data. The "Enable Tool Life Details" option is off by default, in which case the tool life interface returns only normalized data — values converted from the raw data and calculated using a single common definition and unit. With the option enabled, the tool life interface returns both the normalized data and the raw data. For a description of the normalized and raw tool life data, see 4.2.25. ToolLife: Tool Life Data.

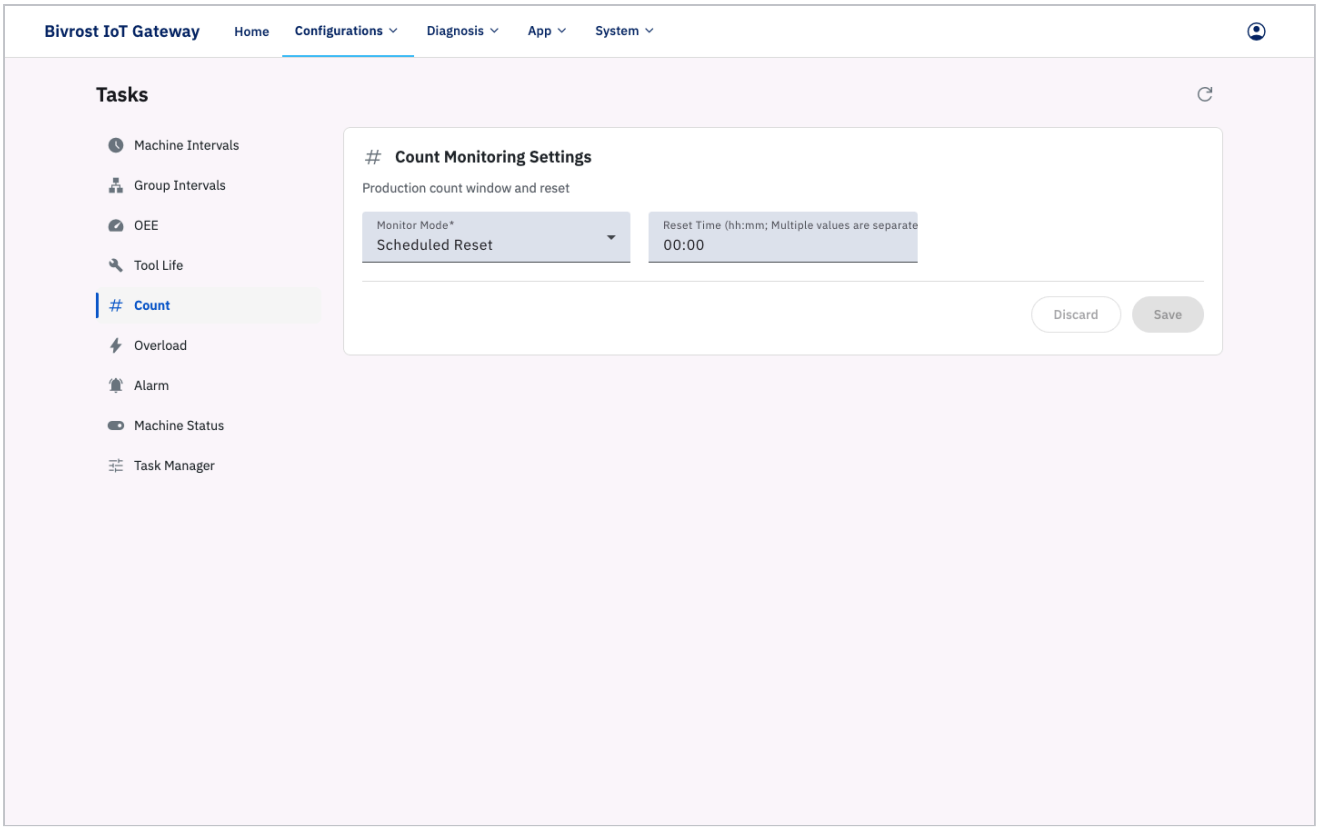


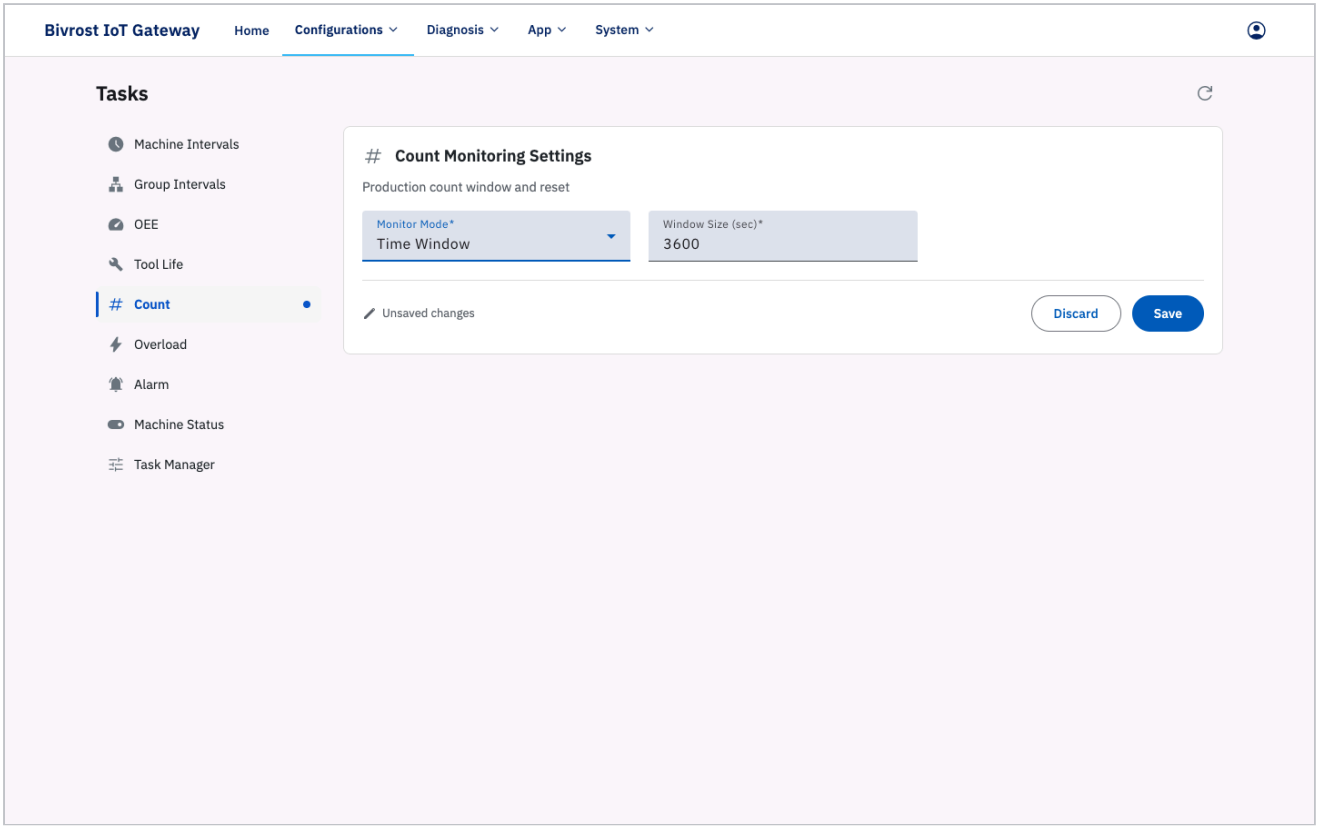
3.5.5. Count Monitoring Settings

“Count Monitoring Settings” defines the monitoring period for the current part count.

The current part count is the machine’s cumulative part count within the monitoring period.

“Monitor Mode” offers “Scheduled Reset” and “Time Window”; the default is “Scheduled Reset” with a default reset time of 00:00. For a description of the two modes, see 3.5.3. OEE Monitoring Settings.

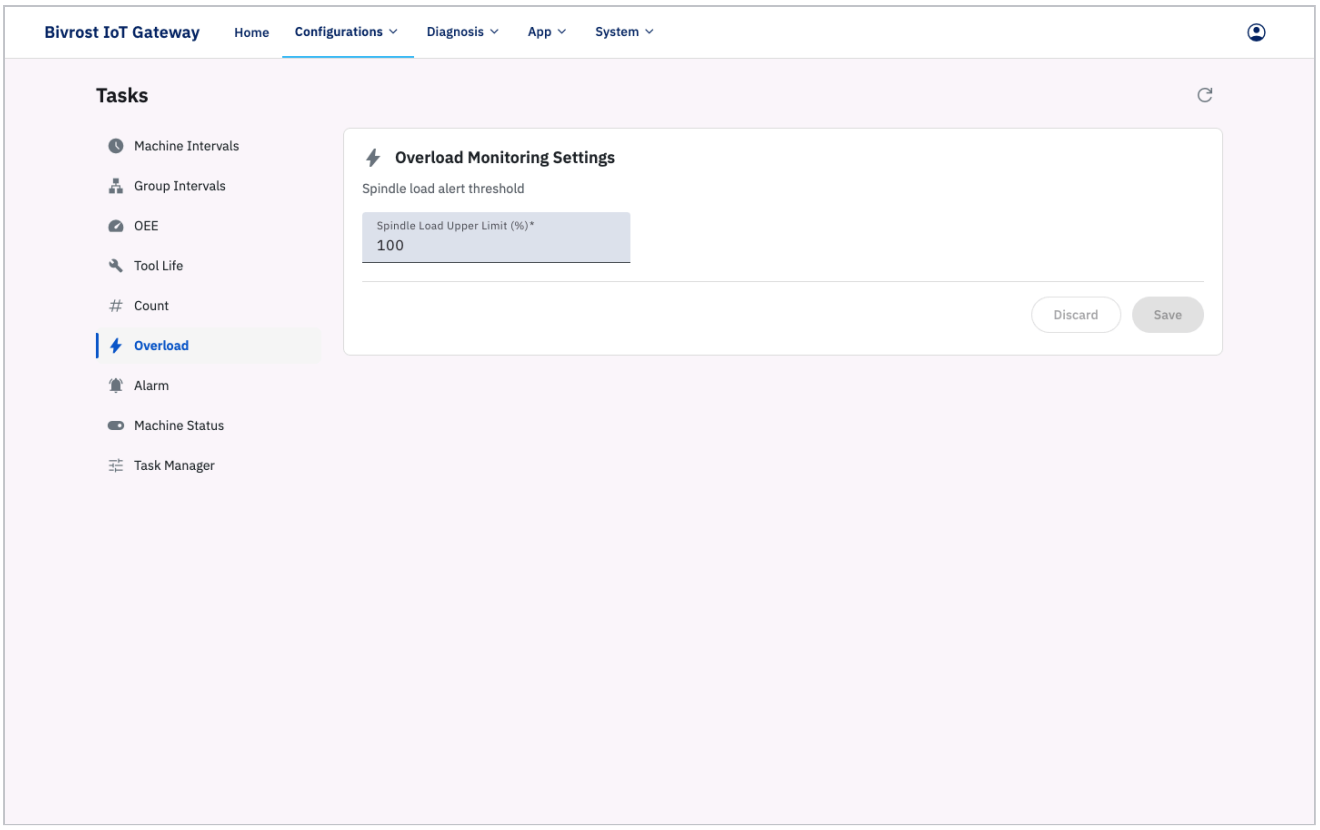




3.5.6. Overload Monitoring Settings

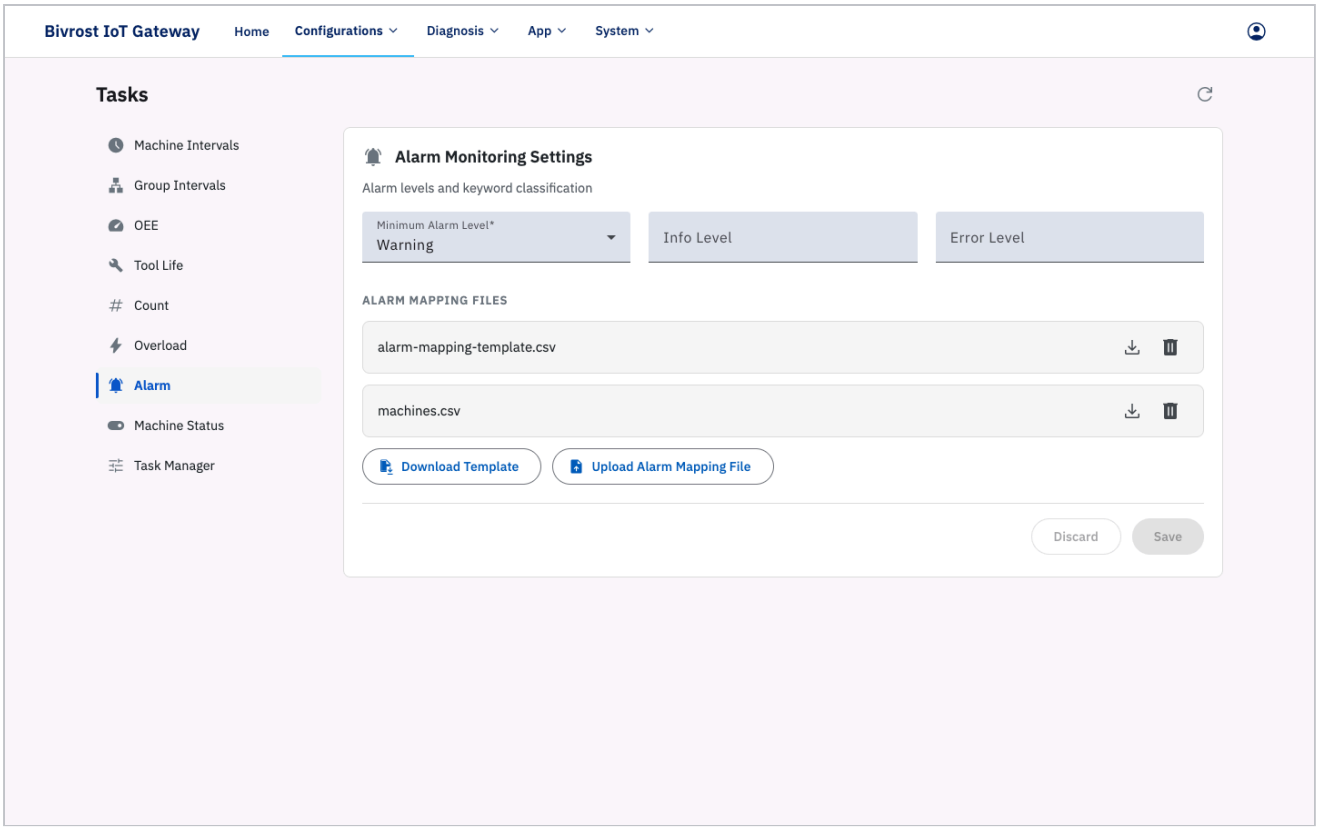
“Overload Monitoring Settings” sets the spindle load upper limit used by the overload task. The default is 100%.

Overload monitoring records the total time the spindle load stays above the limit.



3.5.7. Alarm Monitoring Settings

“Alarm Monitoring Settings” classifies the alarm messages returned by the alarm task into alarm levels and sets the minimum alarm level at which a machine is considered to be in an active alarm state.



Three alarm levels are currently supported, in descending order of priority: Error (ERR), Warning (WRN) and Information (INF). If the minimum alarm level is set to Warning, the gateway automatically ignores information-level alarms and records only error- and warning-level alarms, and uses those two levels to decide whether the machine is in an active alarm state.

In the "Info Level" and "Error Level" input boxes you can enter alarm keywords, separating multiple entries with ";". Any alarm containing one of those keywords is assigned to the corresponding level.

If no error-level keyword is found in an alarm, the alarm defaults to warning level.

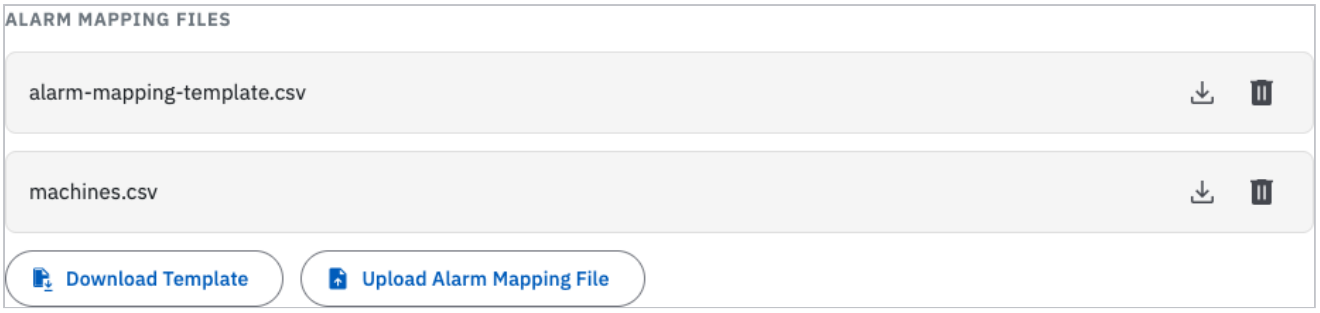
By default, the minimum alarm level for treating a machine as being in an active alarm state is warning level.

3.5.7.1. Alarm Mapping Files

Alarm Mapping maps the alarm numbers reported by a machine to your own alarm messages. It is useful for machines that report only an alarm number and no alarm text, or when alarm descriptions need to be standardized.

Use the "Alarm Mapping Files" area below "Alarm Monitoring Settings" to upload and manage mapping files:

- Click **Upload Alarm Mapping File** and select a local CSV file. Only non-empty CSV files are supported. The uploaded file then appears in the list.
- Each file in the list has a download and a delete button on the right.
- Click **Download Template** to get a sample CSV template (`alarm-mapping-template.csv`).
- When no file has been uploaded, the area shows "No alarm mapping files uploaded."



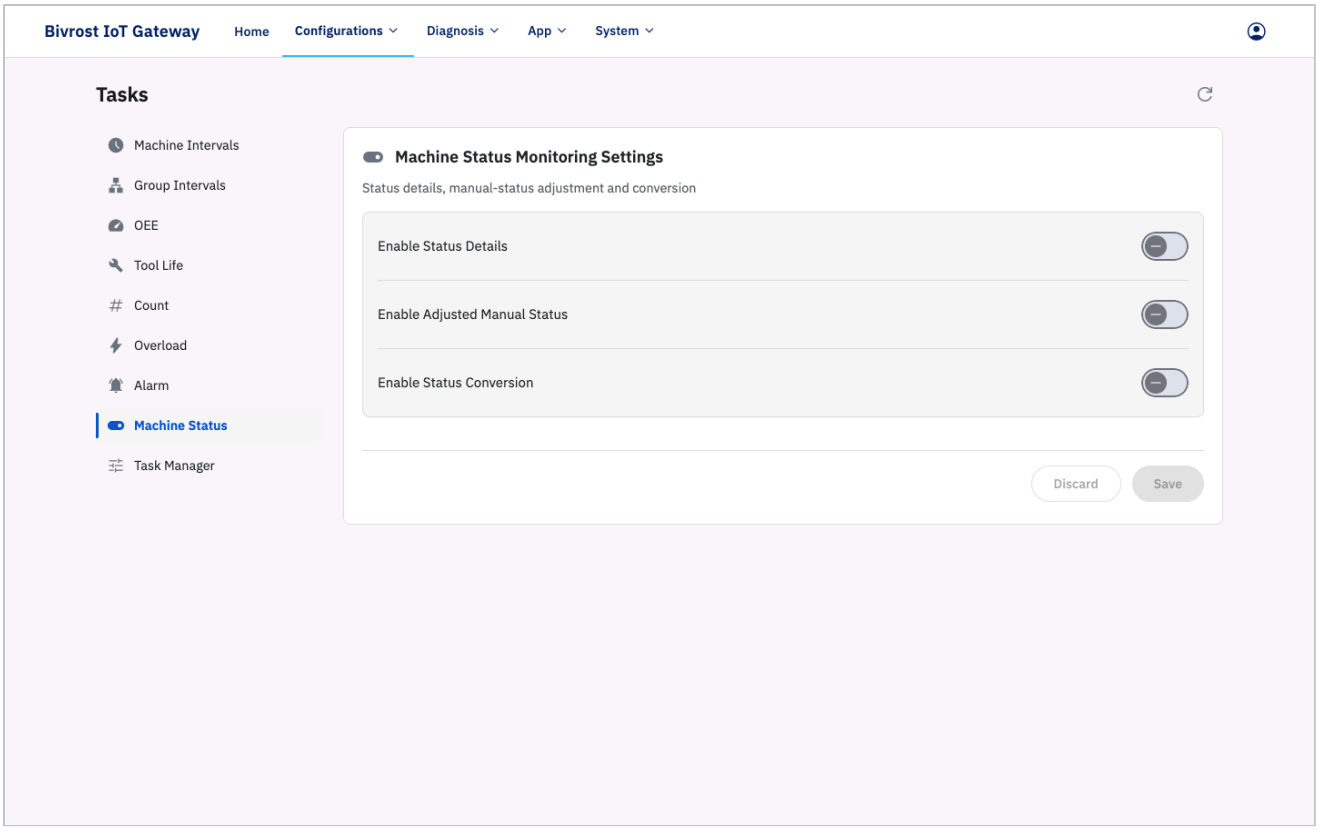
An alarm mapping file is a CSV file with two columns, headed "AlarmNo" and "Content", for example:

AlarmNo	Content
1000	Emergency stop pressed
2001	Spindle overload

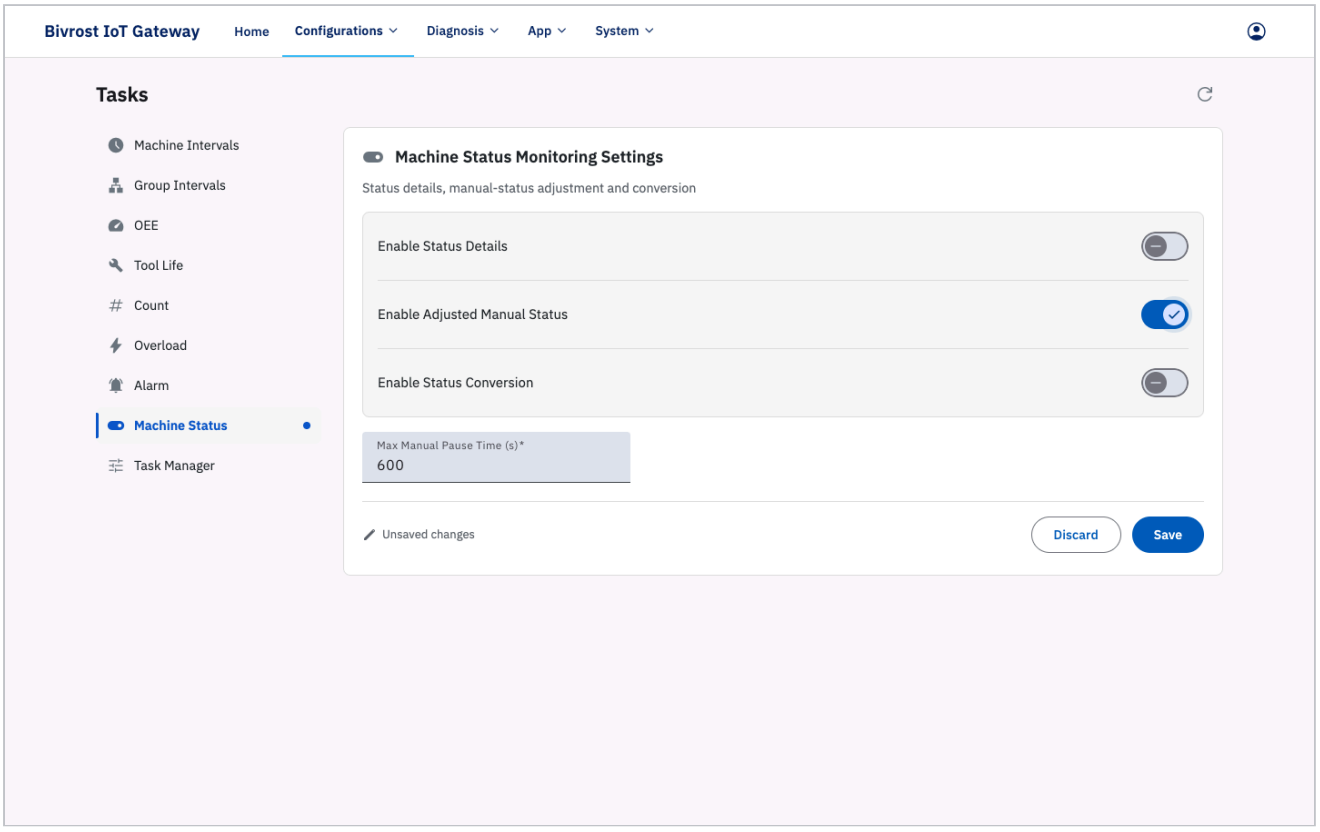
After uploading a mapping file, go to Machines, open the "Alarm" task settings of the machine concerned, enable alarm mapping and select the file; see 3.3.1.2.6. Alarm Mapping.

3.5.8. Machine Status Monitoring Settings

"Machine Status Monitoring Settings" configures the options of the machine status task, including Enable Status Details, Enable Adjusted Manual Status and Enable Status Conversion. For the machine task variables referred to in this section, see 4.2.4. CNCStatus: Machine Status.



The "Enable Status Details" option is off by default, in which case the uploaded status data contains no status details. With the option enabled, the uploaded data includes the status details.

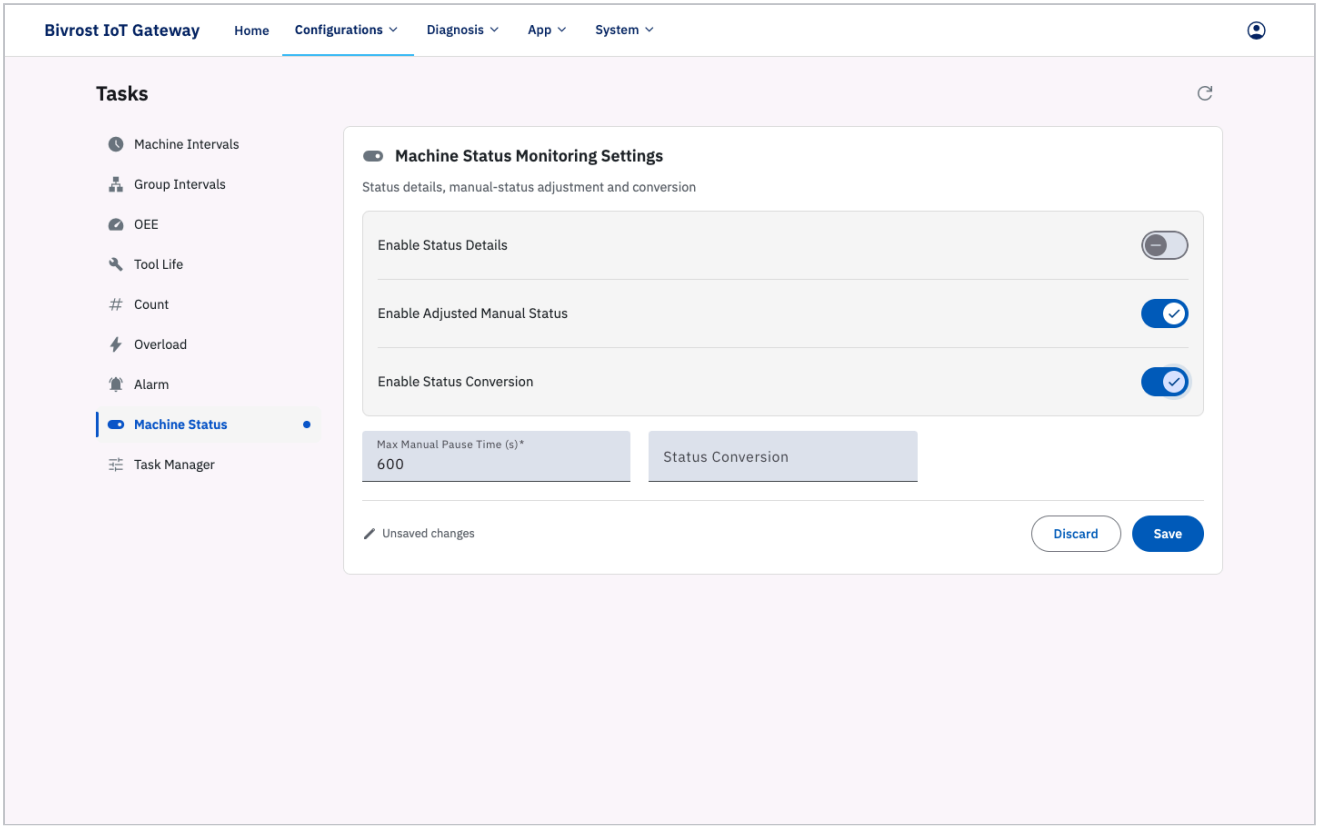


When "Adjusted Manual Status" or "Status Conversion" is enabled, the status is adjusted according to the corresponding rules and reported in adjustedStatus (4.2.4. CNCStatus: Machine Status). Wherever machine status is involved — OEE statistics, cumulative status time and so on — the adjusted status takes precedence. Both options are off by default, in which case adjustedStatus is identical to the real-time status returned by the machine. "Adjusted Manual Status" and "Status Conversion" here are global settings. Enabling the machine-level settings of the same name in 3.3.1.2.5. Special Settings overrides the global settings.

The rules for adjusting the manual status are:

1. The machine is currently in the Manual state;
2. The time since the machine changed from a non-Manual state to the Manual state exceeds the configured Max Manual Pause Time;
3. The machine's coordinate data has remained unchanged for longer than the configured Max Manual Pause Time.

Normally adjustedStatus is identical to the machine's real-time status. If all of the above conditions are met at the same time, adjustedStatus is no longer Manual but is corrected to Wait.



With status conversion enabled, adjustedStatus is set according to the commands you enter, for example:

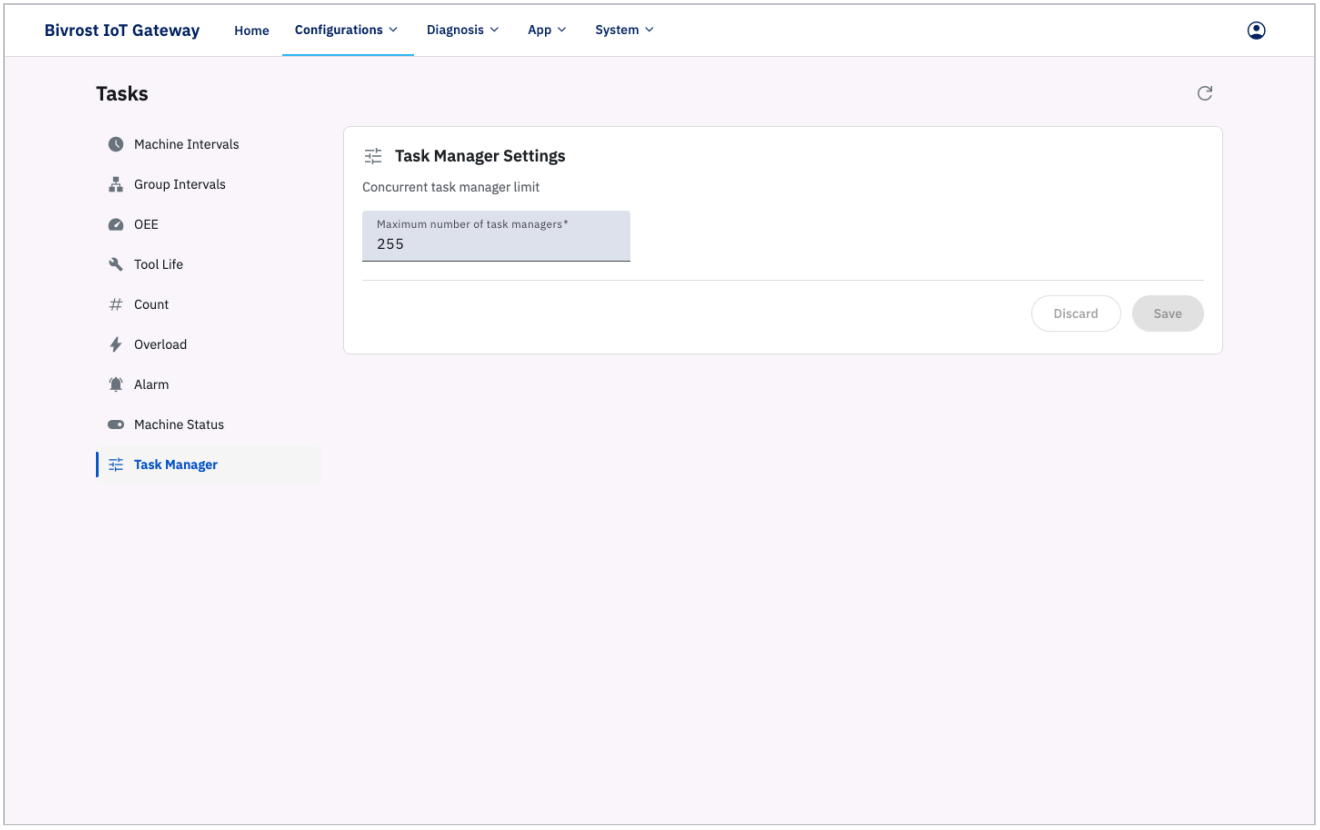
```
CNCStatus_cncStatus = "MANUAL_MDI_RUN" | AUTO_RUN
```

`CNCStatus_cncStatus = "MANUAL_MDI_RUN"` is the precondition (see 11.2.2. Precondition): if the cncStatus (running status, of type String) returned by this machine status task is "MANUAL_MDI_RUN", then adjustedStatus is set to AUTO_RUN. Separate multiple conversion commands with ";".

When both options are enabled, the manual status adjustment runs first and the status conversion second.

3.5.9. Task Manager Settings

Each task manager corresponds to one task thread. "Task Manager Settings" sets the maximum number of task managers, that is, the maximum number of task threads.



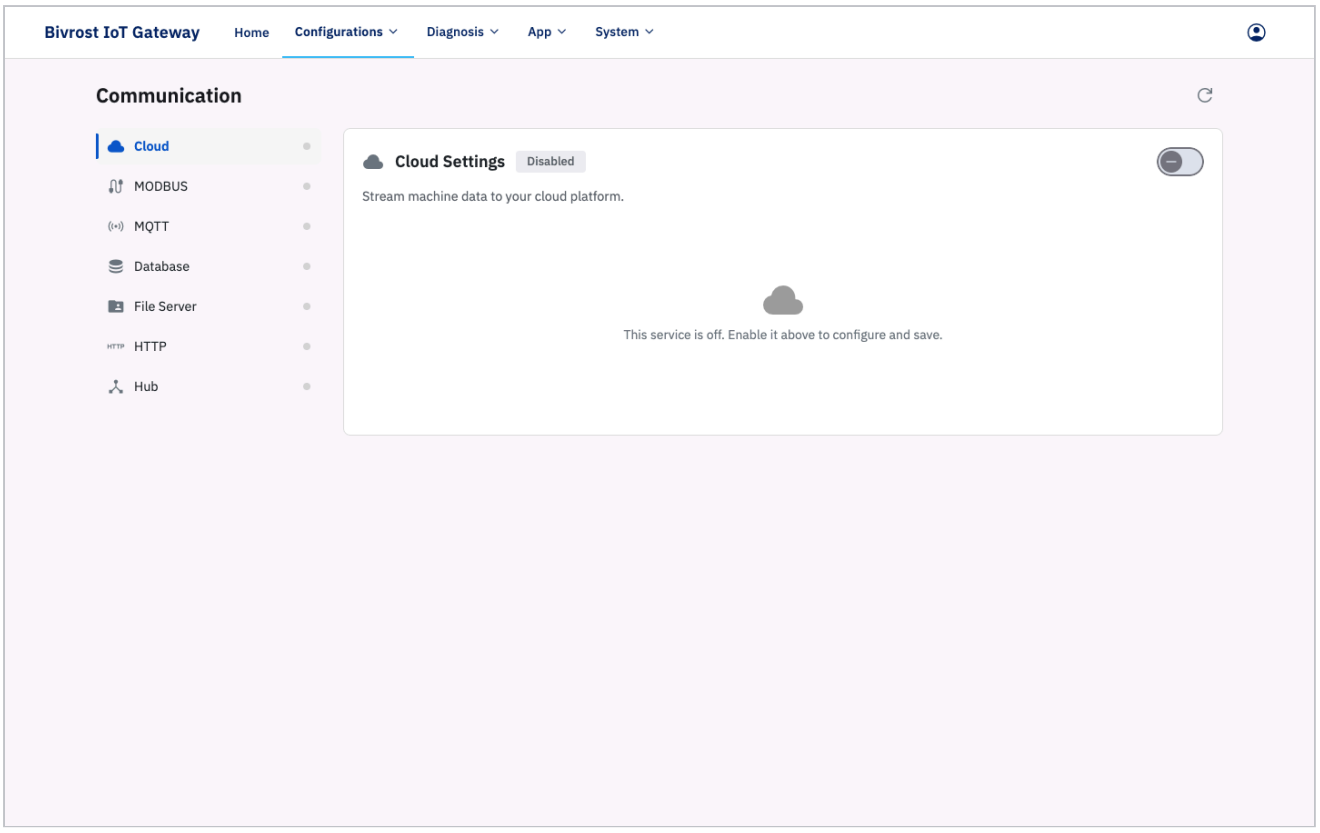
Each machine is allocated one task manager by default. You can assign several independent task managers to improve task throughput (see Parallel Task Processing in 3.3.1.5. Advanced Settings). The more task managers there are, the greater the demands on the hardware, so tune this parameter to your hardware's capabilities. The maximum number of task managers defaults to 255. When the total number of task managers required by the active machines

exceeds this maximum, task managers are merged automatically to bring the total back below the limit, which reduces task throughput.

3.6. Communication

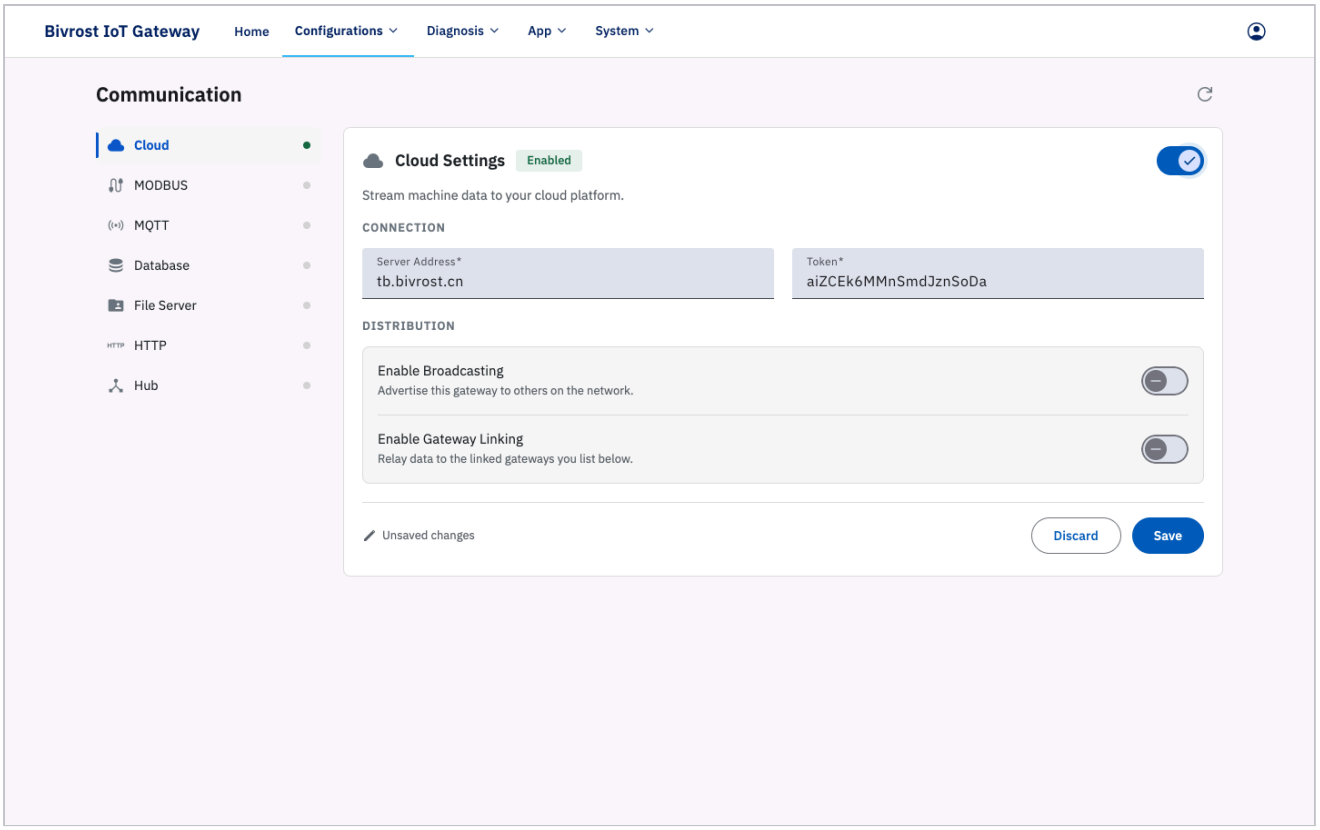
Communication covers Cloud Settings, MODBUS Settings, MQTT Settings, Database Settings, Gateway File Server Settings, HTTP Settings, Hub Settings, MCP Settings and so on. You can turn on one or more of these communication methods as required.

The left side of the **Communication** page lists the available communication methods (and shows whether each one is enabled); the right side holds the settings for the selected method. Turn on the **Enable** switch in those settings to show all of the fields. Except for MCP Settings, after you click **Save** you must go back to Home and click **Restart Service** to apply the changes.



3.6.1. Cloud Settings

Click the **Enable** switch to turn on cloud platform communication.



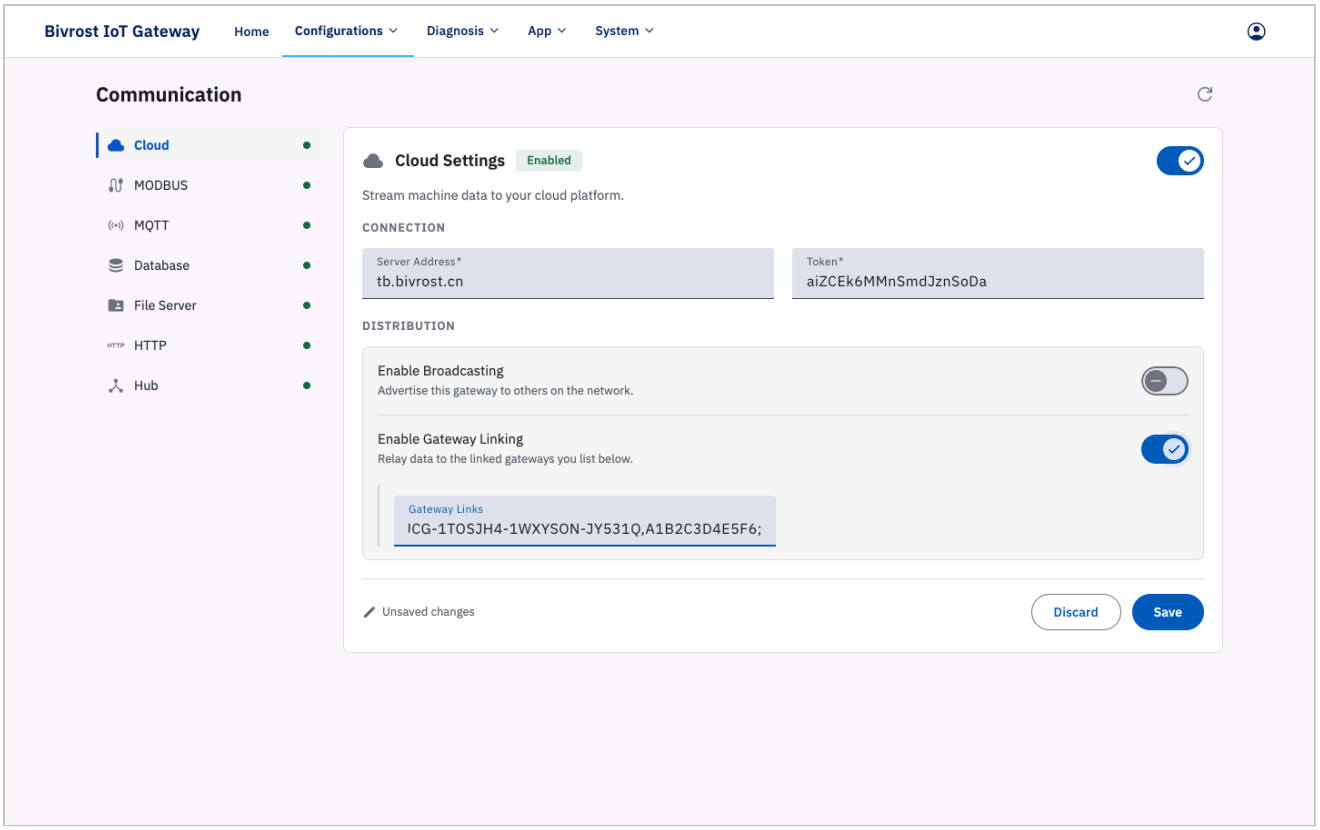
After completing Cloud Settings, saving, restarting the service and confirming that the gateway is connected to the cloud platform, replace the gateway address in the HTTP request URL with the cloud platform server address and add an `AccessToken` request header whose value is the gateway token entered in **Token** on this page. See 5.1.1. Forwarding Requests Through the Cloud Platform or Hub for instructions and an example.

Once cloud communication is enabled, the server settings are complete and the automatic collection tasks of the target machines and groups are switched on (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), task data is uploaded in real time to the designated standard cloud platform over a standard protocol, where it can be stored, analysed and shown on dashboards. Bivrost runs a standard cloud platform server at tb.bivrost.cn; contact support to arrange a trial.

- Enter the cloud platform address in **Broker Address**.
- Enter the token the cloud platform has assigned to the gateway in **Token**.
- With **Enable Broadcasting** on, the gateway broadcasts data through the cloud platform to the other gateways on that platform.
- With **Enable Gateway Linking** on, the gateway receives data broadcast through the cloud platform by the other gateways on that platform. Add the UID and token of each gateway you want to link to under **Gateway Links**, in the following format:

```
UID,token;
```

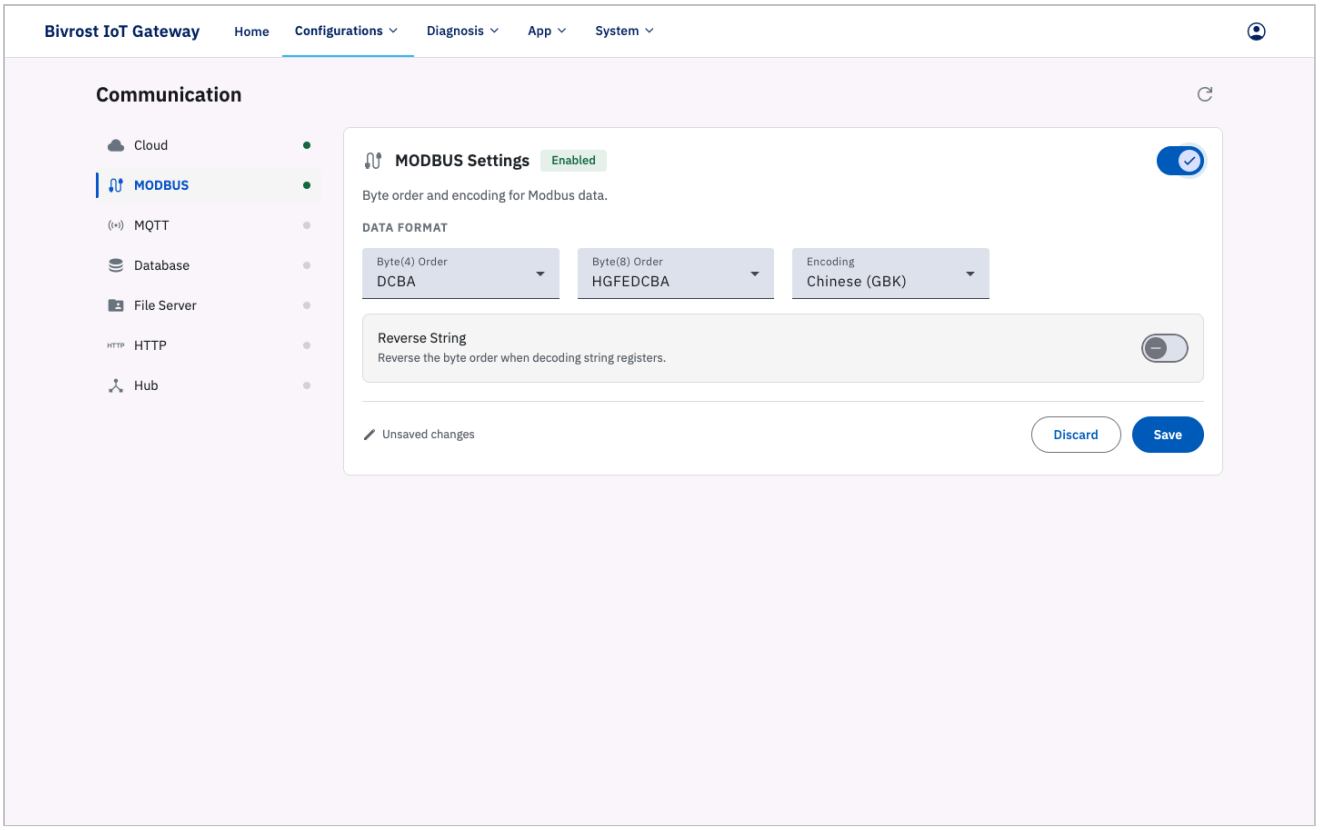
Separate multiple linked gateways with `;`.



> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.2. MODBUS Settings

Click the **Enable** switch to turn on MODBUS communication.



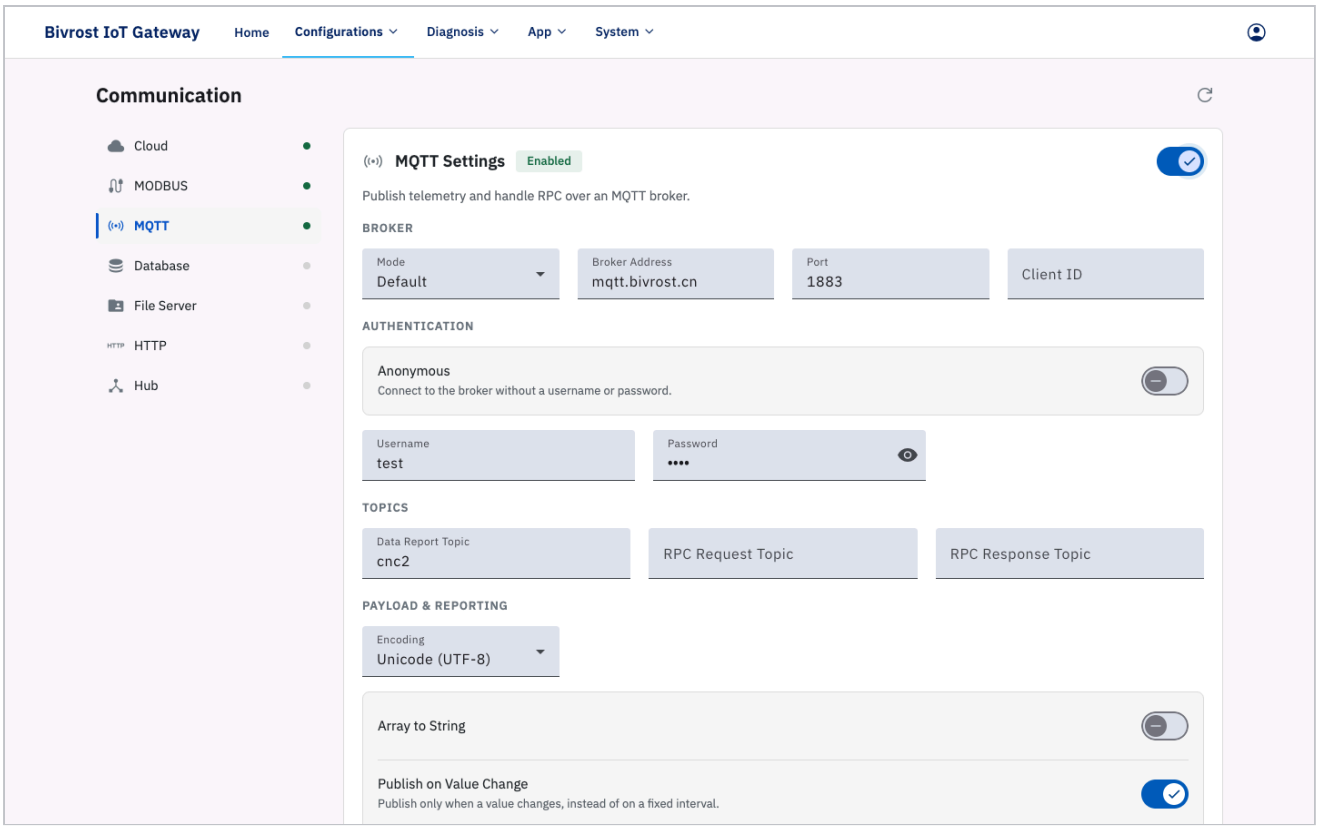
Once MODBUS communication is enabled and the automatic collection tasks of the target machines and groups are switched on (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), the gateway uses the MODBUS protocol to store the task data in the corresponding address registers — see 3, MODBUS Communication,.

- **Byte(4) Order** selects the decoding format used in MODBUS communication for 32-bit data (long integers such as Int32, single-precision Float and any other type that occupies two register addresses). The default is “DCBA”.
- **Byte(8) Order** selects the decoding format used in MODBUS communication for 64-bit data (double-precision Double and any other type that occupies four register addresses). The default is “HGFEDCBA”.
- **Encoding** selects the string encoding used by the MODBUS protocol. The default is “Chinese (GBK)”.
- With **Reverse String** on, the character order of strings is reversed.

> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.3. MQTT Settings

Turn on the **Enable** switch to start MQTT communication.



Once MQTT communication is enabled and the automatic collection tasks of the target machines and groups are switched on (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), the gateway publishes the task data as JSON messages to the designated MQTT broker; see 7. MQTT Communication.

Select a **Mode**, then enter the broker address and port. The screenshot shows a standard MQTT mode, where you can set the Client ID, anonymous access or credentials, and data and RPC topics. Selecting **GeWu** (China Unicom Gewu Cloud) instead shows gateway and subdevice product settings. The gateway generates its Client ID and credentials, and the data topic is fixed by the platform.

- In standard modes, **Data Report Topic** is the topic used for automatic collection results. The default is “r”. GeWu mode uses the platform’s subdevice property topic; see 7.1. Data Upload Message Format.
- In standard modes, **RPC Request Topic** and **RPC Response Topic** configure the RPC interface. Both are blank by default, which disables it. Once set, the gateway subscribes to the request topic and publishes results to the response topic; see 7.2. RPC Interface.
- With **Publish on Value Change** on, the automatic collection tasks enabled in the machine task settings and group task settings only upload a result when the collected value has changed.

Connecting to Gewu Cloud

1. Create the thing model, gateway product, machine product, and group product on Gewu Cloud, then create the gateway device and obtain its device secret. The gateway uses per-device credentials. Machine and group subdevices use weak validation: they need neither preregistration nor individual secrets, but **both subdevice products must allow dynamic registration**.
2. Enable the machines and groups to be reported. Their `machineID` / `groupID` becomes each subdevice' s `deviceKey` and must contain 4–32 ASCII letters, digits, or underscores. Invalid subdevices are skipped.
3. Select **GeWu**, enter the broker address, port, and the fields below, save, then restart the service. The gateway generates its MQTT Client ID, username, and signed password, connects, and adds active machines and groups to its topology before it starts publishing their data.

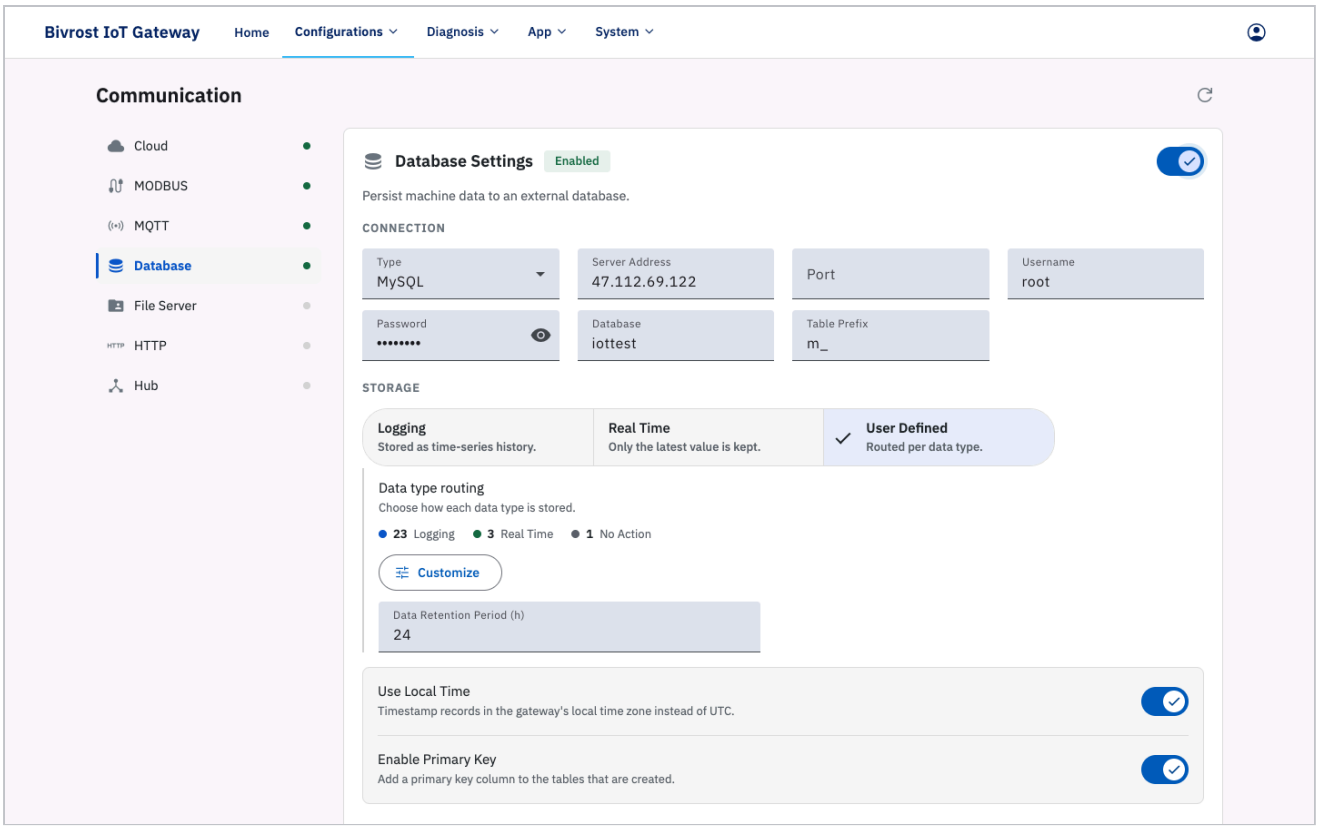
UI field	Value
Gateway Product Key	Product Key of the gateway product on the platform.
Gateway Device Key	Device Key of the gateway device; 4–32 ASCII letters, digits, or underscores.
Gateway Device ID	10–32 ASCII letters, digits, or hyphens.
Gateway Device Secret	Device Secret generated for the gateway device.
Gateway Sign Method	HMAC-SHA256 (0) or HMAC-SM3 (1).
Gateway Operator	0 for no cellular network; 1, 2, 3, or 4 for China Unicom, Mobile, Telecom, or Broadnet.
Machine Product Key	Product Key of the machine product.
Group Product Key	Product Key of the group product.

All three Product Keys are required and cannot contain `/`, `+`, or `#`. If a topology addition fails, the gateway does not begin publishing subdevice data; check dynamic registration on the platform and the service log.

> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.4. Database Settings

Turn on the **Enable** switch to start database communication.

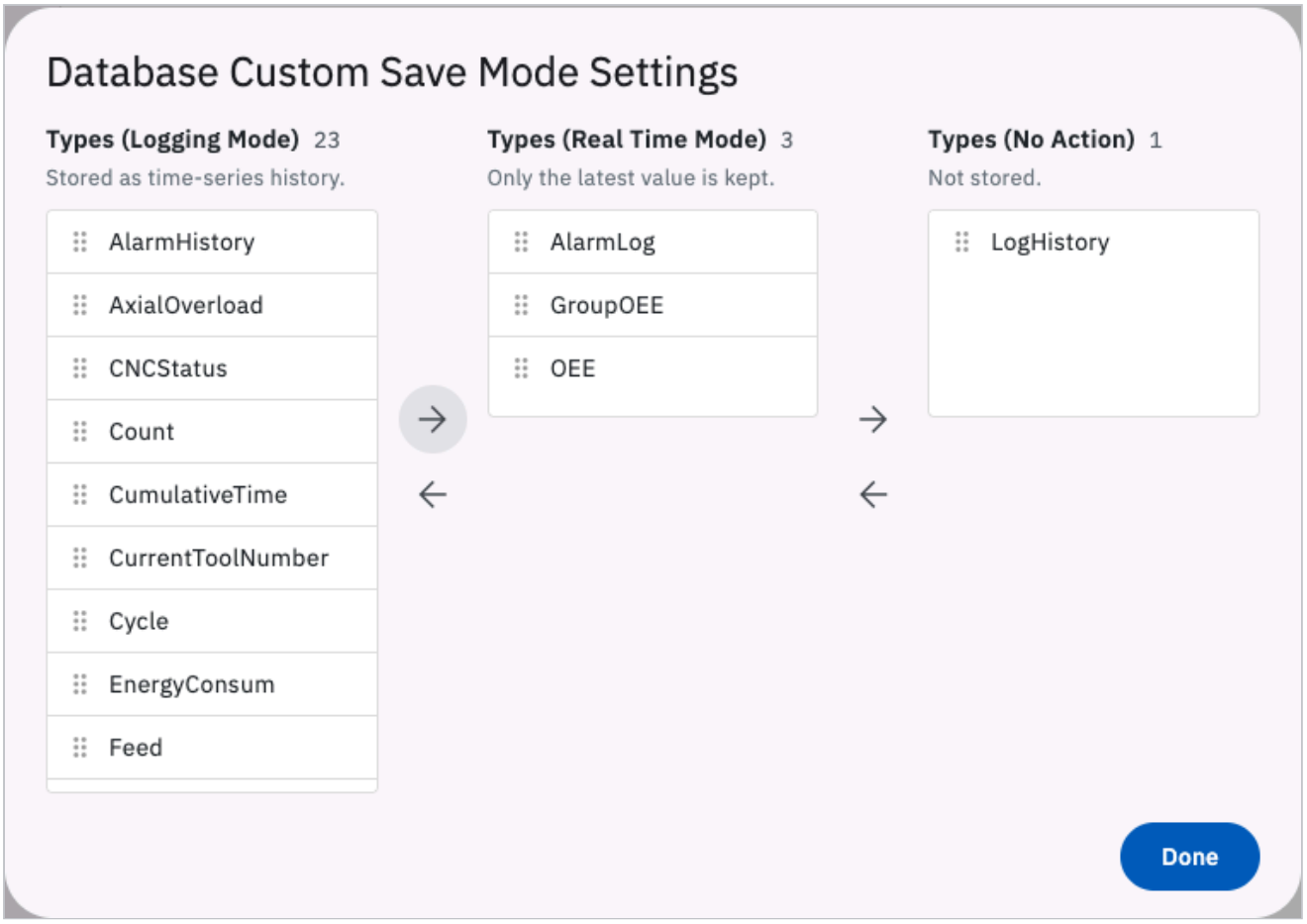


Once database communication is enabled and the automatic collection tasks of the target machines and groups are switched on (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), the gateway writes the results of those tasks to the designated database. The database types currently supported are SQL Server, MySQL, PostgreSQL and InfluxDB v2.x. See 5, Database Communication,.

The database settings are: Type, Broker Address, Port, Username, Password, Bucket, Table Prefix, Storage Mode, Data Retention Period (h), Use Local Time, Enable Primary Key, Write on Value Change and so on.

- **Type** is the type of database: SQL Server, MySQL, PostgreSQL, InfluxDB v2.x and so on.
- **Broker Address** is the IP address of the server the database runs on.
- **Port** is the port of the server the database runs on.
- **Username** and **Password** are used to sign in to the database.
- **Bucket** is the name of the database.
- **Table Prefix** adds a prefix in front of the default table names.
- **Storage Mode** offers Log, Real Time, User Defined and so on.
    - **Log** mode: every time the gateway reads data from a machine it writes a matching record to the database. The database keeps the complete history.

- **Real Time** mode: every time the gateway reads data from a machine it finds the matching record for that machine in the database and updates it with the latest data; no history is kept.
- The default storage mode is **User Defined**: click the **Settings** button on the right and use the "Database Custom Save Mode Settings" dialog to put each individual task into Log mode or Real Time mode. In the dialog, the left column is **Types (Logging Mode)**, the middle column is **Types (Real Time Mode)** and the right column is **Types (No Action)** (not recorded). Drag an interface with the mouse into the mode you want, or click the left and right arrow buttons to move every interface on one side across to the other. Click anywhere outside the dialog to close it, then click **Save** on the page to store your custom settings. By default Alarm, Machine OEE and Group OEE are in Real Time mode and all other data is in Log mode.



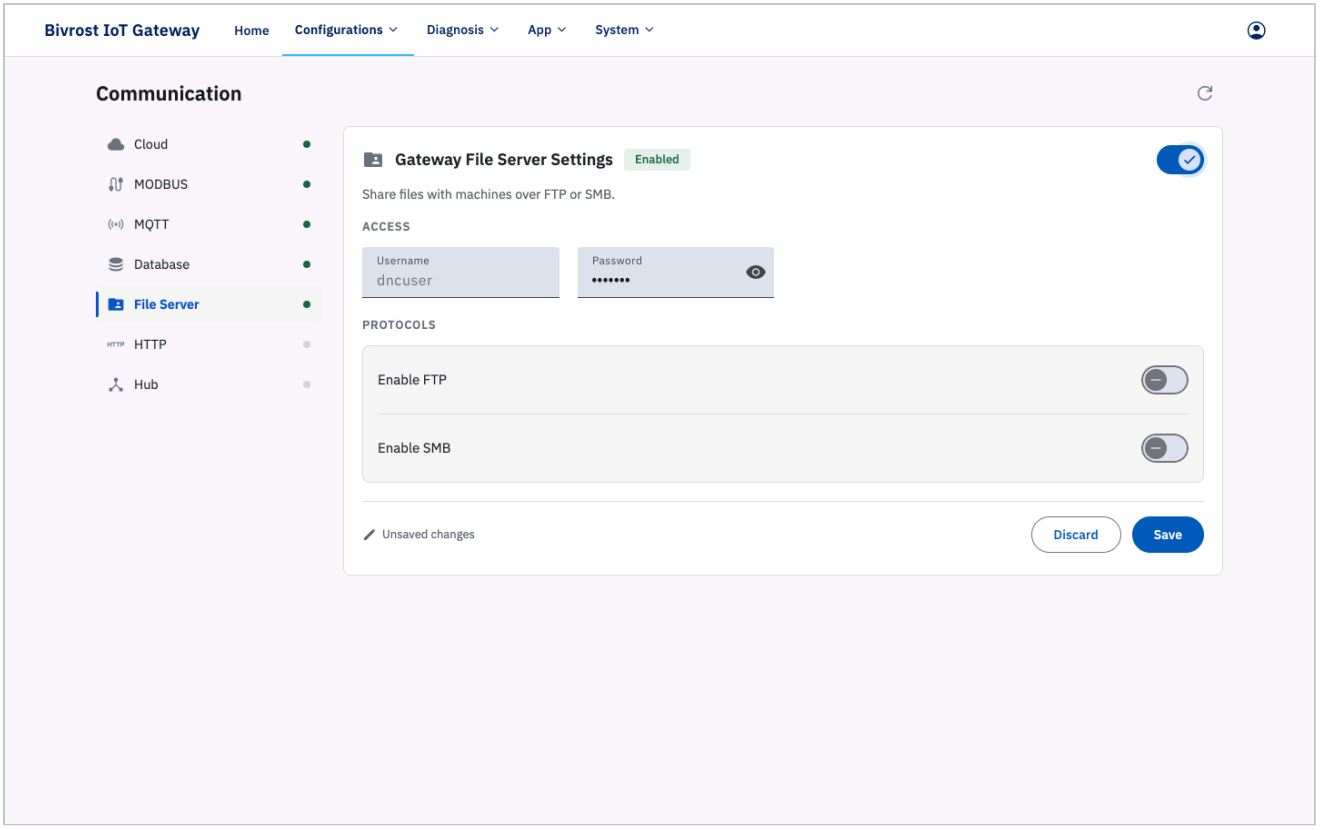
- **Data Retention Period (h)** is how long data is kept in the database, in hours. If it is set to x, the gateway runs a scheduled database cleanup after start-up and automatically deletes data older than x hours. The default is 0, which keeps all data.
- With **Use Local Time** on, data is written using the local time of the time zone set in 3.12. Settings; otherwise UTC is used.
- **Enable Primary Key** adds a primary key column to every table in the database.

- With **Write on Value Change** on, the automatic collection tasks enabled in the machine task settings and group task settings only write a result when the collected value has changed.

> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.5. Gateway File Server Settings

Turn on the **Enable** switch to start the gateway file server. The gateway file server is a file server created locally on the gateway that machines can reach over FTP or as a shared folder. Once it is running, the File Server Type option in 3.3.1.3. DNC Settings can be set to "Gateway File Server".



The username and password are shared by FTP and the shared folder. Both default to dnc_user. At present only the password can be changed.

Turn on **Enable FTP** to allow the gateway file server to be reached over FTP. The FTP address is:

```
ftp://iotgw or ftp://<gateway-IP>
```

Under the FTP home directory each machine has its own folder, named after the machine’s IP address.

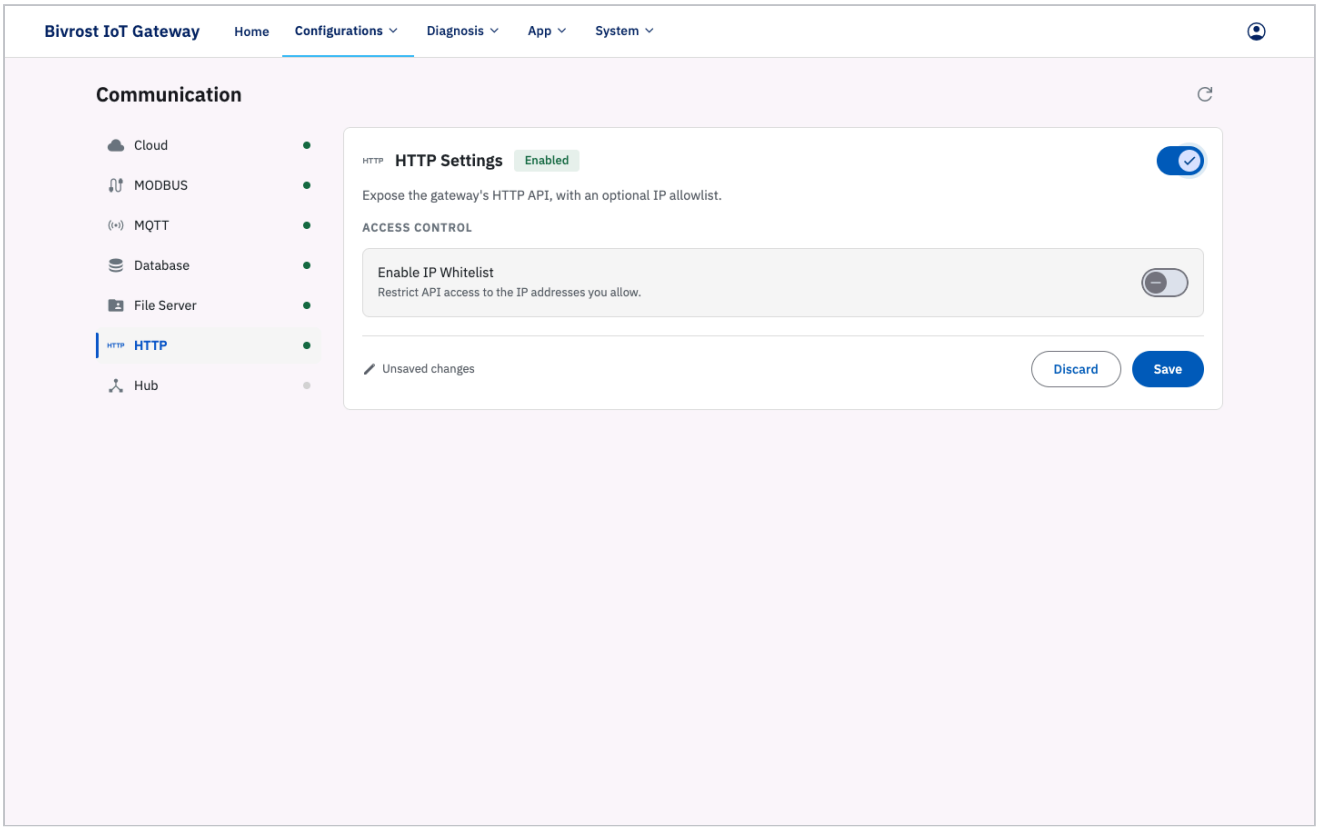
Turn on **Enable SMB** to allow the gateway file server to be reached as a shared folder. Each machine has its own shared folder, named after the machine’s IP address. The shared folder address is:

```
\\iotgw\<machine-IP> or \\<gateway-IP>\<machine-IP>
```

> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.6. HTTP Settings

Turn on the **Enable** switch to start the HTTP service.



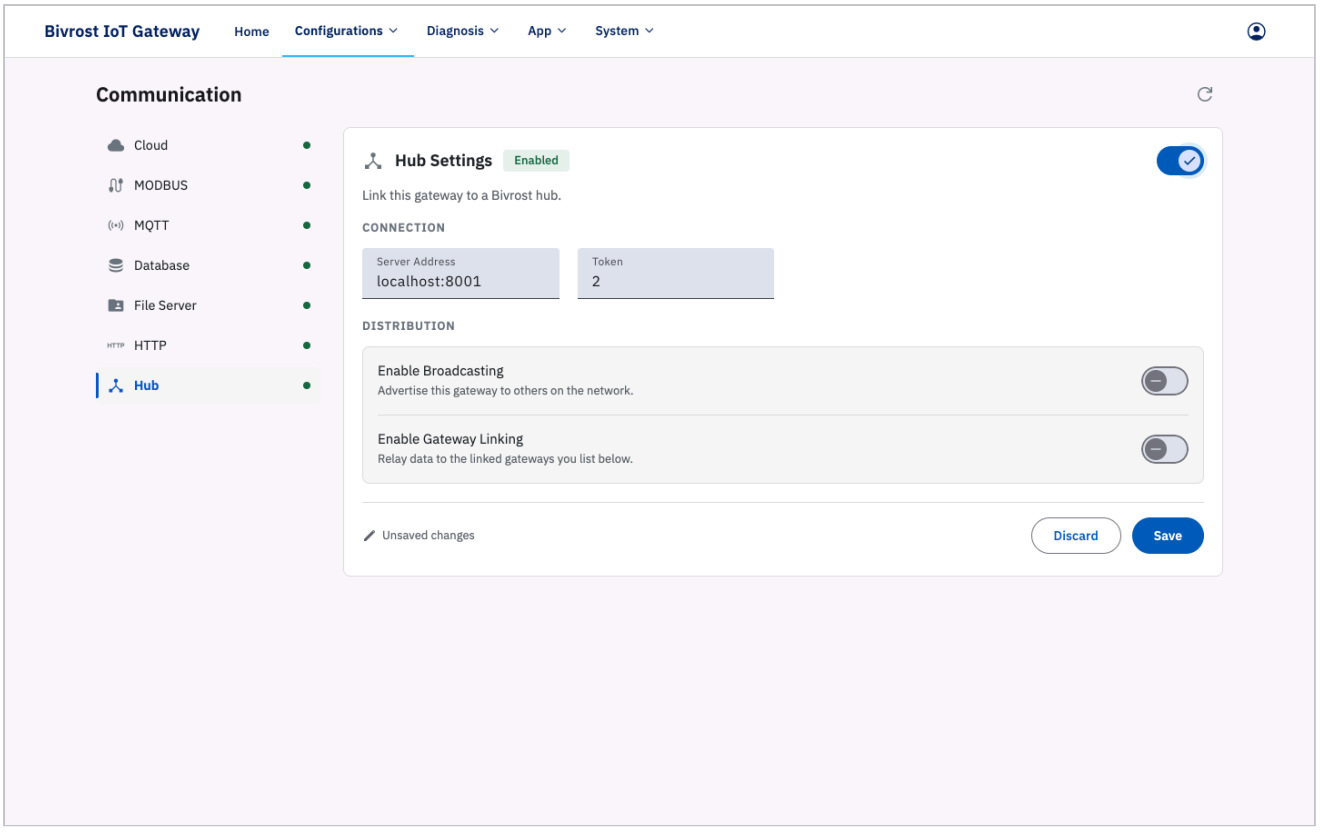
With **Enable IP Whitelist** on, all API requests except those with valid JWT authentication are restricted to whitelisted IP addresses. That is, the login endpoint is not restricted by the whitelist, and requests authenticated with a JWT after login skip the whitelist check; requests

authenticated with a SecretKey must still pass the whitelist check. The **Whitelist** can hold several IP addresses, separated by “;” .

> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.7. Hub Settings

Turn on the **Enable** switch to start the Hub service. A Hub server is used to reach the gateway’s interfaces remotely, so that configuration can be changed remotely and gateways on different sites can exchange data. Contact support to have one deployed.



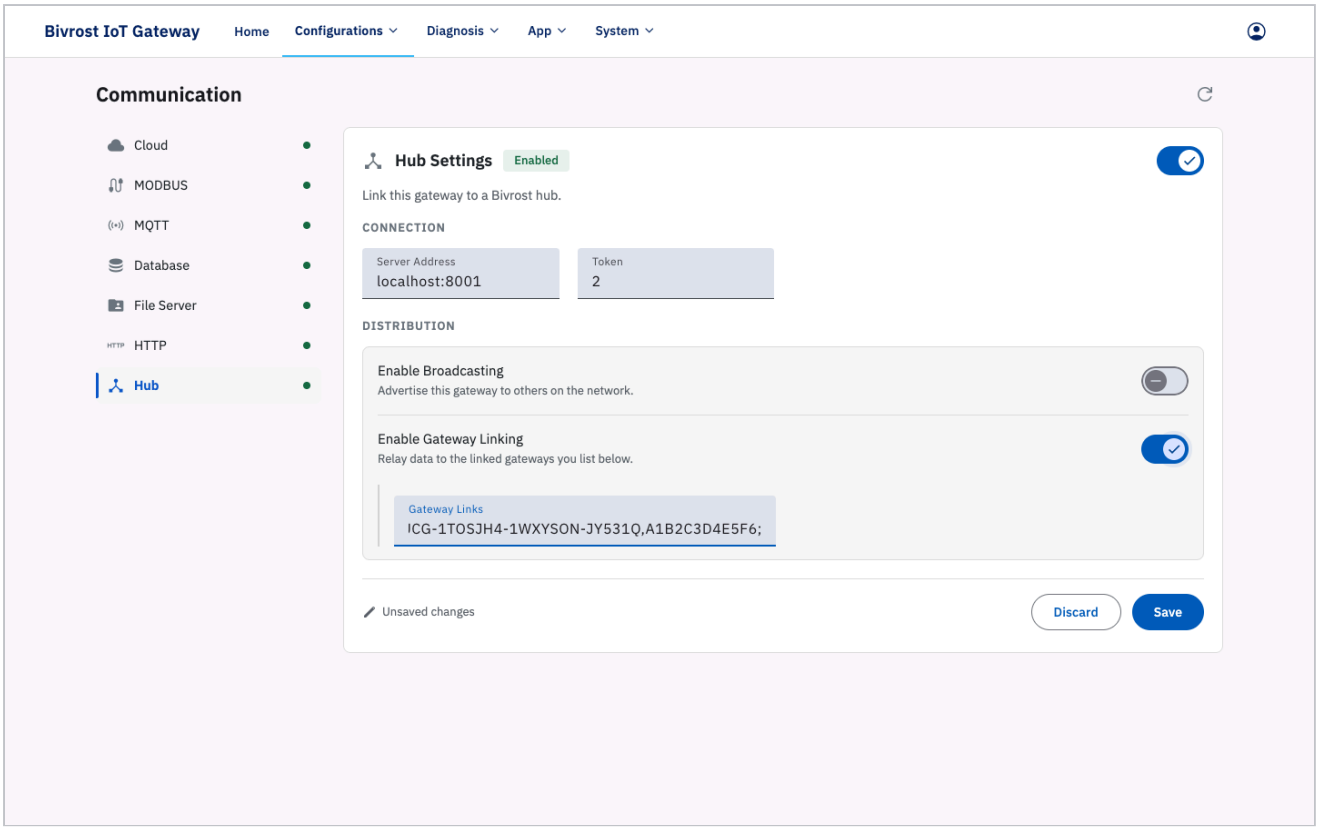
After completing Hub Settings, saving, restarting the service and confirming that the gateway is connected to the Hub, replace the gateway address in the HTTP request URL with the Hub server address and add an `AccessToken` request header whose value is the gateway token entered in **Token** on this page. See 5.1.1. Forwarding Requests Through the Cloud Platform or Hub for instructions and an example.

- Enter the Hub server address in **Broker Address**.
- Enter the token the Hub server has assigned to the gateway in **Token**.

- With **Enable Broadcasting** on, the gateway broadcasts data through the cloud platform to the other gateways on that platform.
- With **Enable Gateway Linking** on, the gateway receives data broadcast by other gateways through the Hub server. Add the UID and token of each gateway you want to link to under **Gateway Links**, in the following format:

```
UID,token;
```

Separate multiple linked gateways with `;`.



> **Note**
>
> When the settings are complete, click **Save**, then go back to Home and click **Restart Service** to apply the changes.

3.6.8. MCP Settings

Turn on the **Enable** switch and click **Save** to start the MCP service. The MCP (Model Context Protocol) server exposes machine data, data analysis and gateway status as "tools" for MCP-capable AI clients (such as Claude Code and Claude Desktop), so machines can be queried in natural language without calling the interfaces by hand.

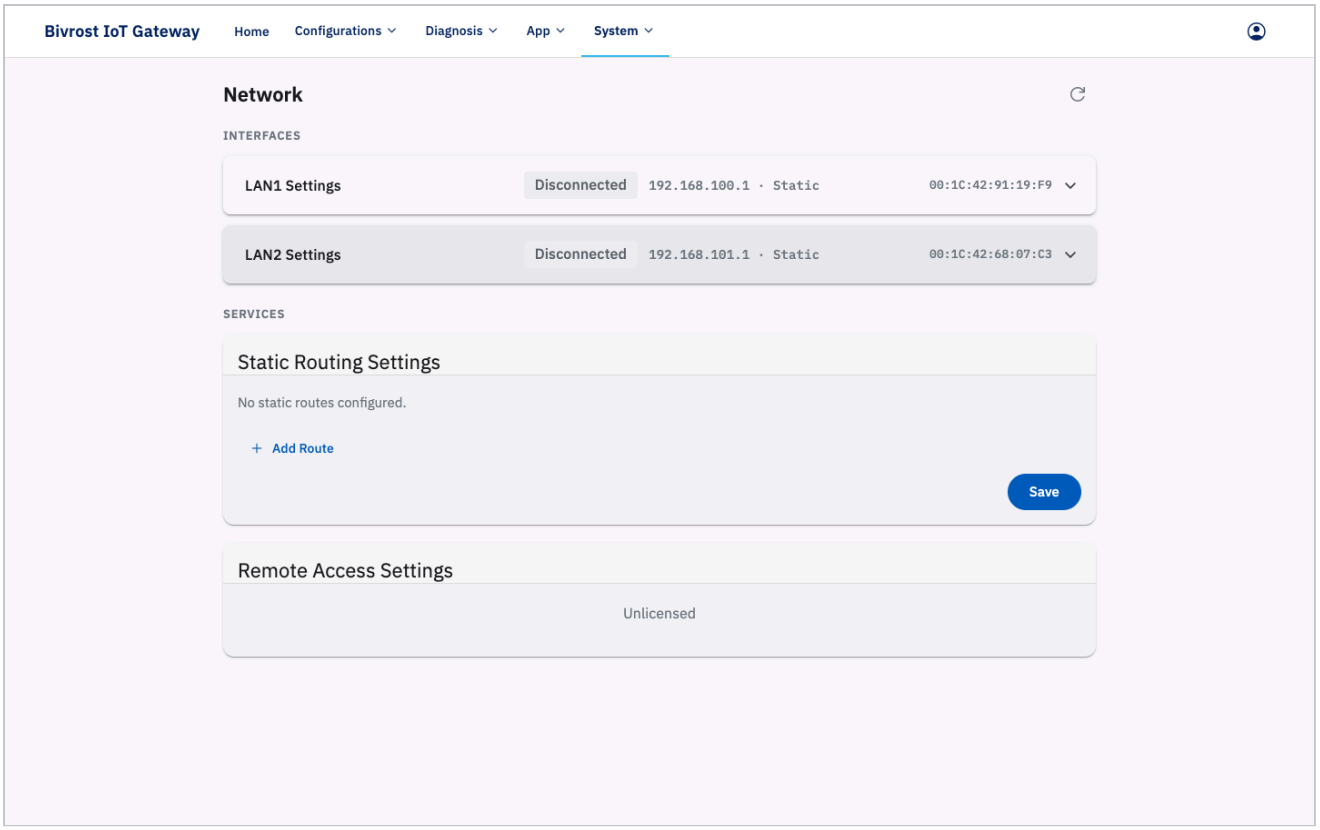
Endpoint shows this gateway’s MCP server address (for example `http://192.168.100.1/mcp`), for use in the MCP client’s configuration. It is read-only.

The MCP service is disabled by default; while it is off, that address returns HTTP 404. It currently exposes 33 read-only tools — anything involving writes, machine control or program transfer has no MCP tool. Authentication and access control follow the HTTP interface settings (secret keys, IP whitelist, authorized APIs) and need no separate configuration on this page. For the full tool list, the authentication methods and client configuration examples, see 9. MCP Service.

> **Note**
>
> Unlike the other communication methods, the MCP switch takes effect as soon as you click **Save** — there is no need to go back to Home and restart the service.

3.7. Network

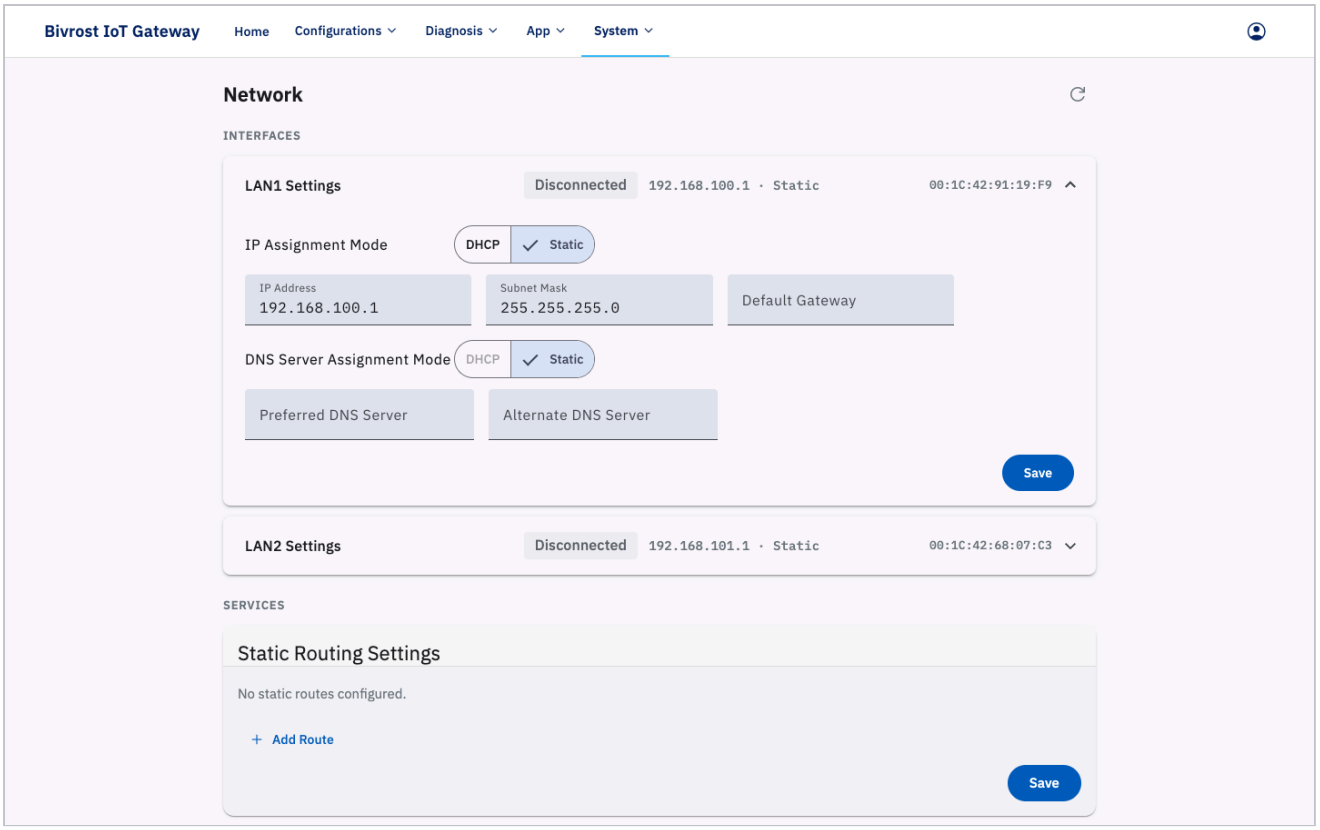
The **Network** page is split into two parts: "Interfaces" (LAN1, LAN2 and the wireless adapter) and "Services" (Static Routing Settings, Remote Access Settings). Click an interface card to expand its settings.



3.7.1. Wired Network

The "Wired Network" section shows the current connection status of the LAN1 port. "Connected" means the LAN adapter has detected a device at the other end of the cable; it does not guarantee that the device can be pinged. You can change the Connection Type of the gateway's LAN1 port, setting the IP Assignment Mode to DHCP (obtained automatically) or Static (entered manually). LAN1 is normally set to Static, in which case you can edit the IP Address, Subnet Mask, Default Gateway and DNS server settings. The default gateway and DNS server only need to be changed when the company network is restricted; otherwise leave them at their defaults. When you are done, click **Save** to apply the settings.

> **Note**
>
> The settings take effect as soon as you click **Save** — no service restart is required. If you were connecting to the gateway on its LAN1 IP address, you must use the new IP address after changing it.



Once the LAN2 Settings option has been enabled under Settings, LAN2 Settings appears below LAN1 Settings and you can change the Connection Type of the LAN2 port; see 3.12. Settings for details.

3.7.2. WiFi Settings

The gateway is fitted with a wireless adapter and can connect to a WiFi network (models on which the wireless adapter is unavailable do not show this section).

While WiFi is disconnected, click the "WiFi Name" field to scan automatically for nearby networks. You can type a name to narrow the list, then click the network you want to join. For a hidden network, type the full WiFi name manually.

Enter the "WiFi Password" and click **Connect**. If the connection succeeds, the "Status" field shows "Connected: [WiFi name][signal strength]" and the **Connect** button changes to **Disconnect**. Unless you click **Disconnect**, the gateway reconnects automatically whenever it

detects that network again. Clicking **Disconnect** drops the current network and stops the gateway from reconnecting to it automatically.

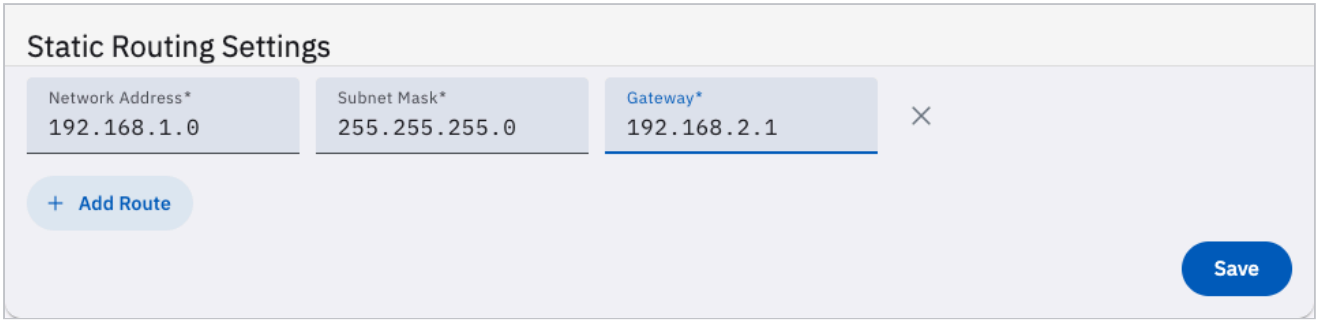
Once connected, click **Configure** to open the "WiFi Settings" dialog. As in 3.7.1. Wired Network, you can change the WiFi IP Address, Subnet Mask and other settings.

3.7.3. Static Routing Settings

You can add static routes here. The route format is:

```
IP network address 1,subnet mask 1,gateway IP address 1;IP network
address 2,subnet mask 2,gateway IP address 2;
```

Separate multiple routes with `;`.



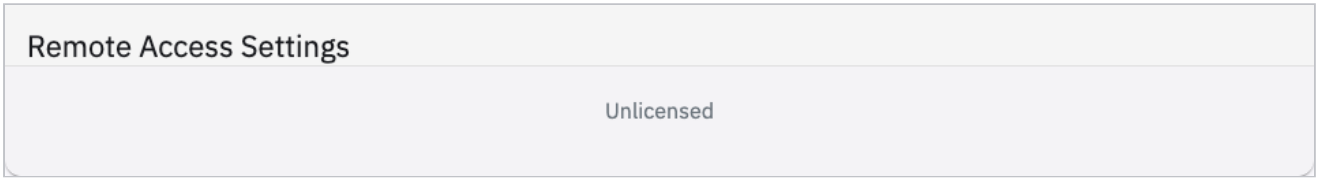
3.7.4. Remote Access Settings

Once remote access is configured, connecting to a remote server provides the following:

1. Remote access to the gateway from outside the local network.
2. Bridging of the LAN1 or LAN2 network, so that debugging software can connect to a machine remotely for commissioning.

Checking the remote access configuration takes a while; the status reads "Detecting" while it runs, as shown below.

This feature is not enabled by default and shows the message below. Contact customer support if you need it enabled.



Once the feature has been enabled, the status is “Disconnected” until it is configured for the first time, as shown below.

Click **Configure** and set up the remote server in the configuration dialog: enter the server address, password and bridge target, then click Confirm. After changing the configuration, you must reconnect to the remote server for it to take effect.

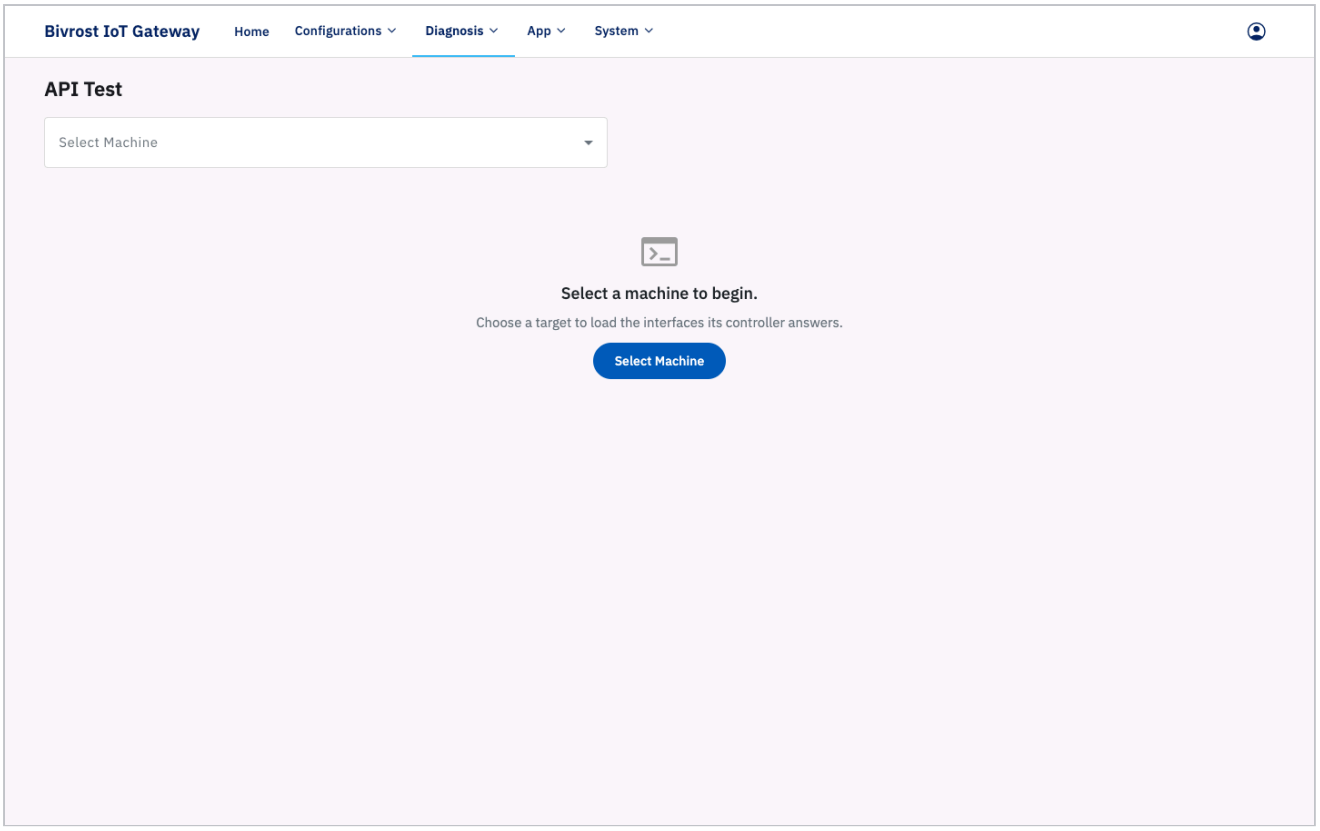
Click the **Connect** button; the gateway starts connecting to the remote server.

Once connected, the gateway connects to the remote server automatically the next time it comes online, as shown below. Click the **Disconnect** button to drop the connection to the remote server immediately and stop it from reconnecting automatically next time.

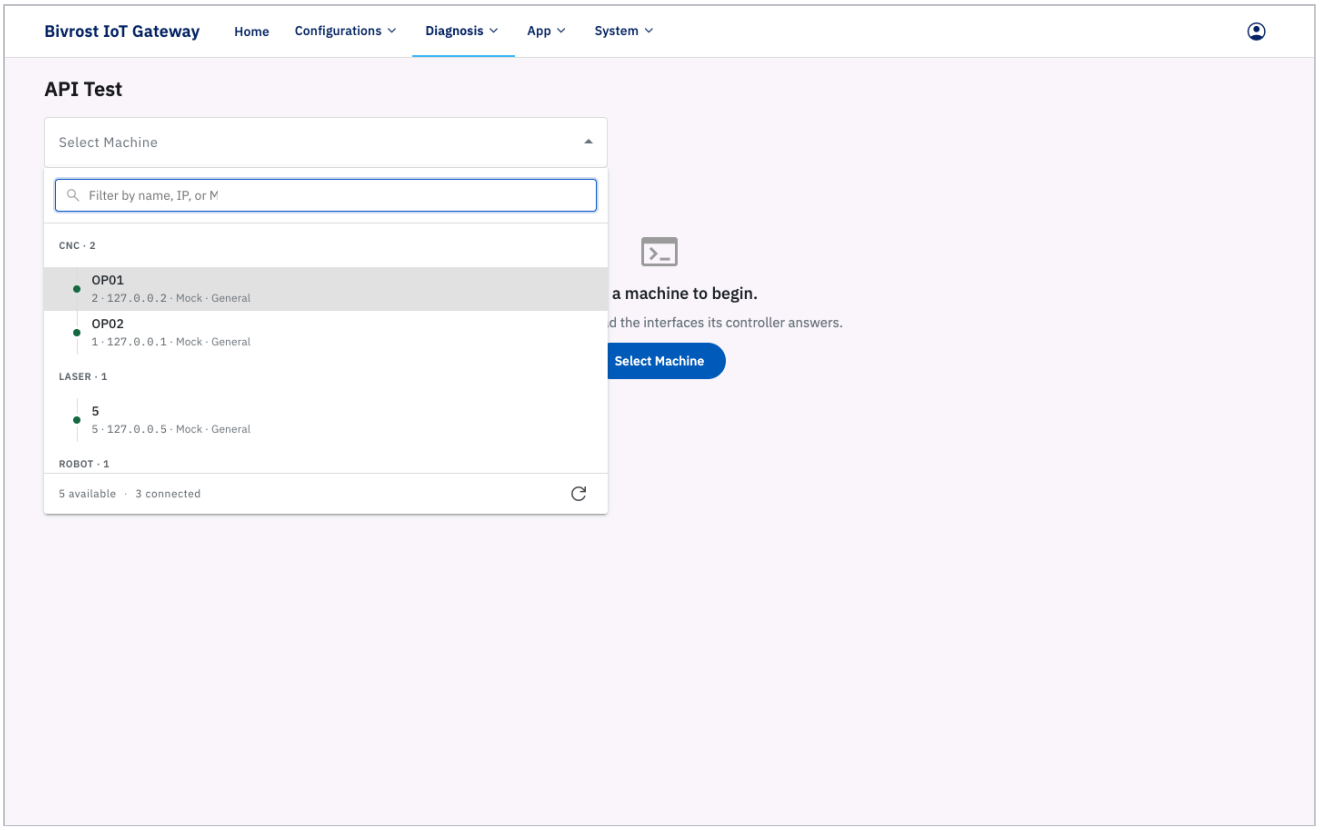
3.8. API Test

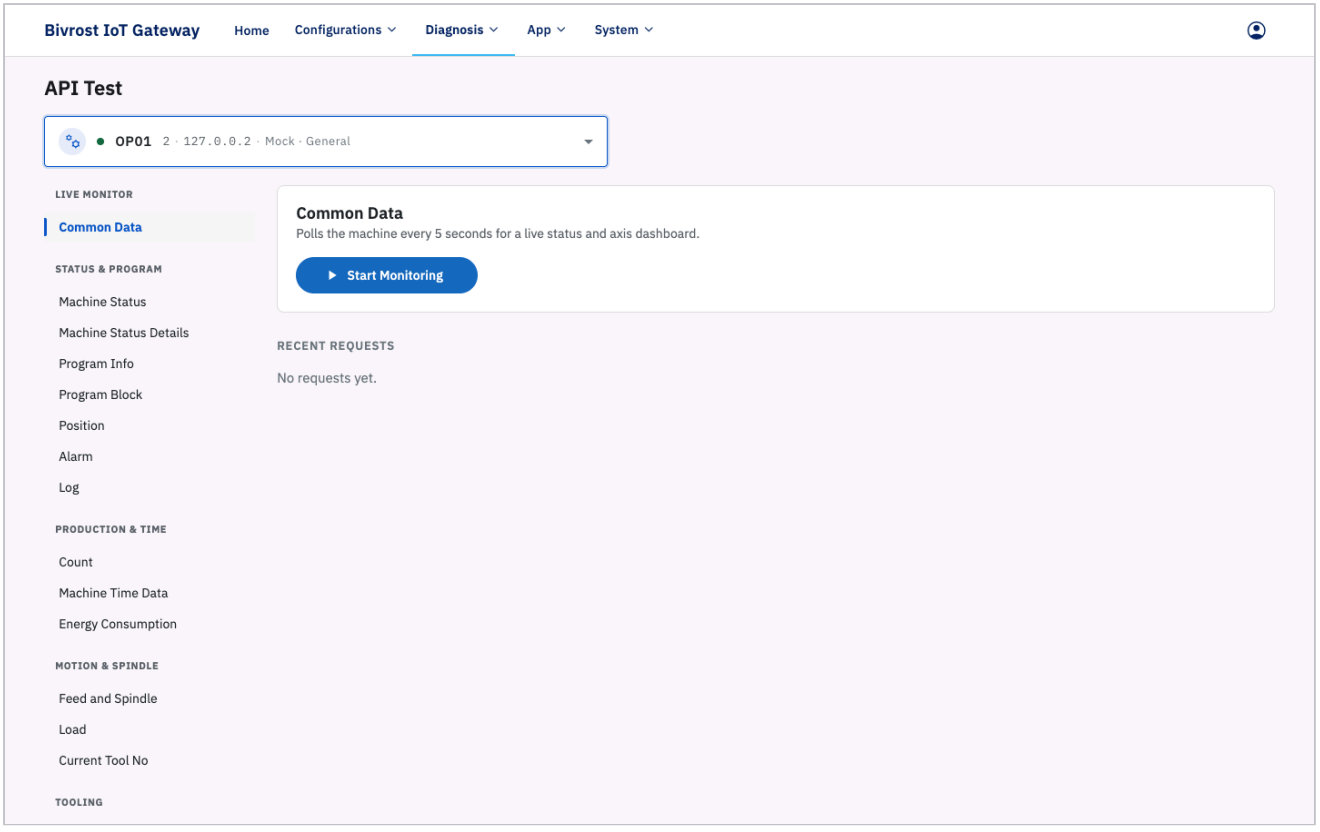
This page exposes a selection of read and write interfaces so you can verify them before going live. Full descriptions of every interface on this page can be found under 5.5.1. Direct Read/Write.

Before running an API test, if you have changed any settings on **Machines**, **Groups**, **Tasks** or **Communication**, click **Restart Service** on **Home** first.



In the **Select Machine** box at the top, pick a machine that is activated on the **Machines** page, then pick the HTTP interface you want to test from the interface list (**Common Data** by default). Click **Monitoring/Send** to retrieve data from that interface on that machine.

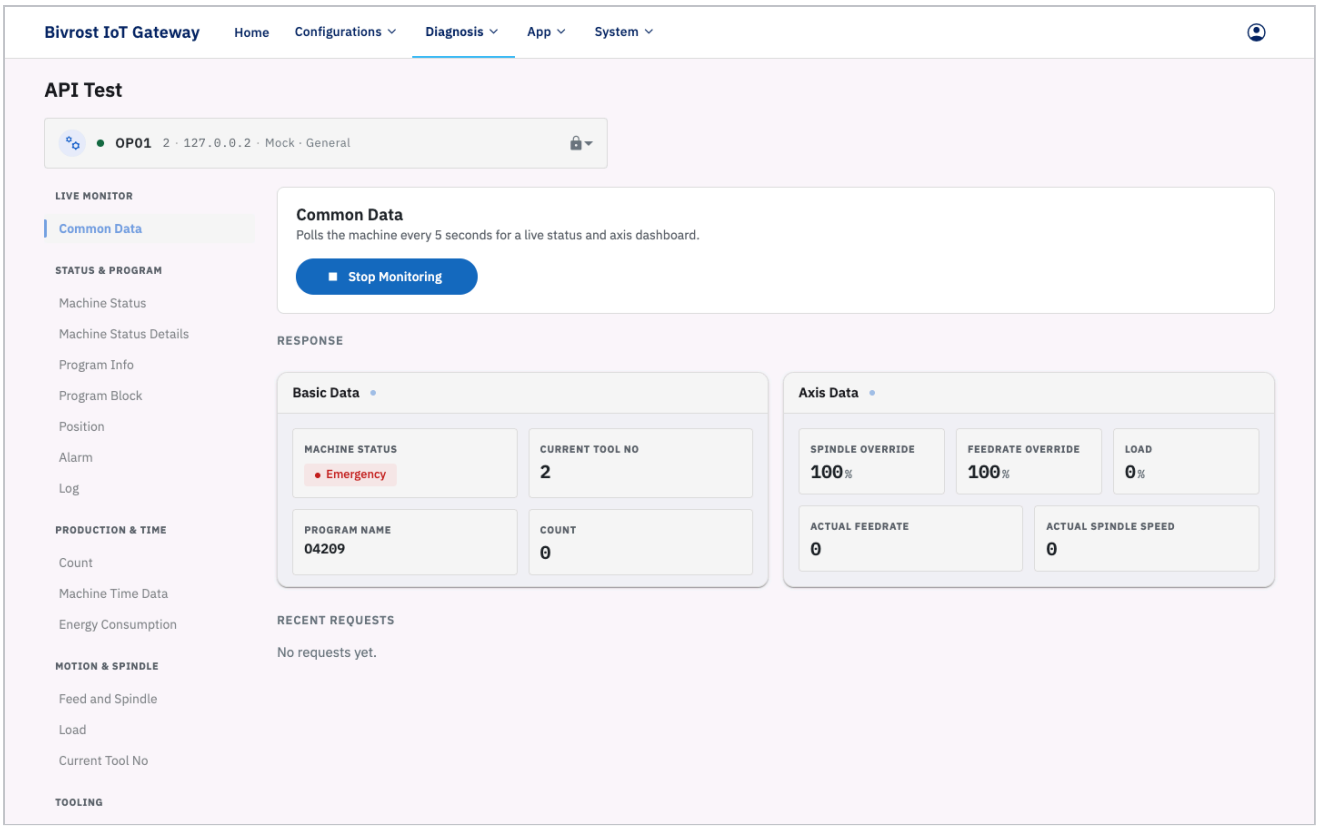




3.8.1. Common Data Interface

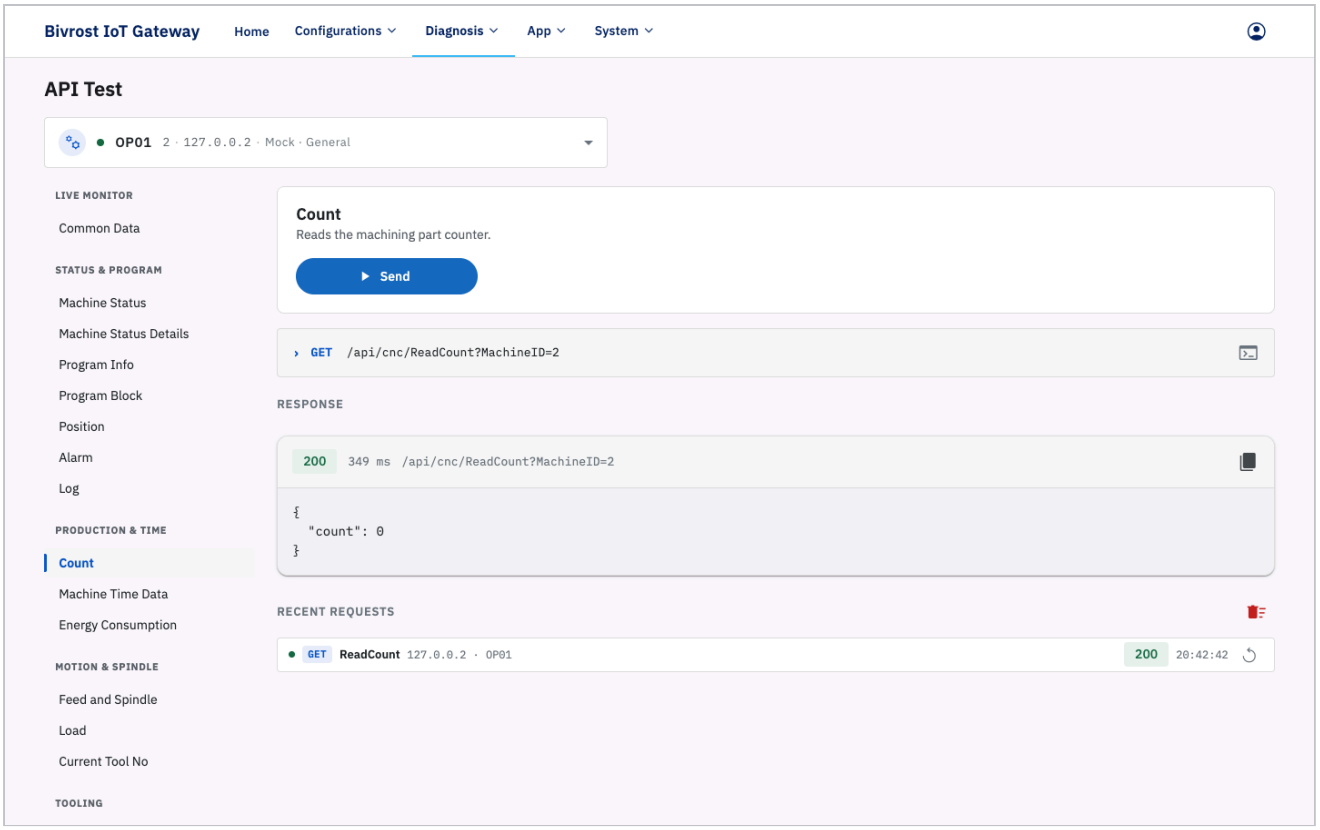
Monitoring common data polls a set of frequently used interfaces at a regular interval (roughly every 5 seconds) — Machine Status, Program Info, Current Tool No, Count, Alarm and so on —

and refreshes the data as shown below. Click **Stop Monitoring** to leave monitoring.



3.8.2. Read Data Interfaces

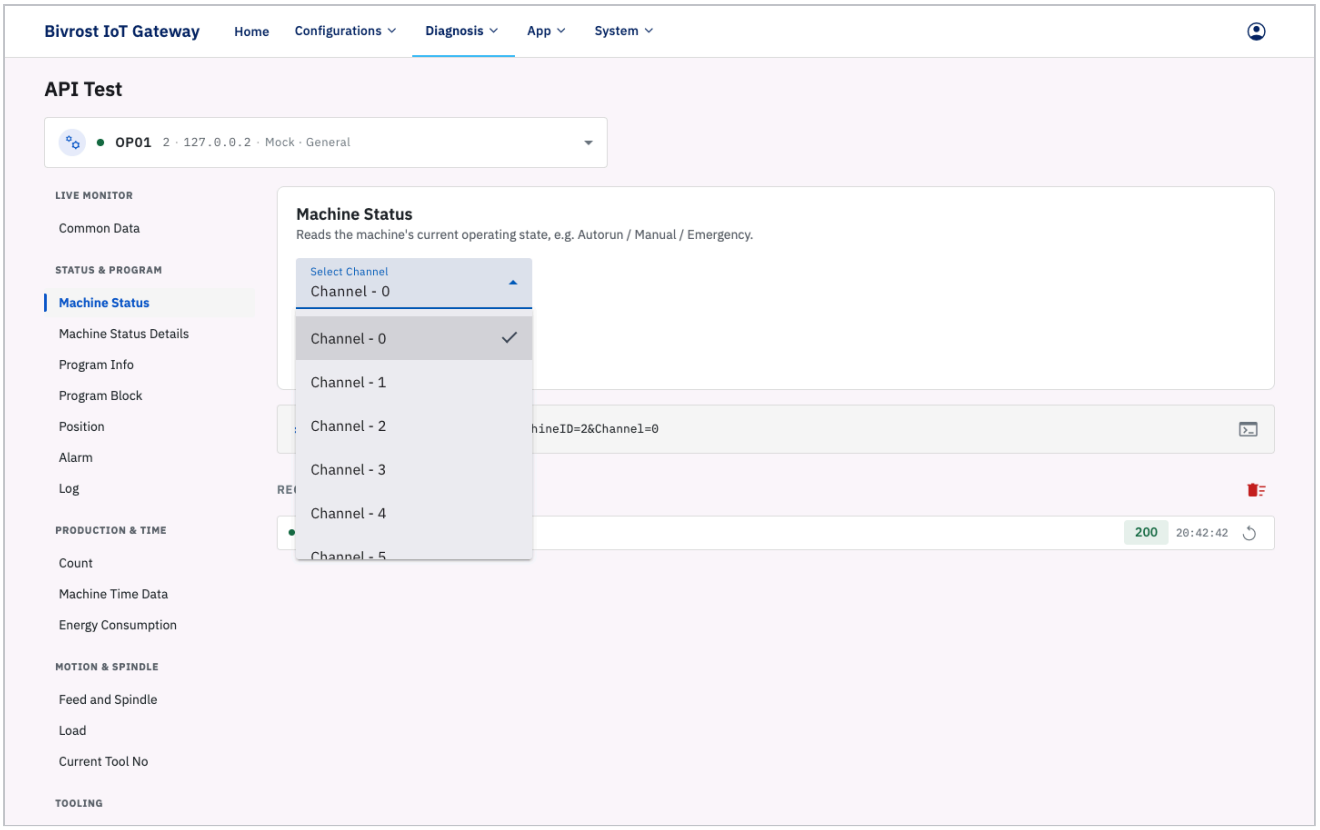
Apart from the **Common Data** interface and the **Write PLC Data** and **Write Offset Data** interfaces, every interface is a read interface. Send a request on a read interface — the **Count** interface, for example — and you get the result shown below. The **Count** interface returns the machine’s part count. The line below that begins with `/api/cnc/` is the HTTP interface address. The result is inside `{}`, where “count” is the name and 2806 is the value for that name. For a detailed description of each interface, see the corresponding entry under 5.5.1. Direct Read/Write.



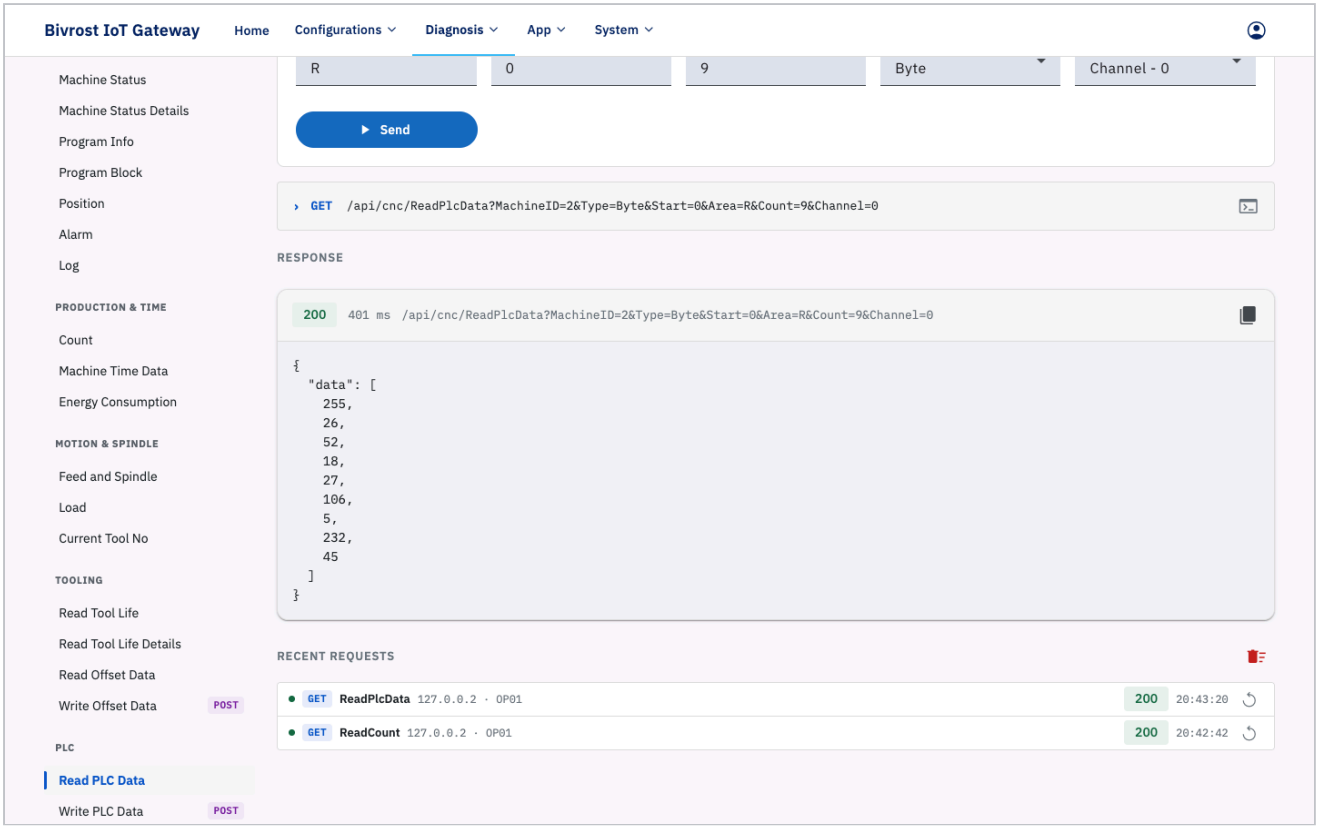

```
/api/cnc/ReadCount?MachineID=2118

{
  "count": 2806
}
```

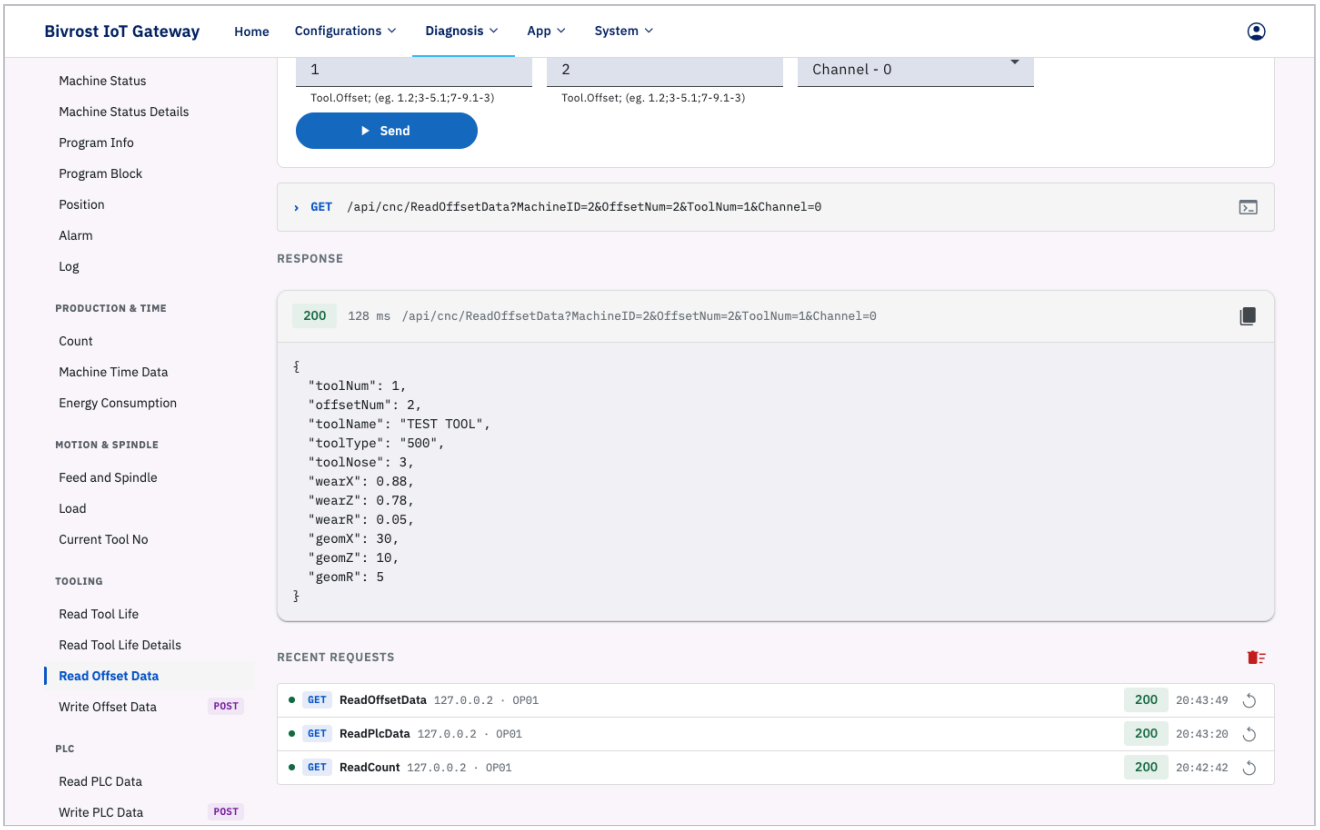
Some interfaces support multi-channel machines and can return data for a specific channel number. The default is channel 0, the main channel. On a single-channel machine, leave it at channel 0. The example below reads the status of a specific channel through the **Machine Status** interface.



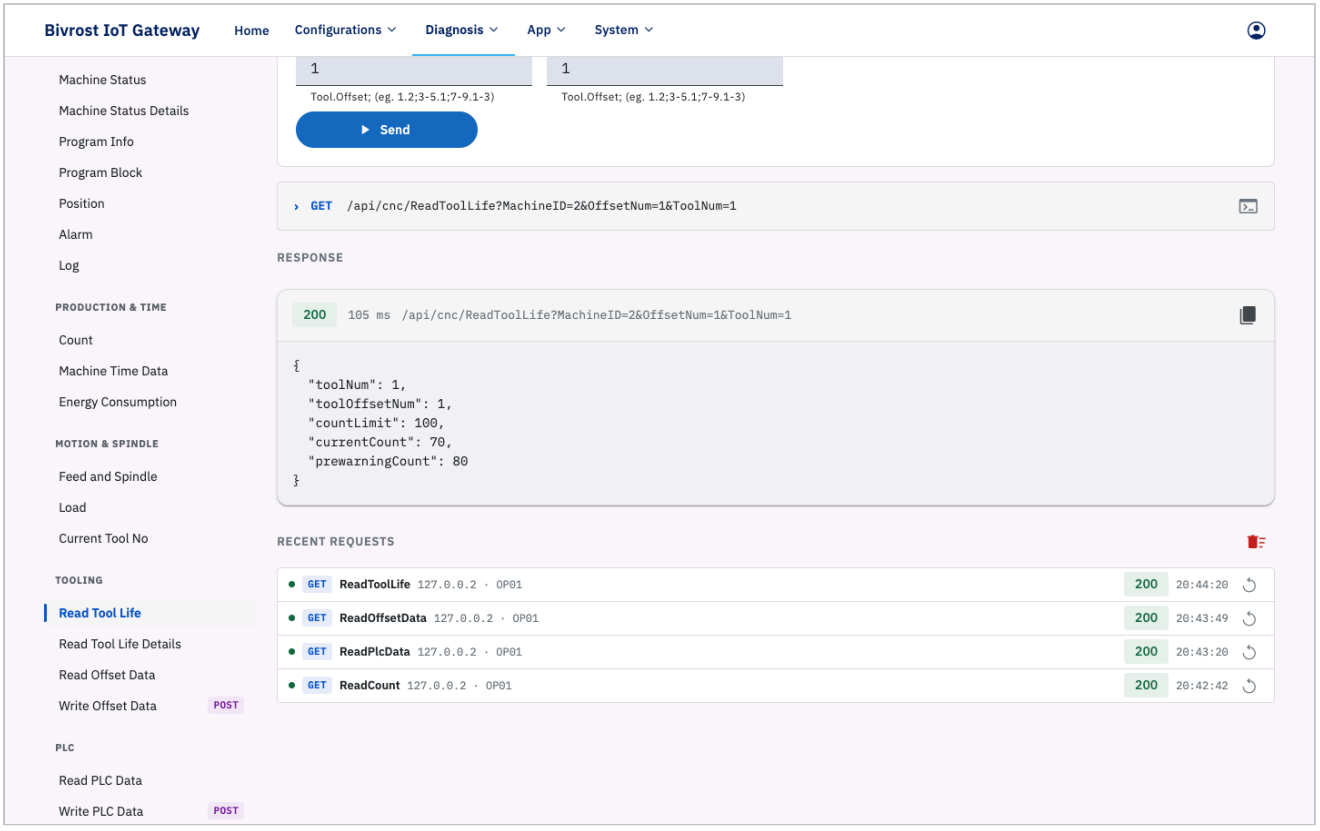
The **Read PLC Data** interface needs additional inputs such as **Area**, **Start Address**, **Data Count** and **Data Type**. See 11.2.1. PLC Data Task and 5.5.1.20. readPlcData — Read PLC Data.



The **Read Offset Data** interface needs inputs such as the **Offset Number**; see 5.5.1.14. readOffsetData — Read Offset Data.



The **Read Tool Life** and **Read Tool Life Details** interfaces need different inputs depending on the machine’s controller model. In this example the controller is a Fanuc, so the tool’s **Group Number** and **Tool Index** within that group are required; see 5.5.1.23. readToolLife — Read Tool Life.

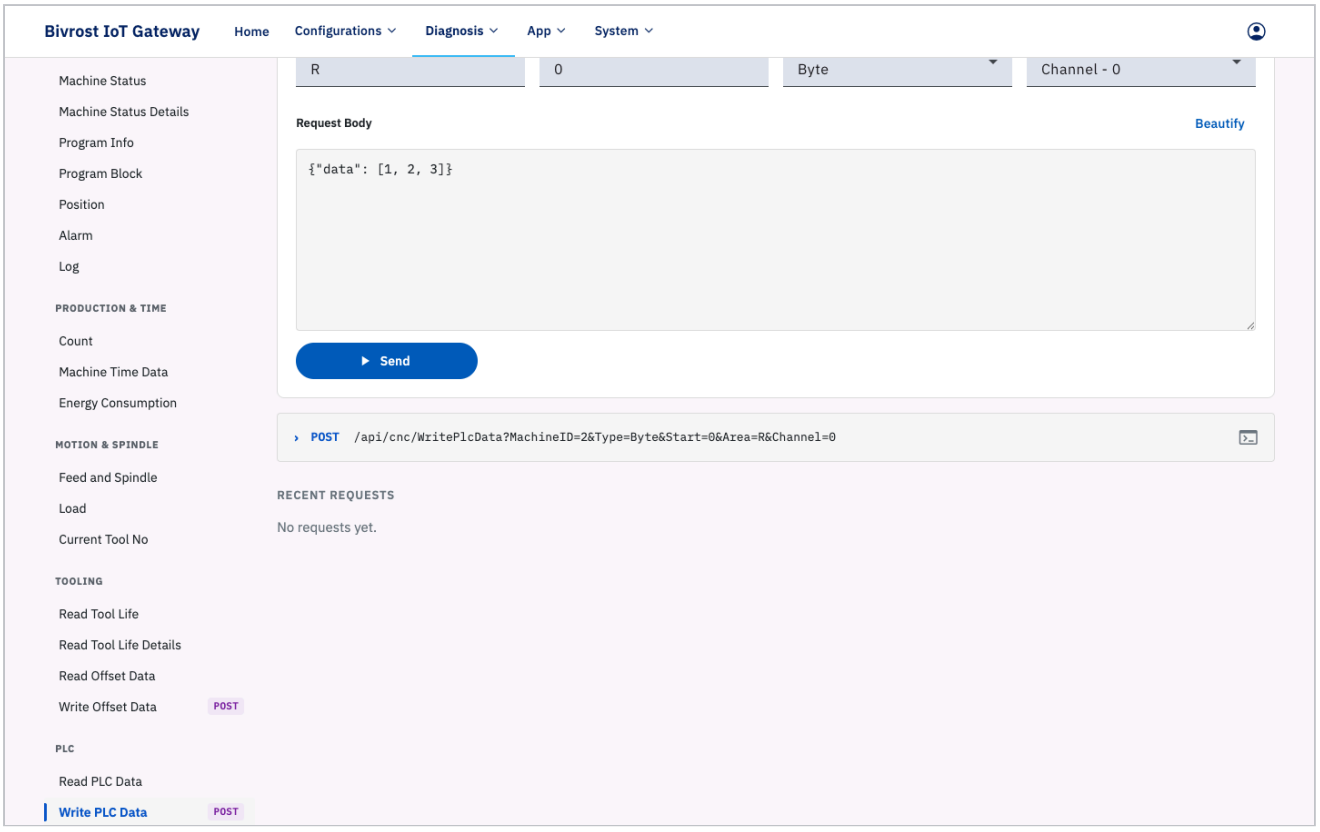


3.8.3. Write Data Interfaces

The write interfaces are **Write PLC Data** and **Write Offset Data**. Like the **Read PLC Data** and **Read Offset Data** interfaces, they need inputs such as **Area**, **Start Address**, **Data Type** and **Offset Number**, and in addition you must supply the data to be written in the request body. See 5.5.1.21. writePlcData — Write PLC Data and 5.5.1.16. writeOffsetData — Write Offset Data. Click **Beautify** to reformat the request body into standard JSON.

Example request body for writing PLC data:

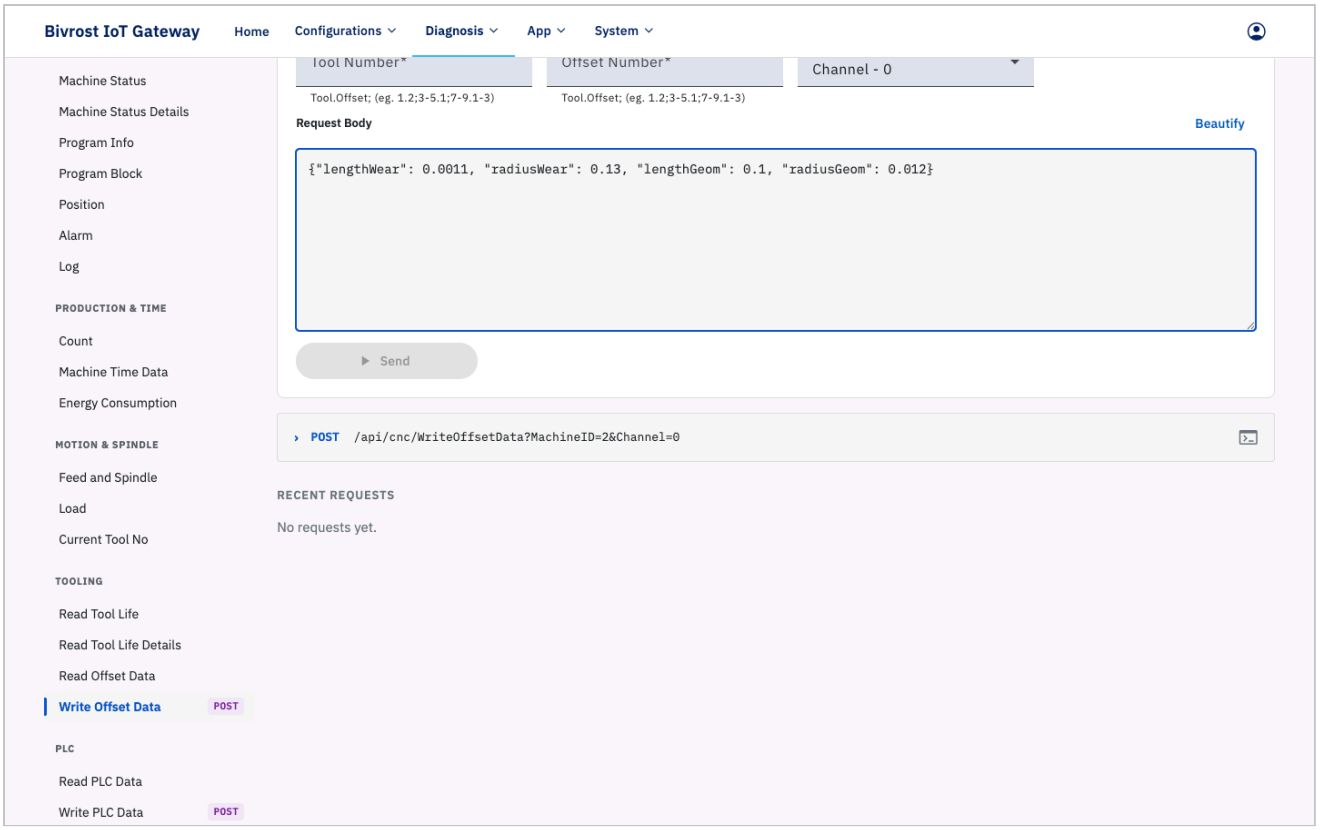
```
{
  "data": [
    1,
    2,
    3
  ]
}
```



Example request body for writing offset data:

```
{
  "lengthWear": 0.0011,
```

```
  "radiusWear": 0.13,
  "lengthGeom": 0.1,
  "radiusGeom": 0.012
}
```

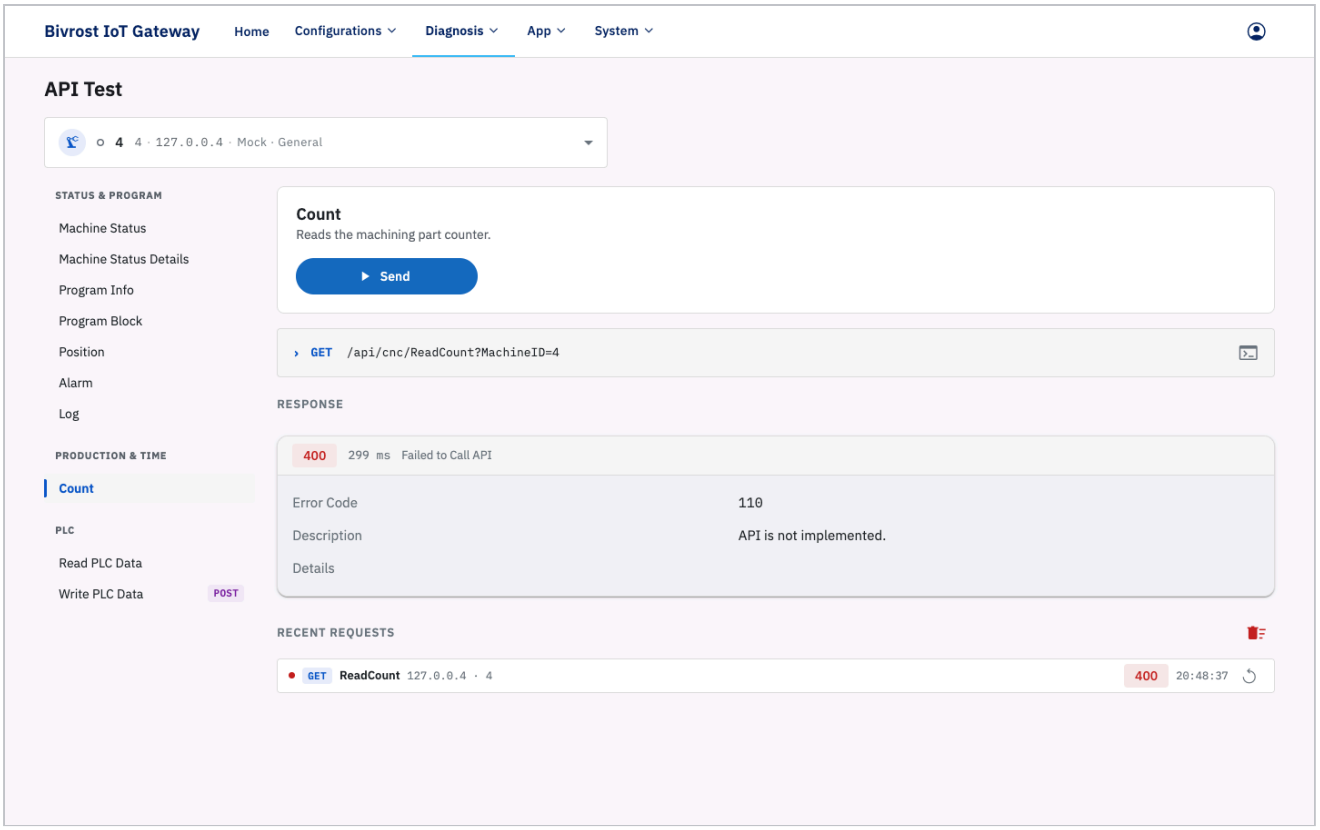


A successful write returns an error code (errorCode, where 0 means success) and an error message (errorMsg); see 5.2. Error Handling.

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

3.8.4. Troubleshooting Test Failures

A failed API test returns **Failed to Call API**, as shown below. The response gives an **Error Code**, a **Description**, **Details** and a suggested **Remedy**.



If an API test fails, check the following:

1. The settings on the **Machines** page;
2. Whether you clicked **Restart Service** on **Home** after changing anything on the **Machines** or **Tasks** page;
3. Whether the network settings on the machine are correct — on some machines the settings only take effect after the power is switched off for three seconds and back on;
4. Whether the hardware — network cable connections, switches and so on — is properly connected;
5. Whether the machine’s IP address is already in use. When office PCs or other devices share the same LAN, the machine’s IP may be taken, which stops the machine and the gateway from communicating. Try giving the machine an IP address that is not in use. Note: an IP address that does not answer a ping is not necessarily free — some devices disable ping replies for security reasons.
6. Another form of address conflict is a subnet conflict. This is common when the data-acquisition network and another internal network share a switch and both use the same subnet. When this happens, move either the conflicting internal devices or the data-acquisition network to a subnet that is not in use.

3.9. DNC (Program Transfer)

This page lets you list, upload, download, delete and back up machine program files and send files to several machines at once, and it has a built-in toolpath viewer. Use it to verify a setup before it goes live, or as a lightweight file manager for your machines.

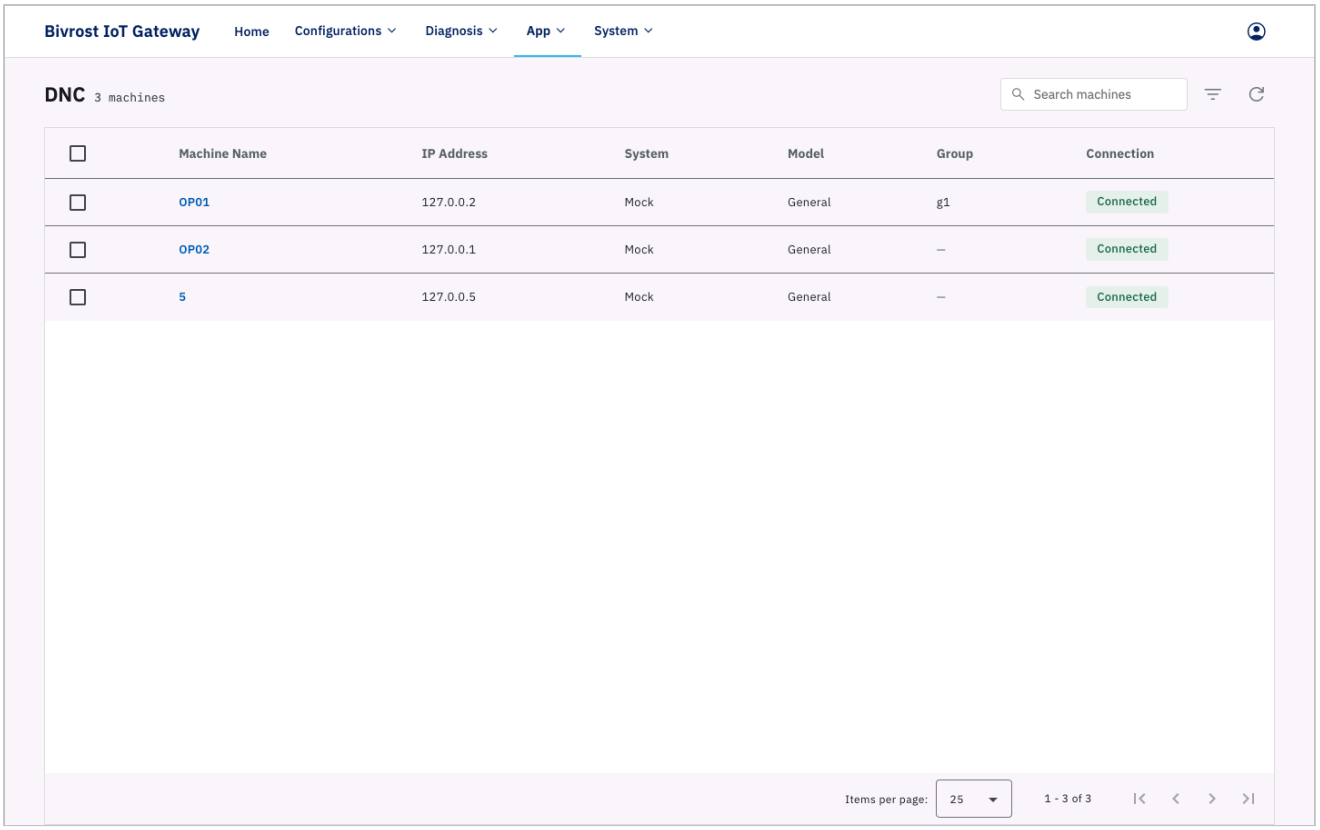
If you have changed any **Machines**, **Groups**, **Tasks** or **Communication** settings, click **Restart Service** on the Home page before transferring programs.

> **Caution**
>
> - A file sent from your computer **cannot overwrite an existing machine file directly**. If you send a file whose name already exists, the gateway asks you to confirm the replacement.
> - Some machines support file locking, write protection and similar features. Once these are enabled, files on the machine cannot be deleted.
> - **A program that is in use cannot be deleted.**

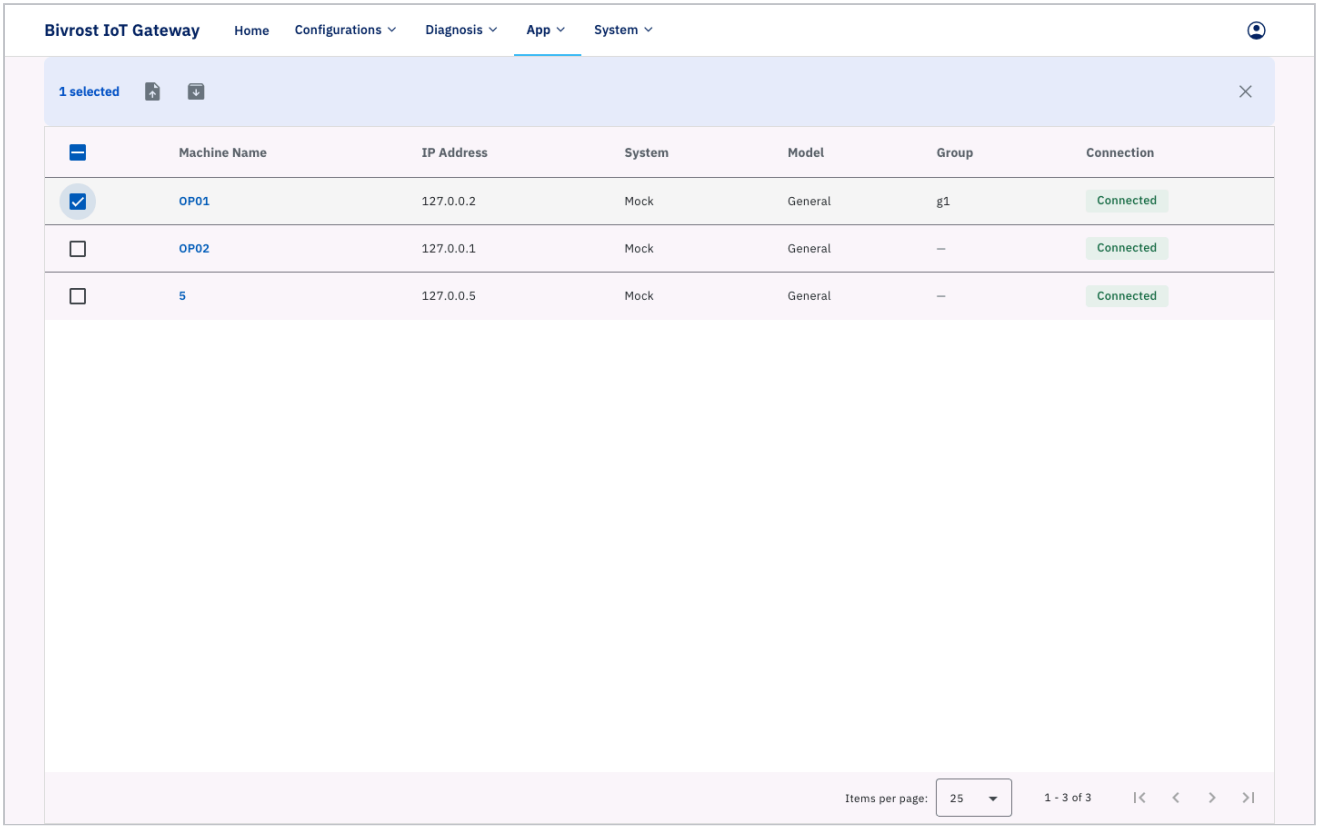
Program transfer uses the file-transfer parts of the HTTP protocol. For details, see 2.6, File Management Interfaces,.

Machine Directory

The **DNC** page opens on the machine directory: every machine that supports program transfer, together with its IP address, system model, group and connection status. DNC is unavailable for offline machines.



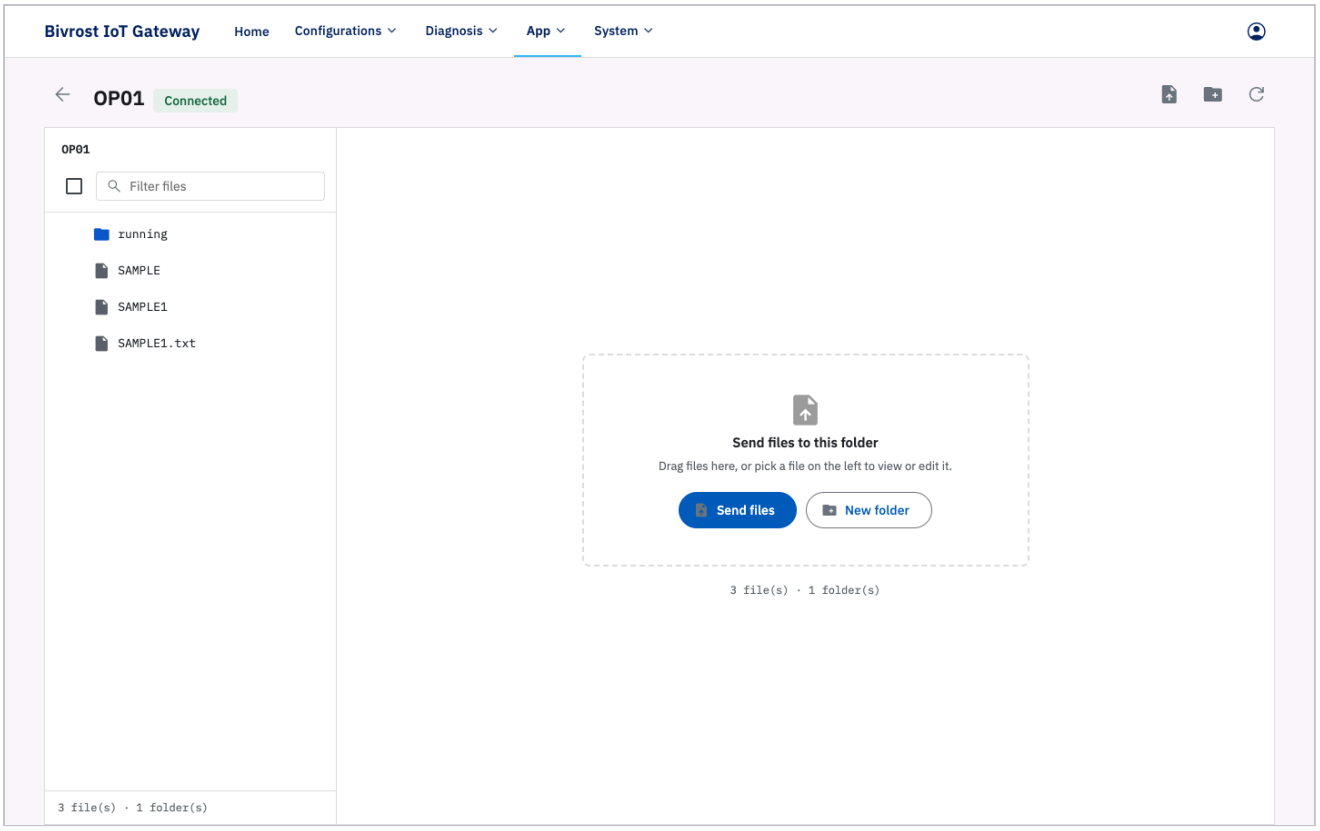
Select one or more machines and the Fleet actions bar appears at the top of the page, offering **Send files** and **Download machine files (backup)** for the selected machines.



Click a machine name to open that machine’s file browser.

3.9.1. Reading a Machine’s File List

The left-hand panel of the file browser lists the folders and files under the machine’s root directory (configured in 3.3.1.3. DNC Settings). Click a folder to open it; use the breadcrumb above the list to go back up. A filter box above the list narrows the view by file name.

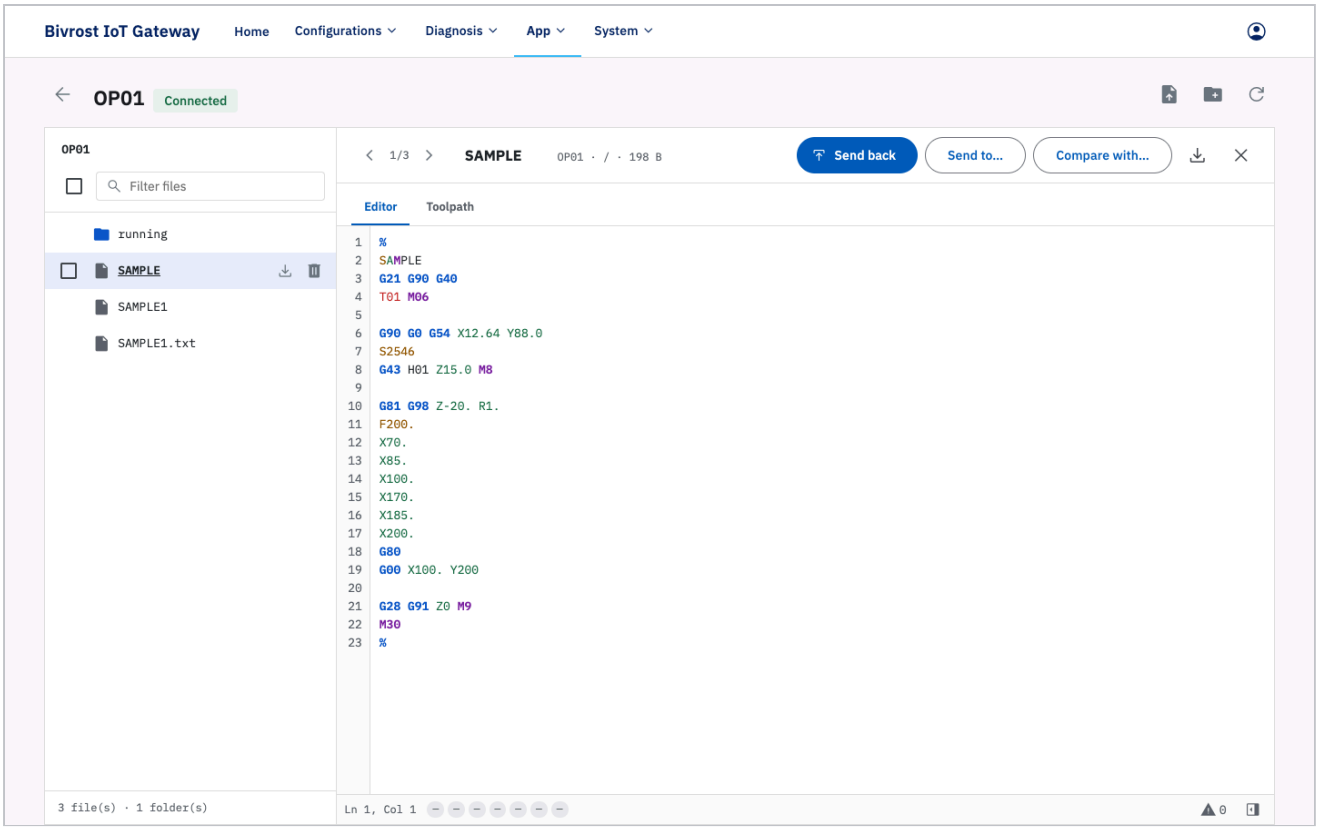


On Mitsubishi controls, a target path holding a large number of files can take a while to list.

If the file list does not respond for a long time and no error appears, go to the API Test page and use the Common Data interface to check that the gateway can talk to the machine.

3.9.2. Receiving a Machine File to Your Computer

Click a file in the left-hand list and the gateway receives it from the machine and shows its contents in the preview pane on the right. The ‹ › buttons at the top left of the preview pane step through the files.



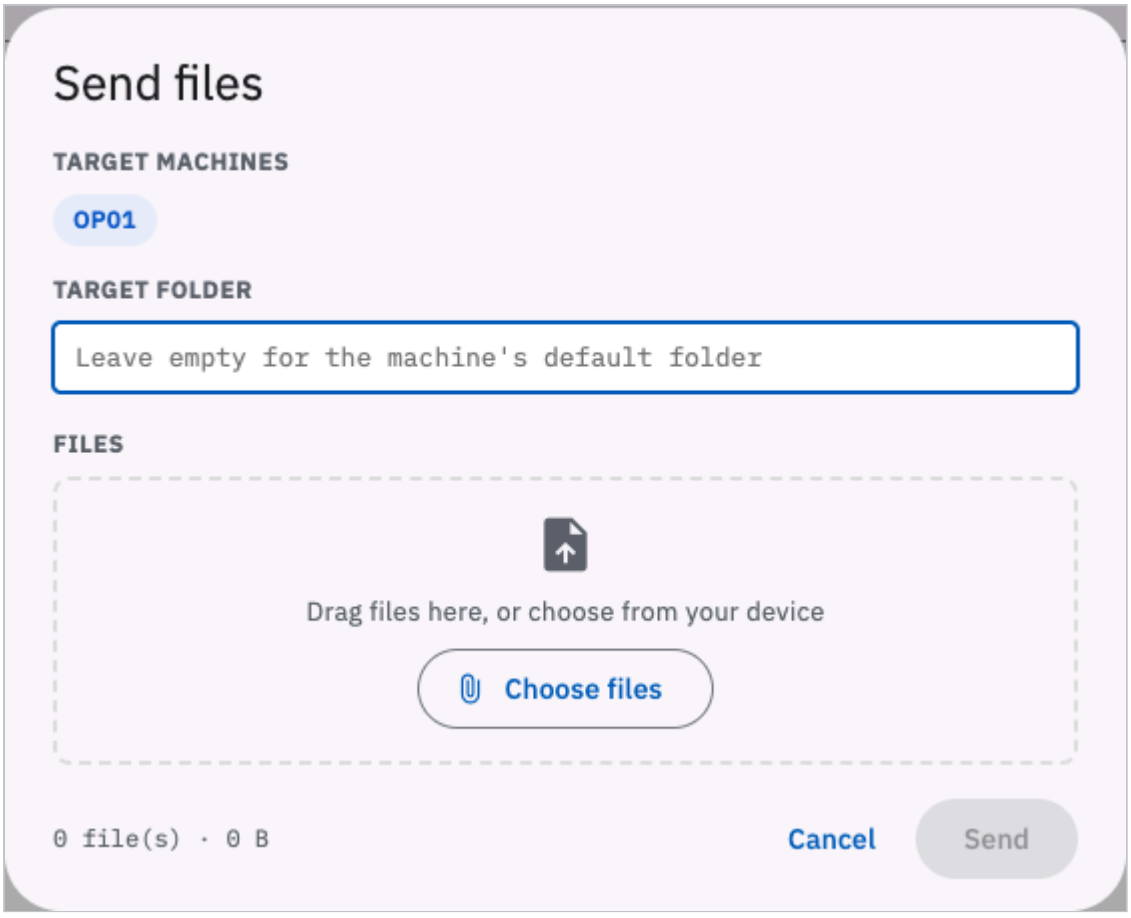
Click the **Download** icon at the top right of the preview pane to save the received file through your browser. The file lands in your browser’s default download folder. You can also open your browser’s download settings to change that folder, or have the browser ask you where to save each download. Alternatively, hover over a file in the list and click the **Download** icon on the right of the row to download it directly.

Binary files cannot be shown in the preview pane, but they can still be downloaded.

3.9.3. Sending a Local File to a Machine

Click the **Upload files** button at the top right of the page (or select machines in the machine directory and click **Send files**) to open the “Send files” dialog:

1. Confirm the target machine and the target folder (leave the folder empty for the machine’s default folder);
2. Drag files into the dialog, or click **Choose files** to pick them from your computer — several files can be selected at once;
3. Click **Send**. The files are sent to the target folder on the machine.



After editing a file in the preview pane, you can also click **Send back** to return the edited contents to the current machine, or **Send to...** to send them to other machines.

If the connection to the machine drops partway through a send, that file shows error code 132 (sending the CNC file was interrupted; the CNC may keep a truncated copy), and the gateway then refreshes the machine’s directory to show the truncated program. Click **Retry failed** to send it again and overwrite it; if you are not resending, delete it from the machine directory by hand so it is not run by mistake.

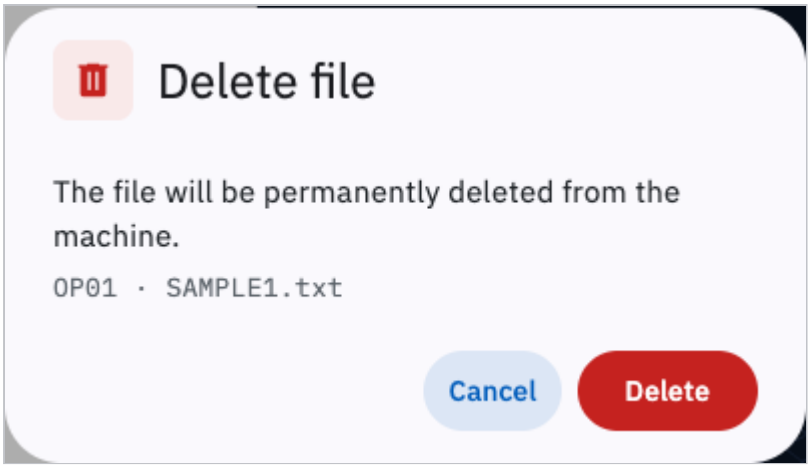
Caution for Fanuc Systems

When you send a file to a Fanuc control, the machine automatically takes the first line of the file’s contents as the file name, not the name the file had on your computer. You must therefore change the first line to the target file name before sending. With the contents below, the file is named O8003 on the machine after a successful send:

```
%
O8003
T1M06
...
```

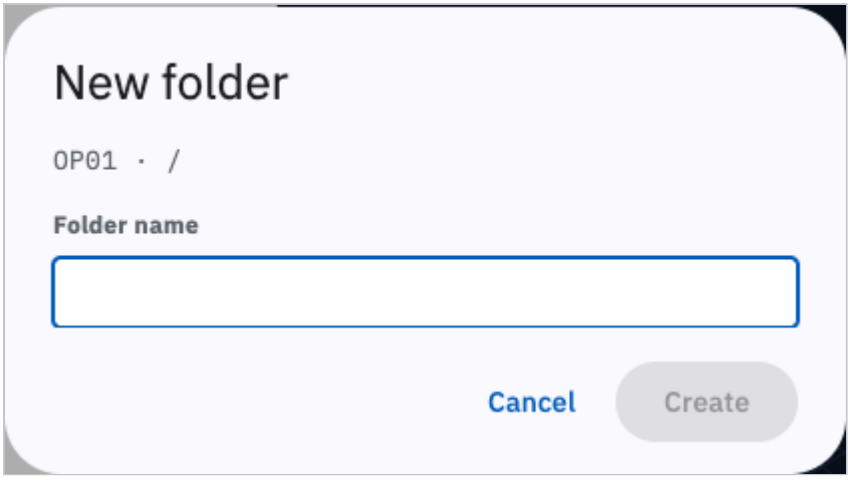
3.9.4. Deleting a Machine File

In the left-hand file list, hover over a file, click the **Delete** icon on the right of the row and confirm in the dialog that appears. The file is then deleted.



3.9.5. Creating a Machine Folder

Click the **New folder** icon at the top right of the page to open the new folder dialog, enter a folder name and click **Create**. The folder is created inside the current folder.



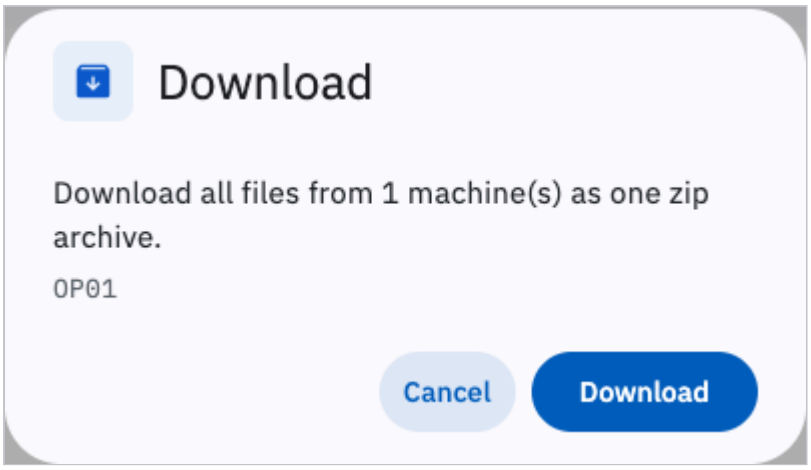
3.9.6. Deleting a Machine Folder

In the left-hand file list, hover over a folder, click the **Delete** icon on the right of the row and confirm in the delete folder dialog. Only empty folders can be deleted at present, so delete all files and subfolders inside it first.

3.9.7. Backing Up Machine Files

This function packs the files of the selected machines — including subfolders and their contents — into a single ZIP archive for you to download. Inside the archive, the first level is a folder named "BackupFiles_" followed by the backup date and time; the second level is one folder per machine root directory, named "machineID_machineName"; below that are each machine's files and folders, with the original folder structure preserved.

In the machine directory, select the machines to back up (tick **All machines** in the table header to select every machine), click **Download machine files (backup)** in the Fleet actions bar, then click **Download** in the confirmation dialog to start the backup. Once the archive is ready, it can be downloaded to your computer.



> **Note**
>
> Backing up a folder that holds a large number of files takes a long time. Back up only the machines you need.

3.9.8. Batch Sending Files

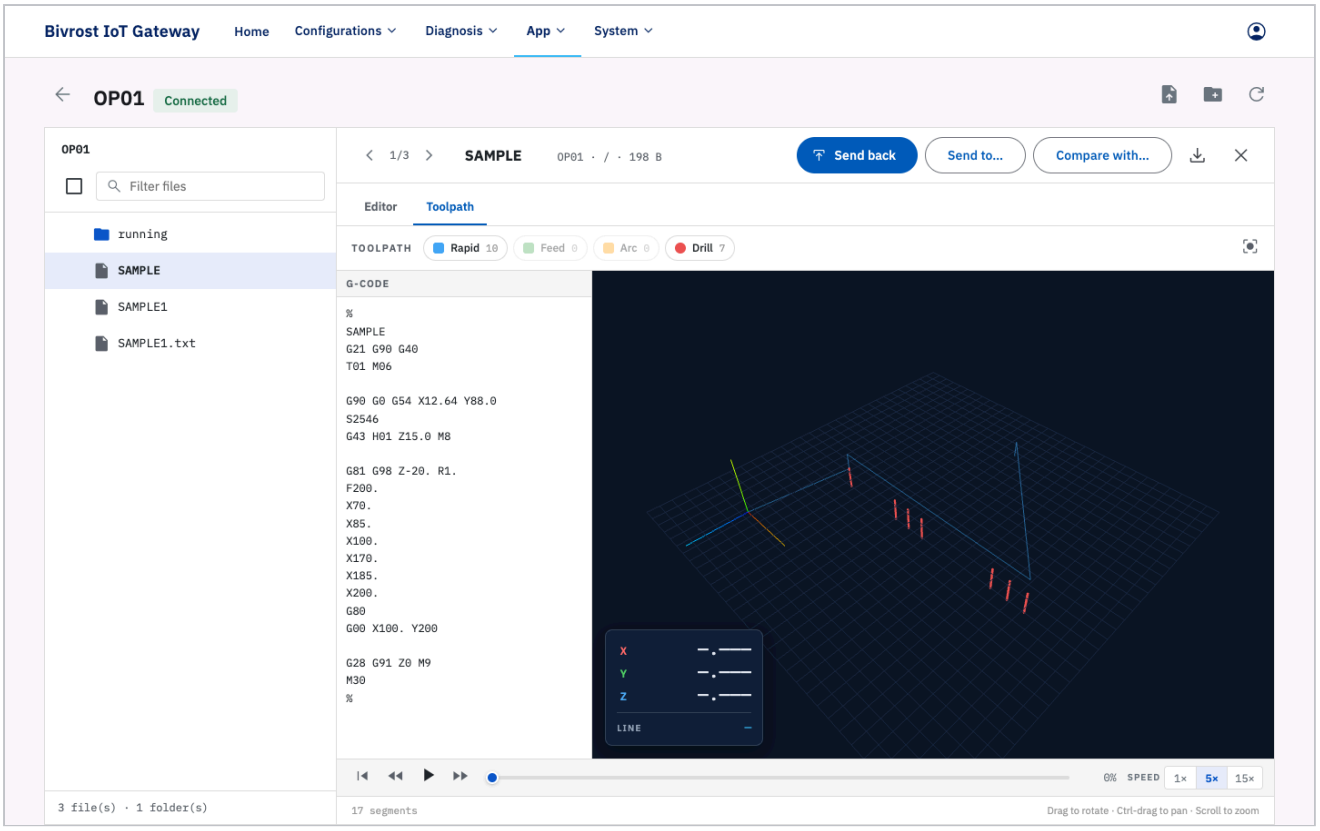
Select several machines in the machine directory and click **Send files** to send the same set of files to a chosen folder on all of them in one go. In the "Send files" dialog you can click **Choose files** repeatedly to add files from different locations; confirm the target folder and click **Send**.

The send runs as a single batch and the gateway transfers to each machine in turn. If a file of the same name already exists at the destination, you are asked to confirm the replacement.

3.9.9. Viewing the Toolpath

Two tabs, **Editor** and **Toolpath**, sit above the file preview pane. Once a G-code file is loaded (received from the machine — see 3.9.2. Receiving a Machine File to Your Computer), switch to the **Toolpath** tab to open the toolpath viewer.

The viewer shows the G-code on the left and an animation of the corresponding toolpath on the right, with X/Y/Z position readouts (DRO) and the current line. The legend above distinguishes toolpath types such as Rapid, Feed, Arc and Drill. Drag to rotate the view, scroll to zoom and adjust the playback speed to study the toolpath directly. If needed, edit the G-code in the code pane on the left and the toolpath on the right updates immediately.

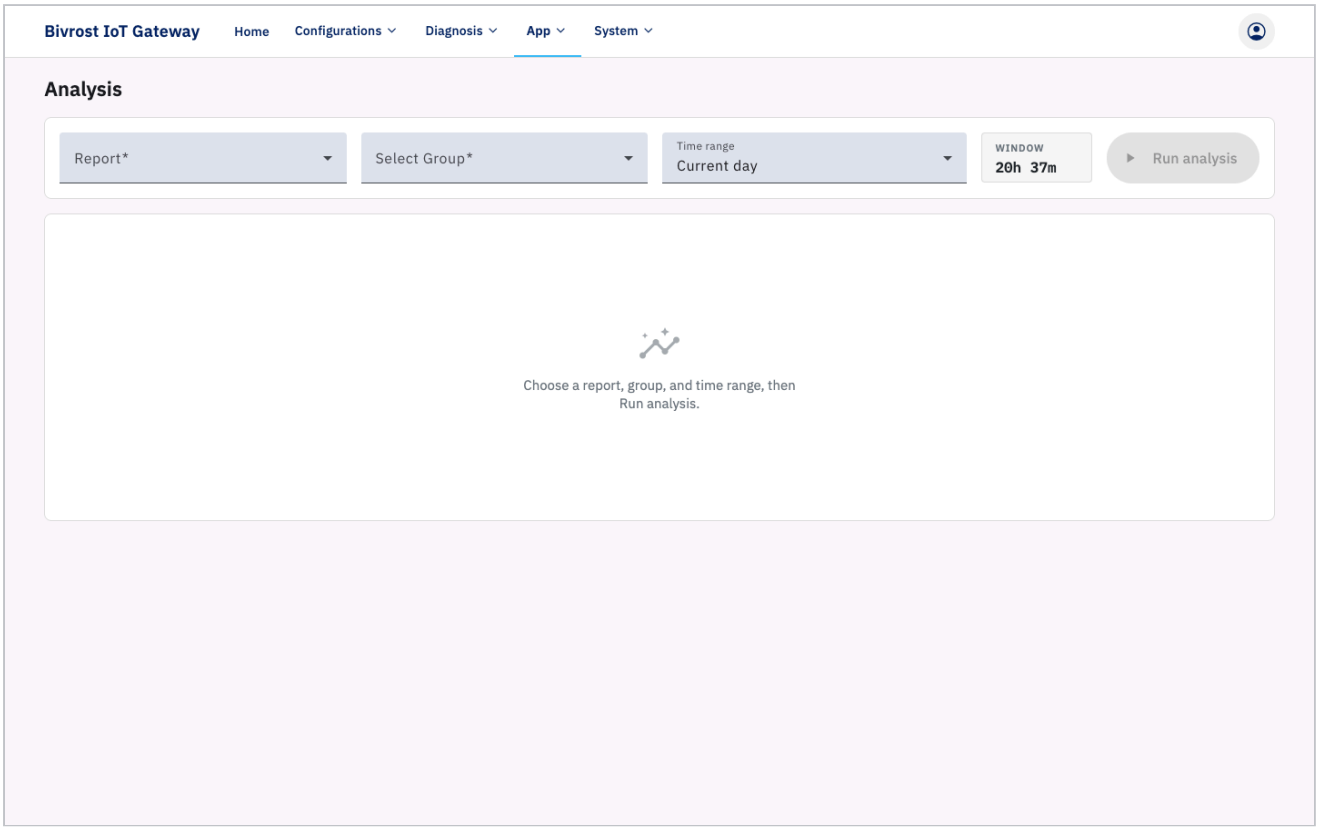


3.10. Analysis

The Analysis page extracts the machine and group data held in the gateway’s local storage and builds a statistics table for a given group, over a given period of time, broken down by grouping interval.

Points to note when using Analysis

- 3.12.4.1. Local Caching must be enabled first, so that machine status history can be kept on the gateway itself. Local history is retained for at most 365 days, so no time option may be set earlier than 365 days before the current time.
- The difference between the start time and the end time (the time range) may not exceed 31 days.
- Machines and groups should already be configured.
- If the end of the time range is later than the current time, the current time is used as the end time.
- If part of the time range contains no machine status data, the status for that gap defaults to **Offline**. Besides the machine actually being off or offline, machine status can be missing for other reasons: local caching was not enabled on the gateway, the gateway lost power, the gateway lost network connectivity, and so on.

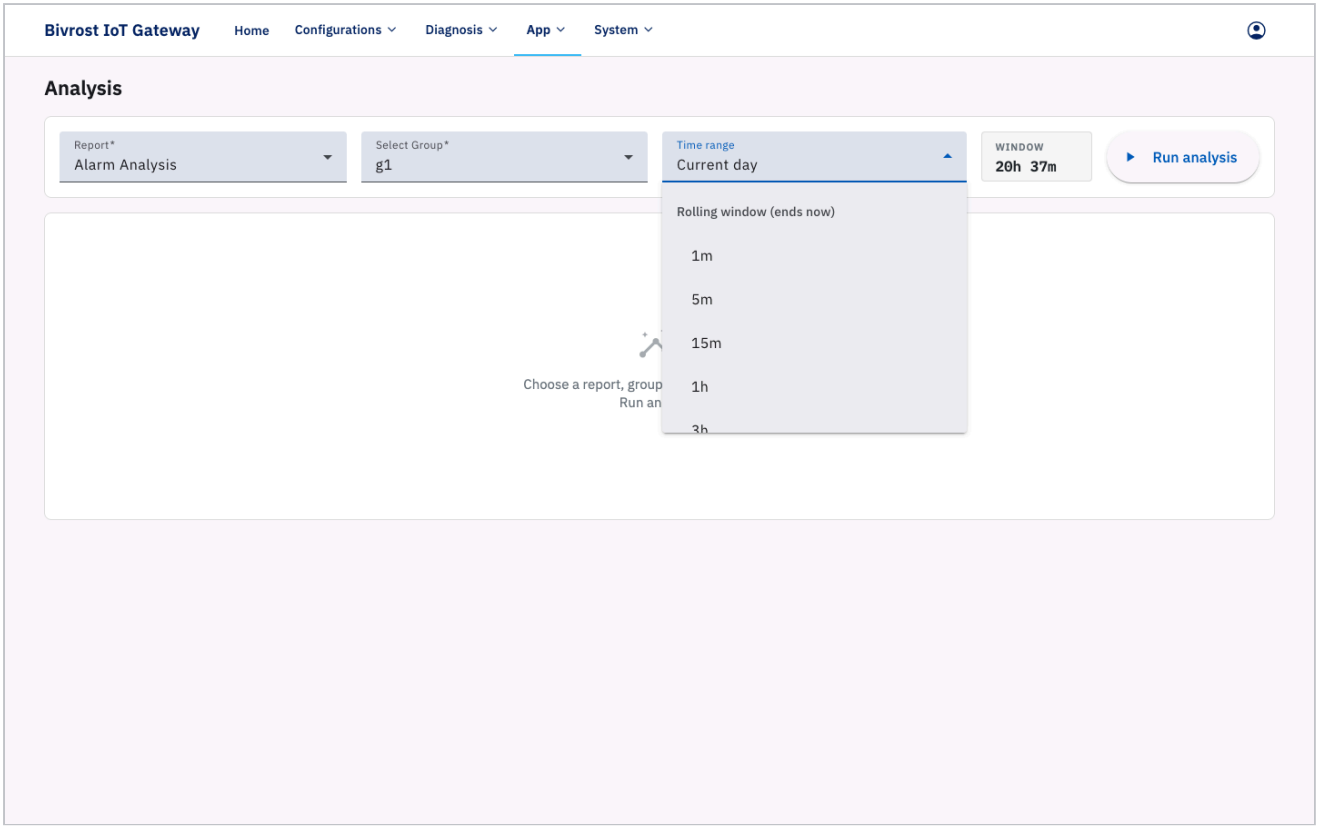


To run an analysis, select an analyzer, select a group, set the time range, and click **Run analysis** to get the result.

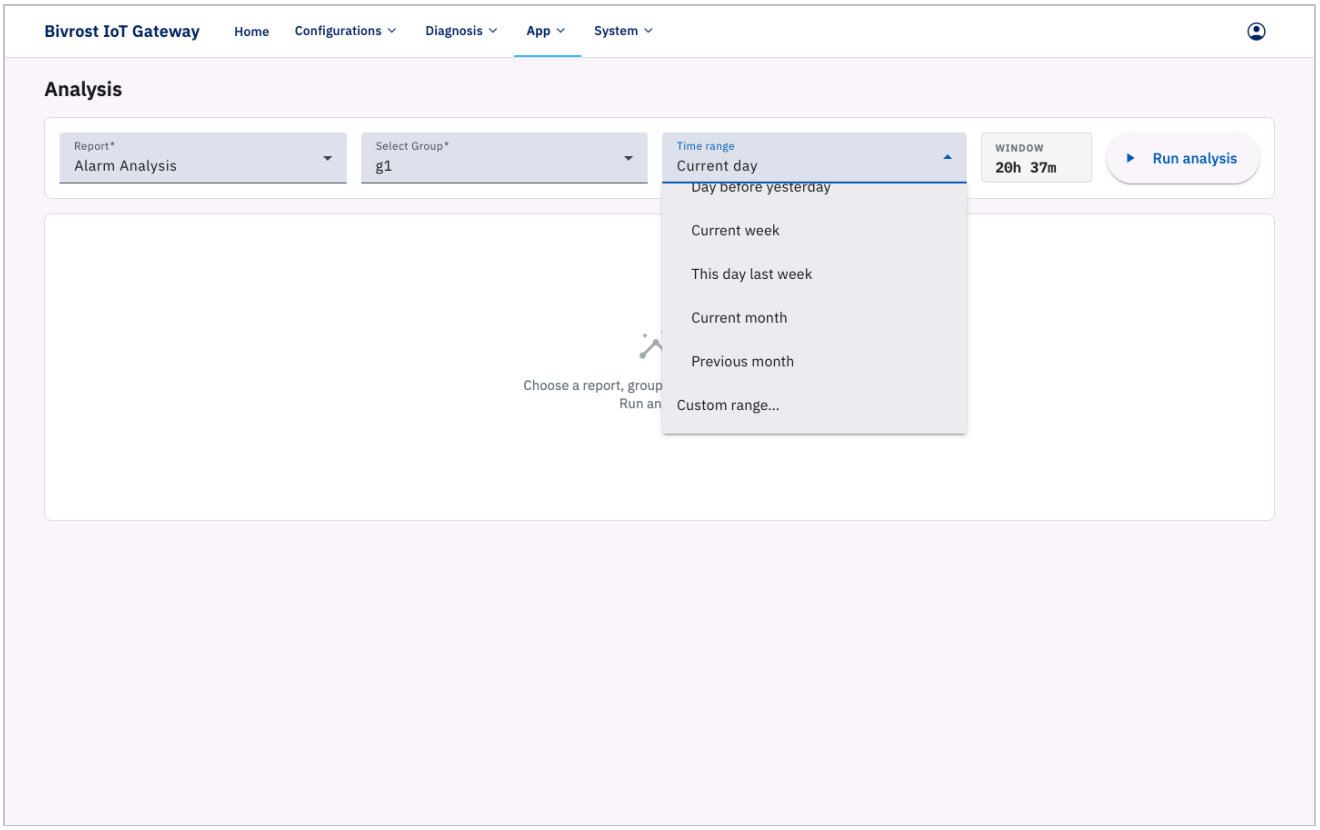
- **Analyzer options**: Alarm Analysis, Count Analysis, Cycle Analysis, OEE Analysis, Overall Analysis, and so on.
- **Group options**: the groups activated on the **Groups** page.

Once the analyzer and the group are selected, the **Time range** list offers three ways to set the period.

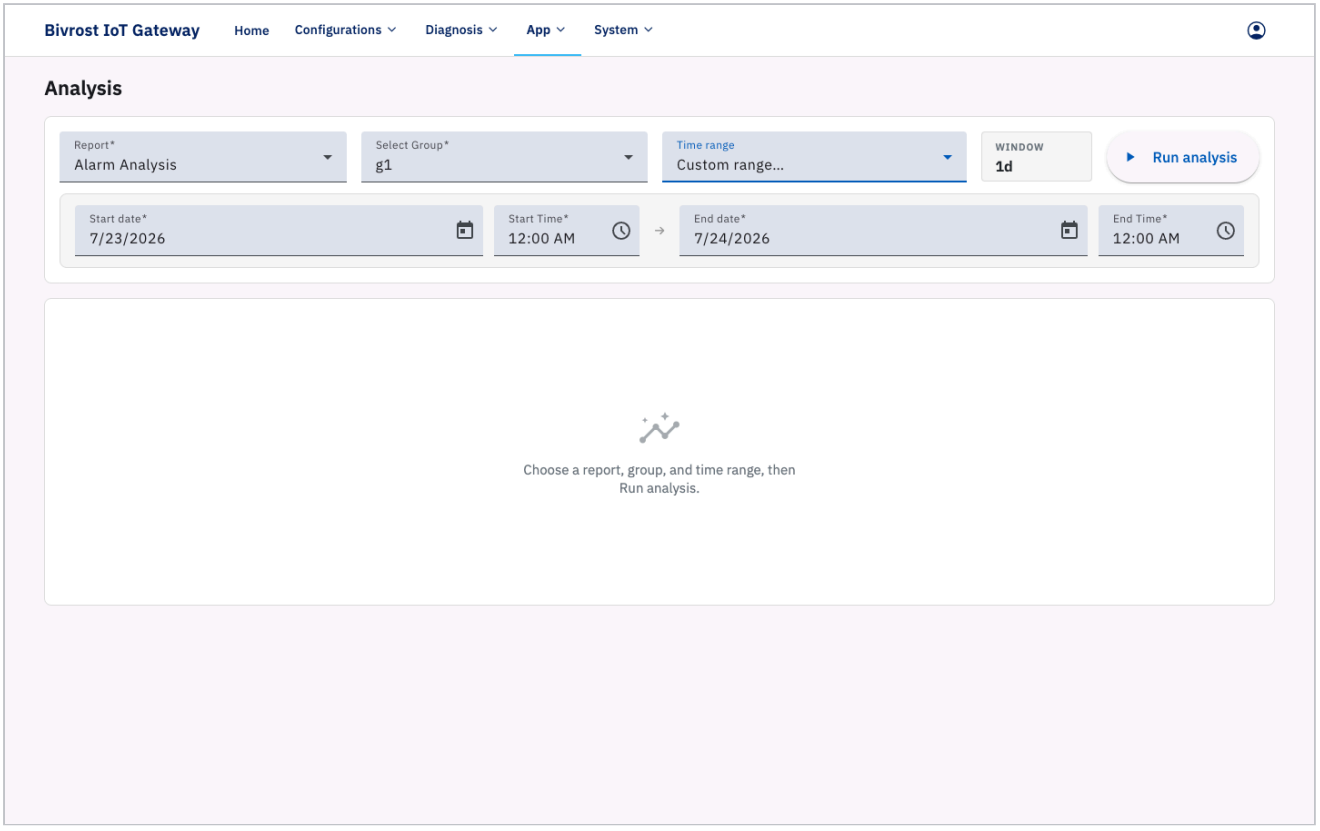
Rolling window (ends now): a period that ends at the current time, chosen from preset lengths such as 5m, 1h, 1d and 30d. Choosing 30d, for example, runs the analysis from 30 days before the current time up to the current time.



Calendar periods: options relative to the present moment, such as Current day, Current week, This day last week and Current month. Current week, for example, sets the start time to midnight on Monday of the current week.



Custom range...: enter the exact start and end date and time by hand.

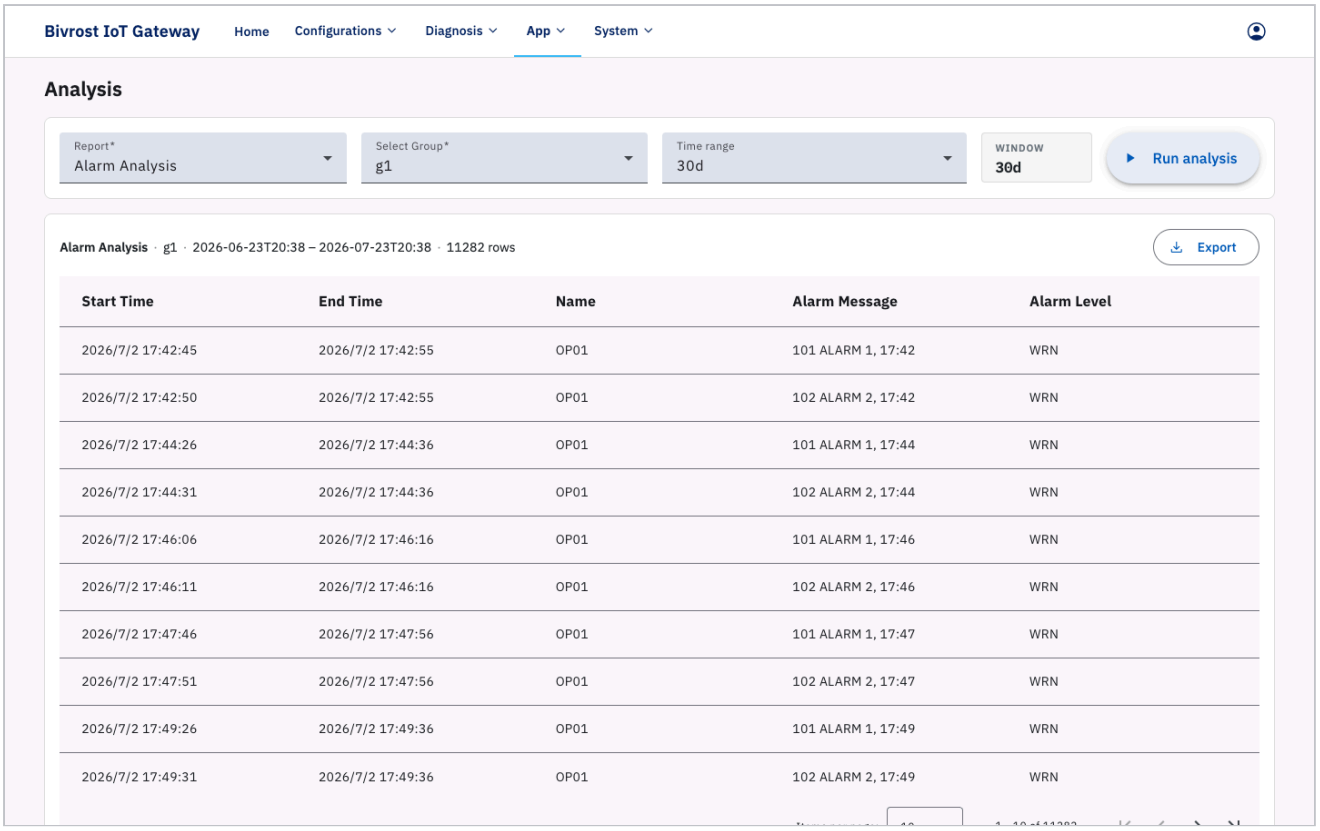


> **Note**
>
> If the end of the time range is later than the current time, it is automatically corrected to the current time.

The options specific to each analyzer, and the results they produce, are described below.

3.10.1. Alarm Analysis

The figure below shows the alarm analysis table for every machine in the selected group, produced by choosing Alarm Analysis, Group 1 and the last 30 days and clicking **Run analysis**. It lists the start and end time of each alarm, its message and its level.

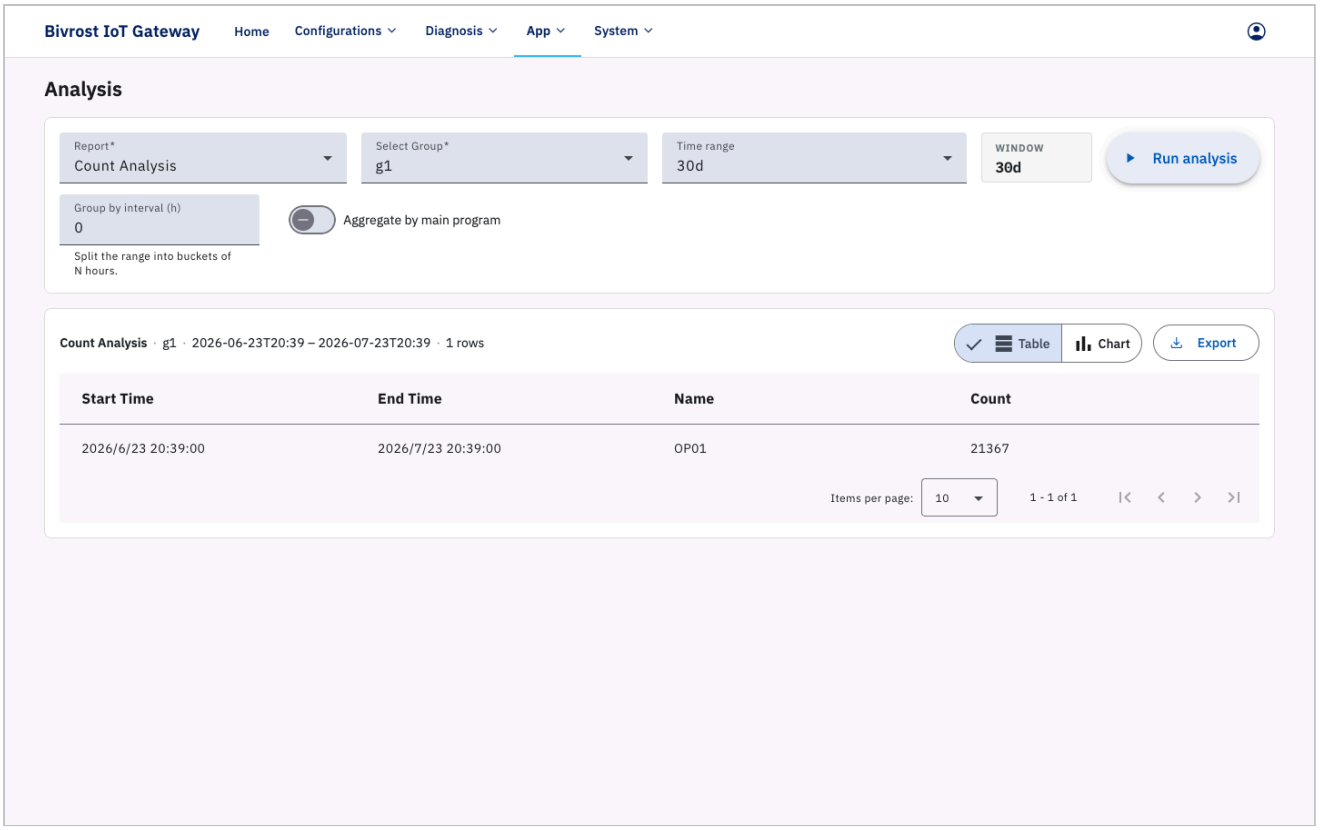


Start Time	End Time	Name	Alarm Message	Alarm Level
2026/7/2 17:42:45	2026/7/2 17:42:55	OP01	101 ALARM 1, 17:42	WRN
2026/7/2 17:42:50	2026/7/2 17:42:55	OP01	102 ALARM 2, 17:42	WRN
2026/7/2 17:44:26	2026/7/2 17:44:36	OP01	101 ALARM 1, 17:44	WRN
2026/7/2 17:44:31	2026/7/2 17:44:36	OP01	102 ALARM 2, 17:44	WRN
2026/7/2 17:46:06	2026/7/2 17:46:16	OP01	101 ALARM 1, 17:46	WRN
2026/7/2 17:46:11	2026/7/2 17:46:16	OP01	102 ALARM 2, 17:46	WRN
2026/7/2 17:47:46	2026/7/2 17:47:56	OP01	101 ALARM 1, 17:47	WRN
2026/7/2 17:47:51	2026/7/2 17:47:56	OP01	102 ALARM 2, 17:47	WRN
2026/7/2 17:49:26	2026/7/2 17:49:36	OP01	101 ALARM 1, 17:49	WRN
2026/7/2 17:49:31	2026/7/2 17:49:36	OP01	102 ALARM 2, 17:49	WRN

Once the result has been calculated, click **Export** to download it as a CSV file. The file is named automatically as `AnalyzerName-GroupName-StartTime-EndTime.csv`.

3.10.2. Count Analysis

The figure below shows the count analysis table for every machine in the selected group, produced by choosing Count Analysis, Group 1 and the last 30 days and clicking **Run analysis**. Count here means the count produced within the specified time range.



Three settings are specific to this analyzer: **Group by interval (h)**, **Aggregate by main program** and **Group total by count multiplier**.

Group by interval (h) splits the time range into buckets: starting from the start time, each interval forms one bucket, and the count is calculated separately for each bucket. The default is 0, which means no grouping — a single bucket. The example below uses a grouping interval of 24 hours, so the count shown is the count within each interval.

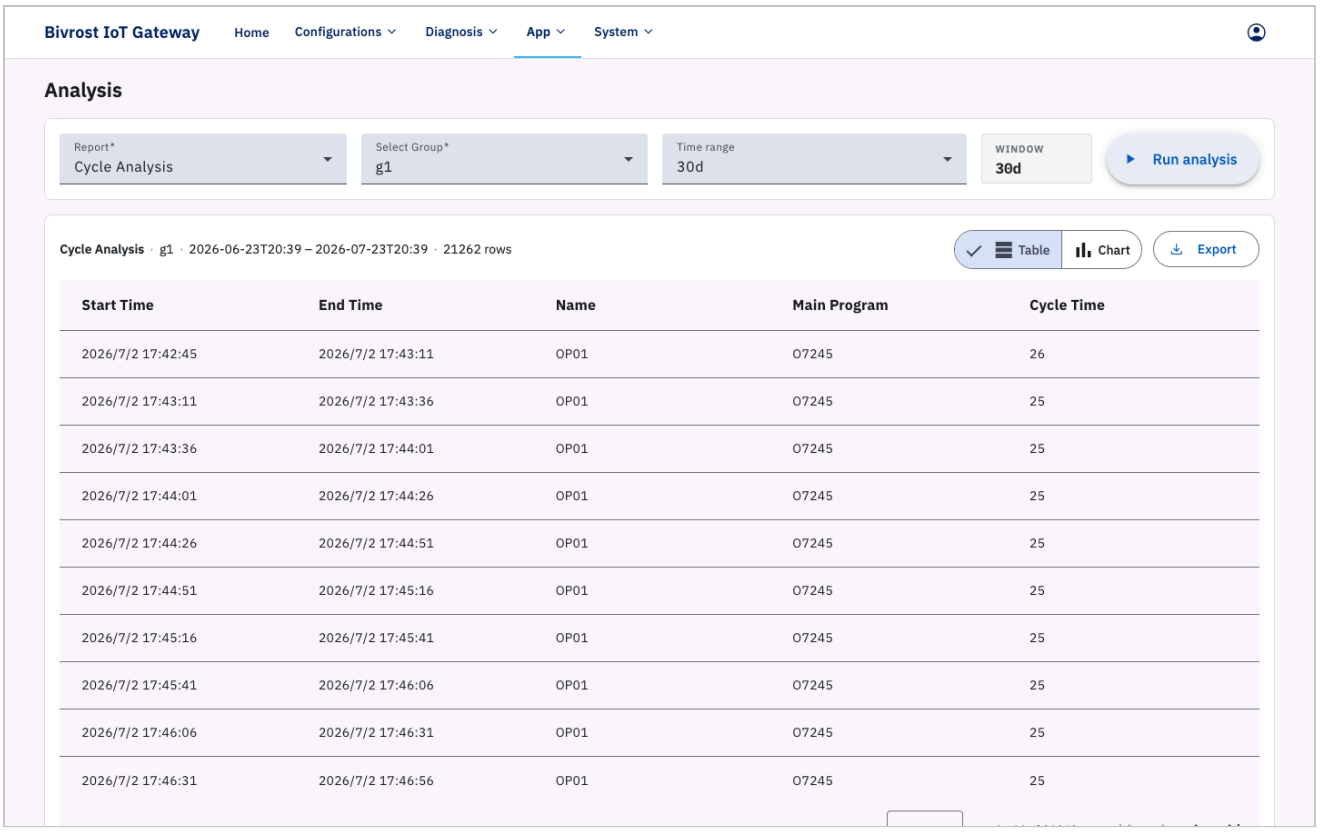
Aggregate by main program breaks the count within each bucket down by the main program that was running at the time. In the figure below, mock machine 1 runs two different main programs, so the count is reported separately for each of them.

Group total by count multiplier outputs a single group row per bucket: the sum of each machine’s count × that machine’s count multiplier, the same calculation the group count monitor uses. Count multipliers are set in 3.4.1.1. Group General Settings. It is exclusive with **Aggregate by main program** — turning one on turns the other off.

Once the result has been calculated, click **Export** to download it as a CSV file. The file is named automatically as `AnalyzerName-GroupName-StartTime-EndTime.csv`.

3.10.3. Cycle Analysis

The figure below shows the cycle analysis table for every machine, produced by choosing Cycle Analysis, Group 1 and the last 30 days and clicking **Run analysis**. It shows, for each machine within the period, the start and end time of every cycle, the main program that was running, and the cycle time.

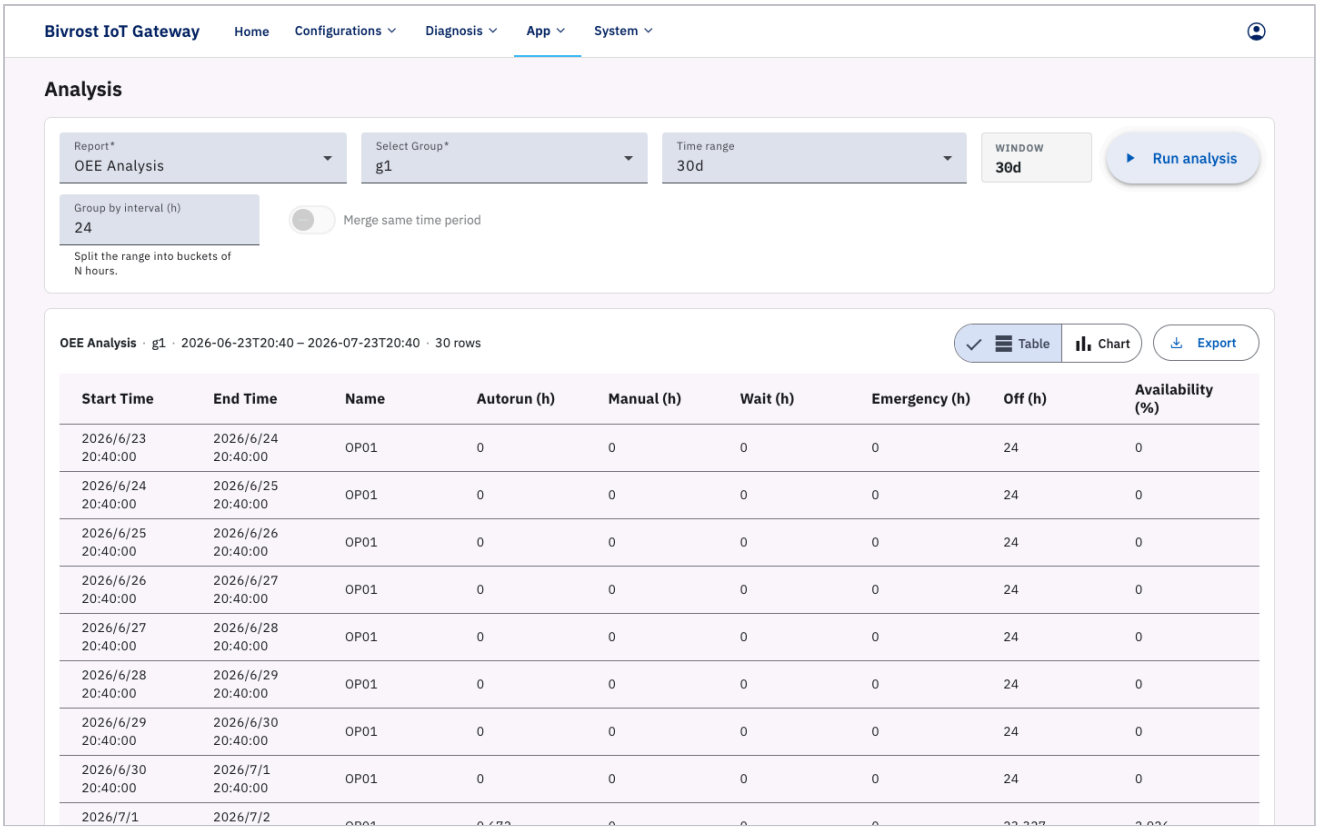


Start Time	End Time	Name	Main Program	Cycle Time
2026/7/2 17:42:45	2026/7/2 17:43:11	OP01	O7245	26
2026/7/2 17:43:11	2026/7/2 17:43:36	OP01	O7245	25
2026/7/2 17:43:36	2026/7/2 17:44:01	OP01	O7245	25
2026/7/2 17:44:01	2026/7/2 17:44:26	OP01	O7245	25
2026/7/2 17:44:26	2026/7/2 17:44:51	OP01	O7245	25
2026/7/2 17:44:51	2026/7/2 17:45:16	OP01	O7245	25
2026/7/2 17:45:16	2026/7/2 17:45:41	OP01	O7245	25
2026/7/2 17:45:41	2026/7/2 17:46:06	OP01	O7245	25
2026/7/2 17:46:06	2026/7/2 17:46:31	OP01	O7245	25
2026/7/2 17:46:31	2026/7/2 17:46:56	OP01	O7245	25

Once the result has been calculated, click **Export** to download it as a CSV file. The file is named automatically as `AnalyzerName-GroupName-StartTime-EndTime.csv`.

3.10.4. OEE Analysis

The figure below shows the OEE analysis table for every machine, produced by choosing OEE Analysis and Group 1, entering an explicit start and end time, setting a grouping interval and clicking **Run analysis**.



Start Time	End Time	Name	Autorun (h)	Manual (h)	Wait (h)	Emergency (h)	Off (h)	Availability (%)
2026/6/23 20:40:00	2026/6/24 20:40:00	OP01	0	0	0	0	24	0
2026/6/24 20:40:00	2026/6/25 20:40:00	OP01	0	0	0	0	24	0
2026/6/25 20:40:00	2026/6/26 20:40:00	OP01	0	0	0	0	24	0
2026/6/26 20:40:00	2026/6/27 20:40:00	OP01	0	0	0	0	24	0
2026/6/27 20:40:00	2026/6/28 20:40:00	OP01	0	0	0	0	24	0
2026/6/28 20:40:00	2026/6/29 20:40:00	OP01	0	0	0	0	24	0
2026/6/29 20:40:00	2026/6/30 20:40:00	OP01	0	0	0	0	24	0
2026/6/30 20:40:00	2026/7/1 20:40:00	OP01	0	0	0	0	24	0
2026/7/1	2026/7/2							

Two settings are specific to this analyzer:

Group by interval (h) works exactly as in Cycle Analysis: it splits the time range into buckets, starting from the start time, and calculates OEE separately for each bucket. The default is 0, which means no grouping — a single bucket.

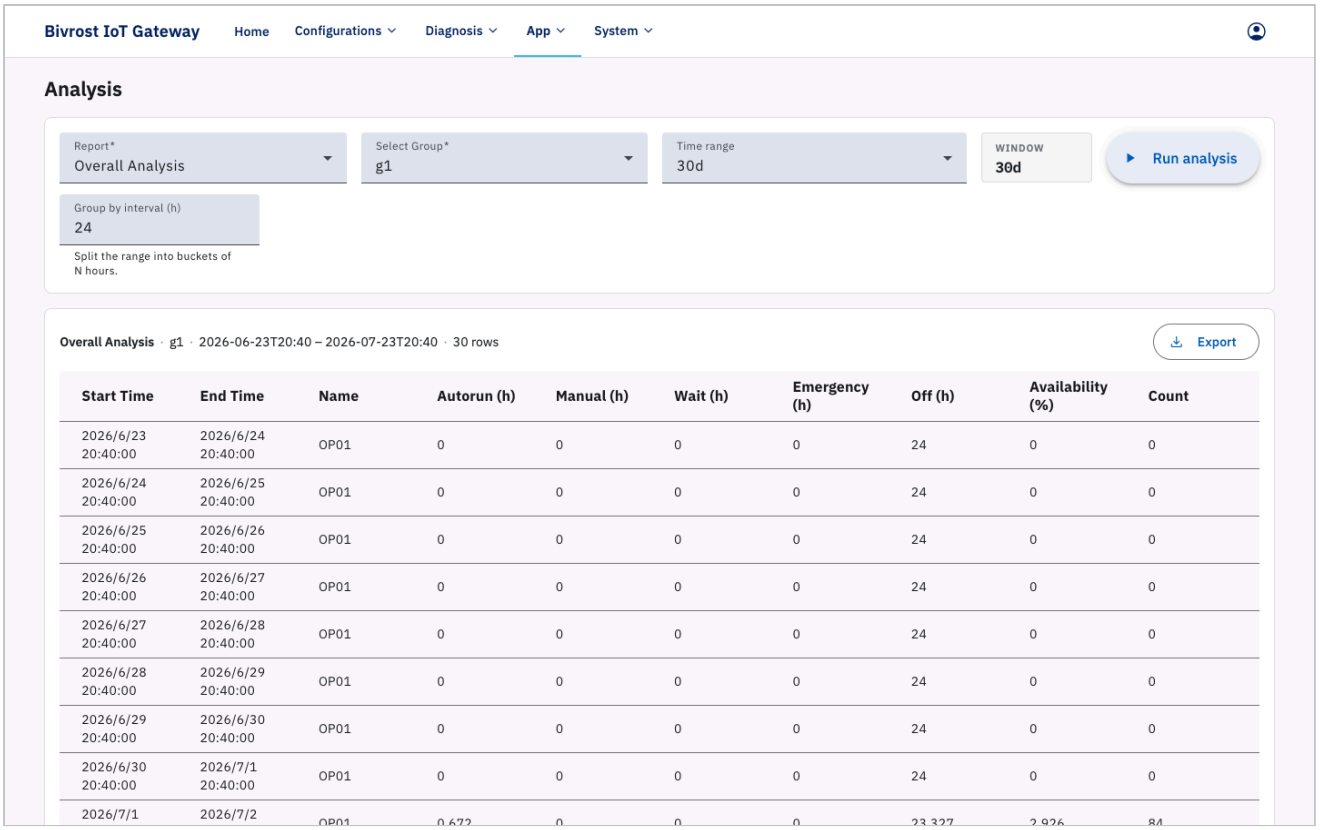
Merge same time period aggregates the OEE data of the buckets that cover the same period of each day. It is typically used to compile OEE data for the several shifts worked each day. You must therefore set the start time to the time a shift begins and set the grouping interval to the length of one shift (greater than 0 and less than 24 hours); only then do you get an OEE summary that matches the shifts. The figure below summarises OEE for two shifts per day (day shift 8:00-20:00, night shift 20:00-8:00) over the specified time range.

Once the result has been calculated, click **Export** to download it as a CSV file. The file is named automatically as `AnalyzerName-GroupName-StartTime-EndTime.csv`.

3.10.5. Overall Analysis

Overall Analysis combines Count Analysis and OEE Analysis.

The figure below shows the OEE data and count data for two shifts per day (day shift 8:00-20:00, night shift 20:00-8:00) over the specified time range.

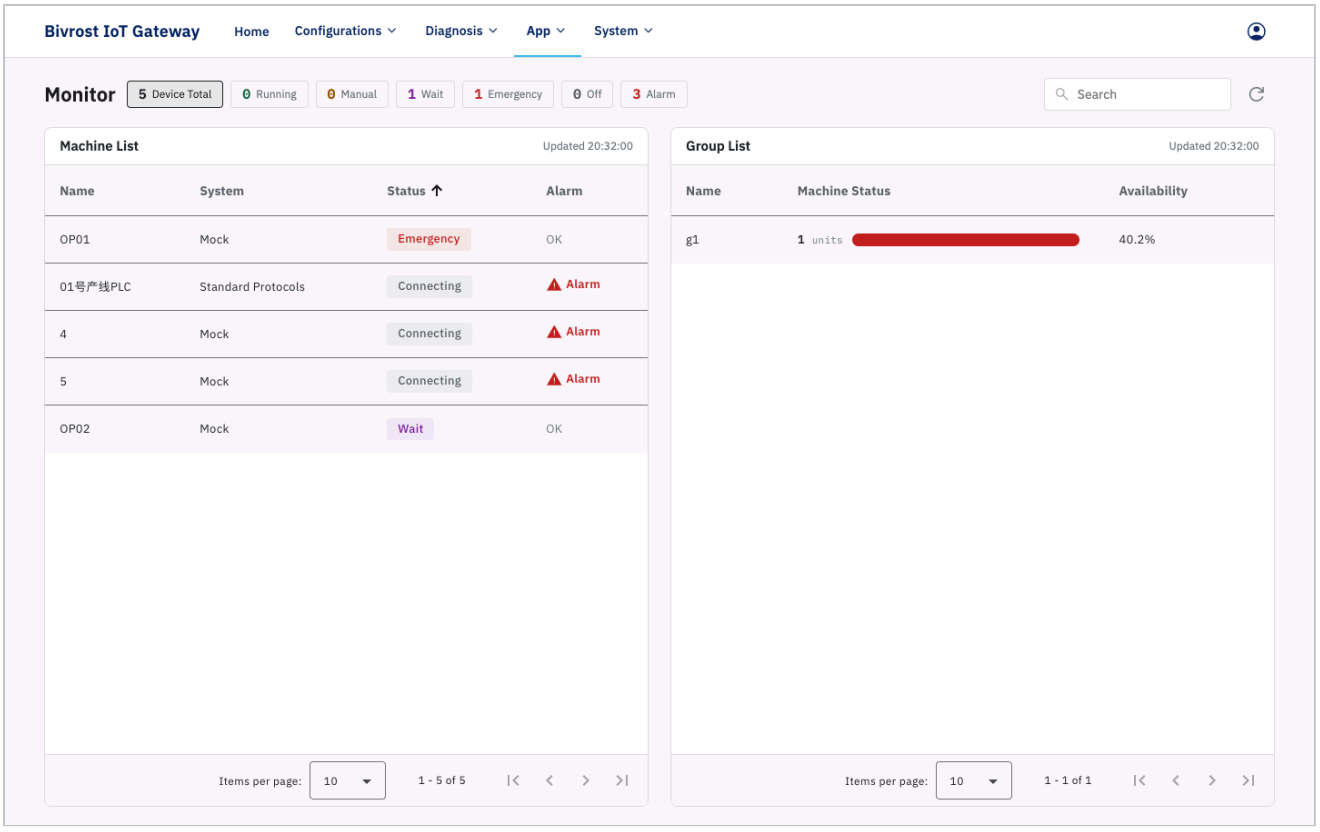


Start Time	End Time	Name	Autorun (h)	Manual (h)	Wait (h)	Emergency (h)	Off (h)	Availability (%)	Count
2026/6/23 20:40:00	2026/6/24 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/24 20:40:00	2026/6/25 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/25 20:40:00	2026/6/26 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/26 20:40:00	2026/6/27 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/27 20:40:00	2026/6/28 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/28 20:40:00	2026/6/29 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/29 20:40:00	2026/6/30 20:40:00	OP01	0	0	0	0	24	0	0
2026/6/30 20:40:00	2026/7/1 20:40:00	OP01	0	0	0	0	24	0	0
2026/7/1	2026/7/2	OP01	0.672	0	0	0	23.327	2.926	84

Once the result has been calculated, click **Export** to download it as a CSV file. The file is named automatically as `AnalyzerName-GroupName-StartTime-EndTime.csv`.

3.11. Monitor

The Monitor page shows machine and group status in real time. At the top of the page are device counts summarised by status (Device Total, Run, Manual, Wait, Emergency, Off, Alarm), each of which can be clicked to filter. Below it are the Machine List (Machine Name, System, Status, Alarm Status) and the Group List.



The layout adapts to the screen size: the Machine List and the Group List may be arranged side by side instead of one above the other.

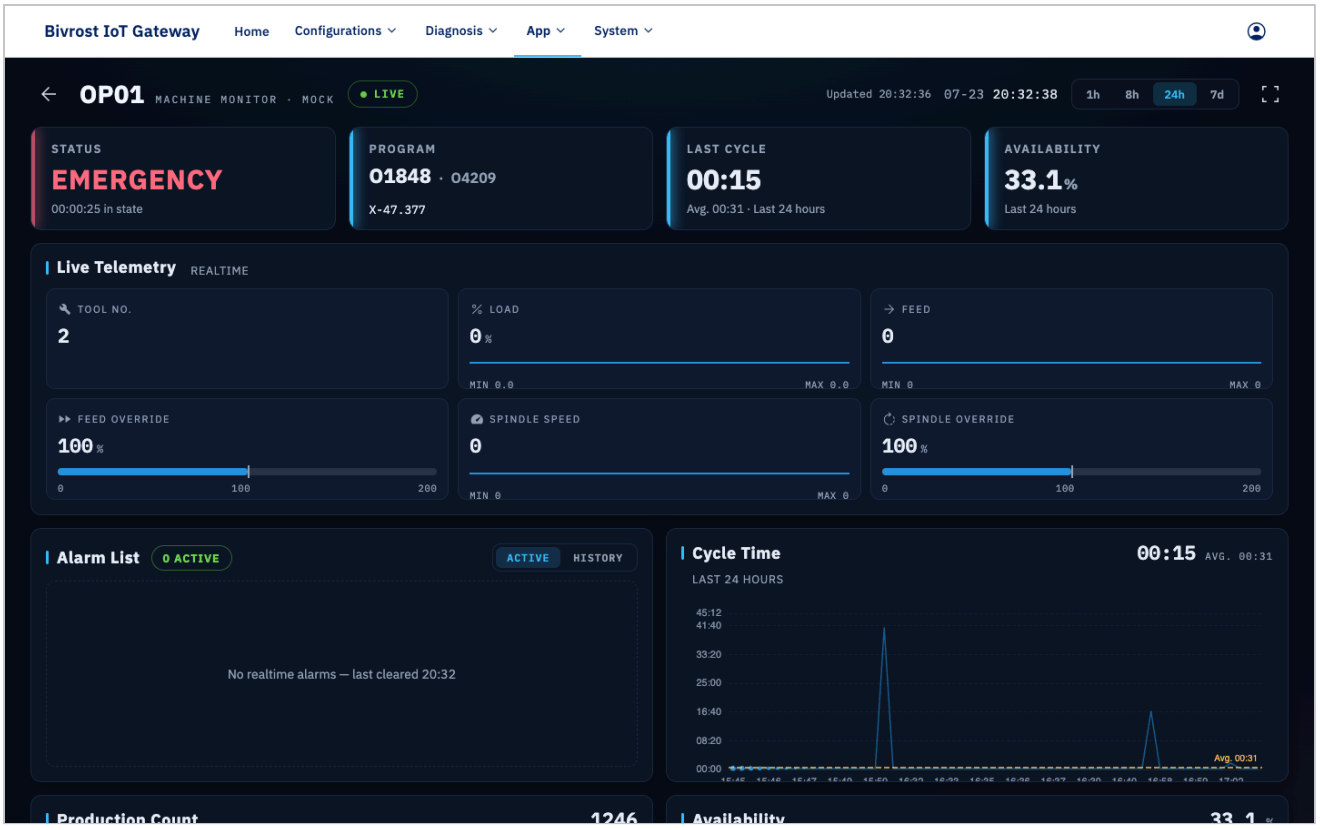
3.11.1. Machine Monitor

Click a machine row in the Machine List to open the Machine Monitor page, which shows the machine’s live data (Main Program, Sub-program, current Alarm Level, Current Tool No, spindle load, feedrate, Feedrate Override, spindle speed, Spindle Override and so on) together with the Count, Availability, status statistics and Alarm History for the last 24 hours.

The large number to the right of the **Production Count** card’s title is the **Current Count**, the live count for the current counting period; the caption beneath it reads “Current Count”, followed by the total for the selected time range. The period mode (Scheduled Reset or Time

Window) and its reset times come from 3.5.5. Count Monitoring Settings; the page no longer repeats them.

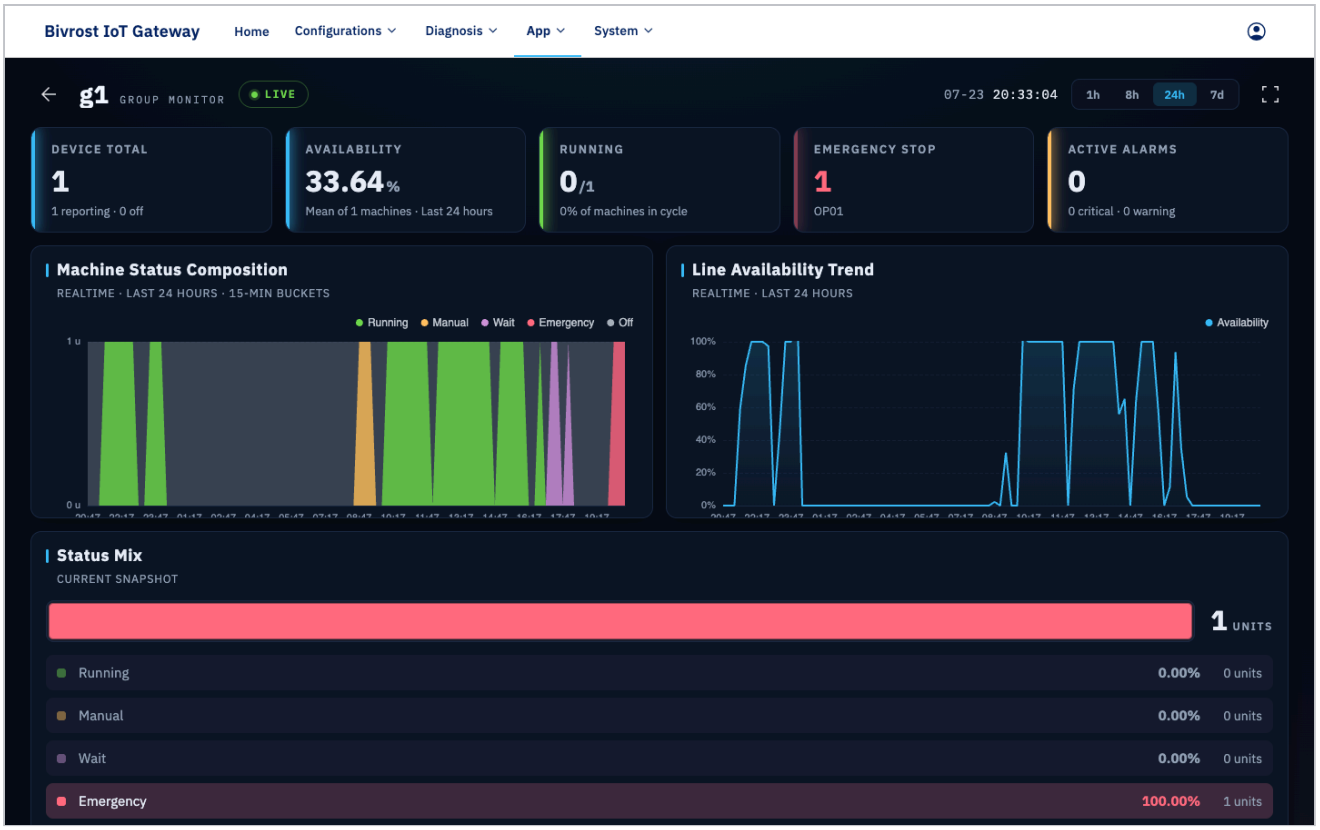
The ‹ button in the top-left corner returns to the Monitor home page.



3.11.2. Group Monitor (Group Overview)

Click a group row in the Group List to open the Group Monitor page, which presents the group’s live data as an Andon wall (Availability, the number of machines in each status, a status card for every machine, alarm severity and time, and so on) along with the availability and status trend charts. For groups with a large amount of data, the trend charts take a moment to load.

The ‹ button in the top-left corner returns to the Monitor home page.



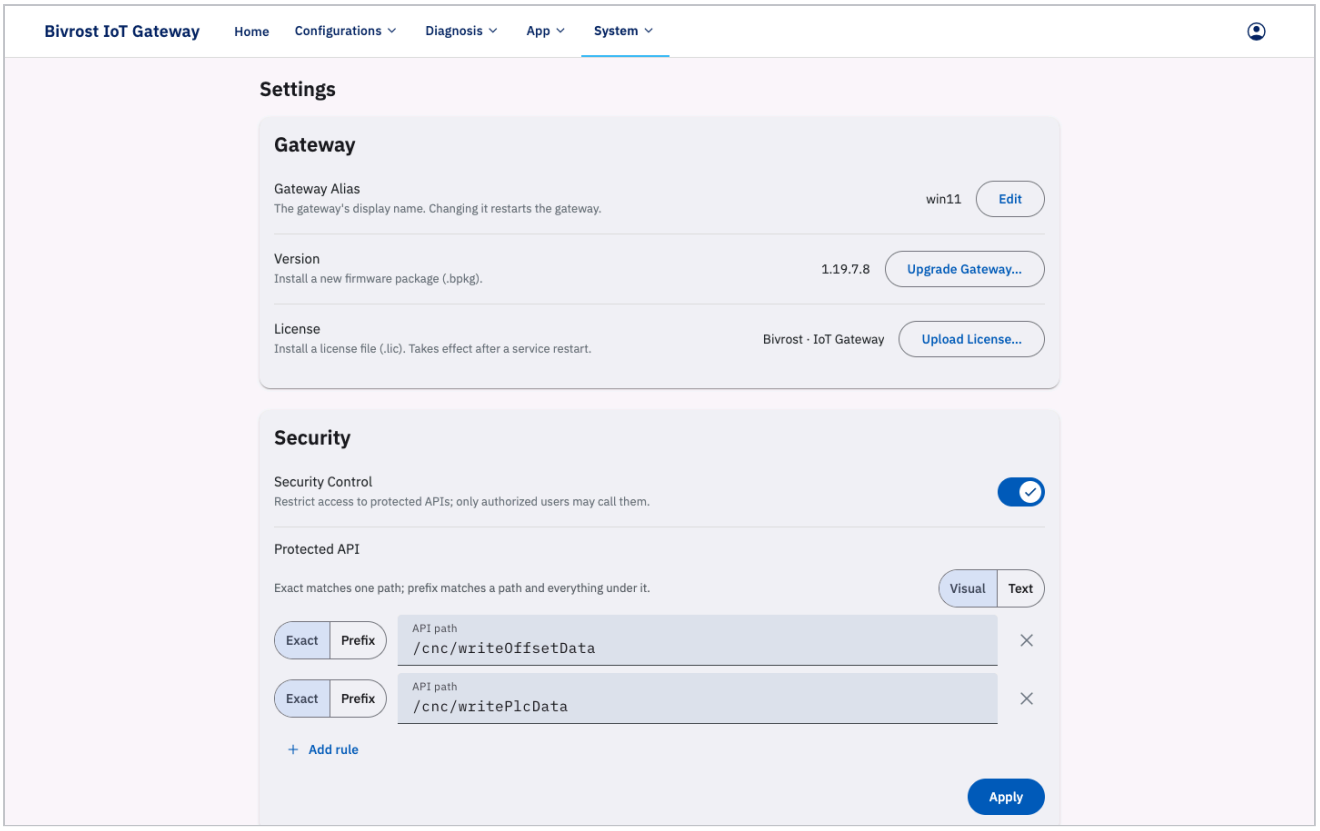
Note

The Availability shown on the page is the cumulative availability over the monitoring period (see 3.5.3. OEE Monitoring Settings). The availability plotted in the availability trend chart is calculated per bucket, using 5-minute buckets within the trend time window.

3.12. Settings

The Settings page is where you configure and maintain the gateway’s advanced functions: user permissions, license updates, software upgrades, API security, time and region settings, and so on.

When you sign in with an administrator account (for example the initial `admin` account), the Settings page shows the following cards from top to bottom: **Gateway** (Gateway Alias, Version, License), **Security**, **Options**, **Time And Region Setup**, **Backup & Restore** and **Users & Access**.

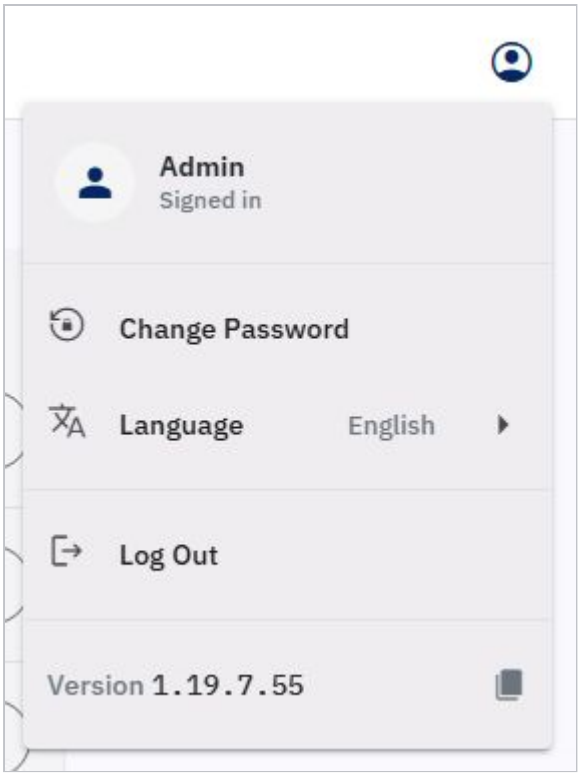


When you sign in with a standard user account, you can change your password, switch language and sign out using the account menu at the top right, and switch language in the Time And Region Setup card.

The instructions below describe the interface as seen by an administrator. Unless stated otherwise, a setting can only be changed while signed in as an administrator.

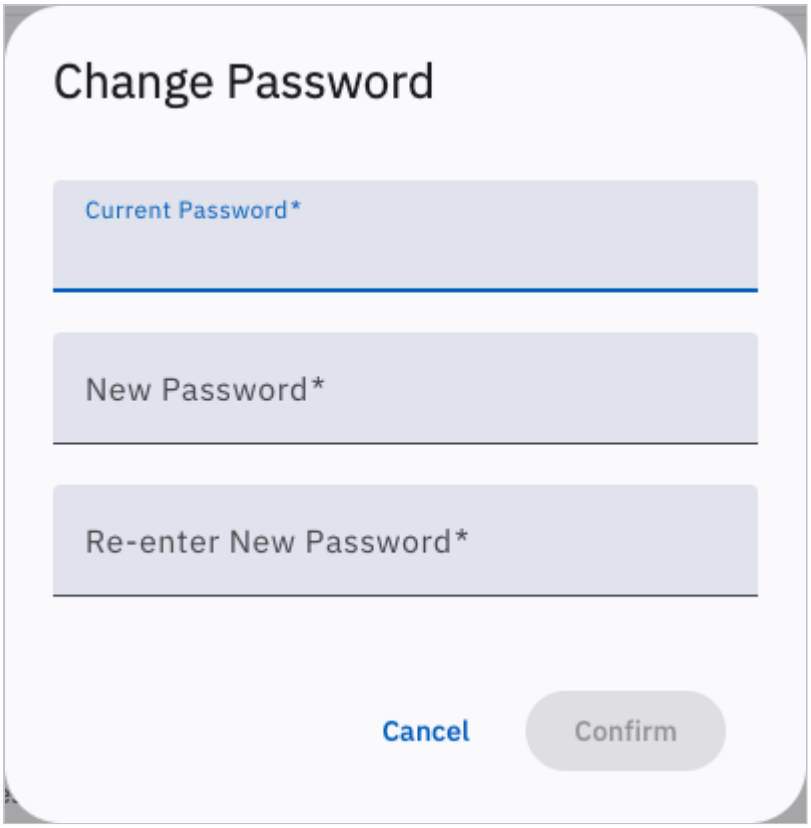
3.12.1. Account Menu

Click **Account** at the top right. The menu shows Change Password, Language, Log Out and the gateway version in that order. These items are available to administrators and standard users.



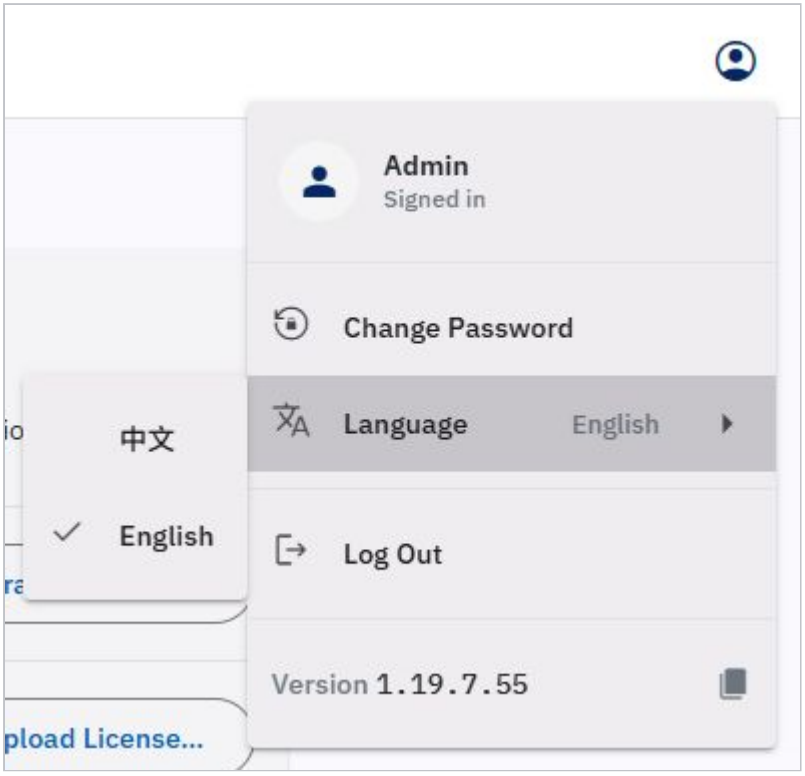
3.12.1.1. Change Password

Click the **Account** button at the top right of the page to open the account menu, choose **Change Password**, enter your current password to verify your identity, then enter the new password and click Confirm to apply the change. Once the change is confirmed, you must use the new password the next time you sign in. This function is available to administrators and standard users alike.



3.12.1.2. Language

Choose **Language**, then select 中文 or English in the submenu. The current language is marked with a checkmark. This changes the web interface language.



3.12.1.3. Log Out

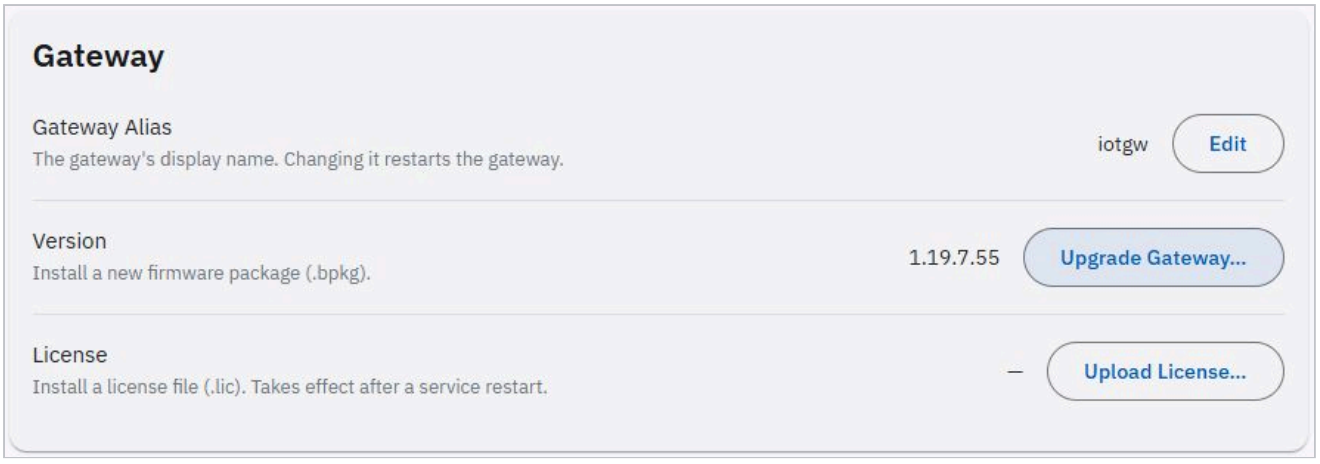
Choose **Log Out** to end the current login session and return to the sign-in page.

3.12.1.4. Version

The bottom of the menu shows the gateway software version. Click it to copy the version to the clipboard when contacting support.

3.12.2. Gateway

The Gateway card shows Gateway Alias, Version and License from top to bottom.



3.12.2.1. Change Gateway Alias

In the "Gateway" card, click the **Edit** button to the right of the Gateway Alias to open a dialog where you can change it. The alias is also the gateway's domain name, which is used both to reach the gateway and for MQTT communication (see 3.6.3. MQTT Settings). The default alias is iotgw. If you reach the gateway by domain name, use the new domain name to sign in after changing the alias.

3.12.2.2. Upgrade Gateway

Upgrade Gateway... in the "Gateway" card uploads an upgrade package (.bpkg) to update the gateway software.

The procedure is similar to "Upload License": contact support to obtain the upgrade package, save it on your computer, then click the **Upgrade Gateway...** button to upload it. An upload progress dialog appears — please wait a moment; you can click **Cancel** at any time to abort the upload. Once the package has uploaded successfully, a confirmation dialog appears; click **Confirm** to start the upgrade. Wait about 10 seconds, then refresh the Home page and check the gateway version under Gateway Details.

3.12.2.3. Upload License

Upload License... in the "Gateway" card uploads a new license file (.lic) to change the license type, raise the license count and so on. If you need to do this, contact support with the gateway UID (see 3.2.3. Gateway Details) and the license type and count you want. Once you have received the new license file on your computer, click the **Upload License...** button, select the license file in the dialog that opens, and click **Open** to upload it. When the upload finishes, click **Restart Service** on the Home page (see 3.2.1. Function Buttons) for it to take effect. After restarting the service and refreshing the Home page, check that "License Type", "License Count" and "License Status" under Gateway Details match what you expect; contact support if anything looks wrong.

3.12.3. Security

3.12.3.1. Security Control

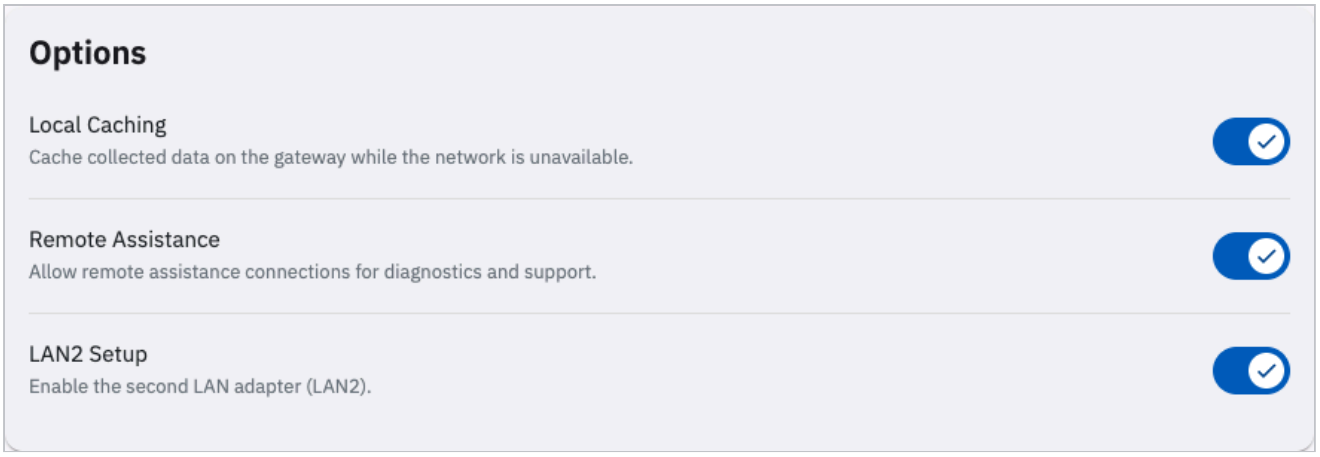
The “Security Control” switch turns API access control on or off; it is on by default. With the switch on, you can edit the protected APIs.

“Protected API” lists the APIs that may not be called. By default, Write Offset Data and Write PLC Data are blocked: writing incorrect data through either interface can cause injury, death or equipment damage, so only use them once you have confirmed it is safe to do so. To change the list, see 11.2.7. Protected API and Authorized APIs, then click **Apply**.

After changing “Security Control” or “Protected API”, click **Restart Service** on the Home page for the change to take effect.

3.12.4. Options

The Options card contains the Local Caching, Remote Assistance and LAN2 Settings switches from top to bottom.



3.12.4.1. Local Caching

The "Local Caching" switch turns local caching on or off; it is on by default. Local caching stores recently collected data and run logs on the gateway so that it can serve historical data and the statistics derived from it, such as machine status statistics and count records for a past period. Data is kept locally for at most 365 days. Turning local caching off disables some gateway functions, so we recommend leaving it on. If you change this switch, click **Restart Service** on the Home page for it to take effect.

3.12.4.2. Remote Assistance

The "Remote Assistance" switch turns remote assistance on or off; it is on by default. When it is on, support staff can reach the gateway over the internet. You can turn this option off for normal operation; if you need remote assistance, contact support first, then turn the option back on once the gateway is connected to the internet. Changes to this switch take effect immediately — no **Restart Service** is needed.

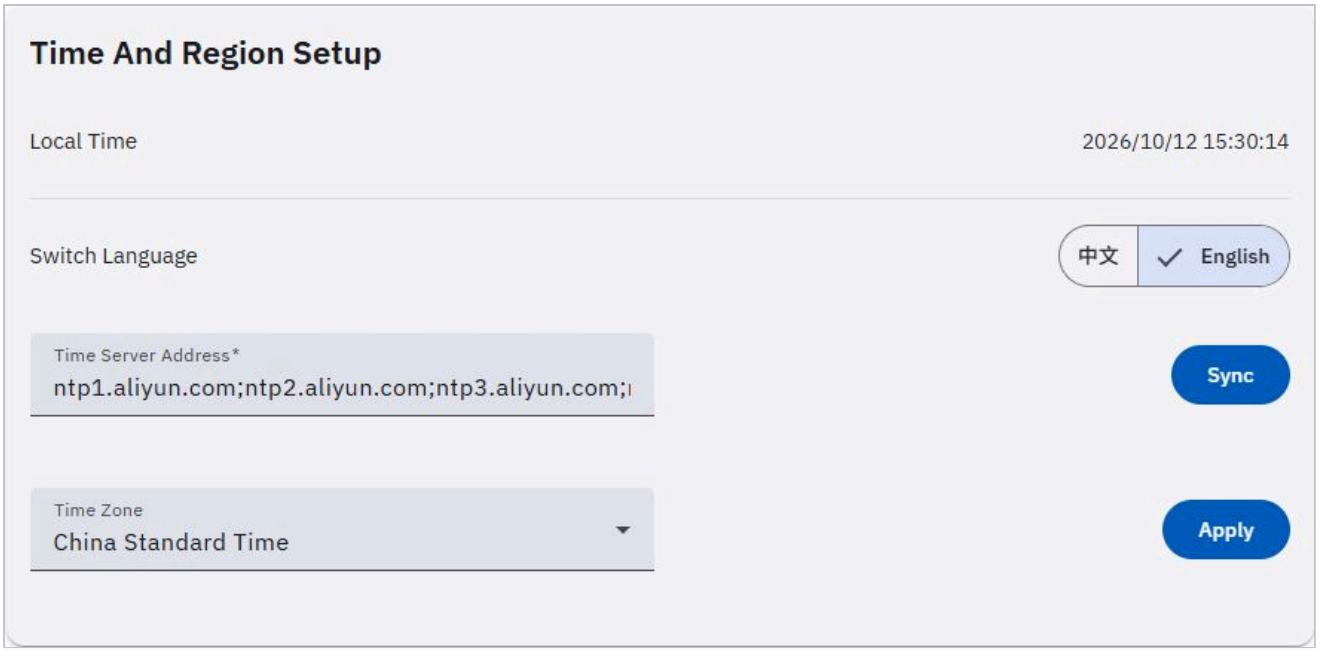
3.12.4.3. LAN2 Settings

The "LAN2 Settings" switch shows or hides the LAN2 settings on the **Network** page; it is off by default. With the switch on, you can change the LAN2 settings on the **Network** page, following the same steps used for LAN1 in 3.7.1. Wired Network.

> **Note**
>
> Turning the "LAN2 Settings" switch off after changing the LAN2 settings does not reset them.

3.12.5. Time And Region Setup

“Time And Region Setup” covers Local Time, Switch Language, Time Server Address and Time Zone.



3.12.5.1. Local Time

“Local Time” shows the gateway’s current time.

3.12.5.2. Switch Language

“Switch Language” switches the gateway’s web interface between Chinese and English. It does the same thing as the language selector at the top right of the sign-in page and the **Language** item in the account menu.

3.12.5.3. Time Server Address

“Time Server Address” takes the addresses of the servers used for time synchronization (separate multiple addresses with “;”). The default is `ntp1.aliyun.com;ntp2.aliyun.com;ntp3.aliyun.com;ntp4.aliyun.com;`, the Alibaba Cloud public time servers, which require an internet connection. Click the **Sync** button on the right to synchronize the time: the gateway starts with the first address and moves on to the next one if it fails, stopping as soon as one succeeds. The current time server address is saved only if synchronization succeeds.

3.12.5.4. Time Zone

“Time Zone” selects the time zone the gateway is in. The default is “China Standard Time”, i.e. Beijing time. Click the **Apply** button to save.

3.12.6. Backup & Restore

The Backup & Restore card provides Backup Config, Restore Config and Factory Reset from top to bottom.

3.12.6.1. Backup Config

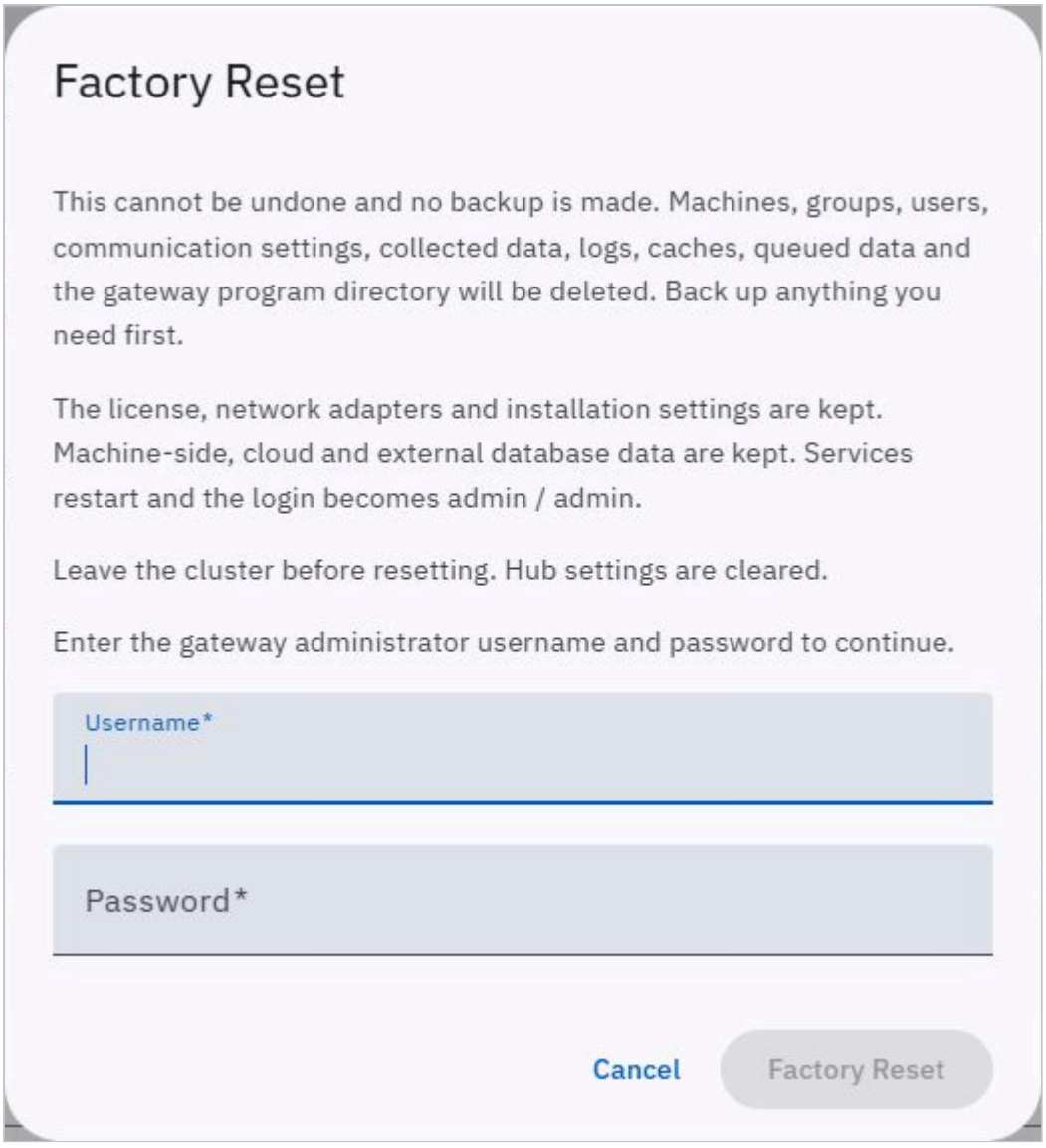
Backup Config backs up the gateway’s current configuration file. Click the **Backup** button to download it.

3.12.6.2. Restore Config

Restore Config restores the gateway configuration from a backup file. Click the **Restore Config...** button and upload a gateway configuration backup file to restore the configuration. This operation cannot be undone.

3.12.6.3. Factory Reset

Click **Factory Reset** in the Backup & Restore card, enter the gateway administrator username and password, then click the **Factory Reset** button to continue. Leave the cluster before resetting a cluster member.



> **This cannot be undone**
> No backup is made. This clears machines, groups, users, communication settings, local collected data, logs, caches, queued data, gateway program files. Back up configuration, data and programs you need first.

The license, network adapters (including saved Wi-Fi networks), system VPN and installation settings are kept. Machine-side, cloud and external database data are kept. The software version stays the same.

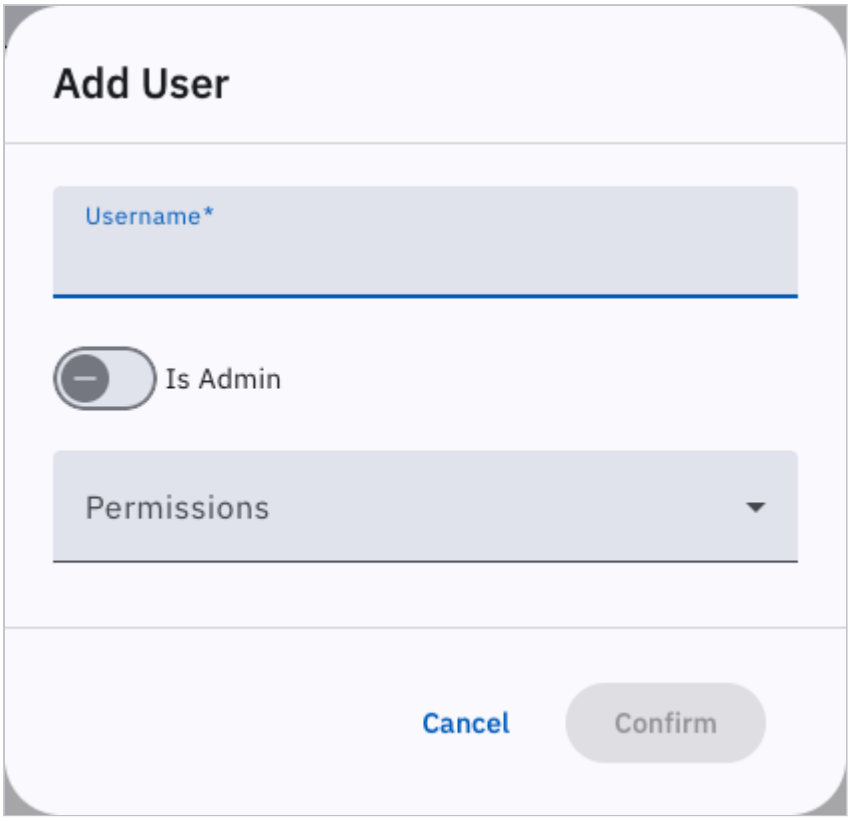
Locally, confirmed completion clears the login session and opens the login page. Sign in with `admin/admin`. Wait a few minutes for the local database to initialize.

3.12.7. Users & Access

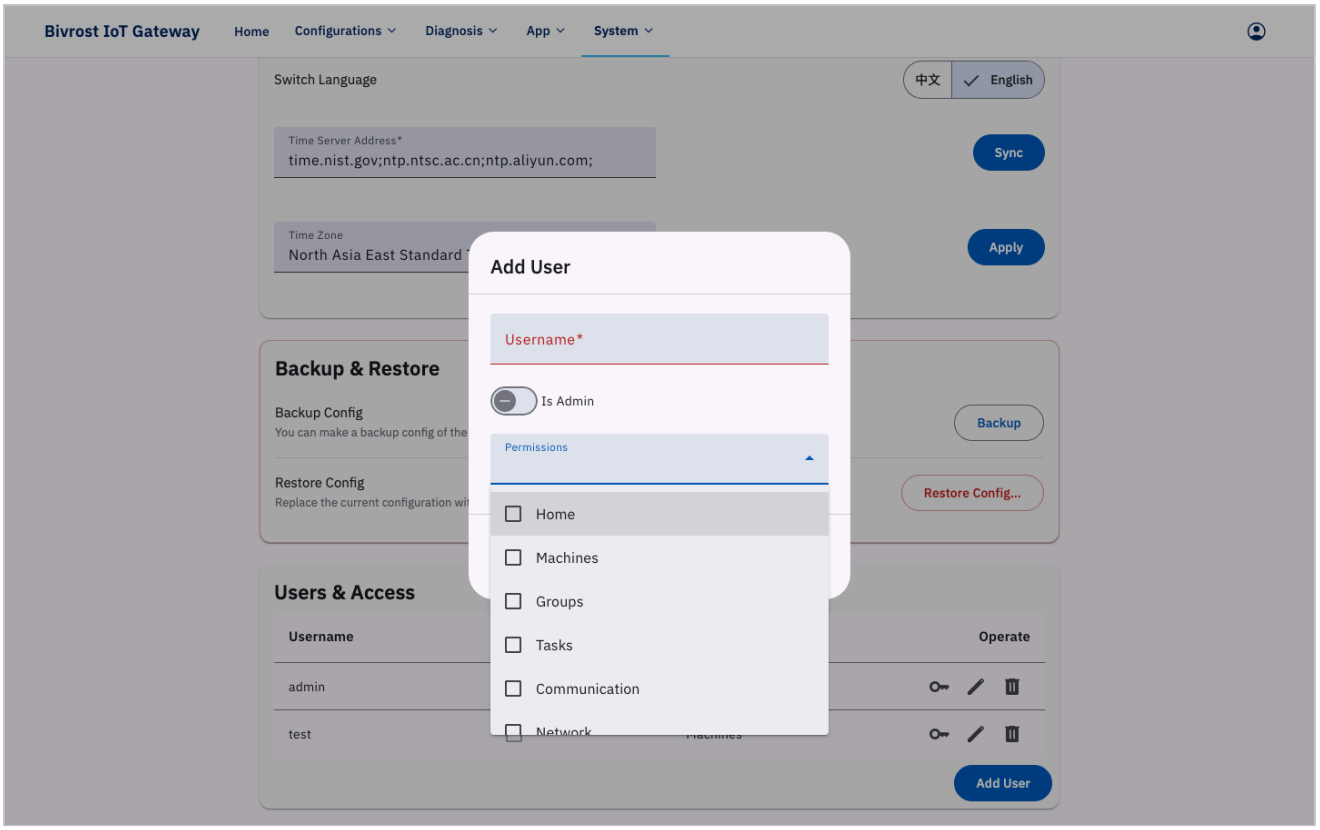
The "Users & Access" card lets you add, edit and delete gateway users, and edit their permissions.

3.12.7.1. Add User

Click the **Add User** button on the card to create a new user.

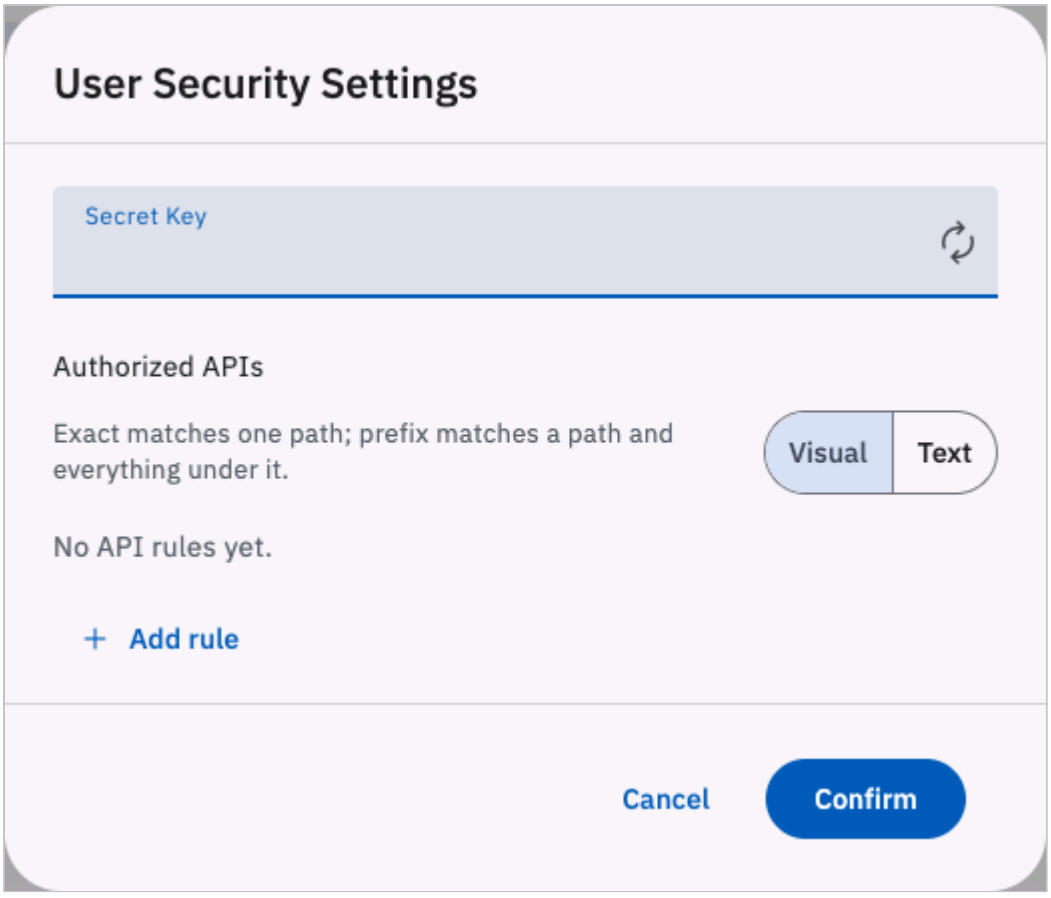


In the dialog that opens, set the username, choose whether the user is an administrator, and assign permissions. A new user' s initial password is the same as the username. There are two user types: administrator and standard user. An administrator has full access to every page and setting on the gateway. A standard user can only open the pages granted to them; tick those pages in the permissions box. Click Confirm when you are done.



3.12.7.2. User Security Settings

In the management dialog, click the shield button 🛡 in the action column of a user’ s row to open the User Security Settings dialog, where you can set a Secret Key and Authorized APIs. User security settings only take effect once the Security Control option is enabled.



The **Secret Key** authenticates a user by key when calling an API; see 5.3.2. Secret Key. The key must start with "sk-". Click the **Generate Secret Key** button 🔄 to the right of the Secret Key field to generate a random key.

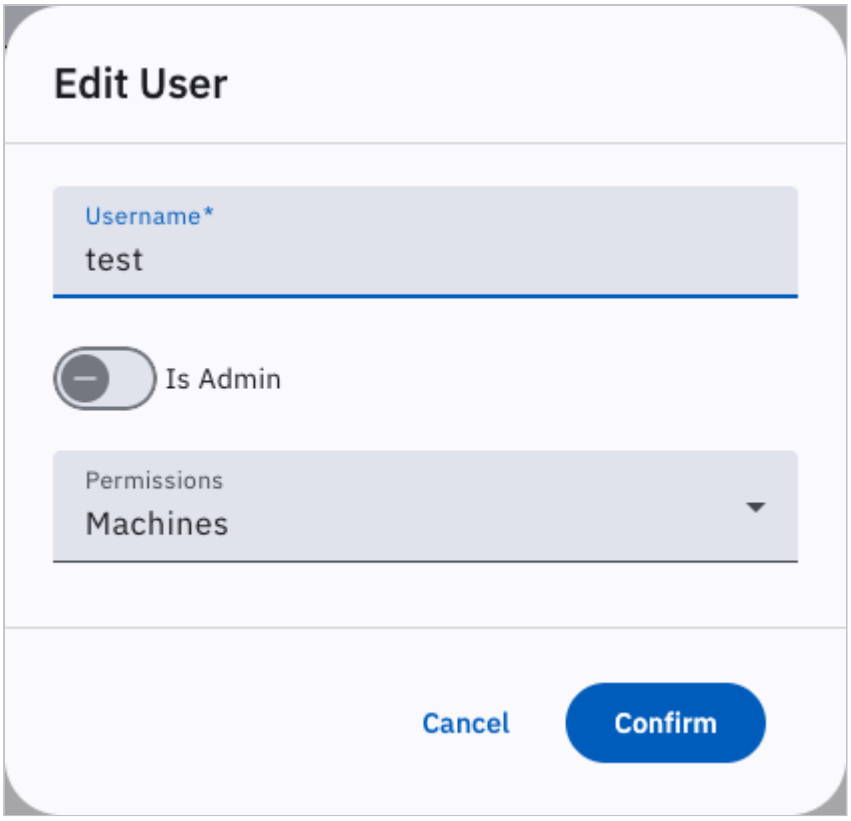
Authorized APIs specifies which interfaces the user may call; see 11.2.7. Protected API and Authorized APIs for how to set them.

> **Note**
>
> Protected APIs take precedence over authorized APIs: if an authorized API is also a protected API, the user cannot call it.

Click Confirm to save your changes.

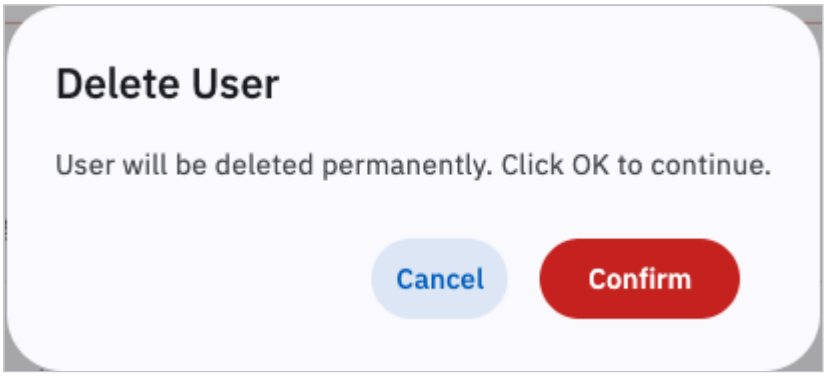
3.12.7.3. Edit User

In the management dialog, click the edit button ✏️ in the action column of a user's row to change the username, administrator status and page permissions. Click Confirm to apply the change.



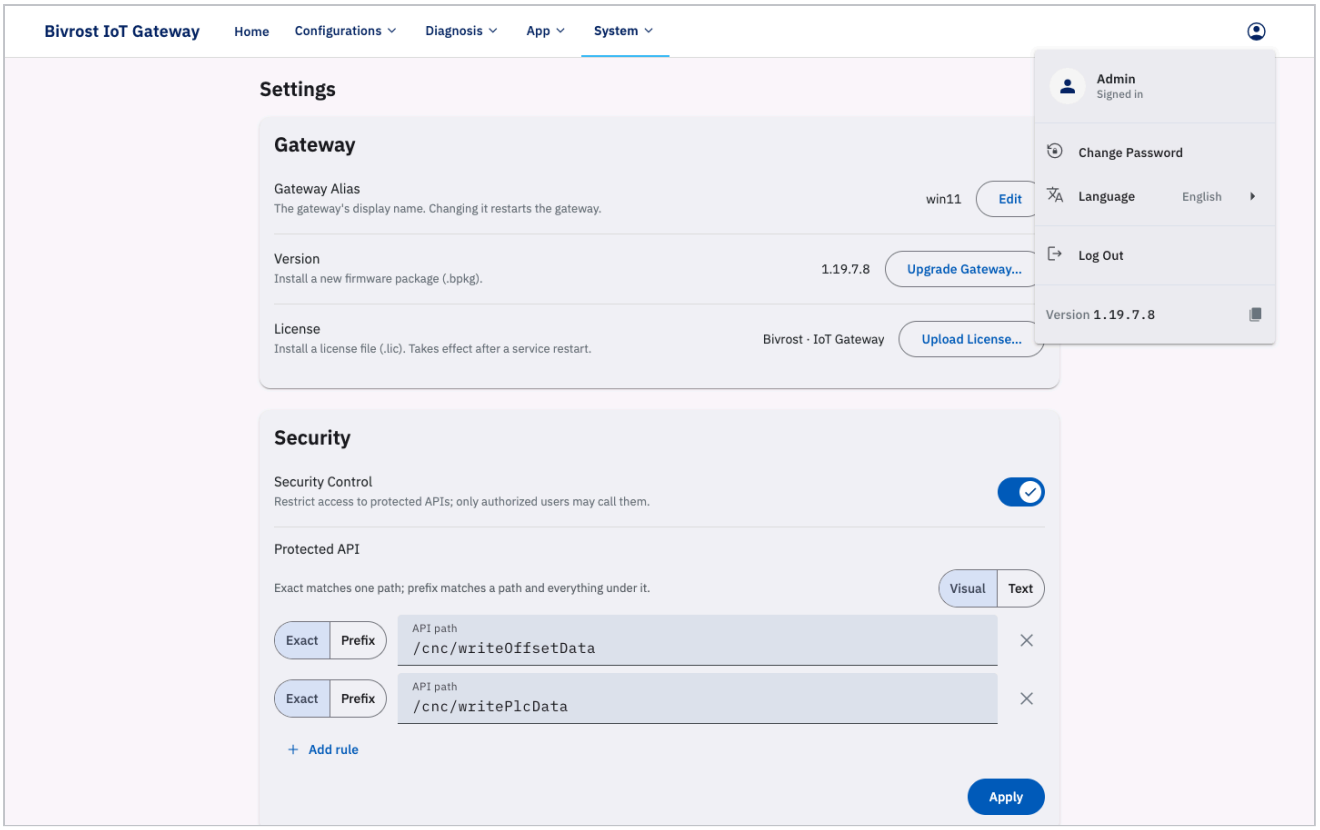
3.12.7.4. Delete User

In the Manage Users dialog, click the delete button 🗑 in the action column of a user’s row and confirm to delete the user.



3.13. Other

The **Account** button is at the top right of the page. Click it to open the account menu, which contains **Change Password**, **Language**, **Log Out** and other options. The current gateway version is shown at the bottom (click to copy it to the clipboard).



Click **Log Out** to sign out of the current session.

4. Protocol Conventions

4.1. Identifiers

For the web management interface referred to in this and the following protocol chapters, see 3. Using the Gateway.

The content explained in this chapter applies to later chapters. Users can look up the explanations in this chapter based on the reference hints in specific use cases.

The identifiers used in communication include machineID (machine identifier), groupID (group identifier), slaveID (slave identifier), ID (database identifier), and uid (gateway identifier).

4.1.1. machineID (Machine Identifier)

The gateway supports connecting to multiple machine devices simultaneously, and a unique machineID must be defined for each connected device. The machineID can be customized by the user on the gateway’s machine configuration page, under Add/Edit Machine - Advanced Settings (see 3.3.1.5. Advanced Settings). By default, the machineID is derived by concatenating the third and fourth octets of the machine’s IP address, then stripping leading zeros. If the third octet is not 0, the fourth octet is padded to three digits. For example, the machineID corresponding to 192.168.0.1 is 1, and the machineID corresponding to 192.168.1.12 is 1012.

4.1.2. groupID (Group Identifier)

The gateway currently supports up to 16 groups, and a unique groupID must be defined for each group of devices. The groupID can be customized by the user on the gateway’s group configuration page, under Add/Edit Machine - Advanced Settings (see 3.4.1.3. Advanced Settings). By default, the groupID consists of the letter “g” followed by the group number, e.g. “g2”.

4.1.3. slaveID (Slave Identifier)

The slaveID is used as the machine data identifier for MODBUS communication, with a range of 1-255. The slaveID can be customized by the user on the gateway’s machine configuration page, under Add/Edit Machine - Advanced Settings (see 3.3.1.5. Advanced Settings). By default, the slaveID is the fourth octet of the machine’s IP address.

4.1.4. ID (Database Identifier)

The ID (database identifier) is the identifier for a user, machine, or group in the local database. The ID is automatically generated in the database when a user, machine, or group is created, and cannot be created or modified by the user.

4.1.5. uid (Gateway Identifier)

The uid (gateway identifier) is the gateway’s own identifier. It defaults to the gateway’s hardware ID and can be viewed via 5.9.1.1. gateway-info Get Gateway Information.

Data collected by the gateway itself has an empty uid. In a cascaded (gatewayLinks) setup, data from a downstream gateway carries that gateway’s uid, therefore:

- The machine/group data and historical data APIs all accept an optional uid parameter that selects the gateway the data came from. When uid is omitted, only this gateway’s own data is returned.
- MQTT payloads and database output also emit uid as a data tag / data column.

4.2. Data Description

The `<type>` data classes are listed in the table below:

No.	type	Description
1	AlarmHistory	Alarm history
2	AlarmLog	Current alarm
3	AxialOverload	Servo axis overload data
4	CNCStatus	Machine status
5	Count	Machining count
6	CurrentToolNumber	Current tool number
7	Cycle	Cycle time data
8	EnergyConsum	Energy consumption data
9	Feed	Feed data
10	FeedAndSpindle	Feed and spindle override data
11	GroupCount	Group machining count
12	GroupCumulativeTime	Group cumulative status time
13	GroupOEE	Group OEE data
14	LaserPower	Laser power
15	Load	Load data
16	LogHistory	Log history
17	OEE	Machine OEE data
18	Offset	Tool offset data
19	PLC	PLC data
20	Position	Position data
21	ProgramBlock	Program block
22	ProgramInfo	Current program
23	SpindleOverload	Spindle overload data
24	TimeData	Machine time data
25	ToolLife	Tool life data

The `<field>` data fields and `<tag>` data tags for each data class are listed below. If a data class does not list `<tag>` data tags, that data class does not require data tags.

4.2.1. AlarmHistory: Alarm History

Alarm history `<field>` data fields:

Field	Type	Description
alarmLevel	String	Alarm level
alarmMsg	String	Alarm message
alarmTime	String	Time the alarm occurred (started)

An AlarmHistory record is produced only when an alarm is cleared, so there is no active/cleared distinction. The alarmTime in the record is the time the alarm occurred (started), while the record’s own timestamp is the time the alarm was cleared.

4.2.2. AlarmLog: Current Alarm

Current alarm `<field>` data fields:

Field	Type	Description
alarmLevel	String[]	Alarm level
alarmMsg	String[]	Alarm message
alarmTime	String[]	Alarm time

4.2.3. AxialOverload: Servo Axis Overload Data

Servo axis overload data `<field>` data fields:

Field	Type	Description
cumulativeTime	UInt32	Cumulative overload time [ms]

The cumulative axis overload time is the axis overload time accumulated since the gateway began automatically collecting data from this device. This value is stored in the gateway, bound to the machineID machine identifier, and is never reset.

Servo axis overload data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	axisNo	Int32	A	Axis number
2	channel	Int32	C	Channel number; if absent, the default channel is implied

4.2.4. CNCStatus: Machine Status

Machine status `<field>` data fields:

Field	Type	Description
alarmStatus	String	Alarm status, see Alarm Status, Alarm Level
alarmLevel	String	Alarm level. If there are multiple alarms, the highest level among them is used. If there is no alarm, this field is not returned. See Alarm Level for details.
cncStatus	String	CNC running status, see Running Status
adjustedStatus	String	Adjusted CNC running status. If status adjustment is enabled (see 3.5.8. Machine Status Monitoring Settings for details), the adjusted machine status is obtained according to the configured rules. If status adjustment is not enabled, this field has the same value as cncStatus. In calculations involving machine status, such as OEE and cumulative status time, adjustedStatus takes priority for determining status.
mode	String	*Operating mode, varies by system model
programStatus	String	*Program status, varies by system model

Field	Type	Description
emergencyStatus	String	*Emergency stop status, see Emergency Stop Status
dryRunStatus	String	*Dry run status, see Dry Run Status
offTime	Int64	Cumulative off time [s]
waitTime	Int64	Cumulative wait time [s]
emergencyTime	Int64	Cumulative emergency stop time [s]
autoRunTime	Int64	Cumulative auto run time [s]
manualTime	Int64	Cumulative manual (setup) time [s]

The cumulative status times are the times accumulated for each status since the gateway began automatically collecting data from this device. These values are stored in the gateway, bound to the machineID machine identifier, and are never reset.

*: Status details are returned in the following cases:

1. Using the 5.5.1.3. readCNCStatusDetails - Read Machine Status Details, 5.5.2.1. readTaskData - Read Machine Task Data, or 5.5.2.2. batchReadTaskData - Batch Read Machine Task Data API.
2. The machine status task is enabled, and status details are enabled in Task Configuration - Machine Status Monitoring Settings; the machine status task then returns these fields.

Machine status `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.5. Count: Machining Count

Machining count `<field>` data fields:

Field	Type	Description
count	Int32	Current number of workpieces machined, as recorded by the machine system
cumulativeCount	Int32	Cumulative machining count
currentCount	Int32	Current production output

The cumulative machining count is the machining count accumulated since the gateway began automatically collecting data from this device. This value is stored in the gateway, bound to the machineID machine identifier, and is never reset. Current production output is the production count for the machine within the monitoring period.

4.2.6. CurrentToolNumber: Current Tool Number

Current tool number `<field>` data fields:

Field	Type	Description
toolNumber	String	Current tool number
toolOffsetNum	String	Current tool offset number

Current tool number `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.7. Cycle: Cycle Time Data

Cycle time data `<field>` data fields:

Field	Type	Description
lastCycleTime	Int64	Last cycle time [s], monitored by the gateway’s machine cycle time data task. If the first cycle has not yet ended after the gateway restarts, this value is 0. The definition of “last cycle time” here is the same as the last cycle time in 1.2.24. TimeData: Machine Time Data; the difference is that the former is computed by the gateway and is supported by all machines that can report production count and status, whereas the latter is provided by the machine itself and is only supported by some machines.
lastTotalTime	Int64	Total time from the end of the last cycle to the end of this cycle [s], including non-running time.
mainPrgmName	String	Main program name of the last cycle
deltaCount	Int32	Change in count between the previous and current cycle. When this value is greater than 1, the cycle time equals the running time during this count change divided by the count change value.

4.2.8. EnergyConsum: Energy Consumption Data

Energy consumption data `<field>` data fields:

Field	Type	Description
allServoAxes	Double	Energy consumption of all servo axes [Wh]
coolantPump	Double	Coolant pump energy consumption [Wh]
ctsPump	Double	CTS pump [Wh]
chipFlusherPump	Double	Chip flusher pump [Wh]

Field	Type	Description
cyclonePump	Double	Cyclone filter pump [Wh]
system24V	Double	24V system [Wh]
ncControl	Double	NC control [Wh]
lcdBacklight	Double	LCD backlight [Wh]
chipConveyor	Double	Chip conveyor [Wh]
screwConveyor	Double	Screw conveyor [Wh]
autoLubrication	Double	Automatic lubrication device (or automatic greasing device) [Wh]
spindleCoolingFan	Double	Spindle cooling fan [Wh]
servoControl	Double	Servo control [Wh]

Currently, this data class is only supported on some newer Brother machine models and the Mock simulated machine.

4.2.9. Feed: Feed Data

Feed data `<field>` data fields:

Field	Type	Description
actFeed	Float	Actual feed rate
cmdFeed	Float	Commanded feed rate
overFeed	Float	Feed override [%]
overRapid	Float	Rapid feed override [%]

Feed data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.10. FeedAndSpindle: Feed and Spindle Override Data

Feed and spindle override data `<field>` data fields:

Field	Type	Description
actFeed	Float	Actual feed rate
actSpindle	Float	Actual speed of the first spindle
cmdFeed	Float	Commanded feed rate
cmdSpindle	Float	Commanded speed of the first spindle
overFeed	Float	Feed override [%]
overRapid	Float	Rapid feed override [%]
overSpindle	Float	Speed override of the first spindle [%]

Feed and spindle override data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.11. GroupCount: Group Machining Count

Group machining count `<field>` data fields:

Field	Type	Description
cumulativeCount	Int32	Group cumulative machining count, i.e., the machining count accumulated since the gateway began automatically collecting data from this group of devices. See 11.1.3. Group Machining Count for details.

4.2.12. GroupCumulativeTime: Group Cumulative Status Time

Group cumulative status time `<field>` data fields:

Field	Type	Description
offTime	Int64	Cumulative off time [s]

Field	Type	Description
waitTime	Int64	Cumulative wait time [s]
emergencyTime	Int64	Cumulative emergency stop time [s]
autoRunTime	Int64	Cumulative auto run time [s]
manualTime	Int64	Cumulative manual (setup) time [s]

The cumulative status times are the sums of each status time, accumulated since the gateway began automatically collecting data from this group, across all devices in the group.

4.2.13. GroupOEE: Group OEE

Group OEE `<field>` data fields:

Field	Type	Description
autoRunCount	Int32	Number of machines currently in auto run within the group
autoRunTime	Int64	Group auto run time [s]
availability	Float	Group availability [%]
emergencyCount	Int32	Number of machines currently in emergency stop within the group
emergencyTime	Int64	Group emergency stop time [s]
manualCount	Int32	Number of machines currently in manual (setup) mode within the group
manualTime	Int64	Group manual (setup) time [s]
offCount	Int32	Number of machines currently off within the group
offTime	Int64	Group off time [s]
waitCount	Int32	Number of machines currently waiting within the group
waitTime	Int64	Group wait time [s]

The group status times here are the sums of each status time, within the monitoring period, for all active machines in the group. Group availability is the availability of the group within the

monitoring period. See 11.1.2. Group OEE Data for details.

4.2.14. LaserPower: Laser Power

Laser power `<field>` data fields:

Field	Type	Description
preset	Float	Preset power [%]
actual	Float	Actual power [%]

4.2.15. Load: Load Data

Load data `<field>` data fields:

Field	Type	Description
axialLoad	Float[]	Servo axis load [%], corresponding in order to the servo axis names in axisName from 1.2.20. Position: Position Data.
spindleLoad	Float[]	Spindle load [%], corresponding in order to the first spindle, second spindle, etc.

Load data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.16. LogHistory: Log History

Log history `<field>` data fields:

Field	Type	Description
msg	String	Log content

4.2.17. OEE: Machine OEE Data

Machine OEE `<field>` data fields:

Field	Type	Description
autoRunTime	Int64	Auto run time [s]
availability	Float	Machine availability [%]
emergencyTime	Int64	Emergency stop time [s]
manualTime	Int64	Manual (setup) time [s]
offTime	Int64	Off time [s]
waitTime	Int64	Wait time [s]

> **Note**
>
> The machine status times here are the status times for the machine within the monitoring period. Machine availability is the availability of the machine within the monitoring period. See 11.1.1. Machine OEE Data for details.

4.2.18. Offset: Tool Offset Data

Tool offset data `<field>` data fields:

Field	Type	Description
toolName	String	Tool name
toolType	String	Tool type
lengthUnit	String	Tool offset length unit
toolNose	Int32	(Imaginary) tool nose position / tool edge type
lengthWear	Double	Length wear
radiusWear	Double	Radius wear
lengthGeom	Double	Length geometry
radiusGeom	Double	Radius geometry
wearX	Double	Length X wear
wearY	Double	Length Y wear
wearZ	Double	Length Z wear

Field	Type	Description
wearR	Double	Radius wear
geomX	Double	Length X
geomY	Double	Length Y
geomZ	Double	Length Z
geomR	Double	Radius

> **Note**
>
> The table lists the common tool offset data fields, which are generally sufficient for most use cases. Some machines support additional, less common data fields that are not described individually here. **When adding a new machine type for the first time, it is recommended to test which fields can be read.** If you need to obtain a field that exists on the machine but is not currently supported, please contact customer support.

Tool offset data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	toolNum	Int32	T	Tool number
2	offsetNum	Int32	O	Offset number
3	channel	Int32	C	Channel number; if absent, represents the default channel

The combination of data tags depends on the machine’s control system; the tag combinations required by different systems are as follows (tool offset data tag combinations):

System Model	toolNum (tool number)	offsetNum (offset number)
Bosunman 博尚	X	O
Brother 兄弟 [General]	X	O
Brother 兄弟 [S series, A00]	O	X
Delta 台达	X	O
Fagor 法格	X	O
Fanuc 发那科	X	O
GSK 广数	X	O
Heidenhain 海德汉	O; “T” column in tool table	X

System Model	toolNum (tool number)	offsetNum (offset number)
Haas 哈斯	X	O
Kede 科德	O	X
Knd 凯恩帝	X	O
Lynuc 铼钠克	X	O
Makino 牧野	X	O
Mazak 马扎克 [Smart, Smooth]	X	O
Mitsubishi 三菱	X	O
Mock 模拟机台	O	O
Okuma 大隈 [P200L, P300L]	X	O; “NO.” column
Okuma 大隈 [P300S(LP)]	O; “TNo” column in the tool data setting table	O; calculated from the “ENo” and “PNo” columns in the tool data setting table: offsetNum = (ENo-1)*20+PNo; if there is no “ENo” column, then offsetNum = PNo
Okuma 大隈 [P200M, P300M, P300S(MP)]	X	O; “NO.” column in the tool length compensation / tool radius compensation table
Siemens 西门子	O; “Position” column in tool table	O; “D” column in tool table
Syntec 新代	X	O

O: required;

X: not required.

If an output quantity has the same name as a tag, the output result will also appear in the tag, but this result does not serve an identifying purpose.

4.2.19. PLC: PLC Data

PLC data includes external PLC data.

PLC data `<field>` data fields:

Field	Type	Description
data	Object[]	PLC data other than String type.
msg	String	String-type PLC data.
pArray	Object[]	Array data obtained through post-processing
pInt64	Int64	Int64 data obtained through post-processing
pDouble	Double	Double data obtained through post-processing
pString	String	String data obtained through post-processing
pBool	Bool	Bool data obtained through post-processing
pTime	String	Time data obtained through post-processing

PLC data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel
2	tag	String	T	Tag name or sequence number

If a tag is defined in the PLC task command (see 11.2.1. PLC Data Task for details), tag takes the value defined in the command; for example, if the tag in the PLC command is “Temperature Data 1”, then tag is “T Temperature Data 1”. If no tag is defined in the PLC task command, tag is T + sequence number, with the sequence number starting from 0. For example, if no tag is defined in the first PLC task command, its sequence number is 0 and tag is T0. When there are multiple PLC acquisition commands, the sequence number accumulates based on the reply data length specified by each command.

If the sequence number of the previous PLC task command is x, then the sequence number of the next one is:

x + the data length obtained by the previous task

Here, data length refers to the number of corresponding 16-bit units for the data.

The main parameters of a PLC task command are: area, start address, data count, and data type.

If the data type is String:

Data length = (data count + 1) / 2, with the result rounded down.

For example:

PLC command `DB1,1,11,String`, data length = (11 + 1) / 2, rounded down to 6.

If the data type is not String:

Data length = single data length * data count, where:

Single data length = (bit count of the single data type + 15) / 16, with the result rounded down.

For example:

PLC command `M,100,5,Bit0`, single data length = (1 + 15) / 16, rounded down to 1. Data length = 1 * 5 = 5.

PLC command `R,100,5,Byte`, single data length = (8 + 15) / 16, rounded down to 1. Data length = 1 * 5 = 5.

PLC command `R,100,5,Int16`, single data length = (16 + 15) / 16, rounded down to 1. Data length = 1 * 5 = 5.

PLC command `R,100,5,Int32`, single data length = (32 + 15) / 16, rounded down to 2. Data length = 2 * 5 = 10.

PLC command `R,100,5,Float`, single data length = (32 + 15) / 16, rounded down to 2. Data length = 2 * 5 = 10.

PLC command `R,100,5,Double`, single data length = (64 + 15) / 16, rounded down to 4. Data length = 4 * 5 = 20.

The sequence number of the first external PLC task continues numbering after the sequence number of the last PLC task.

4.2.20. Position: Position Data

Position data `<field>` data fields:

Field	Type	Description
axisName	String[]	Axis names, e.g. [“x”, “y”, “z”]
abs	Float[]	Absolute coordinates, corresponding in order to the axis names in axisName
dist	Float[]	Remaining distance, corresponding in order to the axis names in axisName
mach	Float[]	Machine coordinates, corresponding in order to the axis names in axisName
rel	Float[]	Relative coordinates, corresponding in order to the axis names in axisName
unit	String[]	Axis coordinate unit
jointCoor	Float[]	Joint coordinates; returned for robot devices only
robotCoor	Float[]	Robot (base) coordinates; returned for robot devices only
userCoor	Float[]	User coordinates; returned for robot devices only
worldCoor	Float[]	World coordinates; returned for robot devices only

Position data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.21. ProgramBlock: Program Block

Program block `<field>` data fields:

Field	Type	Description
currentBlock	String	Content of the currently running program block

Program block `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.22. ProgramInfo: Current Program

Current program `<field>` data fields:

Field	Type	Description
mainPrgmName	String	Current main program name
programSeqNum	String	Sequence number of the currently executing subprogram block
runningPrgName	String	Name of the currently executing subprogram
programStack	String	Program stack; returned on Siemens ([OPC UA], [General] (840D sl / 828D, S7 protocol), [810D/840D]) and Dmg Mori [840D] only

Current program `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.23. SpindleOverLoad: Spindle Overload Data

Spindle overload data `<field>` data fields:

Field	Type	Description
cumulativeTime	UInt32	Cumulative spindle overload time [ms]

The cumulative spindle overload time is the spindle overload time accumulated since the gateway began automatically collecting data from this device. This value is stored in the gateway, bound to the machineID machine identifier, and is never reset.

Spindle overload data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	spindleNo	Int32	S	Spindle number
2	channel	Int32	C	Channel number; if absent, represents the default channel

4.2.24. TimeData: Machine Time Data

Machine time data `<field>` data fields:

Field	Type	Description
cumulativePowerOnTimeMin	Int32	Cumulative power-on time 1 [minutes]
cumulativePowerOnTimeMs	Int32	Cumulative power-on time 2 [ms]
powerOnTimeMin	Int32	Current power-on time 1 [minutes]
powerOnTimeMs	Int32	Current power-on time 2 [ms]
cumulativeOperatingTimeMin	Int32	Cumulative operating time 1 [minutes]
cumulativeOperatingTimeMs	Int32	Cumulative operating time 2 [ms]
operatingTimeMin	Int32	Current operating time 1 [minutes]
operatingTimeMs	Int32	Current operating time 2 [ms]
cumulativeCuttingTimeMin	Int32	Cumulative machining (cutting) time 1 [minutes]
cumulativeCuttingTimeMs	Int32	Cumulative machining (cutting) time 2 [ms]

Field	Type	Description
cuttingTimeMin	Int32	Current machining (cutting) time 1 [minutes]
cuttingTimeMs	Int32	Current machining (cutting) time 2 [ms]
lastCycleTimeMin	Int32	Last cycle time 1 [minutes]
lastCycleTimeMs	Int32	Last cycle time 2 [ms]
currentCycleTimeMin	Int32	Current cycle time 1 [minutes]
currentCycleTimeMs	Int32	Current cycle time 2 [ms]
currentCuttingTimeMin	Int32	Current cutting time 1 [minutes]
currentCuttingTimeMs	Int32	Current cutting time 2 [ms]
remainingCycleTimeMin	Int32	Remaining cycle time 1 [minutes]
remainingCycleTimeMs	Int32	Remaining cycle time 2 [ms]

Each time value is stored as two Int32 values: time 1 is the portion exceeding 1 minute, in minutes; time 2 is the portion under 1 minute, in milliseconds. The actual time value is the sum of time 1 and time 2 after converting them to the same unit. For example:

```
cumulativeOperatingTimeMin = 6683, cumulative operating time 1
cumulativeOperatingTimeMs = 13328, cumulative operating time 2
cumulative operating time = [cumulative operating time 1]*60*1000+
[cumulative operating time 2]
             = 6683*60*1000+13328
             = 400993328 ms
```

For some Siemens systems, the cumulative operating time, cumulative machining (cutting) time, and current machining (cutting) time obtained are always 0, because although the corresponding variables exist in the system, the system does not actually write any value to them.

Some time data can be manually reset to zero on the machine control system.

Which time data each system model supports is listed in 11.5.6.2. Machine Time Data. The endpoints return only the time data the machine’s system supports.

4.2.25. ToolLife: Tool Life Data

Tool life data `<field>` data fields:

Common Data

Field	Type	Description
timeLimit	Int32	Life limit [s], maximum available time
currentTime	Int32	Current life [s], time already used
prewarningTime	Int32	Prewarning life [s]; an alarm is raised when the current life reaches this value
countLimit	Int32	Life limit [count], maximum available count
currentCount	Int32	Current life [count], count already used
prewarningCount	Int32	Prewarning life [count]; an alarm is raised when the current life reaches this value
wearLimit	Double	Life limit [machine’s default length unit], maximum allowable wear
currentWear	Double	Current life [machine’s default length unit], current wear amount
prewarningWear	Double	Prewarning life [machine’s default length unit]; an alarm is raised when the current life reaches this value

Raw Data

Field	Type	Description
rawToolLifeType	String	Tool life type
rawToolLifeUnit	String	Time unit, hereinafter abbreviated as [time]
rawToolLifeStatus	String	Tool life status
rawTimeLimit	Double	Life limit [time]
rawTimeLimit1	Double	Maximum tool life [time]
rawTimeLimit2	Double	Maximum life of the called tool [time]

Field	Type	Description
rawCurrentTime	Double	Current life [time]
rawOverTime	Double	Tool life exceeded [time]
rawOverCount	Int32	Tool life exceeded [count]
rawPrewarningTime	Double	Prewarning life [time]
rawRemainingTime	Double	Remaining life [time]
rawPrewarningRemainingTime	Double	Prewarning remaining life [time]
rawRemainingCount	Int32	Remaining life [count]
rawPrewarningRemainingCount	Int32	Prewarning remaining life [count]
rawRemainingWear	Double	Remaining life [machine’s default length unit]
rawPrewarningRemainingWear	Double	Prewarning remaining life [machine’s default length unit]
inventoryNum	String	Tool identification code

For the meaning of each field, see the examples in 5.5.1.23. readToolLife - Read Tool Life and 5.5.1.25. readToolLifeDetails - Read Tool Life Details.

Raw data is returned in the following cases:

1. Using the 5.5.1.25. readToolLifeDetails - Read Tool Life Details, 5.5.1.26. batchReadToolLifeDetails - Batch Read Tool Life Details, 5.5.2.1. readTaskData - Read Machine Task Data, or 5.5.2.2. batchReadTaskData - Batch Read Machine Task Data API.
2. The tool life task is enabled, and tool life details are enabled in Task Configuration - Tool Life Monitoring Settings; the tool life task then returns these fields.

Tool life data `<tag>` data tags:

No.	Tag	Type	Abbreviation	Description
1	toolGroupNum	Int32	G	Tool group number
2	toolNum	Int32	T	Tool number
3	toolOffsetNum	Int32	O	Offset number
4	toolIndex	Int32	I	Sequence number

The combination of data tags depends on the machine’s control system; the tag combinations required by different systems are as follows (tool life data tag combinations):

System Model	toolGroupNum (tool group number)	toolIndex (index within group)	toolNum (tool number)	toolOffsetNum (offset number)
Brother 兄弟	X	X	O	X
Fanuc 发那科	O	O	X	X
Gsk 广数	O	O	X	X
Heidenhain 海德汉	X	X	O	X
Kede 科德	X	X	O	X
Lynuc 铼钠克	X	X	O	X
Makino 牧野	O	O	X	X
Mazak 马扎克 [Smart, Smooth]	X	X	O; “TNo” column	X
Mock 模拟机台	X	X	O	O
Okuma 大隈 [P200L, P300L]	X	X	O; “TOOL” column	X
Okuma 大隈 [P300S(LP)]	X	X	O; “TNo” column	O; “ENo” column; if there is no “ENo”, then offsetNum = 1
Okuma 大隈 [P200M, P300M, P300S(MP)]	X	X	O; “TOOL NO.” column	X
Siemens 西门子	X	X	O; “Position” column in tool table	O; “D” column in tool table
Syntec 新代	X	X	O	X

O: required;

X: not required.

If an output quantity has the same name as a tag, the output result will also appear in the tag, but this result does not serve an identifying purpose. For example, for Fanuc tool life, the tags toolGroupNum and toolIndex are used to identify the tool, while the tag toolNum is an output quantity; in this case, the result of toolNum will appear in the tag.

Which life types (count, time, wear) each system model supports, and which raw data fields it returns, are listed in 11.5.6.3. Tool Life Data.

4.3. Variable Definitions

4.3.1. Machine Status

4.3.1.1. CNCStatus: Running Status

> **Note**
>
> The variable is named CNCStatus, but it is not limited to CNC machines — it also applies to the status of equipment such as robots and laser cutters.

Running status:

Status Code	Return String	Running Status	Description
0	OFF	Powered off or offline	
1	EMERGENCY	Emergency stop	Emergency stop button pressed
2	WAIT	Standby: general standby	1. In Automatic (including Remote Automatic) mode, the program is not running and the specific status cannot be identified; 2. Unrecognized operating mode
3	WAIT_HOLD	Standby: program paused	Program paused state in Automatic (including Remote Automatic) mode
4	WAIT_STOP	Standby: program stopped	Program stopped state in Automatic (including Remote Automatic) mode
5	WAIT_IDLE	Standby: idle	Idle or ready state in Automatic mode or Remote Automatic mode
10	AUTO_RUN	Automatic running	Program running in Automatic mode
11	AUTO_RUN_REMOTE	Remote automatic running	Program running in Remote Automatic mode

Status Code	Return String	Running Status	Description
20	MANUAL	Setup: general setup	Non-automatic state; the specific setup status cannot be identified
21	MANUAL_EDIT	Setup: program editing	
22	MANUAL_MDI	Setup: manual data input	
23	MANUAL_HANDLE	Setup: manual handwheel feed	
24	MANUAL_HANDLE_TCH	Setup: manual handwheel feed teaching	
25	MANUAL_JOG	Setup: manual jog feed	
26	MANUAL_JOG_TCH	Setup: manual jog feed teaching	
27	MANUAL_INC	Setup: incremental feed/step	
28	MANUAL_REF	Setup: return to reference point	
29	MANUAL_REMOTE	Setup: remote	
30	MANUAL_MSTR	Setup: tool retract/repositioning	
31	MANUAL_DRY_RUN	Setup: dry run	
32	MANUAL_RAPID	Setup: rapid positioning	
33	MANUAL_SINGLE_BLOCK	Setup: single block operation	
34	MANUAL_MDI_RUN	Setup: running program in MDI mode	
35	MANUAL_HOLD	Setup: paused	Paused in non-automatic (including remote automatic) mode

Note

The status code is used only for MODBUS communication.

4.3.1.2. AlarmStatus: Alarm Status

Return String	Alarm Status
ALARM	Alarm present
NO_ALARM	No alarm
UNKNOWN	Unknown (alarm status not obtained; not supported on some models)

4.3.1.3. EmergencyStatus: Emergency Stop Status

Return String	Emergency Stop Status
EMG	Emergency stop
NOT_EMG	Not emergency stop
RESET	Emergency stop reset
WAIT	Wait (FANUC FS35i only)
UNKNOWN	Unknown (emergency stop status not obtained; the device or model does not support reading it)

4.3.1.4. DryRunStatus: Dry Run Status

Return String	Dry Run Status
DRY_RUN	Dry run
NOT_DRY_RUN	Not dry run

4.3.2. Alarm Information

4.3.2.1. AlarmLevel: Alarm Level

Alarm levels, in descending order of priority, are Error (ERR), Warning (WRN), and Message (INF). Users can refer to 3.5.7. Alarm Monitor Settings to set alarm levels by keyword and filter out alarms below the minimum alarm level. Alarms with no configured level default to the WRN (Warning) level.

Return String	Alarm Level
ERR	Error
WRN	Warning
INF	Message

4.3.3. Service Information

4.3.3.1. ServiceRunningStatus: Service Running Status

Running status of each gateway service.

Return Int32	Running Status
0	Disabled
1	Initiated
2	Ready
3	Running
4	Delayed
5	Error

5. HTTP Communication

5.1. Basics

The service runs on the standard HTTP port (port 80). Unless otherwise specified, the base address for requests is **/api/cnc**. A standard request URL has the following format:

```
http://{Gateway IP}/api/cnc/{Interface Name}?MachineID={Target Machine
machineID}
```

For example, if the gateway’s IP address is 192.168.100.1, and you want to obtain the running status of the main channel of a machine tool connected to this gateway with IP 192.168.1.10 and MachineID 1010 (for an explanation of machineID, see 4.1.1. machineID Machine Identifier), you should use the `/readCNCStatus` interface, and the corresponding request URL is:

```
http://192.168.100.1/api/cnc/readCNCStatus?MachineID=1010
```

Request bodies use Content-Type application/json by default. File and stream interfaces (such as sendFileStream, upload-license and upload-alarm-mapping-file) use application/octet-stream or text/plain to upload binary/text content. Only POST/PUT requests read the request body.

Except for a few file-related interfaces, most interfaces return HTTP data of type application/json.

All data must be encoded in UTF-8.

In the descriptions below, some interfaces require additional input parameters in the request URL.

5.1.1. Forwarding Requests Through the Cloud Platform or Hub

HTTP requests can also be forwarded to the target gateway through a cloud platform or Hub server. First complete the gateway’s 3.6.1. Cloud Settings or 3.6.7. Hub Settings, click **Save**, go back to Home and click **Restart Service**, then confirm that the gateway is connected to the corresponding server.

1. Replace the gateway address in the request URL with the corresponding cloud platform or Hub server address. Use the protocol (HTTP or HTTPS) and port configured for that server. Keep the interface path, parameters, request method and request body unchanged.

2. Add an `AccessToken` request header. Set its value to the gateway token entered in **Token** in the corresponding Cloud Settings or Hub Settings. The server uses this token to forward the request to that gateway.

For example, to obtain the running status of a machine’s main channel through the cloud platform or Hub, use the following request URL:

```
http://{Cloud platform or Hub server address}/api/cnc/readCNCStatus?
machineID=1010
```

Add the following request header:

```
AccessToken: {Gateway token}
```

Replace the server address, gateway token and `machineID` in the example with their actual values. Enter the gateway token directly as the `AccessToken` value, without a `Bearer` prefix.

5.2. Error Handling

Whenever an error occurs in a request, an error object is returned as follows:

```
{
  "errorCode": 1,
  "errorMsg": "Unexpected Error."
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message
statusMsg	String	(Optional) Error detail providing additional context; not returned when empty

When the operation succeeds, the corresponding data is returned, and the HTTP status code is 200 in this case.

> **Note**
>
> The HTTP status code is indicative only. Some informational-severity errors are also returned with HTTP 200, so a client must determine success from whether errorCode in the response body is 0, not from the HTTP status code alone.

When an error occurs, an error code is returned, and the HTTP status code is 400 (bad or unsupported request), 401 (unauthorized), 404 (interface or path not found), 409 (conflict), 502 (machine unreachable) or 503 (any other error, the default). Common error codes are as follows:

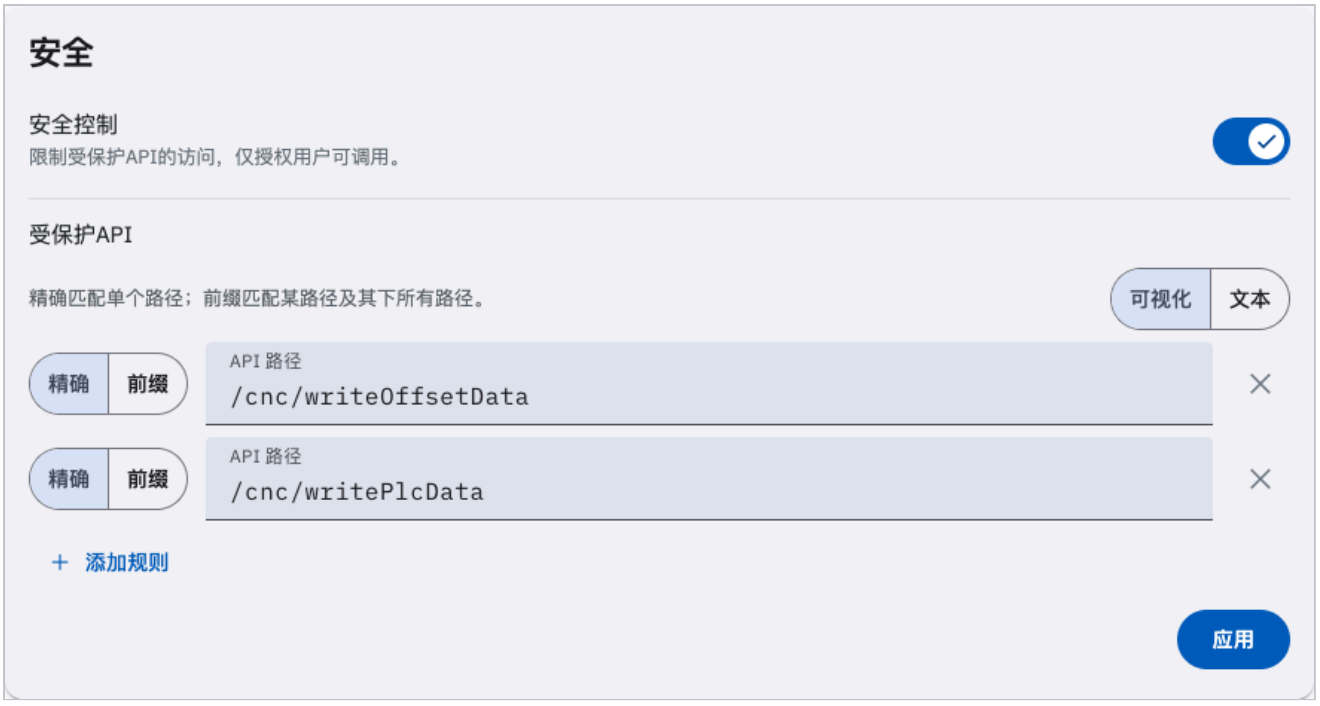
Common Error Codes

Error Code	Message	Description
0	Success	Operation executed successfully
3	Machine off or offline	Unable to communicate with the machine tool; the machine tool is powered off or there is a network hardware fault (machine tool network port, switch, network cable, etc.)
7	No permission	The interface is protected or out of authorization scope, or the authentication information is invalid.
101	Function not supported by the current system	The machine tool control system does not support this interface
102	Machine network fault, unable to obtain data	Able to communicate with the machine tool, but unable to obtain data; there is an issue with the machine tool’s network settings
125	Invalid file path	The input file path is invalid
142	File already exists	A file with the same name already exists in the target directory
143	File is in use	The file to be operated on is currently in use
144	File is write-protected	The file to be operated on is write-protected
158	File does not exist	The target file does not exist in the target directory

Error Code	Message	Description
10003	machineID does not exist	The machineID is incorrect, the corresponding machine is not activated, or the gateway service was not restarted after editing the machine’s IP or activation status
10006	Interface not authorized	To use an unauthorized interface, please contact BIVROST
10017	Invalid request	A request parameter is missing, has the wrong type, or cannot be parsed
10051	WiFi not available	The gateway has no usable wireless adapter or the WLAN service is unavailable; statusMsg carries the cause

5.3. Authentication Methods

When security control is enabled in the gateway settings (enabled by default), users must provide authentication information to use protected APIs. If security control is turned off, user authentication can be skipped and all endpoints can be used directly; however, if the IP white list is also enabled, the source IP must still be in the white list (except for `/api/auth/login`, `/api/misc/product-details`, and JWT-authenticated requests). See 2.3.3. IP White List for details.



The gateway supports the following two authentication methods: the JWT method and the secret key method.

5.3.1. JWT Method

The user logs in with a username and password via the authentication endpoint `/api/auth/login` to obtain a JWT token for that user. The token is valid for 24 hours; once it expires, the user must log in again to obtain a new token. Include `Authorization: Bearer <token>` in the request header to use the protected APIs available to that user’s permissions. If the logged-in user has administrator privileges, all endpoints except protected APIs can be used.

5.3.2. Secret Key Method

Generate a secret key on the gateway’s Settings > Manage Users > User Security Settings page; the secret key is valid permanently. Include `Authorization: Bearer <secret key>` in the request header to use the protected APIs available to that user’s permissions. If the logged-in user has administrator privileges, all endpoints except protected APIs can be used.

5.3.3. IP White List

The IP white list is a second access-control gate, independent of security control. When the HTTP service is enabled and the IP white list is turned on, the source IP of every `/api` request must be in the white list; otherwise HTTP 401 is returned with errorCode 7 and errorMsg `IP not in white list.`.

The following requests are exempt from the IP white list:

- `/api/auth/login`, the login endpoint;
- `/api/misc/product-details`, the product information endpoint;
- requests authenticated with the JWT method (`Authorization: Bearer <token>`).

Requests authenticated with the secret key method are still subject to the IP white list check.

The white list toggle and address list are described by the enableIpWhiteList and ipWhiteList parameters in 5.9.6. Communication Settings.

5.4. Authentication Endpoints

Base address for authentication endpoints: **/api/auth/**.

5.4.1. login

```
POST /api/auth/login
```

Example request body, application/json

```
{
  "username": "username",
  "password": "password"
}
```

Request Parameter	Type	Description
username	String	(Required) Username
password	String	(Required) Password

Example response

```
{
  "token":
"a2JhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJ1bmlxdWV.bmFtZSI6ImFkbWluIiwibmFtZWl
}
```

Response Parameter	Type	Description
token	String	(Required) User token. Valid for 24 hours; once it expires, log in again to obtain a new token.

5.4.2. change-password

This endpoint is only available under the JWT authentication method.

```
POST /api/auth/change-password
```

Example request body, application/json

```
{
  "currentPassword": "currentPassword",
  "newPassword": "newPassword"
}
```

Request Parameter	Type	Description
currentPassword	String	(Required) Current password
newPassword	String	(Required) New password

The username is taken from the JWT token in the request header (`Authorization: Bearer <token>`); it is not passed in the request body.

Example response

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code; 0 indicates success.
errorMsg	String	(Required) Error message

5.5. Data Read/Write Interface

The data read/write interface lets you read or write machine data, or read machine group data, through the gateway. The data read/write interface is divided into two categories: direct read/write and cached read.

5.5.1. Direct Read/Write

When using direct read/write interfaces, the gateway communicates with the target machine immediately to retrieve or write data.

5.5.1.1. readAlarm - Read Alarm Information

```
GET /api/cnc/readAlarm?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response Example

```
{
  "alarmMsg": [
    "警报 1",
    "警报 2"],
  "alarmLevel": [
    "WRN",
    "WRN"]
}
```

Response Parameter	Type	Description
alarmMsg	String[]	(Required) The machine’s current alarm content. If there is no alarm, an empty Array is returned.

Response Parameter	Type	Description
alarmLevel	String[]	(Required) Alarm level, corresponding in order to the alarm content. If there is no alarm, an empty Array is returned. See Alarm Level.

> **Note**
>
> Starting from version 1.8.1, the alarm timestamp has been removed. To obtain the start time of a machine alarm, use the interface 5.7.1.2. alarm - Machine Alarm Data Analysis.

5.5.1.2. readCNCStatus - Read Machine Status

```
GET /api/cnc/readCNCStatus?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "cncStatus": "MANUAL_EDIT",
  "alarmStatus": "ALARM",
  "alarmLevel": "WRN",
  "channel": 2
}
```

Response Parameter	Type	Description
cncStatus	String	(Required) Running status, see Running Status
alarmStatus	String	(Required) Alarm status, see Alarm Status

Response Parameter	Type	Description
alarmLevel	String	Alarm level. If there are multiple alarms, the highest level among them is taken. If there is no alarm, this field is not returned. See Alarm Level
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

The adjusted running status `adjustedStatus`, cumulative shutdown time `offTime`, cumulative standby time `waitTime`, cumulative emergency-stop time `emergencyTime`, cumulative auto-run time `autoRunTime`, cumulative manual-adjustment time `manualTime`, and other values derived by the gateway’s post-processing are not currently supported over HTTP. Once an automatic collection task has been enabled, these can be obtained through the interface 5.5.2.1. readTaskData - Read Machine Task Data or 5.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

5.5.1.3. readCNCStatusDetails - Read Machine Status Details

In addition to the same general machine status data returned by 2.5.1.2. readCNCStatus - Read Machine Status, this interface also returns raw, machine-specific data.

```
GET /api/cnc/readCNCStatusDetails?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "mode": "AUTO_RUN",
  "programStatus": "RUNNING",
  "emergencyStatus": "NOT_EMG",
```

```
  "dryRunStatus": "DRY_RUN",
  "cncStatus": "AUTO_RUN",
  "alarmStatus": "ALARM",
  "alarmLevel": "WRN",
  "channel": 2
}
```

Response Parameter	Type	Description
mode	String	Operating mode
programStatus	String	Program status
emergencyStatus	String	Emergency-stop status, see Emergency-Stop Status
dryRunStatus	String	Dry-run status, see Dry-Run Status
cncStatus	String	(Required) Running status, see Running Status
alarmStatus	String	(Required) Alarm status, see Alarm Status
alarmLevel	String	Alarm level. If there are multiple alarms, the highest level among them is taken. If there is no alarm, this field is not returned. See Alarm Level
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

The adjusted running status `adjustedStatus`, cumulative shutdown time `offTime`, cumulative standby time `waitTime`, cumulative emergency-stop time `emergencyTime`, cumulative auto-run time `autoRunTime`, cumulative manual-adjustment time `manualTime`, and other values derived by the gateway’s post-processing are not currently supported over HTTP. Once an automatic collection task has been enabled, these can be obtained through the interface 5.5.2.1. readTaskData - Read Machine Task Data or 5.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

5.5.1.4. readCount - Read Machining Count

```
GET /api/cnc/readCount?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response Example

```
{
  "count": 1052
}
```

Response Parameter	Type	Description
count	Int32	(Required) Machining count obtained from the machine’s control system

The cumulative machining count `cumulativeCount` computed by the gateway, and the machining count `currentCount` for the current monitoring window, are not currently supported over HTTP. Once an automatic collection task has been enabled, these can be obtained through the interface 5.5.2.1. readTaskData - Read Machine Task Data or 5.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

5.5.1.5. readCurrentToolNumber - Read Current Tool Number

```
GET /api/cnc/readCurrentToolNumber?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "toolNumber": "6",
```

```
  "toolOffsetNum": "6",
  "channel": 2
}
```

Response Parameter	Type	Description
toolNumber	String	(Required) Current tool number
toolOffsetNum	String	Current tool offset number
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

5.5.1.6. readEnergyConsum - Read Energy Consumption Data

Currently supported for some newer Brother machine models and simulated machines only.

```
GET /api/cnc/readEnergyConsum?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response Example

```
{
  "allServoAxes": 127144.0,
  "coolantPump": 41577.0,
  "ctsPump": 6507.0,
  "chipFlusherPump": 0.0,
  "cyclonePump": 0.0,
  "system24V": 46270.0,
  "ncControl": 10099.0,
  "lcdBacklight": 2884.0,
  "chipConveyor": 0.0,
  "screwConveyor": 0.0,
  "autoLubrication": 239.0,
  "spindleCoolingFan": 7212.0,
  "servoControl": 23806.0
```

```
}
```

Response Parameter	Type	Description
allServoAxes	Double	Energy consumption of all servo axes [Wh]
coolantPump	Double	Coolant pump energy consumption [Wh]
ctsPump	Double	CTS pump energy consumption [Wh]
chipFlusherPump	Double	Chip flusher pump energy consumption [Wh]
cyclonePump	Double	Cyclone filter pump energy consumption [Wh]
system24V	Double	24V system energy consumption [Wh]
ncControl	Double	NC control energy consumption [Wh]
lcdBacklight	Double	LCD backlight energy consumption [Wh]
chipConveyor	Double	Chip conveyor energy consumption [Wh]
screwConveyor	Double	Screw conveyor energy consumption [Wh]
autoLubrication	Double	Automatic lubrication device (or automatic greasing device) energy consumption [Wh]
spindleCoolingFan	Double	Spindle cooling fan energy consumption [Wh]
servoControl	Double	Servo control energy consumption [Wh]

5.5.1.7. readFeed - Read Feed Data

Laser cutting/welding machines only.

```
GET /api/cnc/readFeed?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "overFeed": 100.0,
  "actFeed": 80.0,
  "cmdFeed": 80.0,
  "overRapid": 100.0,
  "channel": 2
}
```

Response Parameter	Type	Description
overFeed	Float	Feed override [%]
actFeed	Float	Actual feed rate
cmdFeed	Float	Commanded feed rate
overRapid	Float	Rapid traverse override [%]
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

5.5.1.8. readFeedAndSpindle - Read Feed and Spindle Speed Data

CNC machines only.

```
GET /api/cnc/readFeedAndSpindle?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "overSpindle": 150.0,
  "actSpindle": 3000.0,
  "cmdSpindle": 3000.0,
  "overFeed": 100.0,
  "actFeed": 80.0,
  "cmdFeed": 80.0,
  "overRapid": 100.0,
  "channel": 2
}
```

Response Parameter	Type	Description
overSpindle	Float	Spindle speed override [%]
actSpindle	Float	Actual spindle speed
cmdSpindle	Float	Commanded spindle speed
overFeed	Float	Feed override [%]
actFeed	Float	Actual feed rate
cmdFeed	Float	Commanded feed rate
overRapid	Float	Rapid traverse override [%]
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

When there are multiple spindles, only the speed data of the first spindle is returned.

5.5.1.9. readLaserPower - Read Laser Power

Laser cutting/welding machines only.

```
GET /api/cnc/readLaserPower?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response Example

```
{
  "preset": 100.0,
  "actual": 100.0
}
```

Response Parameter	Type	Description
preset	Float	Preset power [%]
actual	Float	Actual power [%]

5.5.1.10. readLoad - Read Load Data

CNC machines only.

```
GET /api/cnc/readLoad?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "spindleLoad": [
      100.0,
      0.0],
  "axialLoad": [
      50.1,
```

```
        12.4,
        0.0],
    "channel": 2
}
```

Response Parameter	Type	Description
spindleLoad	Float[]	Spindle load [%]
axialLoad	Float[]	Servo axis load [%]
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0

When there are multiple spindles, the entries in the spindle load data correspond in order to the first spindle, second spindle, and so on. The servo axis load values correspond one-to-one, in order, with the `axisName` values returned by 5.5.1.17. readPosition - Read Position Data.

5.5.1.11. readLog - Read Log Information

Currently supported for Fanuc robots, ABB robots, and simulated machines.

```
GET /api/cnc/readLog?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
type	String	Type, possible values: Alarm (alarm log), Operation (operation log), Default (default), defaults to Default. ABB robots support Operation; Fanuc robots support Alarm; simulated machines support all types, and Default is equivalent to Operation. Multiple types can be specified separated by a semicolon `;` (e.g. `type=Alarm;Operation`), in which case the returned msgs is the logs of each type merged in order; an invalid value returns the error code CNCWRAPPER_INVALID_COMMAND.
count	Int32	Number of log entries. Retrieves the specified number of most recent log entries; defaults to 0, meaning all logs are retrieved.
path	String	Log file path. Supported by ABB robots only; if specified, the entire content of that file is returned as the only element of msgs, and the type parameter is ignored.

Response Example (ABB robot, Operation)

```
{
  "msgs": [
    "{\"SeqNo\":\"79\",\"Type\":\"I  \",\"Code\":\"10011\",\"Title\":\" 电机上电(ON) 状态\",\"Date\":\"2025-04-02 T 10:54:40\",\"Description\":\"系统处于电机上电 (ON) 状态。\",\"Args\":[]}",
    "{\"SeqNo\":\"78\",\"Type\":\"I  \",\"Code\":\"10010\",\"Title\":\" 电机下电 (OFF) 状态\",\"Date\":\"2025-04-02 T 10:54:39\",\"Description\":\"系统处于电机下电 (OFF) 状态。从手动模式切换至自动模式，或者程序执行过程中电机上电 (ON) 电路被打开后，系统就会进入此状态。\",\"Args\":[]}"]
}
```

Response Example (Fanuc robot, Alarm)

```
{
  "msgs": [
    "
{\"AlarmID\":24,\"AlarmNumber\":212,\"CauseAlarmID\":0,\"CauseAlarmNumber\":
06-26T04:04:46\",\"AlarmMessage\":\"SYST-212 需要应用 DCS 参数
\",\"CauseAlarmMessage\":\"\",\"SeverityMessage\":\"STOP.G\"}",
    "
{\"AlarmID\":0,\"AlarmNumber\":0,\"CauseAlarmID\":0,\"CauseAlarmNumber\":0,\
26T04:04:46\",\"AlarmMessage\":\"重置\",\"CauseAlarmMessage\":\"\",\"Severity
}
```

Response Example (simulated machine, Operation)

```
{
  "msgs": [
    "2025-07-07 13:24:45 Status: WAIT",
    "2025-07-07 13:24:45 Machine power on"]
}
```

Response Example (simulated machine, Alarm)

```
{
  "msgs": [
    "2025-07-07 13:25:05 Alarm: 102 ALARM 2, 13:25",
    "2025-07-07 13:25:05 Alarm: 101 ALARM 1, 13:25"
  ]
}
```

Response Parameter	Type	Description
msgs	String[]	(Required) Log content. Note: the log content format differs between machine types.

5.5.1.12. readCNCCommonStatus - Read Machine Common Status

In addition to the same machine status data returned by 2.5.1.3. readCNCStatusDetails - Read Machine Status Details, this interface also returns status data specific to certain control systems.

```
GET /api/cnc/readCNCCommonStatus?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel

Response Example

```
{
  "mode": "AUTO_RUN",
  "programStatus": "RUNNING",
  "emergencyStatus": "NOT_EMG",
  "dryRunStatus": "DRY_RUN",
  "cncStatus": "AUTO_RUN",
  "alarmStatus": "ALARM",
  "alarmLevel": "WRN",
  "channel": 2
}
```

Response Parameter	Type	Description
mode	String	Operating mode
programStatus	String	Program status
emergencyStatus	String	Emergency-stop status, see Emergency-Stop Status
dryRunStatus	String	Dry-run status, see Dry-Run Status
cncStatus	String	(Required) Running status, see Running Status
alarmStatus	String	(Required) Alarm status, see Alarm Status
alarmLevel	String	Alarm level. If there are multiple alarms, the highest level among them is taken. If there is no alarm, this field is not returned. See Alarm Level

Response Parameter	Type	Description
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0
mode2	String	Manual mode selection (the secondary mode on Rexroth [OPC UA])
readyStatus	String	Ready status
alarmOnlyStatus	String	Alarm status (excluding warnings)
emergencySetStatus	String	Emergency-stop setting status
lightColor1	String	Color of signal light 1
lightStatus1	String	Status of signal light 1
lightColor2	String	Color of signal light 2
lightStatus2	String	Status of signal light 2
lightColor3	String	Color of signal light 3
lightStatus3	String	Status of signal light 3
lightColor4	String	Color of signal light 4
lightStatus4	String	Status of signal light 4

> **Note**
> mode2 and the parameters below it are returned only on some system models; other machines do not return them, see 11.5.6.1. Machine Status.

5.5.1.13. init Initialize Machine Connection

This interface performs a connection initialization (preprocessing) for the target machine, used to establish or validate communication with the machine before reading or writing data. The interface returns no data, only the execution result.

```
GET /api/cnc/init?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier. See 4.1.1. machineID Machine Identifier

Example response

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

Note

When initialization fails, error code 3 (machine off or offline) is returned, meaning communication with the machine could not be established.

5.5.1.14–5.5.1.22. Direct Read/Write: Tool Offsets, Coordinates, Program & PLC

This page continues from 5.5.1. Direct Read/Write and covers endpoints 5.5.1.14–5.5.1.22: tool offset data, coordinate data, program block/program information, PLC data read/write, and machine time data.

5.5.1.14. readOffsetData - Read Tool Offset Data

```
GET /api/cnc/readOffsetData?
machineID=MACHINEID&channel=CHANNEL&toolNum=TOOLNUM&offsetNum=OFFSETNUM
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.
toolNum, offsetNum	Int32	Supply toolNum (tool number) or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in Tool offset data `<tag>` data tags. Refer to the per-system tag combination table in Tool offset data tag combinations to supply the correct combination of request parameters.

Example 1, Heidenhain:

Request to get the tool offset data for T (tool number) 1 on the machine with machineID 1010:

```
/api/cnc/readOffsetData?machineID=1010&toolNum=1
```

Response example

```
{
  "toolNum": 1,
  "toolName": "MILL_D2_ROUGH",
  "toolType": "9",
  "lengthWear": 0.6,
  "radiusWear": 0.5,
  "lengthGeom": 30.0,
  "radiusGeom": 1.0,
  "radius2Wear": 0.2,
  "radius2Geom": 2.0
}
```

Example 2, Siemens:

Request to get the tool offset data for position (tool number) 2, D (offset number) 1, on the machine with machineID 1010:

```
/api/cnc/readOffsetData?machineID=1010&toolNum=2&offsetNum=1
```

Response example

```
{
  "toolNum": 2,
  "offsetNum": 1,
  "toolName": "ROUGHING_T80 A",
  "toolType": "500",
  "toolNose": 3,
  "wearX": 0.88,
  "wearZ": 0.78,
  "wearR": 0.05,
  "geomX": 55.0,
  "geomZ": 39.0,
  "geomR": 0.8
}
```

Example 3, other systems:

Request to get the tool offset data for offset number 1 on channel 2 of the machine with machineID 1010:

```
/api/cnc/readOffsetData?machineID=1010&offsetNum=1&channel=2
```

Response example

```
{
  "offsetNum": 1,
  "lengthUnit": "MM",
  "toolNose": 3,
  "lengthWear": 0.0001,
  "radiusWear": 0.03,
  "lengthGeom": 5.0,
  "radiusGeom": 0.2,
  "channel": 2
}
```

Response Parameter	Type	Description
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0
toolNum	Int32	Tool number
offsetNum	Int32	Offset number
toolName	String	Tool name
toolType	String	Tool type
lengthUnit	String	Tool offset length unit
toolNose	Int32	(Imaginary) tool nose position / cutting edge type
lengthWear	Double	Length wear
radiusWear	Double	Radius wear
lengthGeom	Double	Length geometry
radiusGeom	Double	Radius geometry
wearX	Double	Length X wear
wearY	Double	Length Y wear
wearZ	Double	Length Z wear
wearR	Double	Radius wear

Response Parameter	Type	Description
geomX	Double	Length X
geomY	Double	Length Y
geomZ	Double	Length Z
geomR	Double	Radius

> **Note**
>
> The table lists the common tool offset parameters, but not all of them. Some machines have less common offset settings that are not individually documented here. **We recommend testing which offset parameters can be read the first time a new machine type is added.** If, in actual use, you need to obtain an offset setting parameter that the machine has but this endpoint does not yet support, please contact support.

5.5.1.15. batchReadOffsetData - Batch Read Tool Offsets

This endpoint is the batch version of 2.5.1.12. readOffsetData - Read Tool Offset Data. By supplying a list of target tools in the request body, you can read multiple sets of data in a single call.

```
POST /api/cnc/batchReadOffsetData?machineID=MACHINEID
```

Request body example, application/json

```
[
  {
    "toolNum": 1,
    "offsetNum": 1,
    "channel": 2
  },
  {
    "toolNum": 1,
    "offsetNum": 2,
    "channel": 2
  }
]
```

The request parameters are the same as 2.5.1.12. readOffsetData - Read Tool Offset Data, but they are supplied in two places: machineID is a query string parameter, while channel, toolNum, and offsetNum are fields of each element in the request body array.

Query string parameters:

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Fields of each element in the request body array:

Request Parameter	Type	Description
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.
toolNum, offsetNum	Int32	Supply toolNum (tool number) or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in Tool offset data `<tag>` data tags. Refer to the per-system tag combination table in Tool offset data tag combinations to supply the correct combination of request parameters.

> **Note**
> channel takes effect only within each element of the request body array. A channel supplied on the query string has no effect for this endpoint and is ignored.

Response example

```
[
  {
    "toolNum": 1,
    "offsetNum": 1,
    "lengthUnit": "MM",
    "toolNose": 3,
    "lengthWear": 0.0001,
```

```
    "radiusWear": 0.03,
    "lengthGeom": 5.0,
    "radiusGeom": 0.2,
    "channel": 2
  },
  {
    "toolNum": 1,
    "offsetNum": 2,
    "lengthUnit": "MM",
    "toolNose": 4,
    "lengthWear": 0.01,
    "radiusWear": 0.02,
    "lengthGeom": 6.0,
    "radiusGeom": 0.3,
    "channel": 2
  }
]
```

The response parameters are the same as 2.5.1.12. readOffsetData - Read Tool Offset Data. If an individual request fails, the corresponding position in the returned array contains the error information for that request.

Response Parameter	Type	Description
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0
toolNum	Int32	Tool number
offsetNum	Int32	Offset number
toolName	String	Tool name
toolType	String	Tool type
lengthUnit	String	Tool offset length unit
toolNose	Int32	(Imaginary) tool nose position / cutting edge type
lengthWear	Double	Length wear
radiusWear	Double	Radius wear
lengthGeom	Double	Length geometry

Response Parameter	Type	Description
radiusGeom	Double	Radius geometry
wearX	Double	Length X wear
wearY	Double	Length Y wear
wearZ	Double	Length Z wear
wearR	Double	Radius wear
geomX	Double	Length X
geomY	Double	Length Y
geomZ	Double	Length Z
geomR	Double	Radius

> **Note**
>
> The table lists the common tool offset parameters, but not all of them. Some machines have less common offset settings that are not individually documented here. **We recommend testing which offset parameters can be read the first time a new machine type is added.** If, in actual use, you need to obtain an offset setting parameter that the machine has but this endpoint does not yet support, please contact support.

5.5.1.16. writeOffsetData - Write Tool Offset Data

By default, the gateway blocks this endpoint (the protected API list in the security settings includes `/cnc/writeOffsetData` by default); before use, enable the corresponding option on the **Settings** page of the gateway management console, otherwise calls return UNAUTHORIZED_ACCESS.

```
POST /api/cnc/writeOffsetData?
machineID=MACHINEID&channel=CHANNEL&toolNum=TOOLNUM&offsetNum=OFFSETNUM
```

Request body example, application/json

```
{
  "toolNose": 3,
  "lengthWear": 0.0001,
  "radiusWear": 0.03,
  "lengthGeom": 5.0,
```

```
  "radiusGeom": 0.2
}
```

> **Note**
> Include only the parameters you want to write in the request body. Parameters that should remain unchanged do not need to be included.

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.
toolNum, offsetNum	Int32	Supply toolNum (tool number) or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in Tool offset data `<tag>` data tags. Refer to the per-system tag combination table in Tool offset data tag combinations to supply the correct combination of request parameters.
toolName	String	Tool name; on Siemens systems this is a read-only parameter and cannot be written
toolType	String	Tool type; on Okuma systems this is a read-only parameter and cannot be written
lengthUnit	String	Tool offset length unit
toolNose	Int32	(Imaginary) tool nose position / cutting edge type
lengthWear	Double	Length wear
radiusWear	Double	Radius wear
lengthGeom	Double	Length geometry
radiusGeom	Double	Radius geometry
wearX	Double	Length X wear

Request Parameter	Type	Description
wearY	Double	Length Y wear
wearZ	Double	Length Z wear
wearR	Double	Radius wear
geomX	Double	Length X
geomY	Double	Length Y
geomZ	Double	Length Z
geomR	Double	Radius

> **Note**
>
> The table lists the common tool offset parameters, but not all of them. Some machines have less common offset settings that are not individually documented here. Generally, any offset parameter that can be read via 2.5.1.12. readOffsetData - Read Tool Offset Data can also be written, but on some machines certain offset parameters are read-only and cannot be written, such as the toolName parameter on Siemens systems. **We recommend first testing reads of tool offset data to determine which offset parameters a machine supports when adding a new machine system model, then testing writes to check for any read-only parameters.** If, in actual use, you need to write an offset parameter that the machine has but this endpoint does not yet support, please contact support.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code; 0 indicates success.
errorMsg	String	(Required) Error message

5.5.1.17. readPosition - Read Coordinate Data

```
GET /api/cnc/readPosition?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.

Response example

```
{
  "axisName": [
        "X",
        "Y",
        "Z"],
  "unit": [
        "mm",
        "mm",
        "mm"],
  "mach": [
        150.12,
        0,
        -31.35],
  "abs": [
        150.12,
        0,
        -31.35],
  "rel": [
        150.12,
        0,
        -31.35],
  "dist": [
        9.88,
        0,
        31.35],
  "channel": 2
```

```
}
```

Response Parameter	Type	Description
axisName	String[]	(Required) Axis name
unit	String[]	Unit
mach	Float[]	Machine coordinates
abs	Float[]	Absolute coordinates
rel	Float[]	Relative coordinates
dist	Float[]	Remaining distance
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0

Each coordinate data array corresponds by index to the axis names in axisName.

5.5.1.18. readProgramBlock - Read Program Block

This endpoint reads the program block currently running on the machine.

```
GET /api/cnc/readProgramBlock?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.

Response example

```
{
  "currentBlock": "Y70.800",
  "channel": 2
}
```

Response Parameter	Type	Description
currentBlock	String	(Required) Content of the currently running program block
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0

5.5.1.19. readProgramInfo - Read Machining Program Information

```
GET /api/cnc/readProgramInfo?machineID=MACHINEID&channel=CHANNEL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.

Response example

```
{
  "mainPrgmName": "1234",
  "runningPrgName": "abcd",
  "programSeqNum": "N30",
  "programStack": "A/B/1234/abcd",
  "channel": 2
}
```

Response Parameter	Type	Description
mainPrgmName	String	(Required) Main program name
runningPrgName	String	(Required) Name of the currently executing subprogram
programSeqNum	String	Sequence number of the currently executing subprogram block

Response Parameter	Type	Description
programStack	String	Program stack; returned on Siemens and Dmg Mori [840D] only, see 4.2.22. ProgramInfo: Current Program
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0

For Fanuc and Makino, a leading zero in a standard program name (letter O + digits) is automatically stripped — for example, a program shown on the machine as O0010 is returned as O10.

5.5.1.20. readPlcData - Read PLC Data

Before using this endpoint, you need to know the PLC area address and type of the target data. Consult the equipment manual or contact the equipment manufacturer for this information.

```
GET /api/cnc/readPlcData?
machineID=MACHINEID&area=AREA&start=START&count=COUNT&type=TYPE
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
area	String	(Required) Area. Format is `{areaName}|{parameter1}|{paramter2}`..., where areaName is the area name and parameter is an additional parameter. areaName is a PLC area such as R, X, Y, etc., obtainable from the machine manual or manufacturer; see PLC Data - Common Areas and Parameters for the area format. When using the standard protocol MODBUS for communication, areaName is a segment name such as 0x, 1x, 3x, 4x, etc. The gateway also supports some special area names, including the $MACRO$ macro variable, $PARAM$ system parameter, and TAG tag, etc.; see PLC Data - Special Areas and Parameters. The parameter field supplies additional data, e.g. axis=AXIS to specify an axis number, level=LEVEL to specify an execution level, type=TYPE to specify a type, and name=NAME to specify a tag name.
start	String	Start address. If not base 10, add the appropriate prefix per PLC Address Numeric Base Prefixes.
count	Int32	(Required) Data count. For non-String types, this is the number of data items; for the String type, this is the length of the target string (in bytes).
type	String	(Required) Data type. See PLC Data Types for the currently supported data types.
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.

PLC Data - Common Areas and Parameters

Device System	Area	area Example	Notes
All device systems	Ordinary area	`X`	The X area of the PLC device
All device systems	Ordinary area	`X|s=S`	To communicate with another PLC connected to the currently connected PLC, you can supply station, slot, or slaveID (MODBUS); if s is not supplied, it defaults to communicating with the currently connected PLC. This parameter is currently supported only by some PLC devices.

PLC Data - Special Areas and Parameters

System Model	Area	area Example	Notes
Bosunman	MODBUS address	`0x` or `1x` or `4x`	
Brother	Macro variable	`$MACRO$`	
Delta	Macro variable	`$MACRO$`	
Fagor [8035,8040,8055]	Macro variable (precise to 5 decimal places)	`$MACRO$|type=local|level=LEVEL`	Local arithmetic parameter; if the nesting level is not 1, supply the execution level, e.g. level=7
Fagor [8035,8040,8055]	Macro variable (precise to 5 decimal places)	`$MACRO$|type=global`	Global arithmetic parameter
Fagor [8035,8040,8055]	Tag (variable name)	`$TAG$|name=NAME`	Supply the name tag (variable name)
Fagor [8060,8065,8070]	Macro variable (precise to 4 decimal places)	`$MACRO$|type=local|level=LEVEL`	Local arithmetic parameter; if the nesting level is not 1, supply the execution level, e.g. level=7

System Model	Area	area Example	Notes
Fagor [8060,8065,8070]	Macro variable (precise to 4 decimal places)	`$MACRO$|type=global`	Global arithmetic parameter
Fagor [8060,8065,8070]	Macro variable (precise to 4 decimal places)	`$MACRO$|type=common`	Common arithmetic parameter
Fagor [8060,8065,8070]	Tag (variable name)	`$TAG$|name=NAME`	Supply the name tag (variable name)
Fanuc	Diagnostic data	`$DIAG$`	
Fanuc	Macro variable	`$MACRO$`	
Fanuc	System parameter	`$PARAM$`	Supports Byte, Int16, Int32, and Double types, among others.
Fanuc	System parameter	`$PARAM$|axis=AXIS`	Some parameters require an axis number, e.g. axis=1
Gsk [980, 988]	Macro variable	`$MACRO$`	
Gsk [Ethernet, serial-to-Ethernet, 986 V4.15]	Macro variable	`$MACRO$`	
Gsk [Ethernet, serial-to-Ethernet, 986 V4.15]	MODBUS address	`0x` or `1x` or `4x`	
Haas [general-purpose]	Macro variable	`$MACRO$`	
Hnc	Macro variable	`$MACRO$`	
Jingdiao	Macro variable	`$MACRO$`	
Kede	Macro variable	`$MACRO$`	# variable
Kede	Tag (variable name)	`$TAG$|name=NAME`	PLC variable, e.g. `$TAG$|name=PLC_PRG.C_Workpiece`
Kede	Tag (variable name)	`$TAG$|name=NAME|type=cncVar`	cncVar-type variable, i.e. a variable used for interaction between the PLC and the CNC system, e.g. `$TAG$|name=PEW4190.1|type=cncVar`
Lynuc	Macro variable	`$MACRO$`	

System Model	Area	area Example	Notes
Lynuc	Tag (variable name)	`$TAG$|name=NAME`	Supply the name tag (variable name), e.g. `$TAG$|name=MOU31.25.28`
Makino	Diagnostic data	`$DIAG$`	
Makino	Macro variable	`$MACRO$`	
Makino	System parameter	`$PARAM$`	Supports Byte, Int16, Int32, and Double types, among others.
Makino	System parameter	`$PARAM$|axis=AXIS`	Some parameters require an axis number, e.g. axis=1
Mazak [Smart, Smooth]	Macro variable	`$MACRO$|type=R`	R variable
Mitsubishi	Macro variable (local or common variable)	`$MACRO$`	Default execution level 0
Mitsubishi	Macro variable (local or common variable)	`$MACRO$|level=LEVEL`	If the execution level is not 0, supply the execution level, e.g. level=1
Mock simulated machine	Macro variable	`$MACRO$`	
Mock simulated machine	System parameter	`$PARAM$`	
Mock simulated machine	Tag (variable name)	`$TAG$|name=NAME`	Supply the name tag (variable name)
Siemens	Macro variable	`$MACRO$|type=R`	R parameter
Siemens	Macro variable	`$MACRO$|type=RG`	RG parameter
Siemens [general-purpose]	Tag (variable name)	`$TAG$|area=AREA|block=BLOCK|name=NAME|areaNo=AREANO|row=ROW|column=COLUMN`	S7 variable; supply the parameters Area, Block (Component), VariableName, AreaNo. (default 1), Row (default 1), Column (default 1), e.g.: `$TAG$|area=C|block=AUXFU|name=status|areaNo=2|row=3`

System Model	Area	area Example	Notes
Siemens [OPC UA]	Tag (variable name)	`$TAG$|name=NAME|ns=NS`	OPC UA server variable address; supply the parameters name (variable name) and ns (name space index, default 2), e.g.: `$TAG$|name=/Channel/State/actParts|ns=2`
Syntec	Macro variable	`$MACRO$`	
Allen-Bradley	Tag (variable name)	`$TAG$|name=NAME`	Supply the name tag (variable name)

Note

Note 1: The $MACRO$ macro variable generally supports only the Double data type.

Note 2: When using the TAG method, the Start start address does not need to be supplied.

PLC Address Numeric Base Prefixes

Base	Prefix (digit 0 + letter)	Example
Base 2	0b	0b1001
Base 8	0	03671
Base 10	none	123
Base 16	0x	0x5A7

PLC Data Types

Data Type	Bit	Byte	SByte	Int16	UInt16	Float	Int32	UInt32	Int64	UInt64	Double	String
Data length	1 bit	8 bits	8 bits	16 bits	16 bits	32 bits	32 bits	32 bits	64 bits	64 bits	64 bits	variable
Bosunman	O	O	O	O	O	O	O	O			O	O
Brother	O			O								
Delta	O	O	O	O	O	O	O	O	O	O	O	O
Fagor [8035,8040,8055]	O	O	O	O	O	O	O	O			O	O
Fagor [8060,8065,8070]							O				O	O
Fanuc		O	O	O	O		O	O			*	
Gsk [988,980]		O									*	

Data Type	Bit	Byte	SByte	Int16	UInt16	Float	Int32	UInt32	Int64	UInt64	Double	String
Gsk [Ethernet, serial-to-Ethernet, 986 V4.15]	O	O	O	O	O	O	O	O			O	O
Haas [general-purpose]				O	O	O	O	O	O	O	O	O
Heidenhain	O		O	O			O					O
Hnc		O		O			O	O			O	
Jingdiao		O		O	O		O	O			O	
Kede	O	O	O	O	O	O	O	O	O	O	O	
Knd		O	O	O	O		O	O				
Lnc	O					O	O					
Lynuc								O			*	
Makino		O	O	O	O		O	O			*	
Mazak [Smart, Smooth]				O	O		O	O			*	
Mitsubishi	O	O		O	O	O	O	O			*	
Mock simulated machine	O	O	O	O	O	O	O	O			O	O
Siemens	O	O		O	O	O	O	O			O	O
Syntec	O	O		O			O				*	
External module					O	O						

O: Supported.

*: This data type is supported only for macro variables (local variables, common variables, etc.).

The Bit type can be written as Bit0–Bit7 to specify the bit position within a byte or word; Bit is equivalent to Bit0.

On Syntec, the I/O/C/S/A bit areas must use the Bit type (Bit0–Bit7).

Request Examples

Examples 1-2 are ordinary area examples.

Example 1, reading an Int32 array:

Request to get 8 Int32 values from the machine with machineID 1010, starting at PLC address R0 (inclusive):

```
/api/cnc/readPLCData?machineID=1010&area=R&start=0&count=8&type=Int32
```

Response format:

```
{
```

```
  "data": [
     2147483647,
     -616240881,
     1351548766,
     1408080714,
     -1677592641,
     681965706,
     1002992212,
     -1588442413]
}
```

Example 2, reading a String:

Request to get a String value of up to 16 bytes from the machine with machineID 1010, starting at PLC address S0 (inclusive):

```
/api/cnc/readPLCData?
machineID=1010&area=S&start=0&count=16&type=String
```

Response format:

```
{
  "msg": "Test string"
}
```

Examples 3-6 are special area examples.

Example 3, reading $MACRO$:

Request to get macro variables (local variable, common variable, etc.) numbered 1 through 3 on the machine with machineID 1010:

```
/api/cnc/readPLCData?
machineID=1010&area=$MACRO$&start=1&count=3&type=Double
```

Response format:

```
{
  "data": [
```

```
    -1.7976931348623157E+308,
    0.0,
    1.2345]
}
```

Here, -1.7976931348623157E+308 is the minimum value of the Double type, indicating the variable has not been set.

Example 4, reading $PARAM$:

Request to get system parameters numbered 100 through 102, of type Byte, on the machine with machineID 1010:

```
/api/cnc/readPLCData?
machineID=1010&area=$PARAM$&start=100&count=3&type=Byte
```

Response format:

```
{
  "data": [
    4,
    1,
    0]
}
```

Example 5, reading a single value by tag:

Request to get 1 value of type UInt32, with tag name A.B, on the machine with machineID 1010:

```
/api/cnc/readPLCData?
machineID=1010&area=$TAG$|name=A.B&count=1&type=UInt32
```

Example 6, reading multiple values from an array by tag:

Request to get elements 0 through 7 of the Float array with tag name AB.C, on the machine with machineID 1010:

```
/api/cnc/readPLCData?
MachineID=1010&area=$TAG$|name=AB.C[0]&count=8&type=Float
```

Response Parameter	Type	Description
data	Array[Object]	When the type parameter in the request is any type other than String, the response body outputs data, an array of the type specified in the request, with length equal to the requested count.
msg	String	When the type parameter in the request is String, the response body outputs msg. Trailing blanks at the end of msg are automatically trimmed. If the actual String length read exceeds the length defined in the request parameters, error errorCode 179 is returned.
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0

Note

Note 1: Some devices impose a limit on the maximum amount of PLC data that can be read in a single request, to avoid affecting machine operation.

Note 2: The Syntec system automatically selects the output type based on the input address. As a result, even if the input type is incorrect, data may still be returned, but its output type will not match the type specified in the request.

Note 3: On some systems, such as Brother, the Int16 type can accommodate results of types such as Bit and Byte.

5.5.1.21. writePlcData - Write PLC Data

Writing PLC data carries a degree of risk. Be sure to understand the risks and write data only when it is safe to do so. By default, the gateway blocks this endpoint; before use, enable the corresponding option on the **Settings** page of the gateway management console.

Before using this endpoint, you need to know the PLC area address and type of the target data. Consult the equipment manual or contact the equipment manufacturer for this information.

```
POST /api/cnc/writePlcData?
machineID=MACHINEID&area=AREA&start=START&type=TYPE
```

Request body example, application/json

When the data being written is not of type String:

```
{
  "data": [
    2147483647,
    -616240881,
    1351548766,
    1408080714,
    -1677592641,
    681965706,
    1002992212,
    -1588442413]
}
```

When the data being written is of type String:

```
{
  "msg": "Test string"
}
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
area	String	(Required) Area. Format is `{areaName}|{parameter1}|{paramter2}`..., where areaName is the area name and parameter is an additional parameter. areaName is a PLC area such as R, X, Y, etc., obtainable from the machine manual or manufacturer; see PLC Data - Common Areas and Parameters for the area format. When using the standard protocol MODBUS for communication, areaName is a segment name such as 0x, 1x, 3x, 4x, etc. The gateway also supports some special area names, including the $MACRO$ macro variable, $PARAM$ system parameter, and TAG tag, etc.; see PLC Data - Special Areas and Parameters. The parameter field supplies additional data, e.g. axis=AXIS to specify an axis number, level=LEVEL to specify an execution level, type=TYPE to specify a type, and name=NAME to specify a tag name.
start	String	Start address. If not base 10, add the appropriate prefix per PLC Address Numeric Base Prefixes.
type	String	(Required) Data type. See PLC Data Types for the currently supported data types.
data	Array[Object]	When the type parameter in the request is any type other than String, supply data in the request body, an array of the type specified in the request.
msg	String	When the type parameter in the request is String, supply msg in the request body.

Request Parameter	Type	Description
channel	Int32	Machine channel number. Defaults to 0 (the main channel) if not supplied.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code; 0 indicates success.
errorMsg	String	(Required) Error message

5.5.1.22. readTimeData - Read Machine Time Data

```
GET /api/cnc/readTimeData?machineID=MACHINEID
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response example

```
{
  "cumulativePowerOnTimeMin": 34194,
  "cumulativePowerOnTimeMs": 0,
  "powerOnTimeMin": 2944,
  "powerOnTimeMs": 0,
  "cumulativeOperatingTimeMin": 6683,
  "cumulativeOperatingTimeMs": 13328,
  "operatingTimeMin": 0,
  "operatingTimeMs": 0,
```

```
  "cumulativeCuttingTimeMin": 0,
  "cumulativeCuttingTimeMs": 0,
  "cuttingTimeMin": 0,
  "cuttingTimeMs": 0,
  "lastCycleTimeMin": 4,
  "lastCycleTimeMs": 43000,
  "currentCycleTimeMin": 0,
  "currentCycleTimeMs": 0,
  "currentCuttingTimeMin": 0,
  "currentCuttingTimeMs": 0,
  "remaingingCycleTimeMin": 0,
  "remaingingCycleTimeMs": 0
}
```

Response Parameter	Type	Description
cumulativePowerOnTimeMin	Int32	Cumulative power-on time, part 1 [minutes]
cumulativePowerOnTimeMs	Int32	Cumulative power-on time, part 2 [milliseconds]
powerOnTimeMin	Int32	Current power-on time, part 1 [minutes]
powerOnTimeMs	Int32	Current power-on time, part 2 [milliseconds]
cumulativeOperatingTimeMin	Int32	Cumulative operating time, part 1 [minutes]
cumulativeOperatingTimeMs	Int32	Cumulative operating time, part 2 [milliseconds]
operatingTimeMin	Int32	Current operating time, part 1 [minutes]
operatingTimeMs	Int32	Current operating time, part 2 [milliseconds]
cumulativeCuttingTimeMin	Int32	Cumulative machining (cutting) time, part 1 [minutes]
cumulativeCuttingTimeMs	Int32	Cumulative machining (cutting) time, part 2 [milliseconds]
cuttingTimeMin	Int32	Current machining (cutting) time, part 1 [minutes]

Response Parameter	Type	Description
cuttingTimeMs	Int32	Current machining (cutting) time, part 2 [milliseconds]
lastCycleTimeMin	Int32	Last cycle time, part 1 [minutes]
lastCycleTimeMs	Int32	Last cycle time, part 2 [milliseconds]
currentCycleTimeMin	Int32	Current cycle time, part 1 [minutes]
currentCycleTimeMs	Int32	Current cycle time, part 2 [milliseconds]
currentCuttingTimeMin	Int32	Current cutting time, part 1 [minutes]
currentCuttingTimeMs	Int32	Current cutting time, part 2 [milliseconds]
remainingCycleTimeMin	Int32	Remaining cycle time, part 1 [minutes]
remainingCycleTimeMs	Int32	Remaining cycle time, part 2 [milliseconds]

This endpoint returns only the time data supported by the machine’s system. Each time value is computed by converting and summing its corresponding time 1 and time 2; see 4.2.24. TimeData: Machine Time Data for the calculation method.

5.5.1.23–5.5.1.26. Direct Read/Write: Tool Life

This page continues 5.5.1. Direct Read/Write, covering tool-life-related endpoints (5.5.1.23–5.5.1.26).

5.5.1.23. readToolLife: Read Tool Life

```
GET /api/cnc/readToolLife?
machineID=MACHINEID&groupNum=GROUPNUM&toolIndex=TOOLINDEX&toolNum=TOOLNUM&of
```

Request Parameter	Type	Description
machineID	String	(Required) Identifier of the target machine, see 4.1.1. machineID: Machine Identifier
groupNum	Int32	Supply groupNum (tool group number), or toolIndex (index within the group), or toolNum (tool number), or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in 4.2.25. ToolLife: Tool Life Data, where the groupNum parameter corresponds to toolGroupNum in the tags, offsetNum corresponds to toolOffsetNum in the tags, and the other parameters share names with the tags. Refer to the per-system tag combination table in 4.2.25. ToolLife: Tool Life Data to determine which combination of request parameters to supply.
toolIndex	Int32	Same as above (described together with the groupNum row)
toolNum	Int32	Same as above (described together with the groupNum row)
offsetNum	Int32	Same as above (described together with the groupNum row)

Examples 1–2 are Fanuc examples.

Example 1, Fanuc count-based:

(The original here shows a screenshot of the Fanuc tool life screen: group 00000001 (max tool count: 32), type 1 (1: count, 2: minutes), life 100, count 21; tool list numbers 01–04, T codes 00000001/00000008/00000002/00000032, H codes 001/000/002/120, D codes 001/000/002/008.)

The type shown above is 1, indicating count-based. To get the tool life data for the machine with machineID 1010 shown above, tool group number 1, index within group 4, the request is as follows:

```
/api/cnc/readToolLife?machineID=1010&groupNum=1&toolIndex=4
```

Response example

```
{
  "toolNum": 32,
  "toolIndex": 4,
  "toolGroupNum": 1,
  "countLimit": 100,
  "currentCount": 21
}
```

toolNum is the T code shown above; toolIndex is the number shown above; toolGroupNum is the group shown above; countLimit is the life shown above, in counts; currentCount is the count shown above, in counts.

Example 2, Fanuc time-based:

(The original here shows a screenshot of the Fanuc tool life screen: group 00000002 (max tool count: 32), type 2 (1: count, 2: minutes), life 1000, count 23; tool list numbers 01–02, T codes 00000004/00000000, H codes 004/000, D codes 004/000.)

The type shown above is 2, indicating time-based. To get the tool life data for the machine with machineID 1010 shown above, tool group number 2, index within group 2, the request is as follows:

```
/api/cnc/readToolLife?machineID=1010&groupNum=2&toolIndex=2
```

Response example

```
{
  "toolNum": 8,
  "toolIndex": 2,
  "toolGroupNum": 1,
  "timeLimit": 60000,
  "currentTime": 1380
}
```

toolNum is the T code shown above; toolIndex is the number shown above; toolGroupNum is the group shown above; timeLimit is the life shown above, in seconds; currentTime is the count shown above, in seconds. The original unit of time shown is minutes; the response data is uniformly converted to seconds.

Examples 3–4 are Siemens examples.

Example 3, Siemens time-based:

(The original here shows a screenshot of the Siemens SINUMERIK OPERATE tool wear screen (MAGAZIN1): position 1, tool name ROUGHING_T80 A, ST 1, D 1, Δ length Z 0.780, Δ radius 0.750, TC column T, tool life 30.0, target value 30.0, warning value 6.2.)

The "TC" column shown above is T, indicating time-based. To get the offset data for the machine with machineID 1010 shown above, position (tool number) 1, D (offset number) 1, the request is as follows:

```
/api/cnc/readToolLife?machineID=1010&toolNum=1&offsetNum=1
```

Response example

```
{
  "toolNum": 1,
  "toolOffsetNum": 1,
  "timeLimit": 1800,
  "currentTime": 0,
  "prewarningTime": 1428
}
```

toolNum is the position shown above; toolOffsetNum is the D shown above; timeLimit is the target value shown above, in seconds; currentTime is the difference between the target value and the tool life shown above, in seconds; prewarningTime is the difference between the target

value and the warning value shown above, in seconds. The original unit of time shown is minutes; the response data is uniformly converted to seconds.

In general, the system issues a warning when currentTime (current life) is greater than or equal to prewarningTime (warning life). On Siemens systems, the configured tool life actually represents remaining usable life, which decreases continuously with machining, and an alarm is raised when the tool life falls below the warning value. To keep the definition consistent, the current life and warning life output by this endpoint have been converted accordingly.

Example 4, Siemens wear-based:

(The original here shows a screenshot of the Siemens SINUMERIK OPERATE tool wear screen (MAGAZIN1): position 3, tool name FINISHING_T35 A, ST 1, D 1, Δ length Z 0.000, Δ radius 0.000, TC column W, wear amount 2.000, rated value 2.000, warning value 1.000.)

The "TC" column shown above is W, indicating wear-based. To get the offset data for the machine with machineID 1010 shown above, position (tool number) 3, D (offset number) 1, the request is as follows:

```
/api/cnc/readToolLife?machineID=1010&toolNum=3&offsetNum=1
```

Response example

```
{
  "toolNum": 3,
  "toolOffsetNum": 1,
  "wearLimit": 2.0,
  "currentWear": 0.0,
  "prewarningWear": 1.0
}
```

toolNum is the position shown above; toolOffsetNum is the D shown above; wearLimit is the rated value shown above; currentWear is the difference between the rated value and the wear amount shown above; prewarningWear is the difference between the rated value and the warning value shown above. Wear-related quantities are in the machine's default length unit, mm or inch.

Response Parameter	Type	Description
toolGroupNum	Int32	Tool group number
toolIndex	Int32	Index within the group

Response Parameter	Type	Description
toolNum	Int32	Tool number
toolOffsetNum	Int32	Offset number
timeLimit	Int32	Life limit [seconds], the maximum usable time
currentTime	Int32	Current life [seconds], the time already used
prewarningTime	Int32	Warning life [seconds], an alarm is raised when the current life reaches this value
countLimit	Int32	Life limit [counts], the maximum usable count
currentCount	Int32	Current life [counts], the count already used
prewarningCount	Int32	Warning life [counts], an alarm is raised when the current life reaches this value
wearLimit	Double	Life limit [machine’s default length unit], the maximum allowable wear
currentWear	Double	Current life [machine’s default length unit], the current wear amount
prewarningWear	Double	Warning life [machine’s default length unit], an alarm is raised when the current life reaches this value

Note

This table lists the generic tool life data. To standardize across all machine systems, some parameters are converted from the raw data. To obtain the raw data, use 2.5.1.23. readToolLifeDetails: Read Tool Life Details.

For the life types supported by each system model, see 4.2.25. ToolLife: Tool Life Data.

5.5.1.24. batchReadToolLife: Batch Read Tool Life

This endpoint is the batch version of 2.5.1.21. readToolLife: Read Tool Life. By supplying a list of target tools in the request body, multiple sets of data can be read in a single call.

```
POST /api/cnc/batchReadToolLife?machineID=MACHINEID
```

Request body example, application/json

```
[
  {
    "groupNum": 1,
    "toolIndex": 4
  },
  {
    "groupNum": 2,
    "toolIndex": 3
  }
]
```

The request parameters are the same as 2.5.1.21. readToolLife: Read Tool Life.

Request Parameter	Type	Description
machineID	String	(Required) Identifier of the target machine, see 4.1.1. machineID: Machine Identifier

Request Parameter	Type	Description
groupNum	Int32	Supply groupNum (tool group number), or toolIndex (index within the group), or toolNum (tool number), or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in 4.2.25. ToolLife: Tool Life Data, where the groupNum parameter corresponds to toolGroupNum in the tags, offsetNum corresponds to toolOffsetNum in the tags, and the other parameters share names with the tags. Refer to the per-system tag combination table in 4.2.25. ToolLife: Tool Life Data to determine which combination of request parameters to supply.
toolIndex	Int32	Same as above (described together with the groupNum row)
toolNum	Int32	Same as above (described together with the groupNum row)
offsetNum	Int32	Same as above (described together with the groupNum row)

Response example

```
[
  {
    "toolNum": 32,
    "toolIndex": 4,
    "toolGroupNum": 1,
    "countLimit": 100,
    "currentCount": 21
  },
  {
    "toolNum": 23,
    "toolIndex": 3,
    "toolGroupNum": 2,
    "countLimit": 200,
```

```
    "currentCount": 58
  }
]
```

The response parameters are the same as 2.5.1.21. readToolLife: Read Tool Life. If an individual request fails, the corresponding position in the returned array holds that request’s error information.

Response Parameter	Type	Description
toolGroupNum	Int32	Tool group number
toolIndex	Int32	Index within the group
toolNum	Int32	Tool number
toolOffsetNum	Int32	Offset number
timeLimit	Int32	Life limit [seconds], the maximum usable time
currentTime	Int32	Current life [seconds], the time already used
prewarningTime	Int32	Warning life [seconds], an alarm is raised when the current life reaches this value
countLimit	Int32	Life limit [counts], the maximum usable count
currentCount	Int32	Current life [counts], the count already used
prewarningCount	Int32	Warning life [counts], an alarm is raised when the current life reaches this value
wearLimit	Double	Life limit [machine’s default length unit], the maximum allowable wear
currentWear	Double	Current life [machine’s default length unit], the current wear amount
prewarningWear	Double	Warning life [machine’s default length unit], an alarm is raised when the current life reaches this value

> **Note**
> This table lists the generic tool life data. To standardize across all machine systems, some data is converted from the raw data.

For the life types supported by each system model, see 4.2.25. ToolLife: Tool Life Data.

5.5.1.25. readToolLifeDetails: Read Tool Life Details

In addition to the generic tool life data also returned by 2.5.1.21. readToolLife: Read Tool Life, this endpoint also returns the unprocessed raw data. Some non-generic tool life data is added to this endpoint’s response parameters.

```
GET /api/cnc/readToolLifeDetails?
machineID=MACHINEID&groupNum=GROUPNUM&toolIndex=TOOLINDEX&toolNum=TOOLNUM&of
```

The request parameters are the same as 2.5.1.21. readToolLife: Read Tool Life.

Request Parameter	Type	Description
machineID	String	(Required) Identifier of the target machine, see 4.1.1. machineID: Machine Identifier
groupNum	Int32	Supply groupNum (tool group number), or toolIndex (index within the group), or toolNum (tool number), or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in 4.2.25. ToolLife: Tool Life Data, where the groupNum parameter corresponds to toolGroupNum in the tags, offsetNum corresponds to toolOffsetNum in the tags, and the other parameters share names with the tags. Refer to the per-system tag combination table in 4.2.25. ToolLife: Tool Life Data to determine which combination of request parameters to supply.

Request Parameter	Type	Description
toolIndex	Int32	Same as above (described together with the groupNum row)
toolNum	Int32	Same as above (described together with the groupNum row)
offsetNum	Int32	Same as above (described together with the groupNum row)

Examples 1–2 are Fanuc examples.

Example 1, Fanuc count-based:

(The original here shows a screenshot of the Fanuc tool life screen: group 00000001 (max tool count: 32), type 1 (1: count, 2: minutes), life 100, count 21; tool list numbers 01–04, T codes 00000001/00000008/00000002/00000032, H codes 001/000/002/120, D codes 001/000/002/008.)

The type shown above is 1, indicating count-based. To get the tool life data for the machine with machineID 1010 shown above, tool group number 1, index within group 4, the request is as follows:

```
/api/cnc/readToolLifeDetails?machineID=1010&groupNum=1&toolIndex=4
```

Response example

```
{
  "toolNum": 32,
  "toolIndex": 4,
  "toolGroupNum": 1,
  "countLimit": 100,
  "currentCount": 21,
  "rawToolLifeType": "Count",
  "rawToolLifeStatus": "Enabled"
}
```

toolNum is the T code shown above; toolIndex is the number shown above; toolGroupNum is the group shown above; countLimit is the life shown above, in counts; currentCount is the count shown above, in counts. rawToolLifeType is the type shown above; rawToolLifeStatus is the status flag shown above.

Example 2, Fanuc time-based:

(The original here shows a screenshot of the Fanuc tool life screen: group 00000002 (max tool count: 32), type 2 (1: count, 2: minutes), life 1000, count 23; tool list numbers 01–02, T codes 00000004/00000000, H codes 004/000, D codes 004/000.)

The type shown above is 2, indicating time-based. To get the tool life data for the machine with machineID 1010 shown above, tool group number 2, index within group 2, the request is as follows:

```
/api/cnc/readToolLifeDetails?machineID=1010&groupNum=2&toolIndex=2
```

Response example

```
{
  "toolNum": 8,
  "toolIndex": 2,
  "toolGroupNum": 1,
  "timeLimit": 60000,
  "currentTime": 1380,
  "rawToolLifeType": "Time",
  "rawToolLifeUnit": "Minute",
  "rawToolLifeStatus": "Enabled",
  "rawTimeLimit": 1000.0,
  "rawCurrentTime": 23.0
}
```

toolNum is the T code shown above; toolIndex is the number shown above; toolGroupNum is the group shown above; timeLimit is the life shown above, in seconds; rawTimeLimit is the life shown above, in minutes; currentTime is the count shown above, in seconds; rawCurrentTime is the count shown above, in minutes. rawToolLifeType is the type shown above; rawToolLifeUnit is the raw data’s time unit; rawToolLifeStatus is the status flag shown above.

Examples 3–4 are Siemens examples.

Example 3, Siemens time-based:

(The original here shows a screenshot of the Siemens SINUMERIK OPERATE tool wear screen (MAGAZIN1): position 1, tool name ROUGHING_T80 A, ST 1, D 1, Δ length Z 0.780, Δ radius 0.750, TC column T, tool life 30.0, target value 30.0, warning value 6.2.)

The “TC” column shown above is T, indicating time-based. To get the offset data for the machine with machineID 1010 shown above, position (tool number) 1, D (offset number) 1, the

request is as follows:

```
/api/cnc/readToolLifeDetails?machineID=1010&toolNum=1&offsetNum=1
```

Response example

```
{
  "toolNum": 1,
  "toolOffsetNum": 1,
  "timeLimit": 1800,
  "currentTime": 0,
  "prewarningTime": 1428,
  "rawToolLifeType": "Time",
  "rawToolLifeUnit": "Minute",
  "rawTimeLimit": 30.0,
  "rawRemainingTime": 30.0,
  "rawPrewarningRemainingTime": 6.2
}
```

toolNum is the position shown above; toolOffsetNum is the D shown above; timeLimit is the target value shown above, in seconds; rawTimeLimit is the target value shown above, in minutes; currentTime is the difference between the target value and the tool life shown above, in seconds; rawRemainingTime is the tool life shown above, in minutes; prewarningTime is the difference between the target value and the warning value shown above, in seconds; rawPrewarningRemainingTime is the warning value shown above, in minutes. rawToolLifeType is the TC shown above; rawToolLifeUnit is the raw data’s time unit.

In general, the system issues a warning when currentTime (current life) is greater than or equal to prewarningTime (warning life). On Siemens systems, the configured tool life actually represents rawRemainingTime, the remaining usable life, which decreases continuously with machining, and an alarm is raised when the tool life falls below rawPrewarningRemainingTime, the warning value.

Example 4, Siemens wear-based:

(The original here shows a screenshot of the Siemens SINUMERIK OPERATE tool wear screen (MAGAZIN1): position 3, tool name FINISHING_T35 A, ST 1, D 1, Δ length Z 0.000, Δ radius 0.000, TC column W, wear amount 2.000, rated value 2.000, warning value 1.000.)

The "TC" column shown above is W, indicating wear-based. To get the offset data for the machine with machineID 1010 shown above, position (tool number) 3, D (offset number) 1, the request is as follows:

```
/api/cnc/readToolLifeDetails?machineID=1010&toolNum=3&offsetNum=1
```

Response example

```
{
  "toolNum": 3,
  "toolOffsetNum": 1,
  "wearLimit": 2.0,
  "currentWear": 0.0,
  "prewarningWear": 1.0,
  "rawToolLifeType": "Wear",
  "rawRemainingWear": 2.0,
  "rawPrewarningRemainingWear": 1.0
}
```

toolNum is the position shown above; toolOffsetNum is the D shown above; wearLimit is the rated value shown above; currentWear is the difference between the rated value and the wear amount shown above; rawRemainingWear is the wear amount shown above; prewarningWear is the difference between the rated value and the warning value shown above; rawPrewarningRemainingWear is the warning value shown above. rawToolLifeType is the TC shown above. Wear-related quantities are in the machine's default length unit, mm or inch.

Response Parameter	Type	Description
Generic Data		
toolGroupNum	Int32	Tool group number
toolIndex	Int32	Index within the group
toolNum	Int32	Tool number
toolOffsetNum	Int32	Offset number
timeLimit	Int32	Life limit [seconds], the maximum usable time
currentTime	Int32	Current life [seconds], the time already used

Response Parameter	Type	Description
prewarningTime	Int32	Warning life [seconds], an alarm is raised when the current life reaches this value
countLimit	Int32	Life limit [counts], the maximum usable count
currentCount	Int32	Current life [counts], the count already used
prewarningCount	Int32	Warning life [counts], an alarm is raised when the current life reaches this value
wearLimit	Double	Life limit [machine’s default length unit], the maximum allowable wear
currentWear	Double	Current life [machine’s default length unit], the current wear amount
prewarningWear	Double	Warning life [machine’s default length unit], an alarm is raised when the current life reaches this value
Raw Data		
rawToolLifeType	String	Tool life type
rawToolLifeUnit	String	Time unit, hereafter abbreviated as [Time]
rawToolLifeStatus	String	Tool life status
rawTimeLimit	Double	Life limit [Time]
rawTimeLimit1	Double	Maximum tool life [Time], Heidenhain only
rawTimeLimit2	Double	Maximum life of the invoked tool [Time], Heidenhain only
rawCurrentTime	Double	Current life [Time]
rawOverTime	Double	Time exceeding tool life [Time], Heidenhain only
rawOverCount	Int32	Count exceeding tool life [counts], Siemens only (user-defined module)

Response Parameter	Type	Description
rawRemainingTime	Double	Remaining life [Time]
rawPrewarningRemainingTime	Double	Warning remaining life [Time]
rawPrewarningTime	Double	Warning life [Time], Brother only
rawRemainingCount	Int32	Remaining life [counts]
rawPrewarningRemainingCount	Int32	Warning remaining life [counts]
rawRemainingWear	Double	Remaining life [machine’s default length unit]
rawPrewarningRemainingWear	Double	Warning remaining life [machine’s default length unit]
inventoryNum	String	Tool identification code, Heidenhain only

> **Note**
>
> The generic data portion of this table is the same as the response parameters in 2.5.1.21. readToolLife: Read Tool Life.

For the life types supported by each system model, see 4.2.25. ToolLife: Tool Life Data.

5.5.1.26. batchReadToolLifeDetails: Batch Read Tool Life Details

This endpoint is the batch version of 2.5.1.23. readToolLifeDetails: Read Tool Life Details. By supplying a list of target tools in the request body, multiple sets of data can be read in a single call.

```
POST /api/cnc/batchReadToolLifeDetails?machineID=MACHINEID
```

Request body example, application/json

```
[
  {
    "groupNum": 1,
    "toolIndex": 4
  },
  {
    "groupNum": 2,
```

```
    "toolIndex": 3
  }
]
```

The request parameters are the same as 2.5.1.23. readToolLifeDetails: Read Tool Life Details.

Request Parameter	Type	Description
machineID	String	(Required) Identifier of the target machine, see 4.1.1. machineID: Machine Identifier
groupNum	Int32	Supply groupNum (tool group number), or toolIndex (index within the group), or toolNum (tool number), or offsetNum (offset number) to identify the target tool. These request parameters correspond to the tags in 4.2.25. ToolLife: Tool Life Data, where the groupNum parameter corresponds to toolGroupNum in the tags, offsetNum corresponds to toolOffsetNum in the tags, and the other parameters share names with the tags. Refer to the per-system tag combination table in 4.2.25. ToolLife: Tool Life Data to determine which combination of request parameters to supply.
toolIndex	Int32	Same as above (described together with the groupNum row)
toolNum	Int32	Same as above (described together with the groupNum row)
offsetNum	Int32	Same as above (described together with the groupNum row)

Response example

```
[
  {
    "toolNum": 32,
    "toolIndex": 4,
    "toolGroupNum": 1,
```

```
    "countLimit": 100,
    "currentCount": 21
  },
  {
    "toolNum": 23,
    "toolIndex": 3,
    "toolGroupNum": 2,
    "countLimit": 200,
    "currentCount": 58
  }
]
```

The response parameters are the same as 2.5.1.23. readToolLifeDetails: Read Tool Life Details. If an individual request fails, the corresponding position in the returned array holds that request’s error information.

Response Parameter	Type	Description
Generic Data		
toolGroupNum	Int32	Tool group number
toolIndex	Int32	Index within the group
toolNum	Int32	Tool number
toolOffsetNum	Int32	Offset number
timeLimit	Int32	Life limit [seconds], the maximum usable time
currentTime	Int32	Current life [seconds], the time already used
prewarningTime	Int32	Warning life [seconds], an alarm is raised when the current life reaches this value
countLimit	Int32	Life limit [counts], the maximum usable count
currentCount	Int32	Current life [counts], the count already used
prewarningCount	Int32	Warning life [counts], an alarm is raised when the current life reaches this value

Response Parameter	Type	Description
wearLimit	Double	Life limit [machine’s default length unit], the maximum allowable wear
currentWear	Double	Current life [machine’s default length unit], the current wear amount
prewarningWear	Double	Warning life [machine’s default length unit], an alarm is raised when the current life reaches this value
Raw Data		
rawToolLifeType	String	Tool life type
rawToolLifeUnit	String	Time unit, hereafter abbreviated as [Time]
rawToolLifeStatus	String	Tool life status
rawTimeLimit	Double	Life limit [Time]
rawTimeLimit1	Double	Maximum tool life [Time], Heidenhain only
rawTimeLimit2	Double	Maximum life of the invoked tool [Time], Heidenhain only
rawCurrentTime	Double	Current life [Time]
rawOverTime	Double	Time exceeding tool life [Time], Heidenhain only
rawOverCount	Int32	Count exceeding tool life [counts], Siemens only (user-defined module)
rawRemainingTime	Double	Remaining life [Time]
rawPrewarningRemainingTime	Double	Warning remaining life [Time]
rawPrewarningTime	Double	Warning life [Time], Brother only
rawRemainingCount	Int32	Remaining life [counts]
rawPrewarningRemainingCount	Int32	Warning remaining life [counts]
rawRemainingWear	Double	Remaining life [machine’s default length unit]
rawPrewarningRemainingWear	Double	Warning remaining life [machine’s default length unit]

Response Parameter	Type	Description
inventoryNum	String	Tool identification code, Heidenhain only

> **Note**
>
> The generic data portion of this table is the same as the response parameters in 2.5.1.21. readToolLife: Read Tool Life.

For the life types supported by each system model, see 4.2.25. ToolLife: Tool Life Data.

5.5.2. Cached Read

When using cache read/write APIs, the gateway returns the corresponding automatic data collection task data from its cache. To use these APIs, the corresponding automatic data collection task must be enabled.

5.5.2.1. readTaskData Read Machine Task Data

This API reads task data for the target machine from the gateway cache, instead of sending a communication request to the machine to read data. As a result, this API can not only read raw data from the machine (such as machine status, alarm information, etc.), but also read data that has been statistically processed by the gateway (overload time, OEE data, etc.), and runs more efficiently. However, the task corresponding to the target data must already be enabled.

```
GET /api/cnc/readTaskData?machineID=MACHINEID&type=TYPE&tag=TAG
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
type	String	(Required) Data class, see 4.2. Data Description
tag	String	Field form of the tag, see 4.2. Data Description

Example

To get the overload data for spindle No. 1 of the machine with machineID 1010, with type=SpindleOverload, tag=S1, the request is as follows:

```
/api/cnc/readTaskData?machineID=1010&type=SpindleOverload&tag=S1
```

If the corresponding machine task has not been enabled, or the task is enabled but has not yet run, the response is:

```
{
  "errorCode": 1303,
```

```
  "errorMsg": "Task not ready."
}
```

When working normally, the data for the specified data class is returned in JSON format, as follows:

```
{
  "cumulativeTime": 14367,
  "spindleNo": 1,
  "time": "2024-03-01T02:26:42.2781587Z"
}
```

Among the response parameters, time is the timestamp when the data was retrieved. For the other response parameters, see the field and tag descriptions for the corresponding type in 4.2. Data Description.

5.5.2.2. batchReadTaskData Batch Read Machine Task Data

This API is the batch version of 2.5.2.1. readTaskData Read Machine Task Data. By supplying a list of target data items in the request body, multiple sets of data can be read in a single call, provided that the tasks corresponding to the target data are already enabled.

```
POST /api/cnc/batchReadTaskData
```

Request body example, application/json

```
[
  {
    "machineID": "1",
    "type": "CNCStatus"
  },
  {
    "machineID": "2110",
    "type": "Count"
  },
  {
    "machineID": "3",
    "type": "PLC",
```

```
    "tag": "TR0"
  }
]
```

The request parameters are the same as 2.5.2.1. readTaskData Read Machine Task Data.

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
Type	String	(Required) Data class, see 4.2. Data Description
Tag	String	Field form of the tag, see 4.2. Data Description

Response example

```
[
  {
    "cncStatus": "AUTO_RUN",
    "alarmStatus": "ALARM",
    "alarmLevel": "WRN",
    "mode": "AUTO_RUN",
    "programStatus": "RUNNING",
    "emergencyStatus": "NOT_EMG",
    "dryRunStatus": "NOT_DRY_RUN",
    "adjustedStatus": "AUTO_RUN",
    "offTime": 2509,
    "waitTime": 2123,
    "emergencyTime": 25,
    "autoRunTime": 1526,
    "manualTime": 2905,
    "time": "2024-03-01T02:26:48.7383566Z"
  },
  {
    "count": 22406,
    "cumulativeCount": 0,
    "currentCount": 0,
    "time": "2024-03-01T02:26:49.8853233Z"
  },
```

```
  {
    "data": [
      32767
    ],
    "tag": "R0",
    "time": "2024-03-01T02:26:49.3324829Z"
  }
]
```

The order of the response results matches the order of the requests. If a given request fails, the corresponding position in the returned array contains the error information for that request.

Among the response parameters, time is the timestamp when the data was retrieved. For the other response parameters, see the field and tag descriptions for the corresponding type in 4.2. Data Description.

5.5.2.3. readGroupTaskData Read Machine Group Task Data

This API reads the latest data from the gateway cache for the target task currently running on the target machine group, provided that the task corresponding to the target data of the target machine group is already enabled.

```
GET /api/cnc/readGroupTaskData?groupID=GROUPID&type=TYPE&tag=TAG
```

Request Parameter	Type	Description
groupID	String	(Required) Target machine group identifier, see 4.1.2. groupID Machine Group Identifier
type	String	(Required) Data class, currently only 4.2.11. GroupCount: Machine Group Processing Count Data, 4.2.12. GroupCumulativeTime: Machine Group Cumulative Status Time, and 4.2.13. GroupOEE: Machine Group OEE are supported
tag	String	Field form of the tag, see 4.2. Data Description

Example

To get the processing count data for the machine group with groupID g2, with type=GroupCount, the request is as follows:

```
/api/cnc/readGroupTaskData?groupID=g2&type=GroupCount
```

If the corresponding machine group task has not been enabled, or the task is enabled but has not yet run, the response is:

```
{
  "errorCode": 1303,
  "errorMsg": "Task not ready."
}
```

When working normally, the data for the specified data class is returned in JSON format, as follows:

```
{
  "cumulativeCount": 127,
  "time": "2024-03-01T02:26:50.1127663Z"
}
```

Among the response parameters, time is the timestamp when the data was retrieved. For the other response parameters, see the field and tag descriptions for the corresponding type in 4.2. Data Description.

5.5.2.4. batchReadGroupTaskData Batch Read Machine Group Task Data

This API is the batch version of 2.5.2.3. readGroupTaskData Read Machine Group Task Data. By supplying a list of target data items in the request body, multiple sets of data can be read in a single call, provided that the tasks corresponding to the target data are already enabled.

```
POST /api/cnc/batchReadGroupTaskData
```

Request body example, application/json

```
[
```

```
  {
    "groupID": "1",
    "type": "GroupOEE"
  },
  {
    "groupID": "2",
    "type": "GroupCount"
  }
]
```

The request parameters are the same as 2.5.2.3. readGroupTaskData Read Machine Group Task Data.

Request Parameter	Type	Description
groupID	String	(Required) Target machine group identifier, see 4.1.2. groupID Machine Group Identifier
type	String	(Required) Data class, currently only 4.2.11. GroupCount: Machine Group Processing Count Data, 4.2.12. GroupCumulativeTime: Machine Group Cumulative Status Time, and 4.2.13. GroupOEE: Machine Group OEE are supported
tag	String	Field form of the tag, see 4.2. Data Description

Response example

```
[
  {
    "autoRunCount": 1,
    "waitCount": 0,
    "emergencyCount": 0,
    "manualCount": 0,
    "offCount": 0,
    "availability": 90.0,
    "offTime": 360,
    "waitTime": 0,
```

```
    "emergencyTime": 0,
    "autoRunTime": 3240,
    "manualTime": 0,
    "time": "2024-03-01T02:27:30.5534659Z"
  },
  {
    "cumulativeCount": 127,
    "time": "2024-03-01T02:27:30.6533778Z"
  }
]
```

The order of the response results matches the order of the requests. If a given request fails, the corresponding position in the returned array contains the error information for that request.

Among the response parameters, time is the timestamp when the data was retrieved. For the other response parameters, see the field and tag descriptions for the corresponding type in 4.2. Data Description.

5.5.2.5. batchReadErrors Batch Read Machine Connection Status

This API is used to batch retrieve the current connection status of specified machines. By supplying a list of target machine identifiers in the request body, the connection status of multiple machines can be read in a single call.

```
POST /api/cnc/batchReadErrors
```

Request body example, application/json

```
[
  {
    "machineID": "1"
  },
  {
    "machineID": "2"
  }
]
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Response example

```
[
  {
    "errorCode": 0,
    "errorMsg": "Success"
  },
  {
    "errorCode": 3,
    "errorMsg": "Machine offline."
  }
]
```

The order of the response results matches the order of the requests.

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code; 0 indicates a successful connection.
errorMsg	String	(Required) Error message

5.5.2.6. readErrorSummary Read Status Summary

This API is used to get a summary of the current connection status of the machines; no request parameters are required.

```
GET /api/cnc/readErrorSummary
```

Response example

```
{
  "success": 8,
  "offline": 1,
  "error": 0
```

```
}
```

Response Parameter	Type	Description
success	Int32	(Required) Number of machines successfully connected
offline	Int32	(Required) Number of offline machines
error	Int32	(Required) Number of machines with connection errors

5.6. File Management Interface

The file management interface can connect to a target file server, enabling reading of file lists, receiving, sending, deleting, locking, and unlocking files, creating and deleting directories, and other functions.

The file server types currently supported for file transfer include “machine storage”, “FTP server”, “shared folder”, etc. (see 3.3.1. Program Transfer Settings), with “machine storage” as the default. Some CNC systems can enable a local FTP server or shared folder option for file transfer. For machines with limited local storage, or machines that do not support direct access to their storage space, an external file server can be used to enable file transfer.

Most file management interfaces accept an additional request parameter, dirAtCNC or subDir, to specify the directory path on the machine or file server. If neither dirAtCNC nor subDir is supplied, the corresponding file operation is performed in the root directory by default. Users can specify a root directory path (see 3.3.1. Program Transfer Settings). If the user has not specified a root directory path, the following default root directory paths are used.

Default Root Directory Paths by System Model

File Server Type	System [Model]	Target Directory Path Supported	Default Root Directory Path
Machine storage	Brother [TC series]	X	`/`
Machine storage	Brother [S series]	O	`/`
Machine storage	Delta [general]	O	`B:\`
Machine storage	Fagor [8035M/T, 8040M/T, 8055M/T]***	O	`MEMORY`
Machine storage	Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]*	O	`//CNC_MEM/`
Machine storage	Fanuc [models other than 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	X	The current foreground directory configured on the machine, not fixed, e.g. `//CNC_MEM/`
Machine storage	Gsk [980, 988]	O	`/user/NCPROG`
Machine storage	Haas [general, MT-CONNECT]	X	None

File Server Type	System [Model]	Target Directory Path Supported	Default Root Directory Path
Machine storage	Heidenhain [TNC640, TNC640 DNC]	O	`TNC:/nc_prog`
Machine storage	Heidenhain [iTNC530]	O	`TNC:/`
Machine storage	HNC [1.24, 1.26.02]	O	`h/lnc8/`
Machine storage	Jingdiao [JD50]	O	`E:\存储文件夹`
Machine storage	Lanhao	O	`昊折\menudir\menu`
Machine storage	Lynuc [all models]	O	`/home/Lynuc/Users/NCFiles`
Machine storage	Makino [general]*	O	`//CNC_MEM/`
Machine storage	Mitsubishi [all models]**	O	`PRG\USER\`
Machine storage	Mock machine	O	`/`
Machine storage	Okuma [all models]	O	`MD1`
Machine storage	Rexroth [OPC UA]	O	`/prog`
Machine storage	Siemens [general]	O	`/nckfs`
Machine storage	Siemens [OPC UA]	O	`Sinumerik/FileSystem/Part Program`
Machine storage	Siemens [810D/840D]	O	`mpf.dir`
Machine storage	Other system models supporting program transfer	X	Machine’s default storage directory
FTP server		O	FTP server root directory
Shared folder		O	None
Shared folder (Win XP)		O	None
Wireless transfer box		O	`/`
Gateway file server		O	`/`

O: Supported; X: Not supported

*: For FANUC [0i-F, 30i, 31i, 32i, 35i, Power Motion i-A] and Makino [general], the target directory path can be changed to an external CF card, e.g. `//MEMCARD/`.

**: For Mitsubishi [M700 series, M800 series], the target directory path can be changed to an external CF card (M700 series) or SD card (M800 series), e.g. IC1.

***: Fagor [8060M/T/L, 8065M/T, 8070M/T/L] does not support program transfer, and does not support specifying a target directory path.

5.6.1. readProgramList — Read Machine File List

```
GET /api/cnc/readProgramList?
machineID=MACHINEID&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Request Parameter	Type	Description
startPrgNo	Int32	Effective only for Fanuc [models other than 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]: reading starts from this program number; 0 (the default) means read all programs. Other system models ignore this parameter.

Response example

```
{
  "dirAtCNC": "Dir/Prog",
  "programs": [
    "1111",
    "2322",
    "3222"
  ],
  "subDirs": [
    "dir1",
    "dir2"
  ]
}
```

Response Parameter	Type	Description
dirAtCNC	String	(Required) Specifies the directory path. Generally matches the dirAtCNC in the request parameters, except: 1. If the device has only one default directory (with no specific path given) for storing files and folders, the returned dirAtCNC is empty; 2. If the request parameter dirAtCNC is not supplied or is empty, the returned dirAtCNC is the default root directory path; 3. If the path specified by the request parameter dirAtCNC exists but does not conform to the system’s path format, the returned dirAtCNC is the automatically corrected path.

Response Parameter	Type	Description
programs	String[]	(Required) List of programs (files) in the specified directory
subDirs	String[]	(Required) List of subfolders in the specified directory. For some system models, the subfolder list cannot be retrieved, and the returned subDirs is empty.

Support for Retrieving the Subfolder List in the Read Machine File List Interface by System Model

File Server Type	System [Model]	Retrieve File List	Retrieve Subfolder List
Machine storage	Brother [all models]	O	O
Machine storage	Fagor [8035M/T, 8040M/T, 8055M/T]	O	X
Machine storage	Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	*	O
Machine storage	Fanuc [models other than 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	O	X
Machine storage	Heidenhain [all models]	O	O
Machine storage	Lynuc [all models]	O	O
Machine storage	Makino [general]	*	O
Machine storage	Mitsubishi [all models]	O	O
Machine storage	Mock machine	O	O
Machine storage	Okuma [all models]	O	O
Machine storage	Rexroth [OPC UA]	O	O
Machine storage	Siemens [all models]	O	O
Machine storage	Other system models supporting program transfer	O	X
FTP server		O	O
Shared folder		O	O

File Server Type	System [Model]	Retrieve File List	Retrieve Subfolder List
Shared folder (Win XP)		O	O
Wireless transfer box		O	O
Gateway file server		O	O

O: Supported; X: Not supported

*: FANUC [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A] and Makino [general] return the standard FANUC program name (letter O + number), automatically stripping leading zeros from the number. For example, if the program name on the machine is 00010, it is returned as O10. For non-standard named programs, the name is returned as-is from the machine, e.g. if the program name is SAMPLE, it is returned as SAMPLE.

5.6.2. receiveFileStream — Receive Machine File (Stream Mode)

This interface can receive both text files and binary files. Compared to 2.6.3. receiveFile — Receive Machine File, it additionally supports files larger than 10MB. Users are recommended to use the /receiveFileStream interface whenever possible.

```
GET /api/cnc/receiveFileStream?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.

Request Parameter	Type	Description
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

If the target file is a text file, the response body example is application/octet-stream

```
%
O0001
G21 G90 G40
T01 M06
G90 G0 G54 X12.64 Y88.0 s2546
G43 H01 Z15.0 M8
G81 G98 Z-20. R1.F200.
x70.
X85.
x170.
G80
G00 x100. Y200
G28 G91 z0 M9
M30
%
```

If the target file is a binary file, it is returned in binary format in the response body.

5.6.3. receiveFile — Receive Machine File

This interface can receive both text files and binary files. Compared to 2.6.2. receiveFileStream — Receive Machine File (Stream Mode), this interface does not support files larger than 10MB. Users are recommended to use the /receiveFileStream interface whenever possible.

```
GET /api/cnc/receiveFile?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.

Request Parameter	Type	Description
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

When the file is in text format:

```
{
  "content": "string",
  "encoding": "us-ascii"
}
```

When the file is in binary format:

```
{
  "base64": "string"
}
```

Response Parameter	Type	Description
content	String	File content
encoding	String	Original file encoding
base64	String	When the file is in binary format, the gateway uses Base64 encoding to convert the binary content into a string. Users can decode it themselves to obtain the binary file.

5.6.4. readCurrentProgram — Receive the Machine’s Currently Running File

```
GET /api/cnc/readCurrentProgram?
machineID=MACHINEID&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. Basic Description for an explanation of machineID.
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

When the file is in text format:

```
{
```

```
  "dirAtCNC": "Dir/Prog",
  "fileName": "O9001",
  "content": "string",
  "encoding": "us-ascii"
}
```

When the file is in binary format:

```
{
  "dirAtCNC": "Dir/Prog",
  "fileName": "Program",
  "base64": "string"
}
```

When there is no file currently running:

```
{
  "errorCode": 196,
  "errorMsg": "Program not selected."
}
```

Response Parameter	Type	Description
dirAtCNC	String	Directory path of the file currently running on the machine
fileName	String	File name of the file currently running on the machine
content	String	File content
encoding	String	Original file encoding
base64	String	When the file is in binary format, the gateway uses Base64 encoding to convert the binary content into a string. Users can decode it themselves to obtain the binary file.
channel	Int32	Machine channel number, present only when `channel` is included in the request and is not 0
errorCode	Int32	Error code, 0 indicates success.

Response Parameter	Type	Description
errorMsg	String	Error message
statusMsg	String	Error details

5.6.5. sendFileStream — Send File to Machine (Stream Mode)

This interface can send both text files and binary files. Compared to 2.6.6. sendFile — Send File to Machine, it additionally supports files larger than 10MB. Users are recommended to use the /sendFileStream interface whenever possible.

Overwriting a file with the same name is not supported. Please delete the file with the same name first, then send.

```
POST /api/cnc/sendFileStream?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request example

If the target file is a text file, the request body example is application/octet-stream

```
%
O0001
G21 G90 G40
T01 M06
G90 G0 G54 X12.64 Y88.0 s2546
G43 H01 Z15.0 M8
G81 G98 Z-20. R1.F200.
x70.
X85.
x100.
x170.
X185.
x200.
G80
G00 x100. Y200
G28 G91 z0 M9
M30
%
```

If the target file is a binary file, it is uploaded to the request body in binary format.

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	File name of the target file on the machine. For some system models, such as Gsk and Siemens, the file name must include the file extension; refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For FANUC systems, fileName is not used; the program name is taken from the content between the first `\n` and the second `\n` in Content. For example, if Content is `"%\nO3(ROUGH)\nT1M06\nG92X0.Y0.Z0. …"`, the program name displayed on the machine will be O0003, with comment (ROUGH); if Content is `"%\<SAMPLE>\nT1M06\nG92X0.Y0.Z0. …"`, the program name displayed on the machine will be SAMPLE.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.

Request Parameter	Type	Description
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.6. sendFile — Send File to Machine

This interface can send both text files and binary files. Compared to 2.6.5. sendFileStream — Send File to Machine (Stream Mode), this interface does not support files larger than 10MB. Users are recommended to use the /sendFileStream interface whenever possible.

Overwriting a file with the same name is not supported. Please delete the file with the same name first, then send.

```
POST /api/cnc/sendFile?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request body example application/json

When the file is in text format:

```
{
  "content": "string",
  "encoding": "us-ascii"
}
```

When the file is in binary format:

```
{
  "base64": "string"
}
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
fileName	String	File name of the target file on the machine. For some system models, such as Gsk and Siemens, the file name must include the file extension; refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For FANUC systems, fileName is not used; the program name is taken from the content between the first `\n` and the second `\n` in Content. For example, if Content is `"%\nO3(ROUGH)\nT1M06\nG92X0.Y0.Z0. …"`, the program name displayed on the machine will be O0003, with comment (ROUGH); if Content is `"%\<SAMPLE>\nT1M06\nG92X0.Y0.Z0. …"`, the program name displayed on the machine will be SAMPLE.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.

Request Parameter	Type	Description
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
content	String	File content. Use this parameter when the file is in text format. When Content is defined, the Base64 parameter is ignored.
encoding	String	Original file encoding. If not specified, the encoding configured on the machine is used by default.
base64	String	Use this parameter when the file is in binary format. Encode the binary content as a string using Base64 encoding before writing it into this parameter.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.7. batchSendFile — Batch Send Files to Machines

This interface is a batch version of 2.6.6. sendFile — Send File to Machine. By supplying target locations in the request body, the same file can be sent, under different file names, to different locations on different machines simultaneously.

Overwriting a file with the same name is not supported. Please delete the file with the same name first, then send.

```
POST /api/cnc/batchSendFile
```

Request body example application/json

When the file is in text format:

```
{
  "content": "string",
  "encoding": "us-ascii",
  "targets": [
    { "machineID": "1010", "fileName": "O8000" },
    { "machineID": "1011", "fileName": "sample.NC", "DirAtCNC": "dir1" },
    { "machineID": "1012", "fileName": "sample.NC", "SubDir": "Dir/Prog" }
  ]
}
```

When the file is in binary format:

```
{
  "base64": "string",
  "targets": [
    { "machineID": "1011", "fileName": "sample.NC", "SubDir": "Dir/Prog" }
  ]
}
```

The request parameters match those in 2.6.6. sendFile — Send File to Machine.

Request Parameter	Type	Description
targets	Array	(Required) Target paths for sending the file to machines

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) File name of the target file on the machine. For some system models, such as Gsk and Siemens, the file name must include the file extension; refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For FANUC systems, fileName is not used; the program name is taken from the content between the first `\n` and the second `\n` in Content. For example, if Content is `"%\nO3(ROUGH)\nT1M06\nG92X0.Y0.Z0. …"`, the program name displayed on the machine will be O0003, with comment (ROUGH).
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.

Request Parameter	Type	Description
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
content	String	File content. Use this parameter when the file is in text format. When content is defined, the Base64 parameter is ignored.
encoding	String	Original file encoding. If not specified, the encoding configured on the machine is used by default.
base64	String	Use this parameter when the file is in binary format. Encode the binary content as a string using Base64 encoding before writing it into this parameter.

Response example

```
[
  {
    "errorCode": 0,
    "errorMsg": "Success"
  },
  {
    "errorCode": 10003,
    "errorMsg": "Machine ID does not exist."
  },
  {
    "errorCode": 142,
    "errorMsg": "Path exists."
```

```
  }
]
```

The response body contains the execution results for each request in the request body, in order.

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.8. deleteFile — Delete Machine File

This interface deletes the specified file on the machine.

```
GET /api/cnc/deleteFile?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.

Request Parameter	Type	Description
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.9. batchDeleteFile — Batch Delete Machine Files

This interface is a batch version of 2.6.8. deleteFile — Delete Machine File. By supplying target locations in the request body, different files on different machines can be deleted in a single call.

```
POST /api/cnc/batchDeleteFile
```

Request body example application/json

```
[
```

```
  { "machineID": "1010", "fileName": "O8000" },
  { "machineID": "1011", "fileName": "sample.NC", "DirAtCNC": "dir1" },
  { "machineID": "1012", "fileName": "sample.NC", "SubDir": "Dir/Prog" }
]
```

The request parameters match those in 2.6.8. deleteFile — Delete Machine File.

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

```
[
  {
    "errorCode": 0,
    "errorMsg": "Success"
  },
  {
    "errorCode": 10003,
    "errorMsg": "Machine ID does not exist."
  },
  {
    "errorCode": 158,
    "errorMsg": "Path is not found."
  }
]
```

The response body contains the execution results for each request in the request body, in order.

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.10. lockFileByRange — Lock/Unlock Machine Program Editing

Currently, this only supports Fanuc and Makino systems, for locking/unlocking editing of program numbers within the ranges 8000-8999, 9000-9999, and 8000-9999 (any other range returns 10017 Invalid API request.), as well as the Mock machine. On some systems, a locked program cannot be edited but can still be deleted from the machine.

```
GET /api/cnc/lockFileByRange?
machineID=MACHINEID&isLock=ISLOCK&lockStart=LOCKSTART&lockEnd=LOCKEND
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
isLock	Bool	(Required) Target lock state. IsLock: true to lock, false to unlock.
lockStart	Int32	(Required) Starting program number of the lock/unlock range
lockEnd	Int32	(Required) Ending program number of the lock/unlock range

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.6.11. createDir — Create Machine Directory

```
GET /api/cnc/createDir?
machineID=MACHINEID&dirName=DIRNAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
dirName	String	(Required) Name of the directory to create

Request Parameter	Type	Description
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

The system models that currently support creating machine directories are listed in the table below.

System Model Support for the Create/Delete Machine Directory Interfaces

File Server Type	System [Model]	Create/Delete Machine Directory
Machine storage	Brother [S series]	O
Machine storage	Delta [general]	O
Machine storage	Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	O
Machine storage	GSK [all models]	X
Machine storage	Haas [general, MT-CONNECT]	X
Machine storage	Heidenhain [all models]	O
Machine storage	Jingdiao [JD50]	O
Machine storage	Lanhao	O
Machine storage	Makino [general]	O
Machine storage	Mitsubishi [M70/M700 L, M70/M700 M, M80/M800 L, M80/M800 M]	*CF card and SD card only (path: IC1)
Machine storage	Mock machine	O
Machine storage	Okuma [all models]	O
Machine storage	Rexroth [OPC UA]	O
Machine storage	Siemens [all models]	O
Machine storage	Other system models supporting program transfer	X
FTP server		O
Shared folder		O
Shared folder (Win XP)		O
Wireless transfer box		O
Gateway file server		O

O: Supported; X: Not supported

5.6.12. deleteDir — Delete Machine Directory

Currently, only empty directories can be deleted. Users must first delete all files and subdirectories within the directory.

```
GET /api/cnc/deleteDir?
machineID=MACHINEID&dirName=DIRNAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
dirName	String	(Required) Name of the directory to delete
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

The system models currently supporting the deletion of machine directories are the same as those listed in System Model Support for the Create/Delete Machine Directory Interfaces.

5.6.13. backupFiles — Back Up Machine Files

This interface packages the files specified in the request body, or all files within a directory (including subfolders and the files within them), into a single ZIP file and returns it via Stream. The top-level directory name in the ZIP file follows the format "BackupFiles_backup date and time"; the second-level directories are the root directories for each machine, named in the format "machineID_machineName"; under each machine's root directory are the specified files and directories for that machine, preserving the original folder structure.

```
POST /api/cnc/backupFiles
```

Request body example application/json

```
[
  { "machineID": "1010", "fileName": "O8000" },
  { "machineID": "1011", "DirAtCNC": "dir1", "includeSubDir": false },
  { "machineID": "1012", "fileName": "sample.NC", "SubDir": "Dir/Prog" }
]
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension. If fileName is not supplied, all files under the directory specified by dirAtCNC or subDir will be backed up.
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
includeSubDir	Bool	When fileName is not supplied, i.e. when downloading all files under the specified directory, if this value is true, all files under all subfolders of the directory are also downloaded. Defaults to true.

Response body example application/octet-stream

```
PK gx BackupFiles_2024.03.07.12.04.46…
```

The response body is the ZIP file returned via Stream.

If any request in the request body fails during execution, the task is terminated and the error information for that request is returned, for example:

```
{
  "errorCode": 10003,
  "errorMsg": "Machine ID does not exist.",
  "statusMsg": "Invalid MachineID (1011)"
}
```

5.6.14. selectProgram — Select Program

This interface selects the specified program as the main program on the machine.

```
GET /api/cnc/selectProgram?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
channel	Int32	Machine channel number. If not specified, it defaults to 0, i.e. the main channel
fileName	String	(Required) File name of the target file, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.

Request Parameter	Type	Description
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
mode	String	Execution mode. Okuma only: for machining centers, “A” (A run), “B” (B run), “S” (S run) are supported; for lathes, “L” (the L spindle of dual spindles) and “R” (single spindle, or the R spindle of dual spindles) are supported.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

Devices currently supported by this interface:

System [Model]	Notes
Bosunman [DF Series Ethernet, DF Series Serial over Ethernet]	
Brother [all models]	Must be in automatic run mode or background edit mode; no program may currently be running.
Delta [general]	
Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	Must be in Auto or Edit mode.
Gsk [980, 988, 25i]	Supported on some versions.
HNC [1.26.02]	
Jingdiao [JD50]	
KND [V4.3.00b, V5.1.00c]	
Lynuc [general]	
Makino [general]	Must be in Auto or Edit mode.
Mock machine	Always returns SUCCESS, but does not change the current program.
Okuma [all models]	The mode execution mode parameter must be supplied.
Syntec	

5.6.15. readAllPrograms — Browse All Files

This interface searches all program files under the specified directory on the machine and returns the file names along with their directory paths.

```
GET /api/cnc/readAllPrograms?
machineID=MACHINEID&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
startPrgNo	Int32	Effective only for Fanuc [models other than 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]: reading starts from this program number; 0 (the default) means read all programs. Other system models ignore this parameter.

Response example

```
{
  "programs": [
    {
      "dirAtCNC": "/",
      "fileName": "SAMPLE"
    },
    {
      "dirAtCNC": "1",
      "fileName": "sample1.nc"
    },
```

```
    {
      "dirAtCNC": "1/1",
      "fileName": "sample2.nc"
    }
  ]
}
```

Response Parameter	Type	Description
programs	Object[]	(Required) Array of program file objects.
fileName	String	Program file name. For some system models, such as Gsk and Siemens, the file name includes the file extension.
dirAtCNC	String	Directory path of the program file.

5.6.16. searchFile — Search Machine Files

This interface searches for the specified file name under the specified machine directory (including subdirectories), and returns the path of the directory containing the file, along with all files (including the matched file) and subdirectories in that directory.

```
GET /api/cnc/searchFile?
machineID=MACHINEID&fileName=FILENAME&dirAtCNC=DIRATCNC&subDir=SUBDIR
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
fileName	String	(Required) Target file name, refer to the file name returned by 2.6.1. readProgramList — Read Machine File List or 2.6.15. readAllPrograms — Browse All Files. For some system models, such as Gsk and Siemens, the file name must include the file extension.

Request Parameter	Type	Description
dirAtCNC	String	Specifies the target directory path. If this parameter is not supplied, the default root directory path is used. See Default Root Directory Paths by System Model for each system model’s default root directory path.
subDir	String	Specifies the target subdirectory path. If dirAtCNC is not provided, the root directory path and subdirectory path are automatically concatenated to form dirAtCNC. Users can refer to 3.3.1. Program Transfer Settings to modify the root directory path; if it has not been modified, see Default Root Directory Paths by System Model for the default root directory path. If dirAtCNC and subDir are both supplied, subDir is ignored.
startPrgNo	Int32	Effective only for Fanuc [models other than 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]: reading starts from this program number; 0 (the default) means read all programs. Other system models ignore this parameter.

Response example

```
{
  "dirAtCNC": "Dir/Prog",
  "programs": [
    "1111",
    "2322",
    "3222"
  ],
  "subDirs": [
    "dir1",
    "dir2"
```

```
    ]
  }
```

Response Parameter	Type	Description
dirAtCNC	String	Directory path containing the target file.
programs	String[]	List of programs (files) in the directory containing the target file
subDirs	String[]	List of subfolders in the directory containing the target file

5.7. Data Analysis Interface

The data analysis interface is used to retrieve data analysis for a target machine or machine group within a specified time range.

Before using the data analysis interface, the gateway’s local cache switch must be enabled (see 3.12. Local Cache), to allow historical machine status data to be saved locally on the gateway.

Using the data analysis interface requires supplying a start timestamp and an end timestamp. When interval is not set, the span between the start and end timestamps must be less than 31 days (exactly 31 days is rejected); when interval is set, this limit applies to each grouping interval rather than to the entire time range.

The maximum local retention period is 365 days. Therefore, the start time cannot be earlier than 365 days before the current time.

If the current timestamp is earlier than the supplied end timestamp, the current timestamp is automatically used as the end timestamp.

If the gateway has not received machine or machine group data for a period of time (data gap), the data returned by the data analysis interface may differ from the actual data. Common causes of data gaps include: the gateway’s local cache not being enabled, the corresponding automatic collection task not being enabled, the gateway losing power, the gateway’s network connection being interrupted, and the machine being offline.

5.7.1. Machine Data Analysis

Machine data analysis is used to retrieve data analysis for a target machine within a specified time range, with base path `/api/analysis`.

5.7.1.1. oee — Machine OEE Data Analysis

Retrieves machine OEE statistics for each grouping interval within a specified time range.

```
GET /api/analysis/oee?
machineID=MACHINEID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "availability": 83.33333,
    "offTime": 100,
    "waitTime": 200,
    "emergencyTime": 300,
    "autoRunTime": 3000,
    "manualTime": 0.0
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "availability": 0.0,
    "offTime": 0,
    "waitTime": 3600,
    "emergencyTime": 0,
    "autoRunTime": 0,
    "manualTime": 0
  }
]
```

Response Parameter	Type	Description
machineName	String	(Required) Machine name, set in the “Add/Edit Device” window on the “Machine Configuration” page.
startTime	String	(Required) Grouping interval start time (UTC)
endTime	String	(Required) Grouping interval end time (UTC)
availability	Float	(Required) Availability rate [%]
offTime	Int64	(Required) Off or offline time [seconds]
waitTime	Int64	(Required) Idle time [seconds]
emergencyTime	Int64	(Required) Emergency stop time [seconds]
autoRunTime	Int64	(Required) Auto-run time [seconds]
manualTime	Int64	(Required) Manual setup time [seconds]

> **Note**
>
> For details on machine OEE data, see 11.1.1. Machine OEE Data. If status data is missing for a period between the start time and end time, the status for that missing period defaults to off or offline.

5.7.1.2. alarm — Machine Alarm Data Analysis

Retrieves the machine alarm history within a specified time range.

```
GET /api/analysis/alarm?
machineID=MACHINEID&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2023-06-16T16:00:00Z",
    "endTime": "2023-06-16T16:00:10Z",
    "alarmMsg": "00:00 ALARM 1",
    "alarmLevel": "WRN"
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2023-06-16T16:00:00Z",
    "endTime": "2023-06-16T16:00:10.001Z",
    "alarmMsg": "00:00 ALARM 2",
    "alarmLevel": "WRN"
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2023-06-16T16:01:00.008Z",
    "endTime": "2023-06-16T16:01:50.019Z",
    "alarmMsg": "00:01 ALARM 1",
    "alarmLevel": "WRN"
  }
]
```

Response Parameter	Type	Description
machineName	String	(Required) Machine name, set in the “Add/Edit Device” window on the “Machine Configuration” page.
startTime	String	(Required) Alarm start time (UTC)
endTime	String	(Required) Alarm end time (UTC)

Response Parameter	Type	Description
alarmMsg	String	(Required) Alarm message
alarmLevel	String	(Required) Alarm level. In descending order of priority: Error (ERR), Warning (WRN), and Info (INF). Users can refer to 3.5.7. Alarm Monitoring Settings to set alarm levels by keyword and filter out alarms below the minimum alarm level; alarms without a configured level default to the Warning level.

Note

An alarm is recorded from the moment it appears until it is cleared, with its start time, end time, alarm message, and alarm level. The alarm start time is the time at which the alarm actually appeared, which may be earlier than the requested start timestamp (it is not clamped to the start timestamp). The query filters on whether the alarm’s clear time falls within the time range.

Only cleared alarms are counted; an alarm that has not yet been cleared at the end timestamp does not appear in the response (its record is written only once the alarm is cleared).

If alarm data is missing between the start time and end time, no alarms are assumed to have occurred during that missing period.

5.7.1.3. count — Machine Count Data Analysis

Retrieves the cumulative machining count for each grouping interval within a specified time range.

```
GET /api/analysis/count?
machineID=MACHINEID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL&en
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier

Request Parameter	Type	Description
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.
enableCountPerProgram	Bool	Groups the machining count by main program. Defaults to false. When true, the response is grouped by main program mainPrgmName, with one record per main program, and count is the sum of the cycle counts deltaCount for that main program; if there is no cycle data within the time range, it is handled as false and a single record is returned (the difference in cumulativeCount).

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 96
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0
  }
]
```

Response Parameter	Type	Description
machineName	String	(Required) Machine name, set in the “Add/Edit Device” window on the “Machine Configuration” page.
startTime	String	(Required) Grouping interval start time (UTC)
endTime	String	(Required) Grouping interval end time (UTC)
count	Int32	(Required) Cumulative machining count, the difference between the machine’s cumulative machining count `cumulativeCount` at the start and end of the grouping interval
mainPrgmName	String	Main program name. Returned only when the request parameter enableCountPerProgram is true and cycle data exists within the time range, in which case one record is returned per main program (cycles without a main program name are grouped under the empty string).

Response example grouped by main program (enableCountPerProgram is true)

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 60,
    "mainPrgmName": "O6495"
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 36,
    "mainPrgmName": "O6496"
```

```
  }
]
```

5.7.1.4. overall — Machine Overall Data Analysis

Retrieves the combined data for each grouping interval within a specified time range, including data from 2.7.1.1. oee — Machine OEE Data Analysis and 2.7.1.3. count — Machine Count Data Analysis.

```
GET /api/analysis/overall?
machineID=MACHINEID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 96,
    "availability": 83.33333,
    "offTime": 100,
    "waitTime": 200,
    "emergencyTime": 300,
    "autoRunTime": 3000,
    "manualTime": 0.0
```

```
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0,
    "availability": 0.0,
    "offTime": 0,
    "waitTime": 3600,
    "emergencyTime": 0,
    "autoRunTime": 0,
    "manualTime": 0
  }
]
```

5.7.1.5. cycle — Machine Cycle Time Data Analysis

Retrieves the machine’s cycle time data within a specified time range.

```
GET /api/analysis/cycle?
machineID=MACHINEID&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2025-05-23T06:35:44.046Z",
    "endTime": "2025-05-23T06:36:25.046Z",
```

```
    "lastCycleTime": 41,
    "mainPrgmName": "O6495"
  },
  {
    "machineName": "模拟机台 1",
    "startTime": "2025-05-23T06:36:25.072Z",
    "endTime": "2025-05-23T06:36:56.072Z",
    "lastCycleTime": 31,
    "mainPrgmName": "O6495"
  }
]
```


Response Parameter	Type	Description
machineName	String	(Required) Machine name, set in the “Add/Edit Device” window on the “Machine Configuration” page.
startTime	String	(Required) Cycle start time (UTC)
endTime	String	(Required) Cycle end time (UTC)
lastCycleTime	Int64	(Required) Cycle duration [seconds]; counts only auto-run time.
mainPrgmName	String	(Required) Name of the main program executed during the cycle

5.7.2. Machine Group Data Analysis

Machine group data analysis is used to retrieve data analysis for a target machine or machine group within a specified time range. Base address: `/api/group-analysis`. It requires the group identifier variable groupID in place of the machine identifier machineID.

5.7.2.1. oee Machine Group OEE Data Analysis

Retrieves the OEE data of all machines in the machine group, for each grouping interval, within the specified time range.

```
GET /api/group-analysis/oee?
groupID=GROUPID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier. See 4.1.2. groupID Group Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	End timestamp [seconds]. If undefined, the current time is used; a value later than the current time is truncated to the current time.
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.

Response example

```
[
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "availability": 83.33333,
```

```
      "offTime": 100,
      "waitTime": 200,
      "emergencyTime": 300,
      "autoRunTime": 3000,
      "manualTime": 0.0
    },
    {
      "machineID": "1010",
      "machineName": "模拟机台 1",
      "startTime": "2022-04-28T15:30:00Z",
      "endTime": "2022-04-28T16:30:00Z",
      "availability": 0.0,
      "offTime": 0,
      "waitTime": 3600,
      "emergencyTime": 0,
      "autoRunTime": 0,
      "manualTime": 0
    },
    {
      "machineID": "1011",
      "machineName": "模拟机台 2",
      "startTime": "2022-04-28T14:30:00Z",
      "endTime": "2022-04-28T15:30:00Z",
      "availability": 80.55556,
      "offTime": 150,
      "waitTime": 250,
      "emergencyTime": 300,
      "autoRunTime": 2900,
      "manualTime": 0.0
    },
    {
      "machineID": "1011",
      "machineName": "模拟机台 2",
      "startTime": "2022-04-28T15:30:00Z",
      "endTime": "2022-04-28T16:30:00Z",
      "availability": 0.0,
      "offTime": 0,
      "waitTime": 3600,
      "emergencyTime": 0,
      "autoRunTime": 0,
```

```
    "manualTime": 0
  }
]
```

Response Parameter	Type	Description
machineID	String	(Required) Machine machineID. See 4.1.1. machineID Machine Identifier
machineName	String	(Required) Machine name, configured in the “Add/Edit Device” dialog on the “Machine Configuration” page.
startTime	String	(Required) Start time of the grouping interval
endTime	String	(Required) End time of the grouping interval
availability	Float	(Required) Availability [%]
offTime	Int64	(Required) Off or offline time [seconds]
waitTime	Int64	(Required) Standby time [seconds]
emergencyTime	Int64	(Required) Emergency stop time [seconds]
autoRunTime	Int64	(Required) Auto-run time [seconds]
manualTime	Int64	(Required) Manual setup/adjustment time [seconds]

5.7.2.2. alarm Machine Group Alarm Data Analysis

Retrieves the alarm history of all machines in the machine group within the specified time range.

```
GET /api/group-analysis/alarm?
groupID=GROUPID&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier. See 4.1.2. groupID Group Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	End timestamp [seconds]. If undefined, the current time is used; a value later than the current time is truncated to the current time.

Response example

```
[
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2023-06-16T16:00:00Z",
    "endTime": "2023-06-16T16:00:10Z",
    "alarmMsg": "00:00 ALARM 1",
    "alarmLevel": "WRN"
  },
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2023-06-17T15:59:48.105Z",
    "endTime": "2023-06-17T16:00:00Z",
    "alarmMsg": "23:59 ALARM 2",
    "alarmLevel": "WRN"
  },
  {
    "machineID": "1011",
    "machineName": "模拟机台 2",
    "startTime": "2023-06-16T16:00:00.399Z",
    "endTime": "2023-06-16T16:00:30.395Z",
    "alarmMsg": "00:00 ALARM 1",
    "alarmLevel": "WRN"
  },
  {
```

```
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2023-06-16T16:01:00.008Z",
    "endTime": "2023-06-16T16:01:50.019Z",
    "alarmMsg": "00:01 ALARM 1",
    "alarmLevel": "WRN"
  },
  {
    "machineID": "1014",
    "machineName": "模拟机台 5",
    "startTime": "2023-06-17T15:59:06.702Z",
    "endTime": "2023-06-17T15:59:16.7Z",
    "alarmMsg": "23:59 ALARM 2",
    "alarmLevel": "WRN"
  }
]
```

Response Parameter	Type	Description
machineID	String	(Required) Target machine identifier. See 4.1.1. machineID Machine Identifier
machineName	String	(Required) Machine name, configured in the “Add/Edit Device” dialog on the “Machine Configuration” page.
startTime	String	(Required) Alarm start time (UTC)
endTime	String	(Required) Alarm end time (UTC)
alarmMsg	String	(Required) Alarm message
alarmLevel	String	(Required) Alarm level. If there are no alarms, an empty Array is returned. Levels are, from highest to lowest priority, Error (ERR), Warning (WRN), and Info (INF). Users can refer to 3.5.7. Alarm Monitor Settings to set alarm levels by keyword and filter out alarms below the minimum alarm level; alarms without a configured level default to Warning.

5.7.2.3. count Machine Group Count Data Analysis

Retrieves the count data of all machines in the machine group, for each grouping interval, within the specified time range.

```
GET /api/group-analysis/count?
groupID=GROUPID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier. See 4.1.2. groupID Group Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	End timestamp [seconds]. If undefined, the current time is used; a value later than the current time is truncated to the current time.
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.
enableCountPerProgram	Bool	Whether to count separately by main program. Defaults to false. When true, each grouping interval returns multiple records grouped by main program mainPrgmName, and count is the sum of the cycle counts deltaCount of that main program; if there is no cycle data within the time range, it falls back to counting by the difference of the cumulative count.

Request Parameter	Type	Description
sumByGroup	Bool	Whether to total the group’s output by count multiplier. Defaults to false. When true, each grouping interval returns a single group record: count is the sum of each machine’s count × that machine’s count multiplier countMultiplier in the group configuration (external machines included, machines with a multiplier of 0 do not count), and machineID and machineName carry the group’s groupID and name; enableCountPerProgram is ignored. For count multipliers see 3.4.1.1. Group General Settings.

Response example

```
[
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 96
  },
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0
  },
  {
    "machineID": "1011",
    "machineName": "模拟机台 2",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 52
  },
```

```
  {
    "machineID": "1011",
    "machineName": "模拟机台 2",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0
  }
]
```

Response Parameter	Type	Description
machineID	String	(Required) Machine machineID. See 4.1.1. machineID Machine Identifier
machineName	String	(Required) Machine name, configured in the “Add/Edit Device” dialog on the “Machine Configuration” page.
startTime	String	(Required) Start time of the grouping interval
endTime	String	(Required) End time of the grouping interval
count	Int32	(Required) Cumulative machining count, the difference between the machine’s cumulative machining count `cumulativeCount` at the start and end of the specified time range
mainPrgmName	String	Main program name. Returned only when the request parameter enableCountPerProgram is true; in that case each grouping interval returns multiple records, one per main program.

Response example totalled by count multiplier (sumByGroup is true; machines 1010 and 1011 have count multipliers 0 and 2)

```
[
  {
    "machineID": "g1",
```

```
    "machineName": "Group 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 104
  },
  {
    "machineID": "g1",
    "machineName": "Group 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0
  }
]
```

5.7.2.4. overall Machine Group Overall Data Analysis

Retrieves the overall data of all machines in the machine group, for each grouping interval, within the specified time range.

```
GET /api/group-analysis/overall?
groupID=GROUPID&startUnix=STARTUNIX&endUnix=ENDUNIX&interval=INTERVAL
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier. See 4.1.2. groupID Group Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	End timestamp [seconds]. If undefined, the current time is used; a value later than the current time is truncated to the current time.
interval	Int32	Grouping interval [seconds]. If undefined, there is only a single grouping interval.

Response example

```
[
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 96,
    "availability": 83.33333,
    "offTime": 100,
    "waitTime": 200,
    "emergencyTime": 300,
    "autoRunTime": 3000,
    "manualTime": 0.0
  },
  {
    "machineID": "1010",
    "machineName": "模拟机台 1",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0,
    "availability": 0.0,
    "offTime": 0,
    "waitTime": 3600,
    "emergencyTime": 0,
    "autoRunTime": 0,
    "manualTime": 0
  },
  {
    "machineID": "1011",
    "machineName": "模拟机台 2",
    "startTime": "2022-04-28T14:30:00Z",
    "endTime": "2022-04-28T15:30:00Z",
    "count": 52,
    "availability": 80.55556,
    "offTime": 150,
    "waitTime": 250,
    "emergencyTime": 300,
    "autoRunTime": 2900,
    "manualTime": 0.0
```

```
  },
  {
    "machineID": "1011",
    "machineName": "模拟机台 2",
    "startTime": "2022-04-28T15:30:00Z",
    "endTime": "2022-04-28T16:30:00Z",
    "count": 0,
    "availability": 0.0,
    "offTime": 0,
    "waitTime": 3600,
    "emergencyTime": 0,
    "autoRunTime": 0,
    "manualTime": 0
  }
]
```

Response Parameter	Type	Description
machineID	String	(Required) Machine machineID. See 4.1.1. machineID Machine Identifier
machineName	String	(Required) Machine name, configured in the “Add/Edit Device” dialog on the “Machine Configuration” page.
startTime	String	(Required) Start time of the grouping interval
endTime	String	(Required) End time of the grouping interval
count	Int32	(Required) Cumulative machining count, the difference between the machine’s cumulative machining count `cumulativeCount` at the start and end of the specified time range
availability	Float	(Required) Availability [%]
offTime	Int64	(Required) Off or offline time [seconds]
waitTime	Int64	(Required) Standby time [seconds]

Response Parameter	Type	Description
emergencyTime	Int64	(Required) Emergency stop time [seconds]
autoRunTime	Int64	(Required) Auto-run time [seconds]
manualTime	Int64	(Required) Manual setup/adjustment time [seconds]

5.7.2.5. cycle Machine Group Cycle Time Data Analysis

Retrieves the cycle data of all machines in the machine group within the specified time range.

```
GET /api/group-analysis/cycle?
groupID=GROUPID&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier. See 4.1.2. groupID Group Identifier
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	End timestamp [seconds]. If undefined, the current time is used; a value later than the current time is truncated to the current time.

Response example

```
[
  {
    "machineName": "模拟机台 1",
    "startTime": "2025-05-23T06:35:44.046Z",
    "endTime": "2025-05-23T06:36:25.046Z",
    "lastCycleTime": 41,
    "mainPrgmName": "O6495",
    "machineID": "1"
  },
```

```
  {
    "machineName": "模拟机台 1",
    "startTime": "2025-05-23T06:36:25.072Z",
    "endTime": "2025-05-23T06:36:56.072Z",
    "lastCycleTime": 31,
    "mainPrgmName": "O6495",
    "machineID": "1"
  },
  {
    "machineName": "模拟机台 2",
    "startTime": "2025-05-23T07:51:50.777Z",
    "endTime": "2025-05-23T07:52:10.777Z",
    "lastCycleTime": 20,
    "mainPrgmName": "O2228",
    "machineID": "2"
  },
  {
    "machineName": "模拟机台 2",
    "startTime": "2025-05-23T07:52:10.842Z",
    "endTime": "2025-05-23T07:52:30.842Z",
    "lastCycleTime": 20,
    "mainPrgmName": "O2228",
    "machineID": "2"
  }
]
```

Response Parameter	Type	Description
machineID	String	(Required) Machine machineID. See 4.1.1. machineID Machine Identifier
machineName	String	(Required) Machine name, configured in the “Add/Edit Device” dialog on the “Machine Configuration” page.
startTime	String	(Required) Cycle start time (UTC)
endTime	String	(Required) Cycle end time (UTC)
lastCycleTime	Int64	(Required) Cycle duration [seconds]; only auto-run time is counted.

Response Parameter	Type	Description
mainPrgmName	String	(Required) Name of the main program executed during the cycle

5.8. History Data Interface

The history data interface is used to retrieve historical data for a target machine or machine group within a specified time range, with base path `/api/db`.

Before using the history data analysis interface, the gateway’s local cache switch must be enabled (see 3.12. Local Cache), to allow historical data to be saved locally on the gateway.

Using the history data analysis interface requires supplying a start timestamp and an end timestamp. The maximum span between the two is 31 days.

The maximum local retention period is 365 days. Therefore, the start time cannot be earlier than 365 days before the current time.

If the current timestamp is earlier than the supplied end timestamp, the current timestamp is automatically used as the end timestamp.

If the gateway has not received machine or machine group data for a period of time (data gap), the data returned by the history data interface may differ from the actual data. Common causes of data gaps include: the gateway’s local cache not being enabled, the corresponding automatic collection task not being enabled, the gateway losing power, the gateway’s network connection being interrupted, and the machine being offline.

5.8.1. machine — Machine History Data

Machine history data is used to retrieve historical data for a target machine within a specified time range.

```
GET /api/db/machine?
machineID=MACHINEID&type=TYPE&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 4.1.1. machineID Machine Identifier
type	String	(Required) Data class, see 4.2. Data Description
startUnix	Int64	(Required) Start timestamp [seconds]

Request Parameter	Type	Description
endUnix	Int64	(Required) End timestamp [seconds]
uid	String	(Optional) UID of the source gateway. When omitted, only data for this gateway’s local machines is queried; when supplied, the query returns data for external machines forwarded by the gateway with that UID
limit	Int32	(Optional) Maximum number of records to return; when omitted, there is no limit. A GET request cannot supply a sort order, so results are returned in ascending order by time, i.e. the earliest records are taken

Example

Retrieve historical machine status data for the machine with machineID 1010, from 10:00 to 11:00 on the morning of March 1, 2024, with type=CNCStatus. The request is as follows:

```
GET /api/db/machine?
machineID=1010&type=CNCStatus&startUnix=1709258400&endUnix=1709262000
```

If no historical data exists for the target machine within the specified time range, an empty array is returned:

```
[]
```

If historical data exists, the data for the specified data class is returned in JSON format:

```
[
  {
    "cncStatus": "MANUAL_INC",
    "adjustedStatus": "MANUAL_INC",
    "alarmStatus": "ALARM",
    "alarmLevel": "WRN",
    "offTime": 2329,
```

```
    "waitTime": 3212,
    "emergencyTime": 232,
    "autoRunTime": 1336,
    "manualTime": 1232,
    "machineID": "2112",
    "time": "2024-03-01T02:22:11.442Z"
  },
  {
    "cncStatus": "MANUAL_INC",
    "adjustedStatus": "MANUAL_INC",
    "alarmStatus": "ALARM",
    "alarmLevel": "WRN",
    "offTime": 2329,
    "waitTime": 3212,
    "emergencyTime": 232,
    "autoRunTime": 1336,
    "manualTime": 1237,
    "machineID": "2112",
    "time": "2024-03-01T02:22:16.44Z"
  },
  ...
]
```

Among the response parameters, machineID is the machine’s machineID (see 4.1. Identifiers), and time is the time the data was captured. For other response parameters, see the `field` and `tag` descriptions for the corresponding `type` in 4.2. Data Description.

5.8.2. group-machine — History Data for All Machines in a Group

Used to retrieve historical data for all machines in a target machine group within a specified time range.

```
GET /api/db/group-machine?
groupID=GROUPID&type=TYPE&startUnix=STARTUNIX&endUnix=ENDUNIX
```

Request Parameter	Type	Description
groupID	String	(Required) Target group identifier, see 4.1.2. groupID Group Identifier

Request Parameter	Type	Description
type	String	(Required) Data class, see 4.2. Data Description
startUnix	Int64	(Required) Start timestamp [seconds]
endUnix	Int64	(Required) End timestamp [seconds]
limit	Int32	(Optional) Maximum number of records to return; when omitted, there is no limit. The limit applies to each data source separately (this gateway’s local machines and each external UID are limited individually), not to the total number of records in the merged result

Example

Retrieve historical machine status data for all machines in the group with groupID g1, from 10:00 to 11:00 on the morning of March 1, 2024, with type=CNCStatus. The request is as follows:

```
GET /api/db/group-machine?
groupID=g1&type=CNCStatus&startUnix=1709258400&endUnix=1709262000
```

If no historical data exists for the target machines within the specified time range, an empty array is returned:

```
[]
```

If historical data exists, the data for the specified data class is returned in JSON format:

```
[
  {
    "cncStatus": "MANUAL_INC",
    "adjustedStatus": "MANUAL_INC",
    "alarmStatus": "ALARM",
    "alarmLevel": "WRN",
    "offTime": 2329,
    "waitTime": 3212,
```

```
    "emergencyTime": 232,
    "autoRunTime": 1336,
    "manualTime": 1232,
    "machineID": "2112",
    "time": "2024-03-01T02:22:11.442Z"
  },
  {
    "cncStatus": "MANUAL_INC",
    "adjustedStatus": "MANUAL_INC",
    "alarmStatus": "ALARM",
    "alarmLevel": "WRN",
    "offTime": 2329,
    "waitTime": 3212,
    "emergencyTime": 232,
    "autoRunTime": 1336,
    "manualTime": 1237,
    "machineID": "2112",
    "time": "2024-03-01T02:22:16.44Z"
  },
  ...
  {
    "cncStatus": "AUTO_RUN",
    "adjustedStatus": "AUTO_RUN",
    "alarmStatus": "NO_ALARM",
    "offTime": 12229,
    "waitTime": 12312,
    "emergencyTime": 2332,
    "autoRunTime": 52332,
    "manualTime": 6132,
    "machineID": "3",
    "time": "2024-03-01T02:19:29.308Z"
  },
  ...
]
```

Among the response parameters, machineID is the machine’s machineID (see 4.1. Identifiers), and time is the time the data was captured. For other response parameters, see the `field` and `tag` descriptions for the corresponding `type` in 4.2. Data Description.

5.8.3. query — Query History Data

Queries historical data matching the specified conditions, similar to a database query command.

```
POST /api/db/query
```

All parameters of this interface are supplied in the application/json request body; parameters in the URL query string are ignored. This interface does not support `groupID` — to restrict a query to a machine group, express the group' s machines as a filter condition, for example `{"key":"machineID","operator":"in","value":["1010","1011"]}`.

Request body example (application/json)

Query the "PLC" class, with timestamps between 1704892831 and 1708893358, where machineID is Simulated Machine 1 and tag is either Ttag1 or Ttag2, returning the earliest 2 records (the first 2 records in ascending order by time).

```
{
  "type": "PLC",
  "startUnix": 1704892831,
  "endUnix": 1708893358,
  "filters": [
    {
      "key": "machineID",
      "operator": "equal",
      "value": "模拟机台 1"
    },
    {
      "key": "tag",
      "operator": "in",
      "value": [
        "Ttag1",
        "Ttag2"
      ]
    }
  ],
  "orderBy": {
    "field": "time",
    "isDesc": false
```

```
  },
  "limit": 2
}
```

Request Parameter	Type	Description
type	String	(Required) Data class, see 4.2. Data Description
startUnix	Int64	(Optional) Start timestamp [seconds]; defaults to 0
endUnix	Int64	(Optional) End timestamp [seconds]; defaults to the current time
filters	Object[]	(Optional) Filter conditions; when omitted, no filtering is applied
key	String	(Required) Key
operator	String	(Optional) Comparison operator; defaults to `equal`. Accepted values: `equal`, `greaterEqual`, `greater`, `less`, `lessEqual`, `in` (value is an array), `null`, `notNull`
value	Object	(Required) Comparison value, of type String or String[]
orderBy	Object	(Optional) Sort order; defaults to ascending by time
field	String	(Required) Sort field; currently only `time` is supported. With any other value the sort is ignored and the data is returned in the database’s default order
isDesc	Bool	(Required) true: descending, false: ascending
limit	Int32	(Optional) Maximum number of records; when omitted, there is no limit

> **Note**
> Every filter condition must supply a non-null `key` and `value`; otherwise the request returns error code 10045 (invalid filter conditions). The `null` and `notNull` operators therefore also require a placeholder `value`, which is ignored.

Response example

```
[
  {
    "data": [
      255
    ],
    "tag": "Ttag1",
    "machineID": "模拟机台 1",
    "time": "2025-04-28T06:30:19.827Z"
  },
  {
    "data": [
      2147483647
    ],
    "tag": "Ttag2",
    "machineID": "模拟机台 1",
    "time": "2025-04-28T06:30:19.848Z"
  }
]
```

Among the response parameters, machineID is the machine’s machineID (see 4.1. Identifiers), and time is the time the data was captured. For other response parameters, see the `field` and `tag` descriptions for the corresponding `type` in 4.2. Data Description.

5.9. Gateway Configuration Interface

The gateway function interface is used to command the gateway to perform specific functions, with base address `/api/config`.

5.9.1. Global Configuration Interface

5.9.1.1. gateway-info Get Gateway Information

This interface has no request parameters.

```
GET /api/config/gateway-info
```

Response example

```
{
  "hid": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
  "uid": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
  "alias": "iotgw"
}
```

Response Parameter	Type	Description
hid	String	(Required) Gateway hardware ID
uid	String	(Required) Gateway UID
alias	String	(Required) Gateway name

5.9.1.2. settings Get Gateway Global Settings

This interface has no request parameters.

```
GET /api/config/settings
```

Response example

```
{
```

```
  "version": "1.19.3.102",
  "enableLocalCaching": true,
  "enableRemoteService": true,
  "enableLan2Setup": false,
  "timeServerAddress":
"ntp1.aliyun.com;ntp2.aliyun.com;ntp3.aliyun.com;ntp4.aliyun.com;"
}
```

Response Parameter	Type	Description
version	String	(Required) Settings version
enableLocalCaching	Bool	(Required) Enable local caching
enableRemoteService	Bool	(Required) Enable remote assistance
enableLan2Setup	Bool	(Required) Enable LAN2 setup
timeServerAddress	String	(Required) Time server address

5.9.1.3. update-settings Update Gateway Global Settings

```
POST /api/config/update-settings
```

Request body example application/json

```
{
  "enableLocalCaching": true,
  "enableRemoteService": true,
  "enableLan2Setup": false,
  "timeServerAddress":
"ntp1.aliyun.com;ntp2.aliyun.com;ntp3.aliyun.com;ntp4.aliyun.com;"
}
```

Request Parameter	Type	Description
enableLocalCaching	Bool	Enable local caching
enableRemoteService	Bool	Enable remote assistance
enableLan2Setup	Bool	Enable LAN2 setup
timeServerAddress	String	Time server address

Response example

```
{
  "version": "1.19.3.102",
  "enableLocalCaching": true,
  "enableRemoteService": true,
  "enableLan2Setup": false,
  "timeServerAddress":
"ntp1.aliyun.com;ntp2.aliyun.com;ntp3.aliyun.com;ntp4.aliyun.com;"
}
```

Response Parameter	Type	Description
version	String	(Required) Settings version
enableLocalCaching	Bool	(Required) Enable local caching
enableRemoteService	Bool	(Required) Enable remote assistance
enableLan2Setup	Bool	(Required) Enable LAN2 setup
timeServerAddress	String	(Required) Time server address

5.9.1.4. security Get Gateway Global Security Settings

This interface has no request parameters.

```
GET /api/config/security
```

Response example

```
{
  "enable": true,
  "protectedApi": "exact,/cnc/writeOffsetData;exact,/cnc/writePlcData"
}
```

Response Parameter	Type	Description
enable	Bool	(Required) Enable global security control

Response Parameter	Type	Description
protectedApi	String	Protected APIs, i.e. APIs forbidden to all users. Not returned means not set. For the API command format, see API Command Format.

API Command Format

API Command Type	exact	prefix
Description	Specifies a single, exact API address	Specifies all API addresses sharing the given prefix
Example	exact,/cnc/writePlcData	prefix,/db

The API address starts right after `/api`. `prefix,/` represents all API addresses.

Multiple API address commands are separated by `;`.

5.9.1.5. update-security Update Global Security Settings

```
POST /api/config/update-security
```

Request body example application/json

```
{
  "enable": true,
  "protectedApi": "exact,/cnc/writeOffsetData;exact,/cnc/writePlcData"
}
```

Request Parameter	Type	Description
enable	Bool	(Required) Enable global security control
protectedApi	String	Protected APIs, i.e. APIs forbidden to all users. For the API command format, see API Command Format.

Response example

```
{
  "enable": true,
  "protectedApi": "exact,/cnc/writeOffsetData;exact,/cnc/writePlcData"
}
```

Response Parameter	Type	Description
enable	Bool	(Required) Enable global security control
protectedApi	String	Protected APIs, i.e. APIs forbidden to all users. Not returned means not set. For the API command format, see API Command Format.

5.9.1.6. backup Back Up Gateway Configuration

Exports the current gateway configuration file. The returned string is the input of the 2.9.1.7. restore Restore Gateway Configuration interface.

This interface has no request parameters.

```
GET /api/config/backup
```

Response example

```
{
  "base64": "XXXXXXXXXXXXXXXXXXXX"
}
```

Response Parameter	Type	Description
base64	String	(Required) Encrypted gateway configuration file content, Base64 encoded.

When the configuration file cannot be read, GENERAL_CANNOT_ACCESS_CONFIG is returned.

5.9.1.7. restore Restore Gateway Configuration

```
POST /api/config/restore
```

Request body example application/json

```
{
  "base64": "XXXXXXXXXXXXXXXXXXXX"
}
```

Request Parameter	Type	Description
base64	String	(Required) Backed-up gateway configuration file content, i.e. the content returned by the 2.9.1.6. backup Back Up Gateway Configuration interface.

On success, no content is returned.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

> **Note**
>
> After a successful restore, the gateway restarts automatically to apply the new configuration.

When the content is empty or cannot be decrypted, GENERAL_API_INVALID_REQUEST is returned; when the configuration file is not accessible, GENERAL_CANNOT_ACCESS_CONFIG is returned.

5.9.1.8. reset Reset Gateway Configuration

This interface has no request parameters.

```
GET /api/config/reset
```

On success, no content is returned.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

Note

This interface clears all gateway configuration (machines, groups, users, communication, etc.) and restores the factory default settings (only the default account is kept). The operation cannot be undone; it is recommended to back up the configuration with the 2.9.1.6. backup Back Up Gateway Configuration interface first.

5.9.2. User Configuration API

5.9.2.1. user — Get user settings

```
GET /api/config/user
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier

Response example

```
{
  "id": 0,
  "username": "admin",
  "isAdmin": true
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
username	String	(Required) Username
isAdmin	Bool	(Required) Whether the user has administrator privileges
roles	String[]	Functional role list (page permissions) for non-administrator users. Not returned if not set.

> **Note**
> Administrator users implicitly have the admin role; `roles` only takes effect for non-administrator users.

5.9.2.2. users — Get all user settings

This API has no request parameters.

```
GET /api/config/users
```

Response example

```
[
  {
    "id": 0,
    "username": "admin",
    "isAdmin": true
  },
  {
    "id": 1,
    "username": "user1",
    "isAdmin": false
  }
]
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
username	String	(Required) Username
isAdmin	Bool	(Required) Whether the user has administrator privileges
roles	String[]	Functional role list (page permissions) for non-administrator users. Not returned if not set.

5.9.2.3. create-user — Create a new user

When no password is provided, the new user’s initial password is the same as the username.

```
POST /api/config/create-user
```

Request body example application/json

```
{
  "username": "user2",
  "isAdmin": false
}
```

Request Parameter	Type	Description
username	String	(Required) Username
isAdmin	Bool	Whether the user has administrator privileges. Defaults to true.
roles	String[]	Functional role list (page permissions) for non-administrator users. Not set if omitted.
password	String	Initial password. Defaults to the username when omitted. The password is never returned in the response.

Response example

```
{
  "id": 2,
  "username": "user2",
  "isAdmin": false
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
username	String	(Required) Username
isAdmin	Bool	(Required) Whether the user has administrator privileges
roles	String[]	Functional role list (page permissions) for non-administrator users. Not returned if not set.

5.9.2.4. update-user — Update user settings

Changing the username does not change the password.

```
POST /api/config/update-user
```

Request body example application/json

```
{
  "id": 2,
  "username": "user2",
  "isAdmin": false
}
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
username	String	Username. Not changed by default.
isAdmin	Bool	Whether the user has administrator privileges. Not changed by default; however, when this user is the last remaining administrator in the system, `isAdmin: true` must be sent explicitly, otherwise INVALID_CONFIG_SETTING (At least one admin.) is returned.
roles	String[]	Functional role list (page permissions) for non-administrator users. Not changed by default.

Response example

```
{
  "id": 2,
  "username": "user2",
  "isAdmin": false
```

```
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
username	String	(Required) Username
isAdmin	Bool	(Required) Whether the user has administrator privileges
roles	String[]	Functional role list (page permissions) for non-administrator users. Not returned if not set.

5.9.2.5. delete-user — Delete a user

```
GET /api/config/delete-user
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code. 0 indicates success.
errorMsg	String	(Required) Error message

5.9.2.6. user-security — Get user security settings

```
GET /api/config/user-security
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier

Response example

```
{
  "id": 2,
  "secretKey": "sk-XXXXXXXXXX",
  "authorizedApis":
"prefix,/cnc;prefix,/db;prefix,/analysis;prefix,/group-
analysis;prefix,/config;prefix,/core;prefix,/gateway;prefix,/auth;"
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
secretKey	String	The secret key, prefixed with “sk-”. Not returned if not set.
authorizedApis	String	Authorized APIs, i.e. the APIs the user is allowed to use. Not returned if not set. For the API command format, see API Command Format.

5.9.2.7. update-user-security — Update user security settings

```
POST /api/config/update-user-security
```

Request body example application/json

```
{
  "id": 2,
  "secretKey": "sk-XXXXXXXXXX",
```

```
  "authorizedApis":
"prefix,/cnc;prefix,/db;prefix,/analysis;prefix,/group-
analysis;prefix,/config;prefix,/core;prefix,/gateway;prefix,/auth;"
}
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
secretKey	String	The secret key, prefixed with “sk-”. The key must be unique across all users; a duplicate returns GENERAL_CONFIG_DUPLICATE_VALUE (Secret key already exists.).
authorizedApis	String	Authorized APIs, i.e. the APIs the user is allowed to use. Not returned if not set. For the API command format, see API Command Format.

Response example

```
{
  "id": 2,
  "secretKey": "sk-XXXXXXXXXX",
  "authorizedApis":
"prefix,/cnc;prefix,/db;prefix,/analysis;prefix,/group-
analysis;prefix,/config;prefix,/core;prefix,/gateway;prefix,/auth;"
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 4.1.4. ID Database Identifier
secretKey	String	The secret key, prefixed with “sk-”. Not returned if not set.
authorizedApis	String	Authorized APIs, i.e. the APIs the user is allowed to use. Not returned if not set. For the API command format, see API Command Format.

5.9.3. Machine Configuration Interface

The machine configuration interface is used to retrieve, add, modify, or delete machine configurations. For more information on machine configuration, see 3.3. Machine Configuration. All configuration parameters for a machine are listed in the table below; these parameters are used as request or response parameters in the machine configuration interface.

Machine Configuration Parameters

Parameter	Type	Description
id	Int32	Database identifier. See 4.1.4. ID Database Identifier
machineType	String	Machine type. See machineType Machine Type Mapping
system	String	System. The value range corresponds to the System options in the Add Machine window under the English UI language.
model	String	Model. The value range corresponds to the Model options in the Add Machine window under the English UI language.
name	String	Machine name
machineID	String	Machine identifier. See 4.1.1. machineID Machine Identifier
slaveID	Int32	Slave identifier. See 4.1.3. slaveID Slave Identifier
ip	String	IP address
port	Int32	Port number. 0 represents each device’s default port.
isActive	Bool	Activation status. true = active, false = inactive
useDefaultMachineID	Bool	Use default machine identifier
useDefaultSlaveID	Bool	Use default slave identifier
customAttributes	String	Custom attributes. Example: `{"attribute": value,…}`

Parameter	Type	Description
username	String	Username. Required for some devices.
permission	String	Permission. Only required for Delta CNC systems. Range: NORMAL, USER_1, USER_2, MACHINE, SYSTEM. Default is NORMAL.
password	String	Password. Required for some devices.
version	String	Version. Required for some devices.
connectionMode	String	Connection mode. Required for some devices.
encoding	String	Encoding, e.g. ASCII, UTF-8, GBK, etc. Default is “default” (auto-detect).
language	String	Alarm language. Only required for CNC-Siemens systems and Robot-ABB (RobotWare 6.0); currently supports English and Chinese. Default is Chinese.
numberOfTasks	Int32	Number of enabled tasks. Read-only.
statusMsg	String	Status message (read-only). Returned when the creation/modification succeeds but a warning applies, e.g. a duplicate slave identifier, or the number of active machines exceeds the license.
taskAlarm	Bool	Alarm information task. true = enabled, false = disabled.
taskAlarmPrecond	String	Precondition for the alarm information task. Not returned if not set.

Parameter	Type	Description
taskAlarmEnableCustomAlarm	Bool	Custom alarm. true = enabled, false = disabled. When enabled, the custom alarm replaces the default alarm. Custom alarm task execution logic: first check the alarm status; if there is an alarm, retrieve the alarm content. If this tag is empty, skip checking the alarm status and retrieve the alarm content directly, determining the alarm status by whether alarm content is present.
taskAlarmCustomAlarmAlarmStatusTag	String	Custom alarm alarmStatus tag. Enter the corresponding PLC data task result tag.
taskAlarmCustomAlarmAlarmMsgTag	String	Custom alarm alarmMsg tag. Enter the corresponding PLC data task result tag.
taskAlarmEnableAlarmMapping	Bool	Alarm mapping. true = enabled, false = disabled.
taskAlarmAlarmMappingFilename	String	Alarm mapping file name.
taskMachineStatus	Bool	Machine status task. true = enabled, false = disabled.
taskChannelMachineStatus	String	Target channel for the machine status task, e.g. “0;1-3”.
taskMachineStatusPrecond	String	Precondition for the machine status task. Not returned if not set.
taskMachineStatusEnableAdjustedManual	Bool	Machine status task setup-status correction. true = enabled, false = disabled.
taskMachineStatusMaxManualPauseTime	Int32	Maximum setup-pause time for the machine status task (seconds).
taskMachineStatusEnableStatusConversion	Bool	Machine status task status conversion. true = enabled, false = disabled.
taskMachineStatusStatusConversionSettings	String	Status to convert to when the machine status task’s condition is met.

Parameter	Type	Description
taskMachineStatusEnableCustomStatus	Bool	Custom machine status task. true = enabled, false = disabled. When enabled, the custom status replaces the default status.
taskMachineStatusCustomStatusCNCStatusTag	String	Custom status cncStatus tag. Enter the corresponding PLC data task result tag.
taskMachineStatusCustomStatusAlarmStatusTag	String	Custom status alarmStatus tag. Enter the corresponding PLC data task result tag.
taskCount	Bool	Machining count task. true = enabled, false = disabled.
taskCountPrecond	String	Precondition for the machining count task. Not returned if not set.
taskCountEnableCustomCount	Bool	Custom machining count task. true = enabled, false = disabled. When enabled, the custom machining count replaces the default machining count.
taskCountCustomCountCountTag	String	Custom machining count tag. Enter the corresponding PLC data task result tag.
taskCurrentToolNo	Bool	Current tool number task. true = enabled, false = disabled.
taskChannelCurrentToolNo	String	Target channel for the current tool number task, e.g. “0;1-3”.
taskCurrentToolNoPrecond	String	Precondition for the current tool number task. Not returned if not set.
taskEnergyConsumption	Bool	Energy consumption task. true = enabled, false = disabled.
taskEnergyConsumptionPrecond	String	Precondition for the energy consumption task. Not returned if not set.
taskFeed	Bool	Feed rate task. true = enabled, false = disabled.
taskChannelFeed	String	Target channel for the feed rate task, e.g. “0;1-3”.

Parameter	Type	Description
taskFeedPrecond	String	Precondition for the feed rate task. Not returned if not set.
taskFeedAndSpindle	Bool	Feed rate and spindle speed task. true = enabled, false = disabled.
taskChannelFeedAndSpindle	String	Target channel for the feed rate and spindle speed task, e.g. “0;1-3”.
taskFeedAndSpindlePrecond	String	Precondition for the feed rate and spindle speed task. Not returned if not set.
taskLaserPower	Bool	Laser power task. true = enabled, false = disabled.
taskLaserPowerPrecond	String	Precondition for the laser power task. Not returned if not set.
taskLoad	Bool	Load data task. true = enabled, false = disabled.
taskChannelLoad	String	Target channel for the load data task, e.g. “0;1-3”.
taskLoadPrecond	String	Precondition for the load data task. Not returned if not set.
taskLogHistory	Bool	Log history task. true = enabled, false = disabled.
taskLogHistoryPrecond	String	Precondition for the log history task. Not returned if not set.
taskOverLoad	Bool	Overload monitoring task. true = enabled, false = disabled.
taskChannelOverLoad	String	Target channel for the overload monitoring task, e.g. “0;1-3”.
taskOverLoadPrecond	String	Precondition for the overload monitoring task. Not returned if not set.
taskPlcData	Bool	PLC data task. true = enabled, false = disabled.
taskPlcDataSettings	String	PLC data task settings
taskPosition	Bool	Position data task. true = enabled, false = disabled.

Parameter	Type	Description
taskChannelPosition	String	Target channel for the position data task, e.g. “0;1-3”.
taskPositionPrecond	String	Precondition for the position data task. Not returned if not set.
taskCurrentProgramBlock	Bool	Current program block task. true = enabled, false = disabled.
taskChannelCurrentProgramBlock	String	Target channel for the current program block task, e.g. “0;1-3”.
taskCurrentProgramBlockPrecond	String	Precondition for the current program block task. Not returned if not set.
taskProgramInfo	Bool	Machining program task. true = enabled, false = disabled.
taskChannelProgramInfo	String	Target channel for the machining program task, e.g. “0;1-3”.
taskProgramInfoPrecond	String	Precondition for the machining program task. Not returned if not set.
taskMachineTimeData	Bool	Machine time task. true = enabled, false = disabled.
taskMachineTimeDataPrecond	String	Precondition for the machine time task. Not returned if not set.
taskToolLife	Bool	Tool life task. true = enabled, false = disabled.
taskToolLifePrecond	String	Precondition for the tool life task. Not returned if not set.
taskToolLifeGroupNumIndex	String	Target tool for the tool life task, in {tool group number.index within group} format.
taskToolLifeIndex	String	Target tool for the tool life task, in {index} format.
taskToolLifeToolNum	String	Target tool for the tool life task, in {tool number} format.
taskToolLifeToolNumOffsetNum	String	Target tool for the tool life task, in {tool number.offset number} format.

Parameter	Type	Description
taskToolOffset	Bool	Tool offset task. true = enabled, false = disabled.
taskChannelToolOffset	String	Target channel for the tool offset task, e.g. “0;1-3”.
taskToolOffsetPrecond	String	Precondition for the tool offset task. Not returned if not set.
taskToolOffsetToolNum	String	Target tool for the tool offset task, in {tool number} format.
taskToolOffsetOffsetNum	String	Target tool for the tool offset task, in {offset number} format.
taskToolOffsetToolNumOffsetNum	String	Target tool for the tool offset task, in {tool number.offset number} format.
taskAlarmHistory	Bool	Alarm history task. true = enabled, false = disabled.
taskAlarmHistoryPrecond	String	Precondition for the alarm history task. Not returned if not set.
taskCycleData	Bool	Cycle time data task. true = enabled, false = disabled.
taskCycleDataPrecond	String	Precondition for the cycle time data task. Not returned if not set.
taskOEE	Bool	OEE monitoring task. true = enabled, false = disabled.
taskOEEPrecond	String	Precondition for the OEE monitoring task. Not returned if not set.
taskExternalPlcDataSettings	String	External PLC task settings
networkErrorAsOffline	Bool	Treat a network error as offline. true = enabled, false = disabled. Default is false.
parallelProcessing	Int	Number of parallel task processes

Parameter	Type	Description
fileServerType	String	File server type (program transfer). Possible values: Machine Memory, Shared Folder, Shared Folder (Win XP), FTP Server, Wireless Disk, Gateway File Server, Scp (Scp is available through the interface only and is not exposed in the UI).
fileServerAddress	String	Server address (program transfer).
fileServerPort	Int32	Custom port (program transfer).
fileServerUsername	String	Username (program transfer).
fileServerPassword	String	Password (program transfer).
fileServerUCode	String	U code (program transfer).
fileServerRootDir	String	Root directory (program transfer).
modbusSlaveID	Int32	Slave ID (general). Required for MODBUS communication. Default is 1.
modbusByteOrder4	String	32-bit format. Required for MODBUS communication. Range: DCBA, BADC, ABCD, CDAB.
modbusByteOrder8	String	64-bit format. Required for MODBUS communication. Range: HGFEDCBA, BADCFEHG, ABCDEFGH, GHEFCDAB.
modbusPLCAddress	Bool	Use PLC addresses for standard MODBUS models and Inovance MODBUS connections. Default: false. When true, addresses start at 1 (address 1 maps to protocol address 0); otherwise, they start at 0. For Inovance, this affects both the 0x/1x/3x/4x areas and element addresses such as D, M and MW.
modbusReverseString	Bool	Reverse string. Required for MODBUS communication. true = enabled, false = disabled.
plcRack	Int32	Rack number. Required for some devices.

Parameter	Type	Description
plcSlot	Int32	Slot number. Required for some devices.
plcStation	Int32	Station number. Required for some devices.
plcCommunicationFormat	String	Communication format. Required for some devices.
plcCommunicationType	String	Communication type. Required for some devices.
plcTsapSystemOfNumeration	String	TSAP numeral system. Required for some devices. Range: Decimal, Hexadecimal.
plcLocalTsap	String	Local TSAP. Required for some devices.
plcRemoteTsap	String	Remote TSAP. Required for some devices.
plcRouter	String	Router. Required for some devices.
plcDA2	Int32	DA2. Required for some devices.
plcSA1	Int32	SA1. Required for some devices.
plcSA2	Int32	SA2. Required for some devices.
plcGCT	Int32	GCT. Required for some devices.
plcSID	Int32	SID. Required for some devices.
plcSumCheck	Bool	Checksum. Required for some devices. true = enabled, false = disabled.
plcReverseString	Bool	Reverse string. Required for some devices. true = enabled, false = disabled.
plcAutoRunValue	Int32	Auto-run status value. Required for some devices. Possible values: 0, 1. Default is 1.
plcUseStation	Bool	Use station number. Required for some devices. true = enabled, false = disabled.

> **Note**
> The interface only returns parameters that have been set. Parameters such as username, version, connectionMode, and the precondition of each task are not returned when not set. The password parameter is returned together with the configuration for devices that require a password (such as Delta, Siemens, Okuma, Fagor, Heidenhain, and LNC); it is not returned for devices that do not require one. Different machineType values support different tasks, and the interface only returns the parameters relevant to the tasks supported by the machine.

machineType Machine Type Mapping

The types currently supported by machineType are listed in the table below.

machineType	Machine Type
CNC	CNC
Laser	Laser cutting machine
PLC	PLC
Robot	Robot

> **Note**
> machineType values are case-sensitive and must match the table exactly. Submitting other casings such as `LASER` or `ROBOT` returns INVALID_CONFIG_SETTING (Invalid machine type).

5.9.3.1. machine Get Machine Configuration

```
GET /api/config/machine?ID=ID&machineID=MACHINEID
```

Request Parameter	Type	Description
id	Int32	Database identifier. See 4.1.4. ID Database Identifier
machineID	String	Target machine identifier. See 4.1.1. machineID Machine Identifier. If both ID and machineID are provided, machineID is ignored.

Response example


```
{
  "id": 1,
  "machineType": "CNC",
  "system": "Mock",
  "model": "General",
  "name": "演示 1",
  "useDefaultMachineID": true,
  "machineID": "1",
  "useDefaultSlaveID": true,
  "slaveID": 1,
  "customAttributes": "",
  "encoding": "Default",
  "ip": "127.0.0.1",
  "port": 0,
  "isActive": true,
  "numberOfTasks": 20,
  "taskAlarm": true,
  "taskAlarmEnableCustomAlarm": false,
  "taskAlarmEnableAlarmMapping": false,
  "taskMachineStatus": true,
  "taskChannelMachineStatus": "0",
  "taskMachineStatusEnableAdjustedManual": false,
  "taskMachineStatusMaxManualPauseTime": 600,
  "taskMachineStatusEnableStatusConversion": false,
  "taskMachineStatusEnableCustomStatus": false,
  "taskCount": true,
  "taskCountEnableCustomCount": false,
  "taskCountPrecond": "CNCStatus_cncStatus = \"AUTO_RUN\" and
Load_spindleLoad[0] > 50",
  "taskCurrentToolNo": true,
  "taskChannelCurrentToolNo": "0",
  "taskEnergyConsumption": true,
  "taskFeedAndSpindle": true,
  "taskChannelFeedAndSpindle": "0",
  "taskLoad": true,
  "taskChannelLoad": "0",
  "taskLogHistory": true,
  "taskOverLoad": true,
```

```
  "taskChannelOverLoad": "0",
  "taskPlcData": true,
  "taskPlcDataSettings": "R,0,10,Int16;Y,0,3,Bit0",
  "taskPosition": true,
  "taskChannelPosition": "0",
  "taskCurrentProgramBlock": true,
  "taskChannelCurrentProgramBlock": "0",
  "taskProgramInfo": true,
  "taskChannelProgramInfo": "0",
  "taskMachineTimeData": true,
  "taskToolLife": true,
  "taskToolOffset": true,
  "taskChannelToolOffset": "0",
  "taskToolOffsetToolNumOffsetNum": "1.2;3-5.1;7-9.1-3",
  "taskAlarmHistory": true,
  "taskCycleData": true,
  "taskOEE": true,
  "networkErrorAsOffline": false,
  "parallelProcessing": 1,
  "fileServerType": "Machine Memory",
  "fileServerRootDir": ""
}
```

See Machine Configuration Parameters for the response parameters.

5.9.3.2. machines Get All Machine Configurations

This interface has no request parameters.

```
GET /api/config/machines
```

Response example

```
[
  {
    "id": 1,
    "machineType": "CNC",
    "system": "Mock",
```

```
    "model": "General",
    "name": "演示 1",
    "useDefaultMachineID": true,
    "machineID": "1",
    "useDefaultSlaveID": true,
    "slaveID": 1,
    "customAttributes": "",
    "encoding": "Default",
    "ip": "127.0.0.1",
    "port": 0,
    "isActive": true,
    "numberOfTasks": 20,
    "taskAlarm": true,
    "taskAlarmEnableCustomAlarm": false,
    "taskAlarmEnableAlarmMapping": false,
    "taskMachineStatus": true,
    "taskChannelMachineStatus": "0",
    "taskMachineStatusEnableAdjustedManual": false,
    "taskMachineStatusMaxManualPauseTime": 600,
    "taskMachineStatusEnableStatusConversion": false,
    "taskMachineStatusEnableCustomStatus": false,
    "taskCount": true,
    "taskCountEnableCustomCount": false,
    "taskCountPrecond": "CNCStatus_cncStatus = \"AUTO_RUN\" and
Load_spindleLoad[0] > 50",
    "taskCurrentToolNo": true,
    "taskChannelCurrentToolNo": "0",
    "taskEnergyConsumption": true,
    "taskFeedAndSpindle": true,
    "taskChannelFeedAndSpindle": "0",
    "taskLoad": true,
    "taskChannelLoad": "0",
    "taskLogHistory": true,
    "taskOverLoad": true,
    "taskChannelOverLoad": "0",
    "taskPlcData": true,
    "taskPlcDataSettings": "R,0,10,Int16;Y,0,3,Bit0",
    "taskPosition": true,
    "taskChannelPosition": "0",
    "taskCurrentProgramBlock": true,
```

```
      "taskChannelCurrentProgramBlock": "0",
      "taskProgramInfo": true,
      "taskChannelProgramInfo": "0",
      "taskMachineTimeData": true,
      "taskToolLife": true,
      "taskToolOffset": true,
      "taskChannelToolOffset": "0",
      "taskToolOffsetToolNumOffsetNum": "1.2;3-5.1;7-9.1-3",
      "taskAlarmHistory": true,
      "taskCycleData": true,
      "taskOEE": true,
      "networkErrorAsOffline": false,
      "parallelProcessing": 1,
      "fileServerType": "Machine Memory",
      "fileServerRootDir": ""
    },
    {
      "id": 2,
      "machineType": "PLC",
      "system": "Standard Protocols",
      "model": "Modbus TCP",
      "name": "2",
      "useDefaultMachineID": true,
      "machineID": "2",
      "useDefaultSlaveID": true,
      "slaveID": 2,
      "customAttributes": "",
      "encoding": "Default",
      "ip": "127.0.0.2",
      "port": 0,
      "isActive": true,
      "numberOfTasks": 2,
      "taskAlarm": false,
      "taskAlarmEnableCustomAlarm": false,
      "taskAlarmEnableAlarmMapping": false,
      "taskMachineStatus": false,
      "taskChannelMachineStatus": "0",
      "taskMachineStatusEnableAdjustedManual": false,
      "taskMachineStatusMaxManualPauseTime": 600,
      "taskMachineStatusEnableStatusConversion": false,
```

```
    "taskMachineStatusEnableCustomStatus": false,
    "taskCount": false,
    "taskCountEnableCustomCount": false,
    "taskPlcData": true,
    "taskPlcDataSettings": "4x,100,2,Int32;0x,100,10,Bit0;",
    "taskAlarmHistory": false,
    "taskOEE": false,
    "networkErrorAsOffline": false,
    "parallelProcessing": 1,
    "modbusSlaveID": 1,
    "modbusByteOrder4": "DCBA",
    "modbusByteOrder8": "HGFEDCBA",
    "modbusPLCAddress": false,
    "modbusReverseString": false
  }
]
```

See Machine Configuration Parameters for the response parameters.

5.9.3.3. create-machine Add Machine Configuration

```
POST /api/config/create-machine
```

Request body example

```
{
  "machineType": "PLC",
  "system": "Standard Protocols",
  "model": "Modbus TCP",
  "name": "150",
  "useDefaultMachineID": false,
  "machineID": "M150",
  "useDefaultSlaveID": true,
  "ip": "127.0.0.150",
  "isActive": true,
  "taskPlcData": true,
  "taskPlcDataSettings": "4x,100,2,Int32;0x,100,10,Bit0;"
```

```
}
```

See Machine Configuration Parameters for the request parameters. Note that the database identifier id does not need to be set in the request body; it is automatically assigned by the gateway and appears in the response body after successful creation.

Response example


```
{
  "id": 21,
  "machineType": "PLC",
  "system": "Standard Protocols",
  "model": "Modbus TCP",
  "name": "150",
  "useDefaultMachineID": false,
  "machineID": "M150",
  "useDefaultSlaveID": true,
  "slaveID": 150,
  "customAttributes": "",
  "encoding": "Default",
  "ip": "127.0.0.150",
  "port": 0,
  "isActive": true,
  "numberOfTasks": 2,
  "taskAlarm": false,
  "taskAlarmEnableCustomAlarm": false,
  "taskAlarmEnableAlarmMapping": false,
  "taskMachineStatus": false,
  "taskChannelMachineStatus": "0",
  "taskMachineStatusEnableAdjustedManual": false,
  "taskMachineStatusMaxManualPauseTime": 600,
  "taskMachineStatusEnableStatusConversion": false,
  "taskMachineStatusEnableCustomStatus": false,
  "taskCount": false,
  "taskCountEnableCustomCount": false,
  "taskPlcData": true,
  "taskPlcDataSettings": "4x,100,2,Int32;0x,100,10,Bit0;",
  "taskAlarmHistory": false,
  "taskOEE": false,
  "networkErrorAsOffline": false,
```

```
  "parallelProcessing": 1,
  "modbusSlaveID": 1,
  "modbusByteOrder4": "DCBA",
  "modbusByteOrder8": "HGFEDCBA",
  "modbusPLCAddress": false,
  "modbusReverseString": false
}
```

See Machine Configuration Parameters for the response parameters.

5.9.3.4. update-machine Modify Machine Configuration

Modifies the configuration of the specified machine and returns the updated machine configuration.

```
POST /api/config/update-machine
```

Request body example

```
{
  "id": 21,
  "name": "newName",
  "useDefaultMachineID": false,
  "machineID": "newMachineID",
  "useDefaultSlaveID": false,
  "slaveID": 192,
  "ip": "127.0.0.192",
  "port": 6000,
  "isActive": false,
  "taskPlcData": false
}
```

See Machine Configuration Parameters for the request parameters. Note: this interface locates the machine by the database identifier id; when id is not provided (or `id <= 0`), it falls back to locating the machine by the machine identifier machineID. At least one of the two must be provided. The machine identifier machineID can be modified through this interface; the machine type machineType cannot — submitting a machineType that differs from the existing type returns an error (statusMsg is “Changing MachineType is forbidden.”).

Response example


```
{
  "id": 21,
  "machineType": "PLC",
  "system": "Standard Protocols",
  "model": "Modbus TCP",
  "name": "newName",
  "useDefaultMachineID": false,
  "machineID": "newMachineID",
  "useDefaultSlaveID": true,
  "slaveID": 192,
  "customAttributes": "",
  "encoding": "Default",
  "ip": "127.0.0.150",
  "port": 6000,
  "isActive": false,
  "numberOfTasks": 0,
  "taskAlarm": false,
  "taskAlarmEnableCustomAlarm": false,
  "taskAlarmEnableAlarmMapping": false,
  "taskMachineStatus": false,
  "taskChannelMachineStatus": "0",
  "taskMachineStatusEnableAdjustedManual": false,
  "taskMachineStatusMaxManualPauseTime": 600,
  "taskMachineStatusEnableStatusConversion": false,
  "taskMachineStatusEnableCustomStatus": false,
  "taskCount": false,
  "taskCountEnableCustomCount": false,
  "taskPlcData": false,
  "taskPlcDataSettings": "4x,100,2,Int32;0x,100,10,Bit0;",
  "taskAlarmHistory": false,
  "taskOEE": false,
  "networkErrorAsOffline": false,
  "parallelProcessing": 1,
  "modbusSlaveID": 1,
  "modbusByteOrder4": "DCBA",
  "modbusByteOrder8": "HGFEDCBA",
  "modbusPLCAddress": false,
  "modbusReverseString": false
```

```
}
```

See Machine Configuration Parameters for the response parameters.

5.9.3.5. delete-machine Delete Machine Configuration

```
GET /api/config/delete-machine?ID=ID&machineID=MACHINEID
```

Request Parameter	Type	Description
id	Int32	Database identifier. See 4.1.4. ID Database Identifier
machineID	String	Target machine identifier. See 4.1.1. machineID Machine Identifier. If both ID and machineID are provided, machineID is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code. 0 indicates success.
errorMsg	String	(Required) Error message

5.9.3.6. batch-delete-machines Batch Delete Machine Configurations

This interface is the batch version of 2.9.3.5. delete-machine Delete Machine Configuration. By including database IDs in the request body, multiple machines can be deleted at once.

```
POST /api/config/batch-delete-machines
```

Request body example application/json

```
{
  "ids": [
    2,
    3,
    4
  ]
}
```

Request Parameter	Type	Description
ids	Int32[]	(Required) Database identifiers. See 4.1.4. ID Database Identifier

Response example

```
{
  "deleted": 3
}
```

The response body contains only deleted, the number of machines successfully deleted. If no machine was deleted successfully, an error code is returned instead.

Response Parameter	Type	Description
deleted	Int32	(Required) Number of machines deleted

5.9.3.7. duplicate-machine Duplicate Machine Configuration

Uses the specified machine as a template and creates count copies of its configuration, assigning IP addresses in ascending order starting from startIP. If the template machine is active, the copies stay active up to the licensed machine count; any beyond the license are created inactive and statusMsg says so.

```
GET /api/config/duplicate-machine?
ID=ID&machineID=MACHINEID&startIP=STARTIP&count=COUNT
```

Request Parameter	Type	Description
id	Int32	Database identifier of the template machine. See 4.1.4. ID Database Identifier
machineID	String	Identifier of the template machine. See 4.1.1. machineID Machine Identifier. At least one of id and machineID must be provided; if both ID and machineID are provided, machineID is ignored.
startIP	String	(Required) Starting IP address. The duplicated machines are assigned IP addresses in order starting from this address.
count	Int32	(Required) Number of copies to create. Must be greater than 0.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code. 0 indicates success.
errorMsg	String	(Required) Error message
statusMsg	String	Status message. Returned as `More activated machines than licensed.` when the copy succeeds but some machines were created inactive because they exceed the license.

5.9.3.8. delete-machines Batch Delete Machine Configurations by IP

Checks count IP addresses in ascending order starting from startIP and deletes the machine configurations on those addresses. IP addresses with no machine on them are skipped.

```
GET /api/config/delete-machines?startIP=STARTIP&count=COUNT
```

Request Parameter	Type	Description
startIP	String	(Required) Starting IP address.
count	Int32	(Required) Number of IP addresses to check, counting from the starting IP address.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code. 0 indicates success.
errorMsg	String	(Required) Error message

5.9.4. Group Configuration API

The group configuration API is used to retrieve, add, modify, or delete group configurations. For details on group configuration, see 3.4. Group Configuration. All configuration parameters for a group are listed in the table below; these parameters are used as request parameters or response parameters in the group configuration API.

Group configuration parameters

Parameter	Type	Description
id	Int32	Group database identifier, see 4.1.4. ID Database Identifier
number	Int32	Group number, valid range 1-16
useDefaultGroupID	Bool	Use the default group ID. When true, the group identifier is generated automatically from the group number (g + group number) and the groupID in the request is ignored; to use a custom groupID, useDefaultGroupID must be set to false.
groupID	String	Group identifier, see 4.1.2. groupID Group Identifier
name	String	Group name; defaults to groupID when left empty
isActive	Bool	Active status, true = active, false = inactive
machines	Object[]	Information about the machines included in the group, see machines Machine List Information.
enableExternalMachines	Bool	Enable external machines

Parameter	Type	Description
externalMachines	String	External machine command. Required when enableExternalMachines is true; leaving it empty returns an error. The format is `uid,machineID[|countMultiplier=1|name=xxx][,machineID...];...`; when the format is invalid, create-group / update-group return an error.
taskCount	Bool	Group production count task, true = enabled, false = disabled.
taskOEE	Bool	Group OEE monitoring task, true = enabled, false = disabled.
taskCumulativeStatusTime	Bool	Group cumulative status time task, true = enabled, false = disabled.

machines Machine list information

machines is the list of machines included in the group, with the parameters shown in the table below. Note that except for id, machineID, and the countMultiplier production coefficient, all other parameters are read-only. Users modify the machines included in a group by adding or removing an id or machineID in the list. In create-group / update-group requests, every machines entry must supply both id (or machineID) and countMultiplier; an entry without countMultiplier is ignored and that machine is not added to the group (the API still returns success). After modifying a machine’s configuration via the machine configuration API, the corresponding machine information in the group’s machine list is updated accordingly.

Parameter	Type	Description
id	Int32	Machine database identifier, see 4.1.4. ID Database Identifier. Add or remove an id in the list to add the specified machine to the group or remove it from the group.
machineType	String	Machine type, see machineType Corresponding Machine Types.

Parameter	Type	Description
system	String	System, same as the system option in the Add Machine dialog under the English UI language
model	String	Model, same as the model option in the Add Machine dialog under the English UI language
name	String	Machine name
machineID	String	Machine identifier, see 4.1.1. machineID Machine Identifier. Add or remove a machineID in the list to add the specified machine to the group or remove it from the group. If both ID and machineID are supplied, machineID is ignored.
slaveID	Int32	Slave identifier, see 4.1.3. slaveID Slave Identifier
ip	String	IP address
port	Int32	Port number, 0 indicates the default port for the given device.
isActive	Bool	Active status, true = active, false = inactive
countMultiplier	Int32	(Required) Production coefficient

5.9.4.1. group Get group configuration

```
GET /api/config/group?ID=ID&groupID=GROUPID
```

Request Parameter	Type	Description
id	Int32	Database identifier, see 4.1.4. ID Database Identifier
groupID	String	Target group identifier, see 4.1.2. groupID Group Identifier. If both ID and groupID are supplied, groupID is ignored.

Response example

```
{
  "id": 1,
  "number": 1,
  "useDefaultGroupID": true,
  "groupID": "g1",
  "name": "g1",
  "isActive": true,
  "machines": [
    {
      "id": 4,
      "machineType": "CNC",
      "system": "Mock",
      "model": "General",
      "name": "演示 1",
      "machineID": "1",
      "slaveID": 1,
      "ip": "127.0.0.1",
      "port": 0,
      "isActive": true,
      "countMultiplier": 1
    },
    {
      "id": 5,
      "machineType": "CNC",
      "system": "Mock",
      "model": "General",
      "name": "演示 2",
      "machineID": "2",
      "slaveID": 2,
      "ip": "127.0.0.2",
      "port": 0,
      "isActive": true,
      "countMultiplier": 0
    }
  ],
  "enableExternalMachines": false,
  "taskCount": true,
  "taskOEE": true,
  "taskCumulativeStatusTime": true
```

```
}
```

For response parameters, see Group Configuration Parameters.

5.9.4.2. groups Get all group configurations

This API has no request parameters.

```
GET /api/config/groups
```

Response example

```
[
  {
    "id": 1,
    "number": 1,
    "useDefaultGroupID": true,
    "groupID": "g1",
    "name": "g1",
    "isActive": true,
    "machines": [
      {
        "id": 4,
        "machineType": "CNC",
        "system": "Mock",
        "model": "General",
        "name": "演示 1",
        "machineID": "1",
        "slaveID": 1,
        "ip": "127.0.0.1",
        "port": 0,
        "isActive": true,
        "countMultiplier": 0
      },
      {
        "id": 5,
        "machineType": "CNC",
        "system": "Mock",
```

```
        "model": "General",
        "name": "演示 2",
        "machineID": "2",
        "slaveID": 2,
        "ip": "127.0.0.2",
        "port": 0,
        "isActive": true,
        "countMultiplier": 0
      }
    ],
    "enableExternalMachines": false,
    "taskCount": true,
    "taskOEE": true,
    "taskCumulativeStatusTime": true
  },
  {
    "id": 3,
    "number": 7,
    "useDefaultGroupID": false,
    "groupID": "group7",
    "name": "g7",
    "isActive": true,
    "machines": [
      {
        "id": 6,
        "machineType": "CNC",
        "system": "Mock",
        "model": "General",
        "name": "演示 3",
        "machineID": "3",
        "slaveID": 3,
        "ip": "127.0.0.3",
        "port": 0,
        "isActive": true,
        "countMultiplier": 23
      },
      {
        "id": 7,
        "machineType": "CNC",
        "system": "Mock",
```

```
        "model": "General",
        "name": "演示 4",
        "machineID": "4",
        "slaveID": 4,
        "ip": "127.0.0.4",
        "port": 0,
        "isActive": true,
        "countMultiplier": 4
      }
    ],
    "enableExternalMachines": false,
    "taskCount": true,
    "taskOEE": true,
    "taskCumulativeStatusTime": true
  }
]
```

For response parameters, see Group Configuration Parameters.

5.9.4.3. create-group Add group configuration

```
POST /api/config/create-group
```

Request body example

```
{
  "number": 7,
  "name": "g7",
  "useDefaultGroupID": false,
  "groupID": "group7",
  "isActive": true,
  "machines": [
    {
      "id": 6,
      "countMultiplier": 23
    },
    {
      "id": 7,
```

```
      "machineID": "4",
      "countMultiplier": 4
    }
  ],
  "enableExternalMachines": false,
  "taskCount": true,
  "taskOEE": true,
  "taskCumulativeStatusTime": true
}
```

For request parameters, see Group Configuration Parameters. Note that the database identifier id does not need to be set in the request body; it is automatically assigned by the gateway and appears in the response body after successful creation. The group number `number` may also be omitted; when it is omitted the gateway automatically assigns the first unused group number in the range 1-16. If all group numbers are already in use, or if number or groupID duplicates an existing group, the API returns an error object (see 5.2. Error Handling).

Response example

```
{
  "id": 3,
  "number": 7,
  "useDefaultGroupID": false,
  "groupID": "group7",
  "name": "g7",
  "isActive": true,
  "machines": [
    {
      "id": 6,
      "machineType": "CNC",
      "system": "Mock",
      "model": "General",
      "name": "演示 3",
      "machineID": "3",
      "slaveID": 3,
      "ip": "127.0.0.3",
      "port": 0,
      "isActive": true,
      "countMultiplier": 23
    },
```

```
    {
      "id": 7,
      "machineType": "CNC",
      "system": "Mock",
      "model": "General",
      "name": "演示 4",
      "machineID": "4",
      "slaveID": 4,
      "ip": "127.0.0.4",
      "port": 0,
      "isActive": true,
      "countMultiplier": 4
    }
  ],
  "enableExternalMachines": false,
  "taskCount": true,
  "taskOEE": true,
  "taskCumulativeStatusTime": true
}
```

For response parameters, see Group Configuration Parameters.

5.9.4.4. update-group Modify group configuration

Modifies the configuration information of the specified group and returns the updated group configuration information.

```
POST /api/config/update-group
```

Request body example

```
{
  "id": 3,
  "number": 6,
  "name": "newName",
  "useDefaultGroupID": false,
  "groupID": "newGroupID",
  "isActive": true,
```

```
  "machines": [
    {
      "id": 6,
      "countMultiplier": 1
    }
  ],
  "taskCount": false,
  "taskOEE": false,
  "taskCumulativeStatusTime": false
}
```

For request parameters, see Group Configuration Parameters. Note: this API uses id as the unique identifier, and the database identifier id must be set in the request body. The group identifier groupID can be modified through this API, but useDefaultGroupID=false must be set in the same request when changing groupID; if an update-group request does not include groupID, the group identifier is reset to its default value.

Response example

```
{
  "id": 3,
  "number": 6,
  "useDefaultGroupID": false,
  "groupID": "newGroupID",
  "name": "newName",
  "isActive": true,
  "machines": [
    {
      "id": 6,
      "machineType": "CNC",
      "system": "Mock",
      "model": "General",
      "name": "演示 3",
      "machineID": "3",
      "slaveID": 3,
      "ip": "127.0.0.3",
      "port": 0,
      "isActive": true,
      "countMultiplier": 1
    }
```

```
  ],
  "enableExternalMachines": false,
  "taskCount": false,
  "taskOEE": false,
  "taskCumulativeStatusTime": false
}
```

For response parameters, see Group Configuration Parameters.

5.9.4.5. delete-group Delete group configuration

```
GET /api/config/delete-group?ID=ID&groupID=GROUPID
```

Request Parameter	Type	Description
ID	Int32	Database identifier, see 4.1.4. ID Database Identifier
groupID	String	Target group identifier, see 4.1.2. groupID Group Identifier. If both ID and groupID are supplied, groupID is ignored.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.9.4.6. batch-delete-groups Batch delete group configurations

This API is the batch version of 2.9.4.5. delete-group Delete Group Configuration; by supplying database IDs in the request body, multiple groups can be deleted at once.

```
POST /api/config/batch-delete-groups
```

Request body example application/json

```
{
  "ids": [
    2,
    3,
    4
  ]
}
```

Request Parameter	Type	Description
ids	Int32[]	(Required) Database identifiers, see 4.1.4. ID Database Identifier

Response example

```
{
  "deleted": 3
}
```

The response body contains only deleted, the number of groups successfully deleted. If ids is empty or missing, or if no group was deleted at all, the API returns an error object (see 5.2. Error Handling) rather than deleted=0.

Response Parameter	Type	Description
deleted	Int32	(Required) Number of groups deleted

5.9.5. Task Configuration Interface

The task configuration interface is used to retrieve or modify task configuration. For details on task configuration, see 3.5. Task Configuration.

5.9.5.1. machine-task-interval-settings Get Machine Task Interval Settings

This interface has no request parameters.

```
GET /api/config/machine-task-interval-settings
```

Response example

```
{
  "alarm": 5000,
  "machineStatus": 5000,
  "count": 5000,
  "currentToolNo": 5000,
  "energyConsumption": 60000,
  "feedAndSpindle": 5000,
  "laserPower": 5000,
  "load": 5000,
  "logHistory": 5000,
  "overload": 100,
  "plcData": 5000,
  "position": 5000,
  "currentProgramBlock": 5000,
  "programInfo": 5000,
  "machineTimeData": 5000,
  "toolLife": 5000,
  "toolOffset": 5000,
  "alarmHistory": 5000,
  "cycleData": 5000,
  "oee": 5000
}
```

The response parameters are shown in the table below:

Machine Task Interval Settings Parameters

Parameter	Type	Description
alarm	Int32	Alarm information interval (milliseconds)
machineStatus	Int32	Machine status interval (milliseconds)
count	Int32	Machining count interval (milliseconds)
currentToolNo	Int32	Current tool number interval (milliseconds)
energyConsumption	Int32	Energy consumption interval (milliseconds)
feedAndSpindle	Int32	Feed rate and spindle speed data interval (milliseconds)
laserPower	Int32	Laser power interval (milliseconds)
load	Int32	Load data interval (milliseconds)
logHistory	Int32	Log history interval (milliseconds)
overload	Int32	Overload monitoring interval (milliseconds)
plcData	Int32	PLC data interval (milliseconds)
position	Int32	Position data interval (milliseconds)
currentProgramBlock	Int32	Current program block interval (milliseconds)
programInfo	Int32	Machining program information interval (milliseconds)
machineTimeData	Int32	Machine time data interval (milliseconds)
toolLife	Int32	Tool life interval (milliseconds)
toolOffset	Int32	Tool offset interval (milliseconds)
alarmHistory	Int32	Alarm history interval (milliseconds)
cycleData	Int32	Cycle time data interval (milliseconds)

Parameter	Type	Description
oee	Int32	OEE monitoring interval (milliseconds)

5.9.5.2. update-machine-task-interval-settings Modify Machine Task Interval Settings

Modifies the machine task interval settings and returns the updated settings.

```
POST /api/config/update-machine-task-interval-settings
```

Request body example

```
{
  "alarm": 6000,
  "machineStatus": 8000
}
```

For request parameters, see Machine Task Interval Settings Parameters. Parameters not included in the request body retain their original values.

Response example

```
{
  "alarm": 6000,
  "machineStatus": 8000,
  "count": 5000,
  "currentToolNo": 5000,
  "energyConsumption": 60000,
  "feedAndSpindle": 5000,
  "laserPower": 5000,
  "load": 5000,
  "logHistory": 5000,
  "overload": 100,
  "plcData": 5000,
  "position": 5000,
  "currentProgramBlock": 5000,
  "programInfo": 5000,
```

```
  "machineTimeData": 5000,
  "toolLife": 5000,
  "toolOffset": 5000,
  "alarmHistory": 5000,
  "cycleData": 5000,
  "oee": 5000
}
```

For response parameters, see Machine Task Interval Settings Parameters.

5.9.5.3. group-task-interval-settings Get Group Task Interval Settings

This interface has no request parameters.

```
GET /api/config/group-task-interval-settings
```

Response example

```
{
  "count": 5000,
  "oee": 5000,
  "cumulativeStatusTime": 5000
}
```

The response parameters are shown in the table below:

Group Task Interval Settings Parameters

Parameter	Type	Description
count	Int32	Machining count interval (milliseconds)
oee	Int32	OEE monitoring interval (milliseconds)
cumulativeStatusTime	Int32	Cumulative status time (milliseconds)

5.9.5.4. update-group-task-interval-settings Modify Group Task Interval Settings

Modifies the group task interval settings and returns the updated settings.

```
POST /api/config/update-group-task-interval-settings
```

Request body example

```
{
  "count": 3000,
  "oee": 2000,
  "cumulativeStatusTime": 1000
}
```

For request parameters, see Group Task Interval Settings Parameters.

Response example

```
{
  "count": 3000,
  "oee": 2000,
  "cumulativeStatusTime": 1000
}
```

For response parameters, see Group Task Interval Settings Parameters.

5.9.5.5. oee-monitoring-settings Get OEE Monitoring Settings

This interface has no request parameters.

```
GET /api/config/oee-monitoring-settings
```

Response example

```
{
  "monitorMode": "Scheduled Reset",
```

```
  "windowSize": 3600,
  "resetTime": "00:00",
  "availabilityMode": "Autorun Time / Total Time",
  "breakTimeInterval": ""
}
```

The response parameters are shown in the table below:

OEE Monitoring Settings Parameters

Parameter	Type	Description
monitorMode	String	Monitoring mode; see monitorMode Monitoring Mode for details.
windowSize	Int32	Window duration (seconds); effective in real-time window mode.
resetTime	String	Reset time (hour:minute); effective in scheduled reset mode.
availabilityMode	String	Availability rate calculation method; see availabilityMode Availability Rate Calculation Method for details.
breakTimeInterval	String	Break time interval (hour:minute-hour:minute)

monitorMode Monitoring Mode

Option	Description
Scheduled Reset	Scheduled reset
Time Window	Real-time window

availabilityMode Availability Rate Calculation Method

Option	Description
Autorun Time / Total Time	Autorun time / total time
Autorun Time / (Total Time - Off Time)	Autorun time / (total time - off time)

Option	Description
Autorun Time / (Total Time - Off Time - Manual Time)	Autorun time / (total time - off time - manual time)
Autorun Time / (Total Time - Break Time)	Autorun time / (total time - break time)

5.9.5.6. update-oee-monitoring-settings Modify OEE Monitoring Settings

Modifies the OEE monitoring settings and returns the updated settings.

```
POST /api/config/update-oee-monitoring-settings
```

Request body example

```
{
  "monitorMode": "Time Window",
  "windowSize": 6000,
  "resetTime": "00:00;22:00;01:00;2:00",
  "availabilityMode": "Autorun Time / (Total Time - Break Time)",
  "breakTimeInterval": "08:00-09:00;11:30-12:30"
}
```

For request parameters, see OEE Monitoring Settings Parameters.

Response example

```
{
  "monitorMode": "Time Window",
  "windowSize": 6000,
  "resetTime": "00:00;22:00;01:00;2:00",
  "availabilityMode": "Autorun Time / (Total Time - Break Time)",
  "breakTimeInterval": "08:00-09:00;11:30-12:30"
}
```

For response parameters, see OEE Monitoring Settings Parameters.

5.9.5.7. tool-life-monitoring-settings Get Tool Life Monitoring Settings

This interface has no request parameters.

```
GET /api/config/tool-life-monitoring-settings
```

Response example

```
{
  "enableToolLifeDetails": false
}
```

The response parameters are shown in the table below:

Tool Life Monitoring Settings Parameters

Parameter	Type	Description
enableToolLifeDetails	Bool	Tool life details; true = enabled, false = disabled.

5.9.5.8. update-tool-life-monitoring-settings Modify Tool Life Monitoring Settings

Modifies the tool life monitoring settings and returns the updated settings.

```
POST /api/config/update-tool-life-monitoring-settings
```

Request body example

```
{
  "enableToolLifeDetails": true
}
```

For request parameters, see Tool Life Monitoring Settings Parameters.

Response example

```
{
```

```
  "enableToolLifeDetails": true
}
```

For response parameters, see Tool Life Monitoring Settings Parameters.

5.9.5.9. count-monitoring-settings Get Machining Count Monitoring Settings

This interface has no request parameters.

```
GET /api/config/count-monitoring-settings
```

Response example

```
{
  "monitorMode": "Time Window",
  "windowSize": 3600,
  "resetTime": "00:00"
}
```

The response parameters are shown in the table below:

Machining Count Monitoring Settings Parameters

Parameter	Type	Description
monitorMode	String	Monitoring mode; see monitorMode Monitoring Mode for details.
windowSize	Int32	Window duration (seconds); effective in real-time window mode.
resetTime	String	Reset time (hour:minute); effective in scheduled reset mode. Separate multiple times with `;`.

5.9.5.10. update-count-monitoring-settings Modify Machining Count Monitoring Settings

Modifies the machining count monitoring settings and returns the updated settings.

```
POST /api/config/update-count-monitoring-settings
```

Request body example

```
{
  "monitorMode": "Scheduled Reset",
  "windowSize": 36000,
  "resetTime": "23:00;3:24;15:28;12:22"
}
```

For request parameters, see Machining Count Monitoring Settings Parameters.

Response example

```
{
  "monitorMode": "Scheduled Reset",
  "windowSize": 36000,
  "resetTime": "23:00;3:24;15:28;12:22"
}
```

For response parameters, see Machining Count Monitoring Settings Parameters.

5.9.5.11. overload-monitoring-settings Get Overload Monitoring Settings

This interface has no request parameters.

```
GET /api/config/overload-monitoring-settings
```

Response example

```
{
  "spindleLoadUpperLimit": 100.0
```

```
}
```

The response parameters are shown in the table below:

Overload Monitoring Settings Parameters

Parameter	Type	Description
spindleLoadUpperLimit	Float	Spindle load upper limit (%)

5.9.5.12. update-overload-monitoring-settings Modify Overload Monitoring Settings

Modifies the overload monitoring settings and returns the updated settings.

```
POST /api/config/update-overload-monitoring-settings
```

Request body example

```
{
  "spindleLoadUpperLimit": 80.0
}
```

For request parameters, see Overload Monitoring Settings Parameters.

Response example

```
{
  "spindleLoadUpperLimit": 80.0
}
```

For response parameters, see Overload Monitoring Settings Parameters.

5.9.5.13. alarm-monitoring-settings Get Alarm Monitoring Settings

This interface has no request parameters.

```
GET /api/config/alarm-monitoring-settings
```

Response example

```
{
  "mininumAlarmLevel": "Warning",
  "alarmMappingFileNames": "a.csv;b.csv;"
}
```

The response parameters are shown in the table below:

Alarm Monitoring Settings Parameters

Parameter	Type	Description
mininumAlarmLevel	String	Minimum alarm level; see mininumAlarmLevel Minimum Alarm Level for details.
infoLevel	String	Info-level keywords; not returned means not set
errorLevel	String	Error-level keywords; not returned means not set
alarmMappingFileNames	String	List of uploaded alarm mapping file names, separated by ; (terminated with ; when not empty). Read-only; ignored if supplied in a request. For uploading, downloading and deleting these files, see 2.9.5.19. upload-alarm-mapping-file Upload Alarm Mapping File, 2.9.5.20. download-alarm-mapping-file Download Alarm Mapping File and 2.9.5.21. delete-alarm-mapping-file Delete Alarm Mapping File.

mininumAlarmLevel Minimum Alarm Level

Option	Description
Information	Information
Warning	Warning
Error	Error
None	None

5.9.5.14. update-alarm-monitoring-settings Modify Alarm Monitoring Settings

Modifies the alarm monitoring settings and returns the updated settings.

```
POST /api/config/update-alarm-monitoring-settings
```

Request body example

```
{
  "mininumAlarmLevel": "Error",
  "infoLevel": "消息;提示;Inf",
  "errorLevel": "错误;錯誤;Err"
}
```

For request parameters, see Alarm Monitoring Settings Parameters.

> **Note**
> Unlike the other modification interfaces on this page, `infoLevel` and `errorLevel` are cleared when they are not supplied in the request body, so they must be submitted in full every time; `mininumAlarmLevel` retains its original value when not supplied.

Response example

```
{
  "mininumAlarmLevel": "Error",
  "infoLevel": "消息;提示;Inf",
  "errorLevel": "错误;錯誤;Err",
  "alarmMappingFileNames": "a.csv;b.csv;"
```

```
}
```

For response parameters, see Alarm Monitoring Settings Parameters.

5.9.5.15. machine-status-monitoring-settings Get Machine Status Monitoring Settings

This interface has no request parameters.

```
GET /api/config/machine-status-monitoring-settings
```

Response example

```
{
  "enableStatusDetails": false,
  "enableAdjustedManualStatus": false,
  "maxManualPauseTime": 600,
  "enableStatusConversion": false
}
```

The response parameters are shown in the table below:

Machine Status Monitoring Settings Parameters

Parameter	Type	Description
enableStatusDetails	Bool	Status details; true = enabled, false = disabled.
enableAdjustedManualStatus	Bool	Manual status adjustment; true = enabled, false = disabled.
maxManualPauseTime	Int32	Maximum manual pause time (seconds)
enableStatusConversion	Bool	Status conversion; true = enabled, false = disabled.
statusConversionSettings	String	Status conversion; not returned means not set.

5.9.5.16. update-machine-status-monitoring-settings Modify Machine Status Monitoring Settings

Modifies the machine status monitoring settings and returns the updated settings.

```
POST /api/config/update-machine-status-monitoring-settings
```

Request body example

```
{
  "enableStatusDetails": true,
  "enableAdjustedManualStatus": true,
  "maxManualPauseTime": 300,
  "enableStatusConversion": true,
  "statusConversionSettings": "CNCStatus_cncStatus = \"MANUAL_MDI_RUN\" |
AUTO_RUN"
}
```

For request parameters, see Machine Status Monitoring Settings Parameters.

Response example

```
{
  "enableStatusDetails": true,
  "enableAdjustedManualStatus": true,
  "maxManualPauseTime": 300,
  "enableStatusConversion": true,
  "statusConversionSettings": "CNCStatus_cncStatus = \"MANUAL_MDI_RUN\" |
AUTO_RUN"
}
```

For response parameters, see Machine Status Monitoring Settings Parameters.

5.9.5.17. task-manager-settings Get Task Manager Settings

This interface has no request parameters.

```
GET /api/config/task-manager-settings
```

Response example

```
{
  "maxTaskManagers": 255
}
```

The response parameters are shown in the table below:

Task Manager Settings Parameters

Parameter	Type	Description
maxTaskManagers	Int32	Maximum number of task managers; default is 255

5.9.5.18. update-task-manager-settings Modify Task Manager Settings

Modifies the task manager settings and returns the updated settings.

```
POST /api/config/update-task-manager-settings
```

Request body example

```
{
  "maxTaskManagers": 128
}
```

For request parameters, see Task Manager Settings Parameters.

Response example

```
{
  "maxTaskManagers": 128
}
```

For response parameters, see Task Manager Settings Parameters.

5.9.5.19. upload-alarm-mapping-file Upload Alarm Mapping File

Uploads an alarm mapping file. The file name is supplied as a query parameter and the file content is sent as binary in the request body (Content-Type application/octet-stream).

```
POST /api/config/upload-alarm-mapping-file?fileName=FILENAME
```

Request Parameter	Type	Description
fileName	String	(Required) Alarm mapping file name. Returns GENERAL_API_INVALID_REQUEST (Invalid fileName.) when empty.

When a file with the same name already exists, the gateway does not overwrite it and returns GENERAL_INVALID_PATH (file already exists.). To replace a file, first call 2.9.5.21. delete-alarm-mapping-file Delete Alarm Mapping File.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.9.5.20. download-alarm-mapping-file Download Alarm Mapping File

Downloads an uploaded alarm mapping file; the response body is a file stream.

```
GET /api/config/download-alarm-mapping-file?fileName=FILENAME
```

Request Parameter	Type	Description
fileName	String	(Required) Alarm mapping file name, which can be obtained from alarmMappingFileNames in Alarm Monitoring Settings Parameters. Returns GENERAL_API_INVALID_REQUEST (Invalid fileName.) when empty.

When the file does not exist, GENERAL_INVALID_PATH (file not found.) is returned.

5.9.5.21. delete-alarm-mapping-file Delete Alarm Mapping File

Deletes an uploaded alarm mapping file.

```
GET /api/config/delete-alarm-mapping-file?fileName=FILENAME
```

Request Parameter	Type	Description
fileName	String	(Required) Alarm mapping file name, which can be obtained from alarmMappingFileNames in Alarm Monitoring Settings Parameters. Returns GENERAL_API_INVALID_REQUEST (Invalid fileName.) when empty.

When the file does not exist, GENERAL_INVALID_PATH (file not found.) is returned.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.9.6. Communication Configuration Interface

The communication configuration interface is used to get or modify the task configuration. For details on communication configuration, see 3.6. Communication Configuration.

5.9.6.1. cloud-settings Get Cloud Platform Settings

This interface takes no request parameters.

```
GET /api/config/cloud-settings
```

Response example

```
{
  "enable": false,
  "serverAddress": "severAddress",
  "token": "tokenABCDEFG",
  "enableBroadcasting": false,
  "enableGatewayLinking": false,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

The response parameters are listed in the table below:

Cloud Platform Settings Parameters

Parameter	Type	Description
enable	Bool	Enable the cloud platform. true = on, false = off.
serverAddress	String	Server address. Not returned if not set.
token	String	Gateway token. Not returned if not set.
enableBroadcasting	Bool	Enable broadcasting. true = on, false = off. When enabled, data is broadcast via the cloud platform.

Parameter	Type	Description
enableGatewayLinking	Bool	Enable gateway linking. true = on, false = off. When enabled, data broadcast by the linked gateways specified on the cloud platform is received.
gatewayLinks	String	Linked gateways. Not returned if not set.

5.9.6.2. update-cloud-settings Update Cloud Platform Settings

Updates the cloud platform settings and returns the updated settings.

```
POST /api/config/update-cloud-settings
```

Request body example

```
{
  "enable": true,
  "serverAddress": "severAddress",
  "token": "tokenABCDEFG",
  "enableBroadcasting": true,
  "enableGatewayLinking": true,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

For request parameters, see Cloud Platform Settings Parameters.

Response example

```
{
  "enable": true,
  "serverAddress": "severAddress",
  "token": "tokenABCDEFG",
  "enableBroadcasting": true,
  "enableGatewayLinking": true,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

For response parameters, see Cloud Platform Settings Parameters.

5.9.6.3. modbus-settings Get MODBUS Settings

This interface takes no request parameters.

```
GET /api/config/modbus-settings
```

Response example

```
{
  "enable": false,
  "byte4Order": "DCBA",
  "byte8Order": "HGFEDCBA",
  "encoding": "GBK",
  "reverseString": false
}
```

The response parameters are listed in the table below:

MODBUS Settings Parameters

Parameter	Type	Description
enable	Bool	Enable MODBUS. true = on, false = off.
byte4Order	String	32-bit format. Supported formats include: ABCD, BADC, CDAB, DCBA, etc.
byte8Order	String	64-bit format. Supported formats include: ABCDEFGH, BADCFEHG, GHEFCDAB, HGFEDCBA, etc.
encoding	String	Encoding, e.g. ASCII, UTF-8, GBK, etc.
reverseString	Bool	Reverse string. true = on, false = off.

5.9.6.4. update-modbus-settings Update MODBUS Settings

Updates the MODBUS settings and returns the updated settings.

```
POST /api/config/update-modbus-settings
```

Request body example

```
{
  "enable": true,
  "byte4Order": "ABCD",
  "byte8Order": "ABCDEFGH",
  "encoding": "UTF-8",
  "reverseString": true
}
```

For request parameters, see MODBUS Settings Parameters.

Response example

```
{
  "enable": true,
  "byte4Order": "ABCD",
  "byte8Order": "ABCDEFGH",
  "encoding": "UTF-8",
  "reverseString": true
}
```

For response parameters, see MODBUS Settings Parameters.

5.9.6.5. mqtt-settings Get MQTT Settings

This interface takes no request parameters.

```
GET /api/config/mqtt-settings
```

Response example

```
{
  "enable": false,
```

```
    "mode": "Default",
    "brokerAddress": "brokerAddress",
    "port": 1111,
    "clientID": "client",
    "username": "username",
    "password": "password",
    "dataReportTopic": "topic1",
    "rpcRequestTopic": "rpcReq",
    "rpcResponseTopic": "rpcRes",
    "encoding": "GBK",
    "allowAnonymous": false,
    "arrayToString": false,
    "publishOnValueChange": false,
    "publishAllOnValueChange": false,
    "timeoutReportingInterval": 0,
    "additionalInfo": {
      "gatewayProductKey": "",
      "gatewayDeviceKey": "",
      "gatewayDeviceID": "",
      "gatewayDeviceSecret": "",
      "gatewaySignMethod": 0,
      "gatewayOperator": 0,
      "machineProductKey": "",
      "groupProductKey": ""
    }
  }
```

The response parameters are listed in the table below:

MQTT Settings Parameters

Parameter	Type	Description
enable	Bool	Enable MQTT. true = on, false = off.
mode	String	Mode. See MQTT mode for details.
brokerAddress	String	Server address
port	Int32	Server port
clientID	String	Client ID

Parameter	Type	Description
username	String	Username
password	String	Password
dataReportTopic	String	Data report topic
rpcRequestTopic	String	RPC request topic
rpcResponseTopic	String	RPC response topic
encoding	String	Encoding, e.g. ASCII, UTF-8, GBK, etc.
allowAnonymous	Bool	Allow anonymous login. true = on, false = off.
arrayToString	Bool	Convert array to string. true = on, false = off.
publishOnValueChange	Bool	Write on value change. true = on, false = off. Default is false.
publishAllOnValueChange	Bool	Upload all content on change. Effective only when publish-on-value-change is enabled. Default is false, meaning only the changed portion is uploaded when a value changes.
timeoutReportingInterval	Int32	Timeout reporting interval. Effective only when publish-on-value-change is enabled. If the time since the last upload exceeds this interval, data is uploaded once regardless of whether the value changed. Unit: seconds. Default is 0, meaning timeout reporting is disabled.
additionalInfo	Object	Gewu-specific settings. GET and update responses always include this object; other modes can ignore it. Updates apply these fields only in `GeWu` mode.

MQTT mode

Option
Default

Option
TB2
Brm
IoTDA
WisIoT
MKT
GeWu
TB

GeWu additionalInfo parameters

When `GeWu` mode is enabled, all fields below are required; `gatewaySignMethod` and `gatewayOperator` may be 0 but must stay within their documented ranges. The gateway generates the MQTT Client ID, username, and signed password, so ordinary `clientID`, `username`, `password`, and `dataReportTopic` need not be submitted. None of the three Product Keys may contain `/`, `+`, or `#`.

Parameter	Type	Description
gatewayProductKey	String	Required gateway product key; cannot contain `/`, `+`, or `#`.
gatewayDeviceKey	String	Gateway device key, 4-32 ASCII letters, digits, or underscores.
gatewayDeviceID	String	Gateway device ID, 10-32 ASCII letters, digits, or hyphens.
gatewayDeviceSecret	String	Required gateway device secret.
gatewaySignMethod	Int32	Signature method: 0 = HMAC-SHA256, 1 = HMAC-SM3.
gatewayOperator	Int32	Carrier: 0 = none, 1 = China Unicom, 2 = China Mobile, 3 = China Telecom, 4 = China Broadnet.
machineProductKey	String	Required machine product key; cannot contain `/`, `+`, or `#`.
groupProductKey	String	Required group product key; cannot contain `/`, `+`, or `#`.

5.9.6.6. update-mqtt-settings Update MQTT Settings

Updates the MQTT settings and returns the updated settings.

```
POST /api/config/update-mqtt-settings
```

Request body example

```
{
  "enable": true,
  "mode": "GeWu",
  "brokerAddress": "mqtt.example.com",
  "port": 1883,
  "publishOnValueChange": true,
  "publishAllOnValueChange": false,
  "timeoutReportingInterval": 0,
  "additionalInfo": {
    "gatewayProductKey": "gatewayProduct",
    "gatewayDeviceKey": "gateway_0001",
    "gatewayDeviceID": "gateway-device-0001",
    "gatewayDeviceSecret": "secret",
    "gatewaySignMethod": 0,
    "gatewayOperator": 1,
    "machineProductKey": "machineProduct",
    "groupProductKey": "groupProduct"
  }
}
```

For request parameters, see MQTT Settings Parameters. In `GeWu` mode, `additionalInfo` may contain only the fields being changed; omitted fields retain their current values. Enabling MQTT requires the merged Gewu settings to pass the checks above and a nonempty broker address. To disable MQTT, submit only `{"enable": false}`; existing Gewu settings are retained. Other modes need not submit `additionalInfo`.

Response example

```
{
  "enable": true,
  "mode": "GeWu",
```

```
  "brokerAddress": "mqtt.example.com",
  "port": 1883,
  "publishOnValueChange": true,
  "publishAllOnValueChange": false,
  "timeoutReportingInterval": 0,
  "additionalInfo": {
    "gatewayProductKey": "gatewayProduct",
    "gatewayDeviceKey": "gateway_0001",
    "gatewayDeviceID": "gateway-device-0001",
    "gatewayDeviceSecret": "secret",
    "gatewaySignMethod": 0,
    "gatewayOperator": 1,
    "machineProductKey": "machineProduct",
    "groupProductKey": "groupProduct"
  }
}
```

For response parameters, see MQTT Settings Parameters.

5.9.6.7. database-settings Get Database Settings

This interface takes no request parameters.

```
GET /api/config/database-settings
```

Response example

For an InfluxDB v2.x type database, the response is as follows:

```
{
  "enable": true,
  "type": "InfluxDB v2.x",
  "serverAddress": "192.168.100.101",
  "port": "12345",
  "username": "username",
  "password": "password",
  "bucket": "Bucket",
  "tablePrefix": "Prefix",
  "organization": "Organization",
```

```
  "dataRetentionPeriod": 0,
  "writeOnValueChange": true,
  "timeoutReportingInterval": 0
}
```

For a non-InfluxDB v2.x type database, the response is as follows:

```
{
  "enable": true,
  "type": "MySQL",
  "serverAddress": "192.168.100.101",
  "port": "12345",
  "username": "username",
  "password": "password",
  "database": "Database",
  "tablePrefix": "Prefix",
  "storageMode": "User Defined",
  "dataRetentionPeriod": 0,
  "useLocalTime": false,
  "enablePrimaryKey": false,
  "writeOnValueChange": false,
  "timeoutReportingInterval": 0,
  "loggingModeTypes": [
    "AlarmHistory",
    "AxialOverload",
    "CNCStatus",
    "Count",
    "CurrentToolNumber",
    "Cycle",
    "EnergyConsum",
    "Feed",
    "FeedAndSpindle",
    "GroupCount",
    "GroupCumulativeTime",
    "LaserPower",
    "Load",
    "LogHistory",
    "Offset",
    "PLC",
```

```
      "Position",
      "ProgramBlock",
      "ProgramInfo",
      "SpindleOverload",
      "TimeData",
      "ToolLife"
    ],
    "realTimeModeTypes": [
      "AlarmLog",
      "GroupOEE",
      "OEE"
    ],
    "noActionModeTypes": []
  }
```

The response parameters are listed in the table below:

Database Settings Parameters

Parameter	Type	Description
enable	Bool	Enable the database. true = on, false = off.
type	String	Type. See Database Types for details.
serverAddress	String	Server address
port	String	Server port
username	String	Username
password	String	Password
bucket	String	Database bucket. InfluxDB v2.x type only.
organization	String	Organization name. InfluxDB v2.x type only.
database	String	Database. Non-InfluxDB v2.x types only.
storageMode	String	Storage mode. Non-InfluxDB v2.x types only. See Database Storage Modes for details.

Parameter	Type	Description
dataRetentionPeriod	Int32	Data retention period (hours). Not returned when `storageMode` is `Real Time`; returned in all other cases (including InfluxDB v2.x).
loggingModeTypes	String[]	Effective when the storage mode is User Defined. See the `<type>` data classes in 4.2. Data Description for details.
noActionModeTypes	String[]	Effective when the storage mode is User Defined. See the `<type>` data classes in 4.2. Data Description for details.
realTimeModeTypes	String[]	Effective when the storage mode is User Defined. See the `<type>` data classes in 4.2. Data Description for details.
useLocalTime	Bool	Use local time. Non-InfluxDB v2.x types only.
enablePrimaryKey	Bool	Enable primary key. Non-InfluxDB v2.x types only.
tablePrefix	String	Table prefix
writeOnValueChange	Bool	Write on value change. true = on, false = off.
timeoutReportingInterval	Int32	Timeout reporting interval. Effective when write-on-value-change is enabled. If the time since the last write exceeds this interval, data is uploaded once regardless of whether the value changed. Unit: seconds. Default is 0, meaning timeout writing is disabled.

Database Types

Option
InfluxDB v2.x
MySQL

Option
SQL Server
PostgreSQL

Database Storage Modes

Option	Description
Logging	Logging
Real Time	Real time
User Defined	User defined

5.9.6.8. update-database-settings Update Database Settings

Updates the database settings and returns the updated settings.

```
POST /api/config/update-database-settings
```

Request body example

```
{
  "enable": true,
  "type": "SQL Server",
  "serverAddress": "192.168.100.201",
  "port": "23456",
  "username": "username2",
  "password": "password2",
  "database": "Database2",
  "tablePrefix": "Prefix2",
  "storageMode": "Real Time",
  "useLocalTime": true,
  "enablePrimaryKey": true,
  "writeOnValueChange": true,
  "timeoutReportingInterval": 60
}
```

For request parameters, see Database Settings Parameters.

Response example


```
{
  "enable": true,
  "type": "SQL Server",
  "serverAddress": "192.168.100.201",
  "port": "23456",
  "username": "username2",
  "password": "password2",
  "database": "Database2",
  "tablePrefix": "Prefix2",
  "storageMode": "Real Time",
  "useLocalTime": true,
  "enablePrimaryKey": true,
  "writeOnValueChange": true,
  "timeoutReportingInterval": 60,
  "loggingModeTypes": [
    "AlarmHistory",
    "AxialOverload",
    "CNCStatus",
    "Count",
    "CurrentToolNumber",
    "Cycle",
    "EnergyConsum",
    "Feed",
    "FeedAndSpindle",
    "GroupCount",
    "GroupCumulativeTime",
    "LaserPower",
    "Load",
    "LogHistory",
    "Offset",
    "PLC",
    "Position",
    "ProgramBlock",
    "ProgramInfo",
    "SpindleOverload",
    "TimeData",
    "ToolLife"
  ],
```

```
  "realTimeModeTypes": [
    "AlarmLog",
    "GroupOEE",
    "OEE"
  ],
  "noActionModeTypes": []
}
```

For response parameters, see Database Settings Parameters.

5.9.6.9. gateway-file-server-settings Get Gateway File Server Settings

This interface takes no request parameters.

```
GET /api/config/gateway-file-server-settings
```

Response example

```
{
  "enable": false,
  "enableFtp": false,
  "enableSmb": false,
  "username": "dncuser",
  "password": "dncuser"
}
```

The response parameters are listed in the table below:

Gateway File Server Settings Parameters

Parameter	Type	Description
enable	Bool	Enable the gateway file server. true = on, false = off.
enableFtp	Bool	Enable FTP. true = on, false = off.
enableSmb	Bool	Enable shared folder. true = on, false = off.

Parameter	Type	Description
username	String	Username. Read-only, cannot be modified.
password	String	Password

5.9.6.10. update-gateway-file-server-settings Update Gateway File Server Settings

Updates the gateway file server settings and returns the updated settings.

```
POST /api/config/update-gateway-file-server-settings
```

Request body example

```
{
  "enable": true,
  "enableFtp": true,
  "enableSmb": true,
  "username": "dncuser",
  "password": "password"
}
```

For request parameters, see Gateway File Server Settings Parameters.

Response example

```
{
  "enable": true,
  "enableFtp": true,
  "enableSmb": true,
  "username": "dncuser",
  "password": "password"
}
```

For response parameters, see Gateway File Server Settings Parameters.

5.9.6.11. http-settings Get HTTP Settings

This interface takes no request parameters.

```
GET /api/config/http-settings
```

Response example

```
{
  "enable": true,
  "enableIpWhiteList": true,
  "ipWhiteList": "192.168.100.200;192.168.100.201;"
}
```

The response parameters are listed in the table below:

HTTP Settings Parameters

Parameter	Type	Description
enable	Bool	Enable HTTP. true = on, false = off.
enableIpWhiteList	Bool	Enable IP whitelist. true = on, false = off.
ipWhiteList	String	Whitelist entries (separated by ;). Not returned if not set.

5.9.6.12. update-http-settings Update HTTP Settings

Updates the HTTP settings and returns the updated settings.

```
POST /api/config/update-http-settings
```

Request body example

```
{
  "enable": true,
  "enableIpWhiteList": true,
  "ipWhiteList": "192.168.100.200;192.168.100.201;"
```

```
}
```

For request parameters, see HTTP Settings Parameters.

Response example

```
{
  "enable": true,
  "enableIpWhiteList": true,
  "ipWhiteList": "192.168.100.200;192.168.100.201;"
}
```

For response parameters, see HTTP Settings Parameters.

5.9.6.13. hub-settings Get Hub Settings

This interface takes no request parameters.

```
GET /api/config/hub-settings
```

Response example

```
{
  "enable": false,
  "server": "serverAddress",
  "accessToken": "accessToken",
  "enableBroadcasting": false,
  "enableGatewayLinking": false,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

The response parameters are listed in the table below:

Hub Settings Parameters

Parameter	Type	Description
enable	Bool	Enable Hub. true = on, false = off.

Parameter	Type	Description
server	String	Server address. Not returned if not set.
accessToken	String	Gateway token. Not returned if not set.
enableBroadcasting	Bool	Enable broadcasting. true = on, false = off. When enabled, data is broadcast via the Hub.
enableGatewayLinking	Bool	Enable gateway linking. true = on, false = off. When enabled, data broadcast by the linked gateways specified under the Hub is received.
gatewayLinks	String	Linked gateways. Not returned if not set.

5.9.6.14. update-hub-settings Update Hub Settings

Updates the Hub settings and returns the updated settings.

```
POST /api/config/update-hub-settings
```

Request body example

```
{
  "enable": true,
  "server": "serverAddress",
  "accessToken": "accessToken",
  "enableBroadcasting": true,
  "enableGatewayLinking": false,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

For request parameters, see Hub Settings Parameters.

Response example

```
{
```

```
  "enable": true,
  "server": "serverAddress",
  "accessToken": "accessToken",
  "enableBroadcasting": true,
  "enableGatewayLinking": false,
  "gatewayLinks": "uid1,token1;uid2,token2;"
}
```

For response parameters, see Hub Settings Parameters.

5.9.6.15. remote-access Get Remote Access Settings

This interface takes no request parameters.

```
GET /api/config/remote-access
```

Response example

```
{
  "host": "serverUrl",
  "hub": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
  "password": "password",
  "bridgeNetwork": "LAN1;",
  "state": "Established",
  "expiration": "2999-12-31T23:59:59"
}
```

The response parameters are listed in the table below:

Remote Access Settings Parameters

Parameter	Type	Description
host	String	Remote server address.
hub	String	Site (read-only), defaults to the gateway UID.
password	String	Password.

Parameter	Type	Description
bridgeNetwork	String	Bridge network, a semicolon-separated list: `""` (no bridging), `"LAN1;"`, `"LAN2;"` or `"LAN1;LAN2;"`.
state	String	Connection state (read-only): Connecting, Negotiation, Retry (the above three are all "connecting" states), Auth (authenticating), Established (connected), Idle (not connected)
expiration	String	Expiration date (read-only), returned when state is not Idle (not connected).

5.9.6.16. update-remote-access Update Remote Access Settings

Updates the remote access settings and returns the updated settings.

```
POST /api/config/update-remote-access
```

Request body example

```
{
  "host": "serverUrl2",
  "password": "password2",
  "bridgeNetwork": "LAN1;"
}
```

For request parameters, see Remote Access Settings Parameters.

Response example

```
{
  "host": "serverUrl2",
  "hub": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
  "password": "password2",
  "bridgeNetwork": "LAN1;",
```

```
  "state": "Established",
  "expiration": "2999-12-31T23:59:59"
}
```

For response parameters, see Remote Access Settings Parameters.

5.10. Gateway Function Interfaces

The gateway function interfaces are used to command the gateway to perform specific functions, including the Gateway service function interfaces and the Core service function interfaces.

5.10.1. Core Service Function Interfaces

Base address **/api/core**.

5.10.1.1. info - Get Core Service Information

This interface has no request parameters.

```
GET /api/core/info
```

Response example

```
{
  "version": "1.19.4.18"
}
```

Response Parameter	Type	Description
version	String	(Required) Version number

5.10.1.2. license-info - Get License Information

Retrieves license details, including the expiration date, license count, license type, license status, and more. This interface has no request parameters.

```
GET /api/core/license-info
```

Response example

```
{
```

```
  "isValid": true,
  "license": {
    "company": "Bivrost",
    "product": "IoT Gateway",
    "machineCount": 255,
    "plcCount": 40,
    "robotCount": 60,
    "cncCount": 150,
    "laserCount": 5,
    "featureAppDNC": true,
    "licType": "Full",
    "uid": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
    "expiration": "2999-12-31T23:59:59"
  }
}
```

Response Parameter	Type	Description
isValid	Bool	(Required) Whether the license is valid; true = valid, false = invalid.
license	object	(Required) License details
company	String	Name of the company that issued the license. Not returned in the neutralized (white-label) version.
product	String	(Required) Name of the product the license applies to.
machineCount	Int32	(Required) Total number of licenses.
plcCount	Int32	Number of PLC licenses. If not returned, the PLC license count equals the total license count.
robotCount	Int32	Number of robot licenses. If not returned, the robot license count equals the total license count.
cncCount	Int32	Number of CNC licenses. If not returned, the CNC license count equals the total license count.

Response Parameter	Type	Description
laserCount	Int32	Number of laser cutter licenses. If not returned, the laser cutter license count equals the total license count.
featureAppDNC	Bool	(Required) Whether the professional file transfer edition is enabled; true = enabled, false = disabled.
licType	String	(Required) License type: Basic = Basic edition, Standard = Standard edition, Full = Extended edition, DNC = DNC edition, PLC = PLC edition.
uid	String	(Required) Gateway UID this license applies to.
expiration	String	(Required) License expiration date.

5.10.1.3. service-status - Get Service Status

This interface retrieves the status of services including the cloud platform, MODBUS, MQTT, the database, the local cache, and more. This interface has no request parameters.

```
GET /api/core/service-status
```

Response example

```
{
  "mqtt": 3,
  "queueSizeMqtt": 9,
  "modbus": 0,
  "queueSizeModbus": 0,
  "tbCloud": 0,
  "queueSizeTbCloud": 0,
  "localDB": 3,
  "queueSizeLocalDB": 0,
  "writeData": 2,
  "queueSizeWriteData": 0,
```

```
  "localDBLog": 0,
  "queueSizeLocalDBLog": 0,
  "hubBroadcasting": 2,
  "queueSizeHubBroadcasting": 0,
  "tbHubBroadcasting": 0,
  "queueSizeTbHubBroadcasting": 0,
  "needRestart": false
}
```

Response Parameter	Type	Description
mqtt	Int32	(Required) MQTT status, see Service Running Status.
queueSizeMqtt	Int32	(Required) MQTT queue length
modbus	Int32	(Required) MODBUS status, see Service Running Status.
queueSizeModbus	Int32	(Required) MODBUS queue length
tbCloud	Int32	(Required) Cloud platform status, see Service Running Status.
queueSizeTbCloud	Int32	(Required) Cloud platform queue length
localDB	Int32	(Required) Local cache status, see Service Running Status.
queueSizeLocalDB	Int32	(Required) Local cache queue length
writeData	Int32	(Required) Database status, see Service Running Status.
queueSizeWriteData	Int32	(Required) Database queue length
localDBLog	Int32	(Required) Log database status, see Service Running Status.
queueSizeLocalDBLog	Int32	(Required) Log database queue length
hubBroadcasting	Int32	(Required) Hub broadcast status, see Service Running Status.
queueSizeHubBroadcasting	Int32	(Required) Hub broadcast queue length

Response Parameter	Type	Description
tbHubBroadcasting	Int32	(Required) Cloud platform broadcast status, see Service Running Status.
queueSizeTbHubBroadcasting	Int32	(Required) Cloud platform broadcast queue length
needRestart	Bool	(Required) Whether the service needs to be restarted to apply the settings

5.10.1.4. upload-license - Upload Gateway License

Uploads a gateway license as a string stream.

```
POST /api/core/upload-license
```

Request body example

```
{
  "base64": "string"
}
```

Request Parameter	Type	Description
base64	String	(Required) Base64-encoded content of the license file.

The gateway license file should be converted to Base64 string format before uploading; the gateway reads the file content using the Base64 string format.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.1.5. log-level - Switch the Core Log Level

Switches the log level of the Core service at runtime. It takes effect immediately and does not require a service restart. The switch is not written to the configuration file; after the Core service restarts, the log level configured in the configuration file is restored.

```
GET /api/core/log-level?level=LEVEL
```

Request Parameter	Type	Description
level	String	(Required) Log level, one of: Verbose (0), Debug (1), Information (2), Warning (3), Error (4), Fatal (5). Either the level name or its numeric value can be used. A value outside this range returns GENERAL_API_INVALID_REQUEST.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.1.6. need-restart - Flag That a Restart Is Required

Flags the Core service as needing a restart to apply the settings. Once flagged, the needRestart field returned by 2.10.1.3. service-status - Get Service Status becomes true. This interface only sets the flag; it does not restart the service. This interface has no request parameters.

```
GET /api/core/need-restart
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.1.7. settings - Get the Core Runtime Configuration

Retrieves the runtime configuration currently loaded by the Core service. This interface has no request parameters.

```
GET /api/core/settings
```

Response example

```
{
  "logLevel": 2,
  "logFileSize": 10000000,
  "logFileCount": 60,
  "coreBaseUrl": "http://localhost:18100",
  "gatewayIPEndPoint": "127.0.0.1:18101",
  "uid": "XXXXXX-XXXXXX-XXXXXX-XXXXXX",
  "modbusPort": 502,
  "serial": {
    "coM1": "LOCAL,COM1",
    "coM2": null,
```

```
    "coM3": null,
    "coM4": null,
    "coM5": null,
    "coM6": null
  },
  "fileServerDir": "E:\\dnc",
  "smbGroup": "dcom",
  "vhdDir": "C:\\iotgw\\vhd",
  "coreContentDir": "C:\\iotgw\\core",
  "gatewayContentDir": "C:\\iotgw\\gateway",
  "showCreateCnc": true,
  "showCreateLaser": true,
  "showCreateRobot": true,
  "showCreatePlc": true,
  "enableInfluxLog": false,
  "enableMqttDefaultCustomAttributes": false,
  "maxAllowedWrapperInstanceCount": 3,
  "serviceQueueCapacity": 100000,
  "enableQueueOverflowToDisk": true,
  "queueOverflowMemoryCapacity": 1000,
  "queueOverflowStoragePath": null,
  "queueOverflowSegmentSize": 5000,
  "maxQueueOverflowBytes": 2147483648,
  "queueOverflowDropWarnIntervalMs": 5000,
  "taskManagerLaunchingOffset": 100,
  "processMetricsIntervalMs": 60000,
  "watchdogEnabled": true,
  "watchdogUnhealthyMinutes": 3,
  "watchdogCooldownMinutes": 10,
  "watchdogProbeTimeoutMs": 10000,
  "internetProbeTargets":
"223.5.5.5:53,114.114.114.114:53,www.baidu.com:80",
  "internetProbeIntervalMs": 15000,
  "internetProbeTimeoutMs": 2000,
  "internetProbeFailuresBeforeOffline": 3,
  "fanucRobotAlarmLogCount": 10,
  "fanucRobotCurAlarmCount": 10,
  "fanucRobotUseWebAlarm": false,
  "mqttMaxBatchSize": 1000,
  "tbCloudServiceIgnoreUID": false,
```

```
  "uiPath": "/app/gateway",
  "bypassRoutes": null,
  "agents": [],
  "tbCloudAddress": null,
  "tbCloudUsername": null,
  "tbCloudPassword": null,
  "tbxAddress": null,
  "hubContentDir": "C:\\iotgw\\core\\"
}
```

Response Parameter	Type	Description
logLevel	Int32	Log level at startup; the values are the same as the numeric level values of 2.10.1.5. log-level - Switch the Core Log Level.
logFileSize	Int32	Maximum size of a single log file (bytes).
logFileCount	Int32	Number of log files to retain.
coreBaseUrl	String	Listening address of the Core service.
gatewayIPEndPoint	String	Address and port of the Gateway service.
uid	String	Gateway UID.
modbusPort	Int32	Listening port of the MODBUS service.
serial	Object	Serial port usage configuration.
coM1 ~ coM6	String	Usage configuration of serial ports COM1 to COM6; null when not configured. The fields are serialized with camel-case naming, so the actual keys are coM1 to coM6.
fileServerDir	String	Root directory of the file server.
smbGroup	String	Workgroup the file server share belongs to.
vhdDir	String	Virtual disk directory.
coreContentDir	String	Data directory of the Core service.

Response Parameter	Type	Description
gatewayContentDir	String	Data directory of the Gateway service.
showCreateCnc	Bool	Whether creating CNC machines is allowed.
showCreateLaser	Bool	Whether creating laser cutters is allowed.
showCreateRobot	Bool	Whether creating robots is allowed.
showCreatePlc	Bool	Whether creating PLCs is allowed.
enableInfluxLog	Bool	Whether Influx logging is enabled.
enableMqttDefaultCustomAttributes	Bool	Whether custom attributes are included in MQTT output by default.
maxAllowedWrapperInstanceCount	Int32	Maximum number of acquisition process instances.
serviceQueueCapacity	Int32	Fixed in-memory capacity limit of a service queue when disk overflow is disabled; it no longer grows with the process’s memory usage.
enableQueueOverflowToDisk	Bool	Whether a queue overflows to disk once its in-memory capacity is full.
queueOverflowMemoryCapacity	Int32	Number of items kept in memory per queue before spilling to disk.
queueOverflowStoragePath	String	Storage path for overflow data; when null, the queue-overflow directory under coreContentDir is used.
queueOverflowSegmentSize	Int32	Number of items per overflow segment file.
maxQueueOverflowBytes	Int64	Byte cap of overflow data per queue; writing stops once the cap is reached.
queueOverflowDropWarnIntervalMs	Int32	Minimum interval between drop and disk I/O warnings (milliseconds).

Response Parameter	Type	Description
taskManagerLaunchingOffset	Int32	Interval between task launches (milliseconds).
processMetricsIntervalMs	Int32	Output interval of the process metrics log line ([Metrics]) in milliseconds, rounded down to whole minutes; 0 disables it.
watchdogEnabled	Bool	Whether Core and Gateway watch each other: every minute each probes the other’s service and starts / force-restarts it once it has been stopped or unresponsive for watchdogUnhealthyMinutes (a .dmp of a hung process is written to its log directory first). Never active in development mode.
watchdogUnhealthyMinutes	Int32	Consecutive minutes the peer service may be stopped or unresponsive before the watchdog acts.
watchdogCooldownMinutes	Int32	Minimum interval between two watchdog actions on the same service (minutes).
watchdogProbeTimeoutMs	Int32	Timeout of each watchdog HTTP probe (milliseconds).
internetProbeTargets	String	Internet probe targets, a comma-separated host:port list; empty disables probing.
internetProbeIntervalMs	Int32	Internet probe interval (milliseconds).
internetProbeTimeoutMs	Int32	Connect timeout per internet probe target (milliseconds).
internetProbeFailuresBeforeOffline	Int32	Number of consecutive all-failed probe rounds before the gateway is considered offline.
fanucRobotAlarmLogCount	Int32	Number of historical alarm records read from FANUC robots.
fanucRobotCurAlarmCount	Int32	Number of current alarm records read from FANUC robots.

Response Parameter	Type	Description
fanucRobotUseWebAlarm	Bool	Whether FANUC robot alarms are read over the web interface.
mqttMaxBatchSize	Int32	Maximum number of items sent in a single MQTT batch.
tbCloudServiceIgnoreUID	Bool	Whether the cloud platform service ignores the gateway UID.
uiPath	String	Deployment path of the web interface.
bypassRoutes	String	Hub only. Routes that are passed through without proxying. Core returns null.
agents	Object[]	Hub only. List of registered gateway agents. Core returns an empty array.
tbCloudAddress	String	Hub only. Cloud platform address. Core returns null.
tbCloudUsername	String	Hub only. Cloud platform user name. Core returns null.
tbCloudPassword	String	Hub only. Cloud platform password. Core returns null.
tbxAddress	String	Hub only. TBX service address. Core returns null.
hubContentDir	String	Hub only. Data directory of the Hub service.

> **Note**
> The response contains credential fields such as tbCloudUsername and tbCloudPassword. Do not expose this interface to untrusted callers.

5.10.1.8. drop-service-queue - Discard a Service’s Send Queue

Permanently discards the data a service has queued for sending, both in memory and on disk (the backlog counted by the queueSize fields of 2.10.1.3. service-status - Get Service Status). Use it when the target server has been unreachable for a long time and the backlog no longer needs to be delivered. Discarded data cannot be recovered; only the send queue is affected, not machine data acquisition.

```
GET /api/core/drop-service-queue?service=SERVICE
```

Request Parameter	Type	Description
service	String	(Required) Service name, case-insensitive, one of: mqtt, tbCloud, modbus, localDB, writeData, hubBroadcasting, tbHubBroadcasting, or all for every service. A missing or unknown value returns GENERAL_API_INVALID_REQUEST.

Response example

```
{
  "droppedTotal": 1500,
  "dropped": {
    "mqtt": 1500
  }
}
```

Response Parameter	Type	Description
droppedTotal	Int32	(Required) Total number of items discarded.
dropped	Object	(Required) Number of items discarded per service, keyed by service name; services the gateway has not enabled are absent.

5.10.2. Gateway Service Function Interfaces

Base URL `/api/gateway`.

5.10.2.1. alias - Get gateway name

This interface takes no request parameters.

```
GET /api/gateway/alias
```

Response example

```
{
  "alias": "iotgw"
}
```

Response Parameter	Type	Description
alias	String	(Required) Gateway name

5.10.2.2. update-alias - Update gateway name

Note: this change takes effect after the hardware is rebooted.

```
POST /api/gateway/update-alias
```

Request body example application/json

```
{
  "alias": "newAlias"
}
```

Request Parameter	Type	Description
alias	String	(Required) Gateway name

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.3. reboot - Reboot gateway hardware

Equivalent to clicking the “Power” - “Reboot” button on the gateway management page. This interface takes no request parameters.

```
GET /api/gateway/reboot
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.4. restart - Restart all services

This interface restarts both the Core service and the Gateway service. This restart differs in function from the “Restart Service” button on the gateway home page. This interface takes no request parameters.

```
GET /api/gateway/restart
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.5. restart-service - Restart the Core service

After a user modifies machine configuration, machine group configuration, task configuration, or communication configuration, this interface must be called; it is equivalent to clicking the "Restart Service" button on the gateway home page. This interface takes no request parameters.

```
GET /api/gateway/restart-service
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.6. shut-down - Shut down the gateway

Equivalent to clicking the “Power” - “Shut Down” button on the gateway management page. This interface takes no request parameters.

```
GET /api/gateway/shut-down
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.7. internet-connection - Get gateway network status

Gets the gateway’s connection status to the internet. This interface takes no request parameters.

The gateway probes the targets in `InternetProbeTargets` in parallel every `InternetProbeIntervalMs` (15s by default); any one succeeding means online. It only reports offline after `InternetProbeFailuresBeforeOffline` (3 by default) consecutive all-failed rounds, while one success restores online immediately. This interface reads that cached verdict and never probes on the request itself.

```
GET /api/gateway/internet-connection
```

Response example

```
{
  "isOnline": true,
  "checkedAtUnix": 1757376000,
```

```
  "lastOkUnix": 1757376000
}
```

Response Parameter	Type	Description
isOnline	Bool	Whether connected to the internet, true = connected, false = not connected; null before the first probe round produces a verdict, or when probing is disabled.
checkedAtUnix	Int64	Unix timestamp (seconds) of the last completed probe round; null when none has run.
lastOkUnix	Int64	Unix timestamp (seconds) of the last round that reached a target; null when none ever did.

5.10.2.8. hardware-resources - Get gateway hardware resources

Gets the gateway’s CPU, memory, and disk usage, for monitoring gateway operating load. This interface takes no request parameters.

```
GET /api/gateway/hardware-resources
```

Response example

```
{
  "cpuUsage": 12.5,
  "physicalMemoryUsage": 48.3,
  "pagingMemoryUsage": 35.1,
  "virtualMemoryUsage": 22.7,
  "diskTotalBytes": 500000000000,
  "diskUsedBytes": 205800000000,
  "diskUsage": 41.16
}
```

Response Parameter	Type	Description
cpuUsage	Float	CPU usage (percentage, 0-100).

Response Parameter	Type	Description
physicalMemoryUsage	Float	Physical memory usage (percentage, 0-100).
pagingMemoryUsage	Float	Paging memory (page file) usage (percentage, 0-100).
virtualMemoryUsage	Float	Virtual memory usage (percentage, 0-100).
diskTotalBytes	Int64	Total disk capacity (bytes).
diskUsedBytes	Int64	Used disk capacity (bytes).
diskUsage	Float	Disk usage (percentage, 0-100).

> **Note**
>
> All of the response parameters above are optional. CPU usage is one item; physical memory, paging memory, and virtual memory form one group of memory metrics; total disk capacity, used capacity, and disk usage form one group of disk metrics. If the gateway cannot obtain a given group of metrics (e.g. because the host system does not support it), that group of fields is omitted entirely from the response.

5.10.2.9. time - Get the gateway’s current time

This interface takes no request parameters.

```
GET /api/gateway/time
```

Response example

```
{
  "localTime": "2025-06-30T05:51:19.286Z"
}
```

Response Parameter	Type	Description
localTime	String	(Required) The gateway’s current time, output in UTC (ISO 8601, format yyyy-MM-ddTHH:mm:ss.fffZ).

5.10.2.10. sync-time - Synchronize gateway time

```
GET /api/gateway/sync-time
```

Request Parameter	Type	Description
timeServerAddress	String	(Required) Time server address(es). Multiple addresses are separated by “;” ; the gateway first attempts to synchronize with the first address, and if that fails, tries the next address. Example: ntp1.aliyun.com;ntp2.aliyun.com; ntp3.aliyun.com;ntp4.aliyun.com;

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.11. time-zone - Get gateway time zone

This interface takes no request parameters.

```
GET /api/gateway/time-zone
```

Response example

```
{
  "timeZoneID": "China Standard Time"
```

```
}
```

Response Parameter	Type	Description
timeZoneID	String	(Required) Gateway time zone (Microsoft Windows time zone ID)

5.10.2.12. time-zones - Get time zone options

This interface takes no request parameters.

```
GET /api/gateway/time-zones
```

Response example

```
{
  "timeZoneIDs": [
  "Afghanistan Standard Time",
  "Alaskan Standard Time",
  "Aleutian Standard Time",
  "…",
  "China Standard Time",
  "…"
  ]
}
```

Response Parameter	Type	Description
timeZoneIDs	String[]	(Required) Available gateway time zone options (Microsoft Windows time zone IDs). The list is sorted alphabetically by time zone ID.

5.10.2.13. update-time-zone - Update gateway time zone

```
POST /api/gateway/update-time-zone
```

Request body example application/json

```
{
  "timeZoneID": "China Standard Time"
}
```

Request Parameter	Type	Description
timeZoneID	String	(Required) Gateway time zone (Microsoft Windows time zone ID). Available options can be obtained via 2.10.2.12. time-zones - Get time zone options.

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.14. network-adapters - Get gateway network adapter list

This interface takes no request parameters.

```
GET /api/gateway/network-adapters
```

Response example

```
{
  "names": [
  "LAN1",
  "LAN2",
  "WLAN"
  ]
```

```
}
```

Response Parameter	Type	Description
names	String[]	(Required) Network adapter names

5.10.2.15. lan - Get wired network settings

```
GET /api/gateway/lan
```

Request Parameter	Type	Description
name	String	(Required) Network adapter name, valid range: LAN1 or LAN2.

Response example

```
{
  "name": "LAN1",
  "macAddress": "00:E0:71:BC:D2:53",
  "state": "Disconnected",
  "isDHCPEnabled": false,
  "ipAddress": "192.168.100.1",
  "subMask": "255.255.0.0",
  "defaultGateway": "",
  "isDNSServerDHCPEnabled": false,
  "dnsServer1": "",
  "dnsServer2": ""
}
```

Response Parameter	Type	Description
name	String	(Required) Network adapter name, valid range: LAN1 or LAN2.
macAddress	String	(Required) MAC address
state	String	(Required) State, valid range: Connected, Disconnected.

Response Parameter	Type	Description
isDHCPEnabled	Bool	(Required) Obtain IP address automatically
ipAddress	String	(Required) IP address
subMask	String	(Required) Subnet mask
defaultGateway	String	(Required) Default gateway
isDNSServerDHCPEnabled	Bool	(Required) Obtain DNS server address automatically
dnsServer1	String	(Required) Preferred DNS server
dnsServer2	String	(Required) Alternate DNS server

5.10.2.16. update-lan - Update gateway wired network settings

```
POST /api/gateway/update-lan
```

Request body example application/json

```
{
  "name": "LAN1",
  "isDHCPEnabled": false,
  "ipAddress": "192.168.100.1",
  "subMask": "255.255.0.0",
  "defaultGateway": "",
  "isDNSServerDHCPEnabled": false,
  "dnsServer1": "",
  "dnsServer2": ""
}
```

Request Parameter	Type	Description
name	String	(Required) Target network adapter name, valid range: LAN1 or LAN2.
isDHCPEnabled	Bool	Obtain IP address automatically
ipAddress	String	IP address
subMask	String	Subnet mask
defaultGateway	String	Default gateway

Request Parameter	Type	Description
isDNSServerDHCPEnabled	Bool	Obtain DNS server address automatically
dnsServer1	String	Preferred DNS server
dnsServer2	String	Alternate DNS server

Response example

```
{
  "name": "LAN1",
  "macAddress": "00:E0:71:BC:D2:53",
  "state": "Disconnected",
  "isDHCPEnabled": false,
  "ipAddress": "192.168.100.1",
  "subMask": "255.255.0.0",
  "defaultGateway": "",
  "isDNSServerDHCPEnabled": false,
  "dnsServer1": "",
  "dnsServer2": ""
}
```

Response Parameter	Type	Description
name	String	(Required) Network adapter name, valid range: LAN1 or LAN2.
macAddress	String	(Required) MAC address
state	String	(Required) State, valid range: Connected, Disconnected.
isDHCPEnabled	Bool	(Required) Obtain IP address automatically
ipAddress	String	(Required) IP address
subMask	String	(Required) Subnet mask
defaultGateway	String	(Required) Default gateway
isDNSServerDHCPEnabled	Bool	(Required) Obtain DNS server address automatically
dnsServer1	String	(Required) Preferred DNS server
dnsServer2	String	(Required) Alternate DNS server

5.10.2.17. wifi - Get wireless network settings

This interface takes no request parameters.

```
GET /api/gateway/wifi
```

Response example

```
{
  "wifiName": "myWifi-5G",
  "signalStrength": 99,
  "macAddress": "00:A0:71:BD:E2:63",
  "state": "Connected",
  "isDHCPEnabled": true,
  "ipAddress": "192.168.1.88",
  "subMask": "255.255.255.0",
  "defaultGateway": "192.168.1.1",
  "isDNSServerDHCPEnabled": true,
  "dnsServer1": "192.168.211.1",
  "dnsServer2": "8.8.8.8"
}
```

Response Parameter	Type	Description
wifiName	String	Wireless network SSID
signalStrength	Int32	Range: 0-100, the higher the value the stronger the signal
macAddress	String	(Required) MAC address
state	String	(Required) State, valid range: Connected, Disconnected.
isDHCPEnabled	Bool	(Required) Obtain IP address automatically
ipAddress	String	(Required) IP address
subMask	String	(Required) Subnet mask
defaultGateway	String	(Required) Default gateway
isDNSServerDHCPEnabled	Bool	(Required) Obtain DNS server address automatically
dnsServer1	String	(Required) Preferred DNS server

Response Parameter	Type	Description
dnsServer2	String	(Required) Alternate DNS server

> **Note**
> wifiName and signalStrength are returned only when state is Connected; when the gateway is not connected to a wireless network, these two fields are omitted from the response.

When the gateway has no usable wireless adapter, error code 10051 (GENERAL_WIFI_NOT_AVAILABLE) is returned, with the cause in statusMsg.

5.10.2.18. update-wifi - Update wireless network settings

```
POST /api/gateway/update-wifi
```

Request body example application/json

```
{
  "isDHCPEnabled": true,
  "ipAddress": "192.168.1.88",
  "subMask": "255.255.255.0",
  "defaultGateway": "192.168.1.1",
  "isDNSServerDHCPEnabled": true,
  "dnsServer1": "192.168.211.1",
  "dnsServer2": "8.8.8.8"
}
```

Request Parameter	Type	Description
isDHCPEnabled	Bool	Obtain IP address automatically
ipAddress	String	IP address
subMask	String	Subnet mask
defaultGateway	String	Default gateway
isDNSServerDHCPEnabled	Bool	Obtain DNS server address automatically
dnsServer1	String	Preferred DNS server
dnsServer2	String	Alternate DNS server

Response example

```
{
  "wifiName": "myWifi-5G",
  "signalStrength": 99,
  "macAddress": "00:A0:71:BD:E2:63",
  "state": "Connected",
  "isDHCPEnabled": true,
  "ipAddress": "192.168.1.88",
  "subMask": "255.255.255.0",
  "defaultGateway": "192.168.1.1",
  "isDNSServerDHCPEnabled": true,
  "dnsServer1": "192.168.211.1",
  "dnsServer2": "8.8.8.8"
}
```

Response Parameter	Type	Description
wifiName	String	Wireless network SSID
signalStrength	Int32	Range: 0-100, the higher the value the stronger the signal
macAddress	String	(Required) MAC address
state	String	(Required) State, valid range: Connected, Disconnected.
isDHCPEnabled	Bool	(Required) Obtain IP address automatically
ipAddress	String	(Required) IP address
subMask	String	(Required) Subnet mask
defaultGateway	String	(Required) Default gateway
isDNSServerDHCPEnabled	Bool	(Required) Obtain DNS server address automatically
dnsServer1	String	(Required) Preferred DNS server
dnsServer2	String	(Required) Alternate DNS server

Note

wifiName and signalStrength are returned only when state is Connected; when the gateway is not connected to a wireless network, these two fields are omitted from the response.

5.10.2.19. search-wifi - Search for wireless networks

This interface takes no request parameters.

```
GET /api/gateway/search-wifi
```

Response example

```
[
  {
    "wifiName": "myWifi-5G",
    "signalStrength": 99,
    "state": "Connected"
  },
  {
    "wifiName": "myWifi-2.4G",
    "signalStrength": 99,
    "state": "Disconnected"
  }
]
```

Response Parameter	Type	Description
wifiName	String	(Required) Wireless network SSID
signalStrength	Int32	Range: 0-100, the higher the value the stronger the signal
state	String	(Required) State, valid range: Connected, Disconnected.

When the gateway has no usable wireless adapter, error code 10051 (GENERAL_WIFI_NOT_AVAILABLE) is returned, with the cause in statusMsg.

5.10.2.20. connect-wifi - Connect to a wireless network

```
GET /api/gateway/connect-wifi
```

Request Parameter	Type	Description
wifiName	String	(Required) Target wireless network SSID
wifiPassword	String	Target wireless network password

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.21. disconnect-wifi - Disconnect from wireless network

This interface takes no request parameters.

```
GET /api/gateway/disconnect-wifi
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.22. static-routing - Get static routing settings

This interface takes no request parameters.

```
GET /api/gateway/static-routing
```

Response example

```
{
  "staticRouting": "0.0.0.0, 0.0.0.0, 192.168.100.1;"
}
```

Response Parameter	Type	Description
staticRouting	String	(Required) Static routing. Multiple routes are separated by “;”, and each route has the format “IP, subnet mask, gateway”.

5.10.2.23. update-static-routing - Update static routing settings

```
POST /api/gateway/update-static-routing
```

Request body example application/json

```
{
  "staticRouting": "0.0.0.0, 0.0.0.0, 192.168.100.1;"
}
```

Request Parameter	Type	Description
staticRouting	String	(Required) Static routing. Multiple routes are separated by “;”, and each route has the format “IP, subnet mask, gateway”. Passing an empty string clears all persistent routes. An invalid format returns error code 10012 (invalid IP address).

Response example

```
{
  "staticRouting": "0.0.0.0, 0.0.0.0, 192.168.100.1;"
}
```

Response Parameter	Type	Description
staticRouting	String	(Required) Static routing. Multiple routes are separated by “;” , and each route has the format “IP, subnet mask, gateway” .

5.10.2.24. connect-remote-host - Connect to remote server

This interface takes no request parameters.

```
GET /api/gateway/connect-remote-host
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.25. disconnect-remote-host - Disconnect from remote server

This interface takes no request parameters.

```
GET /api/gateway/disconnect-remote-host
```

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.26. file-server-items - Get gateway file server item list

Gets the list of files and subdirectories under the root directory of the gateway file server. Subdirectories correspond to machines, named after the machine’s IP address. This interface takes no request parameters.

```
GET /api/gateway/file-server-items
```

Response example

```
{
  "totalSpace": "10.00 GB",
  "usedSpace": "120 B",
  "files": [
  ],
  "subDirs": [
    {
      "name": "127.0.0.1",
      "size": "20 B",
      "machineName": "1",
      "machineID": "1",
      "ip": "127.0.0.1",
      "status": "Activated"
    },
    {
```

```
        "name": "127.0.0.2",
        "size": "100 B",
        "machineName": "2",
        "machineID": "2",
        "ip": "127.0.0.2",
        "status": "Activated"
      }
    ]
  }
```

Response Parameter	Type	Description
totalSpace	String	(Required) Total space
usedSpace	String	(Required) Used space
files	Object[]	(Required) List of files under the root directory, usually empty.
subDirs	Object[]	(Required) List of subdirectories under the root directory.
name	String	File/directory name
size	String	File/directory size
machineName	String	Associated machine name
machineID	String	Associated machine identifier
ip	String	Associated machine IP
status	String	Status, valid range: Unlinked, Activated, Unactivated.

5.10.2.27. delete-file-server-item - Delete a gateway file server item

Deletes the specified file or subdirectory under the root directory.

```
GET /api/gateway/delete-file-server-item
```

Request Parameter	Type	Description
fileName	String	File name
dirName	String	Subdirectory name

Response example

```
{
  "errorCode": 0,
  "errorMsg": "Success"
}
```

Response Parameter	Type	Description
errorCode	Int32	(Required) Error code, 0 indicates success.
errorMsg	String	(Required) Error message

5.10.2.28. ping - Test network reachability

Sends ICMP echo requests from the gateway to a target host, to test reachability from the gateway to that host. Each attempt times out after 1000 ms.

```
GET /api/gateway/ping
```

Request Parameter	Type	Description
host	String	(Required) Target IP address or host name
count	Int32	Number of attempts, default 4, maximum 10; exceeding the maximum returns an error.

Response example

```
{
  "isReachable": true,
  "repliesMs": [
    12,
    11,
    null,
    13
  ]
}
```

Response Parameter	Type	Description
isReachable	Bool	(Required) Whether the target is reachable; true if any attempt received a reply.
repliesMs	Int64[]	(Required) Round-trip time of each attempt (milliseconds); null when that attempt timed out without a reply.

5.10.2.29. telnet - Test port connectivity

Opens a TCP connection from the gateway to the specified port on a target host, to test whether the port is reachable. The connection times out after 3000 ms.

```
GET /api/gateway/telnet
```

Request Parameter	Type	Description
host	String	(Required) Target IP address or host name
port	Int32	(Required) Target port, valid range: 1-65535.

Response example

```
{
  "isConnected": true
}
```

Response Parameter	Type	Description
isConnected	Bool	(Required) Whether a TCP connection can be established within 3 seconds.

5.10.2.30. traceroute - Trace network route

Traces the network path from the gateway to a target host hop by hop. Each hop times out after 3000 ms.

```
GET /api/gateway/traceroute
```

Request Parameter	Type	Description
host	String	(Required) Target IP address or host name
maxHops	Int32	Maximum number of hops, default 30, maximum 64; exceeding the maximum returns an error.

Response example

```
{
  "reachedTarget": true,
  "hops": [
    {
      "hop": 1,
      "address": "192.168.1.1",
      "rttMs": 1
    },
    {
      "hop": 2,
      "address": null,
      "rttMs": null
    }
  ]
}
```

Response Parameter	Type	Description
reachedTarget	Bool	(Required) Whether the target host was reached.
hops	Object[]	(Required) List of per-hop results.
hop	Int32	Hop number (TTL), starting from 1.
address	String	IP address of the node that responded at this hop; null when the hop did not respond.

Response Parameter	Type	Description
rttMs	Int64	Round-trip time of this hop (milliseconds); null when the hop did not respond.

5.10.2.31. scan-ports - Scan ports

Opens a TCP connection to each port in a set of ports on the target host, to detect whether the ports are open. Each port connection times out after 1000 ms.

```
POST /api/gateway/scan-ports
```

Request body example application/json

```
{
  "host": "192.168.1.1",
  "startPort": 80,
  "endPort": 88,
  "ports": [
    443,
    8080
  ]
}
```

Request Parameter	Type	Description
host	String	(Required) Target IP address or host name
startPort	Int32	Start of the port range (inclusive); must not be greater than endPort.
endPort	Int32	End of the port range (inclusive).
ports	Int32[]	List of ports to scan; merged with the port range and deduplicated.

> **Note**
> At least one of the port range and ports must be provided; an error is returned when the merged, deduplicated set is empty. Ports must be between 1 and 65535, and the merged, deduplicated set must not exceed 1024 ports.

Response example

```
{
  "ports": [
    {
      "port": 80,
      "isOpen": true
    },
    {
      "port": 443,
      "isOpen": false
    }
  ]
}
```

Response Parameter	Type	Description
ports	Object[]	(Required) List of port scan results.
port	Int32	Port number
isOpen	Bool	Whether the port is open

5.10.2.32. lookup-dns - Resolve a host name

Resolves the specified host name on the gateway and returns the corresponding list of IP addresses.

```
GET /api/gateway/lookup-dns
```

Request Parameter	Type	Description
host	String	(Required) Host name to resolve

Response example

```
{
  "addresses": [
    "142.250.196.238"
  ]
}
```

Response Parameter	Type	Description
addresses	String[]	(Required) List of resolved IP addresses; an empty array is returned when the host name cannot be resolved, with no error code.

6. MODBUS Communication

After enabling MODBUS communication and turning on the automatic data collection task for the target machine/group (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), the gateway stores the automatic data collection task results in the corresponding register addresses based on the MODBUS protocol. The MODBUS communication address is `<Gateway IP>`, and the port is 502. To use MODBUS communication, the machine tool must have a slave ID configured (see 3.3.1.5. Advanced Settings); by default, the last segment of the IP address is used as the slave ID. For example, if the machine tool’s IP is 192.168.100.2, its default slave ID is 2. The slave ID cannot be set to 0 (0 is reserved for the gateway), and the slave IDs of all machine tools connected under the same gateway (default: the last segment of the IP) must be unique from one another; otherwise data identification errors will occur. Users can configure the MODBUS service by following the instructions in 3.6.2. MODBUS Configuration.

> **Note**
>
> The gateway’s MODBUS server register addresses are 0-based. If the client uses 1-based addressing, add 1 to the corresponding address.

6.1. Machine-Related Data

The MODBUS data addresses are as follows:

Address	Name	Data Type	Unit	Remarks
400010	Connection status	UInt16	-	1-Connected;2-Disconnected;3-Error
400011	Alarm count	UInt16	count	Current number of alarms
400012	CNC operating status	Int16	-	Current operating status code, see Operating Status

Address	Name	Data Type	Unit	Remarks
400020	Raw program status value	Int16	-	Raw program status code returned by the machine tool’s system; -1 if the machine tool’s system does not support it or the data is unavailable
400021	Raw operating mode value	Int16	-	Raw mode code returned by the machine tool’s system; -1 if the machine tool’s system does not support it or the data is unavailable
400040~400075	Axis 1 to Axis 18 name	String(4)	-	Converted to a 4-byte array (2 register addresses) using the encoding set in the MODBUS configuration.
400100	Machining count	UInt32	pieces	Machining count obtained from the machine tool’s system
400102	Spindle override	Float	%	Spindle speed override
400104	Actual spindle speed	Float	same as machine setting	
400106	Feed override	Float	%	Feed rate override
400108	Actual feed rate	Float	same as machine setting	
400110~400115	Spindle 1 to Spindle 3 load	Float	%	
400120	Rapid feed override	Float	%	
400122	Commanded feed rate	Float	same as machine setting	

Address	Name	Data Type	Unit	Remarks
400124	Commanded spindle speed	Float	same as machine setting	
400172	Cumulative power-on time 1	Int32	minutes	See 4.2.24. TimeData: Machine Time Data for how the time is calculated. If the value is -2147483648, it indicates the machine tool’s system model does not support this time data.
400174	Cumulative power-on time 2	Int32	milliseconds	
400176	Current power-on time 1	Int32	minutes	
400178	Current power-on time 2	Int32	milliseconds	
400180	Cumulative running time 1	Int32	minutes	
400182	Cumulative running time 2	Int32	milliseconds	
400184	Current running time 1	Int32	minutes	
400186	Current running time 2	Int32	milliseconds	
400188	Cumulative machining time 1	Int32	minutes	
400190	Cumulative machining time 2	Int32	milliseconds	
400192	Current machining time 1	Int32	minutes	
400194	Current machining time 2	Int32	milliseconds	
400196	Last cycle time 1	Int32	minutes	
400198	Last cycle time 2	Int32	milliseconds	

Address	Name	Data Type	Unit	Remarks
400200	Current cycle time 1	Int32	minutes	
400202	Current cycle time 2	Int32	milliseconds	
400204	Current cutting time 1	Int32	minutes	
400206	Current cutting time 2	Int32	milliseconds	
400208	Remaining cycle time 1	Int32	minutes	
400210	Remaining cycle time 2	Int32	milliseconds	
400220	Current tool number	String(16)	-	Converted to a 16-byte array (8 register addresses) using the encoding set in the MODBUS configuration.
400228	Running program name	String(32)	-	Name of the currently executing subprogram
400244	Main program name	String(32)	-	Name of the current main program
400260	Running program block sequence number	String(16)	-	Sequence number of the currently executing subprogram block
400268	Current program block	String(70)	-	Content of the currently running program block
400498	Current machining count	UInt32	pieces	Machining count within the current monitoring period
400500	Cumulative machining count	UInt32	pieces	
400502	Shutdown time	UInt32	seconds	

Address	Name	Data Type	Unit	Remarks
400504	Emergency stop time	UInt32	seconds	
400506	Standby time	UInt32	seconds	
400508	Automatic running time	UInt32	seconds	
400510	Setup time	UInt32	seconds	
400512	Machine monitored utilization rate	Float	%	
400520	Cumulative shutdown time	UInt32	seconds	
400522	Cumulative emergency stop time	UInt32	seconds	
400524	Cumulative standby time	UInt32	seconds	
400526	Cumulative automatic running time	UInt32	seconds	
400528	Cumulative setup time	UInt32	seconds	
400546	Last takt time	UInt32	seconds	
400600	Program stack	String(128)	-	Converted to a 128-byte array (64 register addresses) using the encoding set in the MODBUS configuration.
400664	Current tool offset number	String(16)	-	
400900	Preset power	Float	%	Preset power of the laser cutting machine
400902	Actual power	Float	%	Actual power of the laser cutting machine

Address	Name	Data Type	Unit	Remarks
401100~401135	Axis 1 to Axis 18 load	Float	%	
401136~401171	Axis 1 to Axis 18 machine coordinate	Float	same as machine setting	
401172~401207	Axis 1 to Axis 18 relative coordinate	Float	same as machine setting	
401208~401243	Axis 1 to Axis 18 remaining distance	Float	same as machine setting	
401244~401279	Axis 1 to Axis 18 absolute coordinate	Float	same as machine setting	
401400~401405	Spindle 1 to Spindle 3 cumulative overload time	UInt32	milliseconds	
401410~401445	Axis 1 to Axis 18 cumulative overload time	UInt32	milliseconds	
405000~406499	Alarm 1 to Alarm 15 (each alarm occupies 100 register addresses; the start address of alarm n is 405000 + (n-1)×100)	String(200)	-	Includes alarm content, time, and level
420000~429999	PLC data bits 1 to 10000	UInt16	-	PLC task data and external PLC task data, with PLC task data first followed by external PLC task data, arranged in task order. 32-bit data, 64-bit data, and String-type data are converted to UInt16 using the method set in the MODBUS configuration.

Note 1

The cumulative machining count is the accumulated machining count since the gateway started automatically collecting data from this device. This value is stored in the gateway, bound to the `machineID` (i.e., the last two segments of the machine tool’s IP address), and is never reset.

Note 2

400502-400510: the machine’s state times are the durations of each state within the monitoring period. 400512: the machine’s utilization rate is the utilization rate within the monitoring period. See 11.1.1. Machine OEE Data for details.

Note 3

`String(200)` means this string is converted to a 200-byte array using the configured encoding (see 3.6.2. MODBUS Configuration), which corresponds to 100 MODBUS register addresses (UInt16).

Note 4

The last takt time is monitored by the gateway’s machine takt data task. If the gateway restarts and the first takt cycle has not yet finished, this value is 0. The last takt time is defined the same way as the last cycle time, but the former is calculated by the gateway and is supported by any machine whose output and status can be tracked, while the latter is provided by the machine tool itself and is only supported by some machines.

6.1.1. Multi-Channel Machine Data

The addresses above are the data addresses for channel 0 (the default channel). When a data collection task runs on a channel whose channel number is greater than 0 (the channel number is set with the `taskChannel*` parameters in the machine configuration, see 2.9.3. Machine Configuration Parameters), the channel-related data listed in the table below is written to a separate address area:

```
Channel data address = 430000 + 2000 × (channel number - 1) + (base address - 400000)
```

For example, the base address of the Axis 1 load is 401100, so its address is 431100 for channel 1 and 433100 for channel 2. Addresses 430000~449999 are reserved for channel-related data. Addresses not listed in the table below do not change with the channel number.

Base Address	Name
400040~400075	Axis 1 to Axis 18 name
400102	Spindle override
400104	Actual spindle speed
400106	Feed override
400108	Actual feed rate
400110~400115	Spindle 1 to Spindle 3 load
400120	Rapid feed override
400122	Commanded feed rate
400124	Commanded spindle speed
401100~401135	Axis 1 to Axis 18 load
401136~401171	Axis 1 to Axis 18 machine coordinate
401172~401207	Axis 1 to Axis 18 relative coordinate
401208~401243	Axis 1 to Axis 18 remaining distance
401244~401279	Axis 1 to Axis 18 absolute coordinate
401400~401405	Spindle 1 to Spindle 3 cumulative overload time
401410~401445	Axis 1 to Axis 18 cumulative overload time

6.2. Group-Related Data

Group-related data is stored uniformly under slave ID 0. The data addresses are as follows:

Address	Name	Data Type	Unit	Remarks
400100	Group 1 cumulative machining count	UInt32	pieces	
400102	Current number of shut-down machines in Group 1	Int16	units	
400103	Current number of emergency-stopped machines in Group 1	Int16	units	

Address	Name	Data Type	Unit	Remarks
400104	Current number of standby machines in Group 1	Int16	units	
400105	Current number of automatically running machines in Group 1	Int16	units	
400106	Current number of machines in setup in Group 1	Int16	units	
400110	Group 1 shutdown time	UInt32	seconds	
400112	Group 1 emergency stop time	UInt32	seconds	
400114	Group 1 standby time	UInt32	seconds	
400116	Group 1 automatic running time	UInt32	seconds	
400118	Group 1 setup time	UInt32	seconds	
400120	Group 1 monitored utilization rate	Float	%	
400130	Group 1 cumulative shutdown time	UInt32	seconds	
400132	Group 1 cumulative emergency stop time	UInt32	seconds	
400134	Group 1 cumulative standby time	UInt32	seconds	
400136	Group 1 cumulative automatic running time	UInt32	seconds	
400138	Group 1 cumulative setup time	UInt32	seconds	
400200~400299	Group 2 (corresponding addresses are Group 1’s addresses + 100)			

Address	Name	Data Type	Unit	Remarks
……				
401600~401699	Group 16 (corresponding addresses are Group 1’s addresses + 1500)			

Note 1

400100: the group’s cumulative machining count is the accumulated machining count since the gateway started automatically collecting data from this group of devices. See 11.1.3. Group Machining Count for details.

Note 2

400110-400118: the group’s state times are the durations of each state, within the monitoring period, for all active machines in the group. 400120: the group’s monitored utilization rate is the group’s utilization rate within the monitoring period. See 11.1.2. Group OEE Data for details.

7. MQTT Communication

After enabling MQTT communication and turning on the automatic acquisition task for the target machine/group (see 3.3.1.2. Task Settings and 3.4.1.2. Group Task Settings), the gateway publishes the results as JSON messages to the configured MQTT broker. The default data topic in standard modes is “r”; GeWu mode uses the platform’s topic. The gateway also supports the RPC interface. Configure MQTT as described in 3.6.3. MQTT Configuration.

7.1. Data Upload Message Format

The gateway’s MQTT data upload messages support Default, MKT, TB, TB2, Brm, IoTDA, GeWu, WisIoT, and other modes.

The message content structure for each mode is as follows:

Default Mode

```
{
    "type": <type>,
    "unixtimestamp": <time>,
    "data": {
        "<field>": <value>
    },
    "<tag>": <value>,
    "<entityID>": <value>
}
```

MKT Mode

```
{
    "unixtimestamp": <time>,
    "<entityID>_<fieldID>_<field>": <value>
}
```

TB Mode

```
{
    "<alias>": [
        {
            "ts": <time>,
            "values": {
                "<fieldID>_<field>": <value>
            }
        }
    ]
}
```

TB2 Mode

```
{
    "<entityID>": [
        {
            "ts": <time>,
            "values": {
                "<fieldID>_<field>": <value>
            }
        }
    ]
}
```

Brm Mode

```
[
    {
        "type": <type>,
        "ts": <time>,
        "data": {
            "<field>": <value>
        },
        "<tag>": <value>,
        "<entityID>": <value>
    }
```

```
]
```

IoTDA Mode

```
{
    "devices": [
        {
            "device_id": <entityID>,
            "services": [
                {
                    "service_id": <fieldID>,
                    "properties": {
                        "<field>": <value>
                    },
                    "event_time": <time>
                }
            ]
        }
    ]
}
```

IoTDA mode places the `<fieldID>`, which includes the data class and any tags, in `service_id`. Keys in `properties` are the original `<field>` names, without a tag prefix or separate tag properties.

GeWu Mode

The Gewu Cloud subdevice property topic is generated from the gateway product and device keys: `$sys/<gatewayProductKey>/<gatewayDeviceKey>/subdevice/propertyEvent/pub`. Machines and groups report as gateway subdevices. Before sending their data, the gateway adds active subdevices to its topology; see 3.6.3. MQTT Configuration.

```
{
    "messageId": "<messageId>",
    "params": {
        "subDevices": [
            {
```

```
                "productKey": "<machineProductKey or groupProductKey>",
                "deviceKey": "<entityID>",
                "properties": [
                    {
                        "key": "<fieldID>",
                        "value": { "<field>": <value> },
                        "ts": "<time>"
                    }
                ],
                "events": []
            }
        ]
    }
}
```

`deviceKey` is the machine’s `machineID` or group’s `groupID`. `key` is the `<fieldID>` built from the data class and tags, while the `value` object uses the original field names. `ts` is a millisecond timestamp encoded as a string. Boolean values are sent as lowercase strings `"true"` / `"false"`. A message can contain several subdevices, with at most 10 `properties` items in total; repeated samples with the same `key` remain separate items.

WisIoT Mode

```
{
    "properties": [
        {
            "key": <entityID>_<fieldID>_<field>,
            "name": <fieldID>_<field>,
            "value": <value>,
            "time": <time>
        }
    ]
}
```

> **Note**
>
> If the gateway has custom attribute reporting enabled (off by default), the JSON object configured in the machine's `customAttributes` is merged into the top level of the published message in all of the modes above. This only applies to machine data; group data is unaffected. See 2.9.3. Machine Configuration Parameters for `customAttributes`.

Key Concepts

1. `<entityID>` Machine or group identifier: if the current data belongs to machine data, `<entityID>` is the machine identifier machineID; if the current data belongs to group data, `<entityID>` is the group identifier groupID. See 4.1. Basic Description for machineID and groupID.
2. `<alias>` Machine or group name: the user-defined machine name or group name (see 3.3.1. Adding a Machine and 3.4.1. Adding a Group).
3. `<type>` Class, `<field>` field, and `<tag>` tag: a **class** contains at least one **field** and any number of **tags**. A **field** is the name of a piece of data. When a **class** cannot be distinguished by machine or group alone, we need to add a **tag** to that data type. For example, the **class** SpindleOverload (spindle overload data) contains the **field** CumulativeTime (cumulative overload time) and the **tag** SpindleNo (spindle number).
4. The **field form of a tag** combines its abbreviation and value. For example, S1 represents SpindleNo=1; field forms of multiple tags are joined with underscores.
5. `<fieldID>` Field identifier: if a **class** contains no **tags**, the **field identifier** is the same as the **class**. If a **class** contains **tags**, the **field identifier** is composed of the **class** and the **field forms of the tags**. The **class** and the multiple **tag field forms** are separated by underscores. The order of the **tags** in the field form is determined by the tag's sequence number described below. For example, the **field identifier** SpindleOverload_S1 represents the overload data of the first spindle, where S1 is the field form of SpindleNo=1; the **field identifier** ToolLife_G1_I2 represents the life data of the 2nd tool in tool group 1, where G1_I2 is the field form of GroupNum=1;Index=2.
6. `<time>` Timestamp: indicates the time when the task was executed, in milliseconds.
7. `<value>` Value: a numeric value or content.
8. In Default and MKT mode, each message contains one data class `<type>` from one machine or group `<entityID>`. TB, TB2, IoTDA, Brm, GeWu, and WisIoT can combine data from different classes and entities in one message.

Message Examples

Examples 1–3 are messages in Default mode.

Example 1

```
{
  "type": "CNCStatus",
  "unixtimestamp": 1698908463135,
  "data": {
    "cncStatus": "MANUAL",
    "alarmStatus": "NO_ALARM"
  },
  "machineID": "10"
}
```

Line 2, `"type": "CNCStatus",`

`<type>` Data class: using “CNCStatus” as the keyword, you can find it in the `<type>` data class table in 4.2. Data Description, which leads to 4.2.4. CNCStatus: Machine Status, identifying the data in this message as machine status data.

Line 3, `"unixtimestamp": 1698908463135,`

`<unixtimestamp>` Timestamp: the value 1698908463135 is the timestamp of this message.

Lines 4 to 7,

```
"data": {
    "cncStatus": "MANUAL",
    "alarmStatus": "NO_ALARM"
},
```

`<data>` Data content: you can look up the explanation of the corresponding content through the data class declared by `<type>`. Here, looking up 4.2.4. CNCStatus: Machine Status, “cncStatus” and “alarmStatus” are `<field>` data fields under the CNCStatus data class, indicating that the CNC operating status is a setup status: general setup, and the alarm status is currently no alarm.

Line 8, `"machineID": "10"`

`<entityID>` Machine or group identifier: if the current data belongs to machine data, `<entityID>` is the machine identifier machineID; if the current data belongs to group data,

`<entityID>` is the group identifier groupID. See 4.1. Basic Description for machineID and groupID.

The content of line 8 indicates that the data in this message comes from the machine with machine identifier 10.

Example 2

```
{
  "type": "GroupCount",
  "unixtimestamp": 1698911434710,
  "data": {
    "cumulativeCount": 233
  },
  "groupID": "g1"
}
```

Line 2, `"type": "GroupCount",`

`<type>` Data class: using "GroupCount" as the keyword, you can find it in the `<type>` data class table in 4.2. Data Description, which leads to 4.2.11. GroupCount: Group Machining Count Data, identifying the data in this message as group machining count data.

Line 3, `"unixtimestamp": 1698911434710,`

`<unixtimestamp>` Timestamp: the value 1698911434710 is the timestamp of this message.

Lines 4 to 6,

```
"data": {
    "cumulativeCount": 233
},
```

`<data>` Data content: you can look up the explanation of the corresponding content through the data class declared by `<type>`. Here, looking up 4.2.11. GroupCount: Group Machining Count Data, "cumulativeCount" is a `<field>` data field under the GroupCount data, representing the group's cumulative machining count.

Line 7, `"groupID": "g1"`

`<entityID>` Machine or group identifier: here it is the group identifier groupID.

The content of line 7 indicates that the data in this message comes from the group with group identifier g1.

Example 3

```
{
  "type": "ToolLife",
  "unixtimestamp": 1698908397751,
  "data": {
    "timeLimit": 1800,
    "currentTime": 1620,
    "prewarningTime": 1680
  },
  "toolNum": 9,
  "toolOffsetNum": 2,
  "machineID": "10"
}
```

Line 2, `"type": "ToolLife",`

`<type>` Data class: using "ToolLife" as the keyword, you can find it in the `<type>` data class table in 4.2. Data Description, which leads to 4.2.25. ToolLife: Tool Life Data, identifying the data in this message as tool life data.

Line 3, `"unixtimestamp": 1698908397751,`

`<unixtimestamp>` Timestamp: the value 1698908397751 is the timestamp of this message.

Lines 4 to 8,

```
"data": {
    "timeLimit": 1800,
    "currentTime": 1620,
    "prewarningTime": 1680
},
```

`<data>` Data content: you can look up the explanation of the corresponding content through the data class declared by `<type>`. Here, looking up the `<field>` data fields in 4.2.25. ToolLife: Tool

Life Data, “timeLimit” is the life limit [seconds], “currentTime” is the current life [seconds], and “prewarningTime” is the pre-warning life [seconds].

Lines 9 and 10, `"toolNum": 9,` and `"toolOffsetNum": 2,`

You can look up the explanation of the corresponding content through the data class declared by `<type>`. Here, looking up the `<tag>` data fields in 4.2.25. ToolLife: Tool Life Data, “toolNum” is the tool number, and “toolOffsetNum” is the tool offset number.

The content of lines 9 and 10 indicates that the tool offset data in this message belongs to tool 9, offset 2.

Line 11, `"machineID": "10"`

`<entityID>` Machine or group identifier: here it is the machine identifier machineID, representing the machine that the current data belongs to.

The content of line 11 indicates that the data in this message comes from the machine with machine identifier 10.

Example 4, MKT Mode

```
{
  "unixtimestamp": 1698908397751,
  "10_ToolLife_T9_O2_timeLimit": 1800,
  "10_ToolLife_T9_O2_currentTime": 1620,
  "10_ToolLife_T9_O2_prewarningTime": 1680
}
```

Line 2, `"unixtimestamp": 1698908397751,`

`<unixtimestamp>` Timestamp: the value 1698908397751 is the timestamp of this message.

Line 3, `"10_ToolLife_T9_O2_timeLimit": 1800,`

Its format is `"<entityID>_<fieldID>_<field>": <value>,`

where 10 is the `<entityID>` machine or group identifier: here it is the machine identifier machineID, representing the machine that the current data belongs to.

ToolLife_T9_O2 is the `<fieldID>` data field identifier, where ToolLife is the `<type>` data class; using “ToolLife” as the keyword, you can find it in the `<type>` data class table in 4.2. Data

Description, which leads to 4.2.25. ToolLife: Tool Life Data, identifying the data in this message as tool life data. T9 and O2 are combinations of `<tag>` data tag abbreviations and tag values; looking up the `<tag>` data tags in 4.2.25. ToolLife: Tool Life Data, T is the abbreviation for toolNum (tool number) and O is the abbreviation for toolOffsetNum (tool offset number), indicating that the tool offset data in this message belongs to tool 9, offset 2.

timeLimit is the `<field>` data field; looking up the `<field>` data fields in 4.2.25. ToolLife: Tool Life Data, “timeLimit” is the life limit [seconds].

Summary of this line: tool life data for tool 9, offset 2, from the machine with machine identifier 10; the life limit is 1800 seconds.

Line 4, `"10_ToolLife_T9_O2_currentTime": 1620,`

Similar to line 3; summary of this line: tool life data for tool 9, offset 2, from the machine with machine identifier 10; the current life is 1620 seconds.

Line 5, `"10_ToolLife_T9_O2_prewarningTime": 1680`

Similar to line 3; summary of this line: tool life data for tool 9, offset 2, from the machine with machine identifier 10; the pre-warning life is 1680 seconds.

Example 5, TB Mode

```
{
  "机台10": [ {
    "ts": 1698908397751,
    "values": {
      "ToolLife_T9_O2_timeLimit": 1800,
      "ToolLife_T9_O2_currentTime": 1620,
      "ToolLife_T9_O2_prewarningTime": 1680
    }
  } ]
}
```

Line 2, `"机台10": [ {`

Its format is `"<alias>": [ {`,

where 机台10 is the `<alias>` machine or group name, set by the user in the gateway’s machine configuration page, in the Add/Edit Machine (Group) dialog; if not set, the alias defaults to being

the same as the entityID machine or group identifier.

Lines 3 to 6, `"ts": 1698908397751,`

`<ts>` Timestamp: the value 1698908397751 is the timestamp of this message.

Lines 4 to 8,

```
"values": {
    "ToolLife_T9_O2_timeLimit": 1800,
    "ToolLife_T9_O2_currentTime": 1620,
    "ToolLife_T9_O2_prewarningTime": 1680
}
```

`<values>` Data content. Taking line 5 as an example:

`"ToolLife_T9_O2_timeLimit": 1800,`

Its format is `"<fieldID>_<field>": <value>,`

ToolLife_T9_O2 is the `<fieldID>` data field identifier, where ToolLife is the `<type>` data class; using "ToolLife" as the keyword, you can find it in the `<type>` data class table in 4.2. Data Description, which leads to 4.2.25. ToolLife: Tool Life Data, identifying the data in this message as tool life data. T9 and O2 are combinations of `<tag>` data tag abbreviations and tag values; looking up `<tag>` in 4.2.25. ToolLife: Tool Life Data, you can find that T is the abbreviation for toolNum (tool number) and O is the abbreviation for toolOffsetNum (tool offset number), indicating that the tool offset data in this message belongs to tool 9, offset 2.

timeLimit is the `<field>` data field; looking up the `<field>` data fields in 4.2.25. ToolLife: Tool Life Data, "timeLimit" is the life limit [seconds].

The content of this line: tool life data for tool 9, offset 2; the life limit is 1800 seconds.

Similarly, the content of line 6: tool life data for tool 9, offset 2; the current life is 1620 seconds. The content of line 7: tool life data for tool 9, offset 2; the pre-warning life is 1680 seconds.

Example 6, IoTDA Mode

```
{
  "devices": [
    {
```

```
      "device_id": "10",
      "services": [
        {
          "service_id": "ToolLife_T9_O2",
          "properties": {
            "timeLimit": 1800,
            "currentTime": 1620,
            "prewarningTime": 1680},
          "event_time": 1698908397751
        }
      ]
    }
  ]
}
```

Line 4, `"device_id": "10",`

Its format is `"device_id": <entityID>`,

where 10 is the `<entityID>` machine or group identifier: here it is the machine identifier machineID, representing the machine that the current data belongs to. The content of this line indicates that the data in this message comes from the machine with machine identifier 10.

Line 7, `"service_id": "ToolLife_T9_O2",`

Its format is `"service_id": <fieldID>`. ToolLife is the data class, while T9 and O2 identify tool 9 and offset 2. See the tags in 4.2.25. ToolLife: Tool Life Data.

Lines 8 to 11,

```
"properties": {
    "timeLimit": 1800,
    "currentTime": 1620,
    "prewarningTime": 1680},
```

The tags are already encoded in `service_id`. The keys in `properties` remain the original field names, without a T9_O2 prefix or separate tag fields.

timeLimit is the `<field>` data field; looking up the `<field>` data fields in 4.2.25. ToolLife: Tool Life Data, “timeLimit” is the life limit [seconds], “currentTime” is the current life [seconds], and “prewarningTime” is the pre-warning life [seconds].

Summary of this section: tool life data for tool 9, offset 2; the life limit is 1800 seconds, the current life is 1620 seconds, and the pre-warning life is 1680 seconds.

Line 12, `"event_time": 1698908397751`

Its format is `"event_time": <time>`. The value 1698908397751 is the timestamp of this message.

Example 7, GeWu Mode

Suppose machine `MACHINE_01` reports ToolLife data for tool 9, offset 2, and its product key is `machineProduct`. The gateway publishes this message to `$sys/gatewayProduct/gateway_0001/subdevice/propertyEvent/pub`:

```
{
  "messageId": "12345",
  "params": {
    "subDevices": [
      {
        "productKey": "machineProduct",
        "deviceKey": "MACHINE_01",
        "properties": [
          {
            "key": "ToolLife_T9_O2",
            "value": {
              "timeLimit": 1800,
              "currentTime": 1620,
              "prewarningTime": 1680
            },
            "ts": "1698908397751"
          }
        ],
        "events": []
      }
    ]
  }
}
```

Here `deviceKey` comes from the machine’s `machineID`, T9_O2 in `key` identifies the tool and offset, and the three keys in `value` remain the original ToolLife field names.

7.2. RPC Interface

The gateway supports an RPC interface based on the MQTT protocol. Users must configure the RPC request topic and RPC reply topic on the gateway management page; see 3.6.3. MQTT Configuration for details. Once this is configured, the gateway listens on the RPC request topic and publishes the result of the RPC request to the RPC reply topic. The RPC interface data encoding is the encoding configured in the MQTT settings.

If you need to embed the RPC request identifier in the topic, use the wildcard `${request_id}` at the position where the identifier should be embedded.

For example, in the examples in this section, the RPC request topic is set to `request/${request_id}` and the RPC reply topic is set to `response/${request_id}`. With this configuration, when the gateway receives a request on `request/24`, it publishes the execution result to the topic `response/24`.

It is recommended to place `${request_id}` at the end of the topic. If `${request_id}` needs to be placed in the middle of the topic, it must be preceded by `/` as a separator.

The gateway' s MQTT protocol supports multiple message modes, including Default, MKT, Brm, TB, TB2, and others.

RPC Request Message Format

Data formats for RPC requests in the different modes:

Default/MKT/Brm Mode

```
{
  "id": <request_id>,
  "method": "<method>",
  "params": <params>
}
```

> **Note**
> In Default/MKT/Brm mode, the `params` field must be present and must be a JSON object (or a string containing a JSON object); a request whose `params` is missing, null, an array or a scalar is discarded by the gateway, which only logs a `GENERAL_API_INVALID_REQUEST` error and publishes no reply message. Commands that take no input parameters (such as `settings`, `users`, `time`, and `service-status`) must still send `"params": {}`. In TB mode, a missing or null `params` is replaced with an empty object, and TB2 mode accepts JSON null.

TB/TB2 Mode

```
{
  "device": "<deviceName>",
  "data": {
    "id": "<request_id>",
    "method": "<method>",
    "params": <params>
  }
}
```

WisIoT/IoTDA Mode

WisIoT/IoTDA mode uses the same request message format as Default/MKT/Brm mode.

Where `<request_id>` is the request identifier; `<method>` is the RPC command; `<params>` is the input parameters of the RPC command; `<deviceName>` is the machine name.

> **Note**
> In TB/TB2 mode, the parameters “machineID” and “groupID” in `<params>` do not need to be provided; the gateway overrides both from `<deviceName>`, so the machine and the machine group are instead specified by `<deviceName>`. In TB2 mode, `<deviceName>` is used directly as the entity ID, and a request whose `<deviceName>` is not a known machine or machine group entity ID is discarded: the gateway only logs a `GENERAL_API_INVALID_REQUEST` error and publishes no reply message.

RPC Reply Message Format

Message formats for RPC replies in the different modes:

Default/MKT/Brm Mode

```
{
  "id": <request_id>,
  "data": <response>
}
```

TB/TB2 Mode

```
{
  "device": "<deviceName>",
  "id": <request_id>,
  "data": <response>
}
```

WisIoT/IoTDA Mode

WisIoT/IoTDA mode uses the same reply message format as Default/MKT/Brm mode.

Where `<response>` is the return result of the RPC command; `<deviceName>` is the machine name; `<request_id>` is the custom request identifier.

Supported RPC Commands

The gateway currently supports the following RPC commands:

Data Read/Write Interface - Direct Read/Write, Prefix /cnc/

RPC Command	Description
readAlarm	Read alarm information
readCNCStatus	Read machine status
readCNCStatusDetails	Read machine status details

RPC Command	Description
readCount	Read machining count
readCurrentToolNumber	Read current tool number
readEnergyConsum	Read energy consumption data
readFeed	Read feed rate data
readFeedAndSpindle	Read feed rate and spindle speed data
readLaserPower	Read laser power
readLoad	Read load data
readLog	Read log information
readOffsetData	Read tool offset data
batchReadOffsetData	Batch read tool offset data
writeOffsetData	Write tool offset data
readPosition	Read position data
readProgramBlock	Read current program block
readProgramInfo	Read current program information
readPlcData	Read PLC data
writePlcData	Write PLC data
readTimeData	Read machine time data
readToolLife	Read tool life
batchReadToolLife	Batch read tool life
readToolLifeDetails	Read tool life details
batchReadToolLifeDetails	Batch read tool life details

Data Read/Write Interface - Cached Read, Prefix /cnc/

RPC Command	Description
readTaskData	Read machine task data
batchReadTaskData	Batch read machine task data
readGroupTaskData	Read machine group task data
batchReadGroupTaskData	Batch read machine group task data
batchReadErrors	Batch read machine connection status
readErrorSummary	Read status summary

File Management Interface, Prefix /cnc/

RPC Command	Description
readProgramList	Read machine file list
receiveFile	Receive file from machine
readCurrentProgram	Receive the file currently running on the machine
sendFile	Send file to machine
batchSendFile	Batch send files to machine
deleteFile	Delete machine file
batchDeleteFile	Batch delete machine files
lockFileByRange	Lock/unlock machine program editing
createDir	Create machine directory
deleteDir	Delete machine directory
selectProgram	Select program
readAllPrograms	Browse all files
searchFile	Search machine files

Machine Data Analysis Interface, Prefix /analysis/

RPC Command	Description
oee	Machine OEE data analysis
alarm	Machine alarm data analysis
count	Machine count data analysis
overall	Machine overall data analysis
cycle	Machine cycle time data analysis

Machine Group Data Analysis Interface, Prefix /group-analysis/

RPC Command	Description
oee	Machine group OEE data analysis
alarm	Machine group alarm data analysis
count	Machine group count data analysis
overall	Machine group overall data analysis

RPC Command	Description
cycle	Machine group cycle time data analysis

Historical Data Interface, Prefix /db/

RPC Command	Description
machine	Machine historical data
group-machine	Historical data for all machines in the machine group
query	Query historical data

Global Configuration Interface, Prefix /config/

RPC Command	Description
gateway-info	Get gateway information
settings	Get gateway global settings
update-settings	Modify gateway global settings
security	Get gateway global security settings
update-security	Modify gateway global security settings
backup	Back up gateway configuration
restore	Restore gateway configuration
reset	Reset gateway configuration

User Configuration Interface, Prefix /config/

RPC Command	Description
user	Get user settings
users	Get all user settings
create-user	Create new user
update-user	Modify user settings
delete-user	Delete user
user-security	Get user security settings
update-user-security	Modify user security settings

Machine Configuration Interface, Prefix /config/

RPC Command	Description
machine	Get machine configuration
machines	Get all machine configurations
create-machine	Add machine configuration
update-machine	Modify machine configuration
delete-machine	Delete machine configuration
batch-delete-machines	Batch delete machine configurations

Machine Group Configuration Interface, Prefix /config/

RPC Command	Description
group	Get machine group configuration
groups	Get all machine group configurations
create-group	Add machine group configuration
update-group	Modify machine group configuration
delete-group	Delete machine group configuration
batch-delete-groups	Batch delete machine group configurations

Task Configuration Interface, Prefix /config/

RPC Command	Description
task-manager-settings	Get task manager settings
update-task-manager-settings	Modify task manager settings
machine-task-interval-settings	Get machine task interval settings
update-machine-task-interval-settings	Modify machine task interval settings
group-task-interval-settings	Get machine group task interval settings
update-group-task-interval-settings	Modify machine group task interval settings
oee-monitoring-settings	Get OEE monitoring settings
update-oee-monitoring-settings	Modify OEE monitoring settings
tool-life-monitoring-settings	Get tool life monitoring settings
update-tool-life-monitoring-settings	Modify tool life monitoring settings

RPC Command	Description
count-monitoring-settings	Get machining count monitoring settings
update-count-monitoring-settings	Modify machining count monitoring settings
overload-monitoring-settings	Get overload monitoring settings
update-overload-monitoring-settings	Modify overload monitoring settings
alarm-monitoring-settings	Get alarm monitoring settings
update-alarm-monitoring-settings	Modify alarm monitoring settings
machine-status-monitoring-settings	Get machine status monitoring settings
update-machine-status-monitoring-settings	Modify machine status monitoring settings

Communication Configuration Interface, Prefix /config/

RPC Command	Description
cloud-settings	Get cloud platform settings
update-cloud-settings	Modify cloud platform settings
modbus-settings	Get MODBUS settings
update-modbus-settings	Modify MODBUS settings
mqtt-settings	Get MQTT settings
update-mqtt-settings	Modify MQTT settings
database-settings	Get database settings
update-database-settings	Modify database settings
gateway-file-server-settings	Get gateway file server settings
update-gateway-file-server-settings	Modify gateway file server settings
http-settings	Get HTTP settings
update-http-settings	Modify HTTP settings
hub-settings	Get Hub settings
update-hub-settings	Modify Hub settings
remote-access	Get remote access settings
update-remote-access	Modify remote access settings

Core Service Function Interface, Prefix /core/

RPC Command	Description
info	Get Core service information
license-info	Get license information
service-status	Get service status
upload-license	Upload gateway license
settings	Get Core service settings
log-level	Switch log level
need-restart	Check whether the Core service needs to be restarted
drop-service-queue	Discard a service’s send queue

Gateway Service Function Interface, Prefix /gateway/

RPC Command	Description
alias	Get gateway name
update-alias	Modify gateway name
reboot	Reboot gateway hardware
restart	Restart all services
restart-service	Restart Core service
shut-down	Shut down gateway
internet-connection	Get gateway network status
hardware-resources	Get gateway hardware resources
time	Get gateway current time
sync-time	Sync gateway time
time-zone	Get gateway time zone
time-zones	Get time zone options
update-time-zone	Modify gateway time zone
network-adapters	Get gateway network adapter list
ping	Ping network diagnostic
telnet	Telnet port connectivity diagnostic
traceroute	Traceroute diagnostic
scan-ports	Scan ports
lookup-dns	DNS lookup

RPC Command	Description
lan	Get wired network settings
update-lan	Modify gateway wired network settings
wifi	Get wireless network settings
update-wifi	Modify wireless network settings
search-wifi	Search for wireless networks
connect-wifi	Connect to wireless network
disconnect-wifi	Disconnect wireless network
static-routing	Get static routing settings
update-static-routing	Modify static routing settings
connect-remote-host	Connect to remote server
disconnect-remote-host	Disconnect from remote server
file-server-items	Get gateway file server list
delete-file-server-item	Delete gateway file server item

Log Interface, Prefix /log/

RPC Command	Description
access	Read access log
error	Read error log

Except for the authentication interface and the stream-returning interfaces (`sendFileStream`, `receiveFileStream`, `backupFiles`, `/log/core`, `/log/gateway`), which are not available over RPC, RPC commands correspond one-to-one with the corresponding interfaces in 5. HTTP Communication; input parameters and return results can be found in the interface descriptions in 5. HTTP Communication.

Examples

Example 1: sendFile - Send File to Machine

In Default mode, to send the ASCII-encoded file content “fileContent” to the Dir/Prog directory of the machine with machineID 1010 and save it as 2222, use the request topic:

“request/1”, where 1 is the custom request identifier. The request message, including the RPC command and parameters, is as follows:

```
{
  "method": "sendFile",
  "params": {
    "machineID": "1010",
    "fileName": "2222",
    "dirAtCNC": "Dir/Prog",
    "data": {
      "content": "fileContent",
      "encoding": "us-ascii"
    }
  }
}
```

Where machineID, fileName, and dirAtCNC are the request parameters in the URL of the HTTP interface 5.6.6. sendFile - Send File to Machine, embedded directly in the “params” field. The POST request body should be embedded in the “data” field within “params”.

Reply topic “response/1”; the reply message is as follows:

```
{
  "data": {
    "errorCode": 0,
    "errorMsg": "Success"
  }
}
```

Example 2: writeOffsetData - Write Tool Offset Data

In Default mode, to write tool offset data for offset number 1, use the request topic: “request/2”, where 2 is the custom request identifier. The request message, including the RPC command and parameters, is as follows:

```
{
  "method": "writeOffsetData",
  "params": {
    "machineID": "1010",
```

```
    "offsetNum": 1,
    "data": {
      "toolNose": 3,
      "lengthWear": 0.0001,
      "radiusWear": 0.03,
      "lengthGeom": 0.0,
      "radiusGeom": 0.002
    }
  }
}
```

Where machineID and offsetNum are parameters in the URL of the HTTP interface 5.5.1.16. writeOffsetData - Write Tool Offset Data, embedded directly in the “params” field. The POST request body should be embedded in the “data” field within “params”.

Reply topic “response/2”; the reply message is as follows:

```
{
  "data": {
    "errorCode": 0,
    "errorMsg": "Success"
  }
}
```

8. Database Communication

After enabling database communication and turning on the automatic data collection task for the target machine/group (see 3.3.1.2. Task Settings, and 3.4.1.2. Group Task Settings), the gateway will automatically write the data collection task results to the specified database. Currently supported database types include InfluxDB v2.x, MySQL, SQL Server, PostgreSQL, and others. Refer to 3.6.4. Database Configuration for instructions on configuring the database service.

Database table names take the form [table prefix][data class], where the table prefix is set on the gateway’s communication configuration page (see 3.6.4. Database Configuration); the table name is the same as the `type` of the data class used in MQTT communication (see 4.2. Data Description). The table below shows the names of each database table (using a MySQL database with no table prefix set and log storage mode as an example):

Table Name	Data Class
alarmhistory	AlarmHistory
alarmlog	AlarmLog
axialoverload	AxialOverload
cncstatus	CNCStatus
count	Count
currenttoolnumber	CurrentToolNumber
cycle	Cycle
energyconsum	EnergyConsum
feed	Feed
feedandspindle	FeedAndSpindle
groupcount	GroupCount
groupcumulativetime	GroupCumulativeTime
groupoee	GroupOEE
laserpower	LaserPower
load	Load
loghistory	LogHistory
machine	— (Machine information table)
oee	OEE

Table Name	Data Class
offset	Offset
plc	PLC
position	Position
programblock	ProgramBlock
programinfo	ProgramInfo
spindleoverload	SpindleOverload
timedata	TimeData
toollife	ToolLife

The fields in each database table are the same as the tags and fields of the corresponding data class `type` (see 4.2. Data Description). The table below shows the fields of the `cncStatus` table (using a MySQL database with no table prefix set and log storage mode as an example):

#	Name	Data Type	Length/Decimal Point
1	cncStatus	TEXT	
2	adjustedStatus	TEXT	
3	alarmStatus	TEXT	
4	alarmLevel	TEXT	
5	offTime	BIGINT	20
6	waitTime	BIGINT	20
7	emergencyTime	BIGINT	20
8	autoRunTime	BIGINT	20
9	manualTime	BIGINT	20
10	channel	VARCHAR	128
11	machineID	VARCHAR	128
12	time	DATETIME	3
13	uid	VARCHAR	128

> **Note**
>
> The table above does not include an `id` primary key column. An auto-increment `id` primary key column is created as the first column only when the "Enable primary key" option is switched on in the database configuration (off by default); its SQL data type depends on the database type: `bigint` for MySQL and SQL Server, `bigserial` for PostgreSQL, and `integer` for SQLite.

9. MCP Service

The gateway has a built-in MCP (Model Context Protocol) server that exposes machine data, data analysis, and gateway status as "tools" for AI clients (Claude Code, Claude Desktop, MCP Inspector, and any other MCP-capable application or agent), so machines can be queried in natural language without writing HTTP requests by hand.

Internally, MCP tools go through exactly the same processing pipeline as the HTTP interfaces. Passthrough tools (get_machine, read_tool_life, list_programs, read_task_data) return exactly what the corresponding HTTP interface returns; composite tools (get_fleet_status, read_machine, analyze, query_history, get_system_info, ...) call several interfaces in one go and merge or wrap the results as described in 9.5. Results and Errors. For field definitions, see the sections under 5. HTTP Communication.

9.1. Basics

The MCP server endpoint is **/mcp**, served on the same port as the HTTP interfaces:

```
http://{Gateway IP}/mcp
```

> **Caution**
>
> The MCP service is **disabled by default**; while it is off, `/mcp` returns HTTP 404. Turn the switch on under 3.6.8. MCP Settings and save; it takes effect immediately, with no restart of the gateway or its services.

The transport is the MCP standard Streamable HTTP: the client POSTs JSON-RPC messages with the headers `Content-Type: application/json` and `Accept: application/json, text/event-stream`. The server is stateless — every tool call is an independent HTTP request.

13 **read-only** tools are currently provided, covering machine configuration, live status, cached snapshots, data analysis, and gateway system information; see 9.4. Tool List. No MCP tools are provided for interfaces that write data, control machines, or transfer files — those remain available through the HTTP interfaces only.

Designed for agents:

- The `initialize` response carries server instructions: the recommended call order, time parameter forms, the meaning of machine status values, common error codes with what

to do about them, and the documentation URL.

- Every tool is annotated read-only and idempotent (`readOnlyHint`, `idempotentHint`), so clients that honour the annotations can call them without confirmation.
- All time parameters accept several forms (see 9.4. Tool List); results echo the time window resolved in the gateway’s time zone.
- Error messages carry a hint, and composite tools report errors per section instead of failing as a whole.

> **Note**
> The cloud platform (Hub) also provides an MCP server at `https://{cloud platform address}/mcp`. Its tool list is identical to the gateway’s; calls are forwarded by the cloud platform to the designated gateway for execution. For authentication, see 9.2.2. Cloud Platform Authentication.

9.2. Authentication

9.2.1. Gateway Authentication

The MCP server uses the same authentication as the HTTP interfaces; see 5.3. Authentication Methods. Send `Authorization: Bearer <token>` in the request headers, where the token is either a token obtained via 5.3.1. JWT Method or a key generated via 5.3.2. Secret Key Method (an administrator generates it for the user under 3.12.7.2. User Security Settings). If no credentials or invalid credentials are supplied, HTTP 401 is returned.

> **Caution**
> A JWT token is valid for 24 hours; once it expires, the MCP client keeps failing every call. MCP client configurations are usually kept long-term, so **a permanent secret key is recommended**.

The other access controls apply as well:

- **IP whitelist**: when enabled under 3.6.6. HTTP Settings, the source IP of a `/mcp` request must be in the whitelist, following the same rules as `/api`; see 5.3.3. IP Whitelist.
- **Per-user authorized APIs**: when Security Control is enabled, administrators may call every tool, while other users may only call the interfaces covered by their authorized API list (each interface a tool calls is matched individually, using the same configuration as HTTP requests). In a composite tool an unauthorized section is reported under `errors` as

`Forbidden: ...` while the other sections still return; a single-interface tool returns an error for the call.

- With security control disabled, tools can be called without user authentication (the IP whitelist still applies).

9.2.2. Cloud Platform Authentication

When calling the cloud platform’s MCP server, send `accessToken: <gateway token>` in the request headers — the same rule the cloud platform’s HTTP proxy uses: the token is both the credential and the selector for which gateway executes the tool. The gateway token is the one entered under 3.6.1. Cloud Platform; for the interface see 5.9.6.1. cloud-settings - Get Cloud Platform Settings.

If the token is missing or not registered, the tool call returns `Unauthorized: missing or unknown accessToken header.`; if the token is valid but the corresponding gateway is not currently connected to the cloud platform, it returns `GATEWAY_SERVICE_UNAVAILABLE`.

9.3. Client Configuration

Using Claude Code as an example, add the gateway MCP server:

```
claude mcp add --transport http bivrost-gateway
http://192.168.100.1/mcp --header "Authorization: Bearer <secret key>"
```

Add the cloud platform MCP server:

```
claude mcp add --transport http bivrost-hub
https://cloud.example.com/mcp --header "accessToken: <gateway token>"
```

Other MCP clients generally use a JSON configuration of the following form:

```
{
  "mcpServers": {
    "bivrost-gateway": {
      "type": "http",
```

```
      "url": "http://192.168.100.1/mcp",
      "headers": {
        "Authorization": "Bearer <secret key>"
      }
    }
  }
}
```

To verify directly that the server is reachable, list all tools with a JSON-RPC request:

```
curl -X POST http://192.168.100.1/mcp \
  -H "Authorization: Bearer <secret key>" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -d '{"jsonrpc":"2.0","id":1,"method":"tools/list"}'
```

9.4. Tool List

Parameters marked "optional" below may be omitted, in which case the default is used. For the meaning of machineID and groupID see 4.1.1. Basics; they can be obtained with `list_machines` and `list_groups`.

Time parameters (`start`, `end`) accept the following forms and are resolved in the gateway's time zone: a Unix timestamp (seconds; 13 digits are treated as milliseconds); ISO-8601 with an offset (`2026-09-17T08:00:00+08:00`); ISO-8601 or a bare date without an offset (`2026-09-17T08:00`, `2026-09-17`, interpreted as gateway local time); a relative offset (`-30m`, `-24h`, `-7d`, `-2w`); or the keywords `now`, `today`, `yesterday` (the last two mean midnight of that day). `end` defaults to now.

Recommended call order: `list_machines` → `get_fleet_status` (plant-wide overview) → `read_machine` / `analyze` / `query_history` (details of one machine).

9.4.1. Configuration

Tool	Description	Parameters	Interface
list_machines	List every machine’s identity fields: machineID, name, system, model, machineType, ip, port, isActive	none	machines
list_groups	List every machine group: groupID, name, isActive, member machineIDs	none	groups
get_machine	Full configuration of one machine (credential fields excluded)	machineID	machine

9.4.2. Machine Status

Tool	Description	Parameters	Interface
get_fleet_status	The latest cached status and connection health of every machine (or of one group’s members) in one call, without contacting the machines. Per machine: connection (errorCode 0 = communication OK), status (cncStatus, adjustedStatus, alarmStatus, alarmLevel, mode, programStatus, time) and a derived state (running / idle / alarm / setup / offline / noData / inactive), plus summary counts. When the connection is down the state is offline and the cached status is flagged statusStale. Requires the machine status task; without it the state is noData	groupID (optional)	machines, group, batchReadTaskData, batchReadErrors

Tool	Description	Parameters	Interface
read_task_data	The latest cached sample of one data class (no request to the machine). Machine classes include CNCStatus, OEE, Count, Cycle, TimeData, ToolLife; group classes are GroupCount, GroupCumulativeTime, GroupOEE. An invalid class returns an error listing the valid values	type, machineID / groupID (one of them), tag (optional)	readTaskData, readGroupTaskData
read_machine	Live read of several items from one machine. items may include status (status details), alarm (active alarms), position, load, feedAndSpindle, toolNumber (current tool), timeData, count; default status and alarm. Each item is returned under its own key; failed items are listed under errors	machineID, items (optional), channel (optional)	readCNCStatusDetails, readAlarm, readPosition, readLoad, readFeedAndSpindle, readCurrentToolNumber, readTimeData, readCount
read_tool_life	Read tool life; detailed=true returns the detailed record	machineID, toolNum / groupNum / offsetNum (all optional; omit to read all tools), detailed (optional)	readToolLife, readToolLifeDetails
list_programs	List the part program files on a machine	machineID, dirAtCNC / subDir / startPrgNo (all optional)	readProgramList
read_current_program	Read the currently running program (directory, file name, content); content longer than maxChars is cut and flagged with truncated and totalChars	machineID, dirAtCNC / subDir (both optional), maxChars (optional, default 20000, 0 = unlimited)	readCurrentProgram

9.4.3. Data Analysis

Tool	Description	Parameters	Interface
analyze	Statistics over a time window for a machine or a group; kind is one of oee, alarm, count, cycle, overall. The window must be shorter than 31 days; interval (seconds) splits it into sub-windows. Returns `{kind, machineID or groupID, window, data}`	kind, start, machineID / groupID (one of them), end (optional), interval (optional), enableCountPerProgram (optional, count only), sumByGroup (optional, group count only: the group’s total by count multiplier)	Machine analysis, Group analysis
query_history	Historical rows of a machine or of a group’s machines; type is a data class (e.g. CNCStatus, AlarmLog, AlarmHistory, Count, OEE, Heartbeat). Rows are oldest-first by default; newestFirst=true returns the most recent rows. Returns `{type, machineID or groupID, window, order, count, limitReached, data}`	type, start, machineID / groupID (one of them), end (optional), limit (optional, default 200, 0 = unlimited), newestFirst (optional)	machine, group-machine, query (newestFirst only)

9.4.4. System Information

Tool	Description	Parameters	Interface
get_system_info	One call returning gateway (identity and alias, without accessToken), core (version), license, services (service status and queues), time (current time, gateway time zone and offset) and hardware (resource usage); includeNetwork=true adds network (adapters). Failed sections are reported under errors	includeNetwork (optional)	gateway-info, info, license-info, service-status, time-zone, hardware-resources, network-adapters
get_error_log	Query the gateway’s interface error log; limit keeps the most recent rows. Returns `{window, total, returned, truncated, data}`	start (optional), end (optional), limit (optional, default 200)	`/api/log/error`

9.5. Results and Errors

On success, a passthrough tool returns the response body of the corresponding HTTP interface (a JSON string); see each interface’s section for field descriptions. `password` and `fileServerPassword` in machine configurations and `accessToken` in the gateway information are never returned through MCP.

Composite tools return a merged JSON object: `list_machines` and `list_groups` return compact rows; `get_fleet_status`, `read_machine` and `get_system_info` are organised by section, with failed sections reported as error messages under `errors` (or `statusError` / `connection.error`) while the other sections return normally; `analyze`, `query_history` and `get_error_log` echo the resolved time window under `window` (`startUnix`, `endUnix`, `start`, `end`, `timeZone`) and put the interface’s response under `data`.

On failure, a tool returns an MCP tool error (`isError`) whose message has the format:

```
{interface path} failed: {error code name}({errorCode}) {error
description}. Hint: {what to do next}
```

For example, querying a machine that does not exist:

```
/cnc/readCNCStatusDetails failed:
GENERAL_MACHINE_ID_NOT_EXISTED(10003). Hint: Unknown or inactive
machineID; call list_machines to get valid IDs.
```

The errorCode is exactly the same as that of the HTTP interfaces; see 5.2. Error Handling. Information-level errors that the HTTP interfaces return with status 200 (such as task not ready, 1303) are returned as tool errors through MCP as well. Parameter validation errors (an invalid data class or time form, machineID and groupID given together, ...) list the accepted values in the message.

9.6. Limitations

- All tools are **read-only**; they provide no way to modify configuration, control machines, or transfer files. Use the HTTP interfaces for write operations.
- The following interfaces have no MCP tools: interfaces that return file streams (file transfer, log download) and interfaces carrying sensitive information such as user configuration and security settings. The free-form historical data query interface is not exposed as a tool of its own; only `query_history`’s newestFirst option calls it with fixed filters (with security control enabled, the user must be authorized for `/api/db/query`).
- The amount of data returned depends on machine response and the time window; for analysis tools prefer a short time window, and for historical data queries use the limit parameter.
- If no physical machine tool is available, use a mock machine to test the MCP tools (added under 3.3.1. Adding a Machine).

10. Mock Machine Testing

If you need to quickly test the communication protocol without an actual machine tool connected, you can add a mock machine in the **Machine Configuration** section of the gateway management page: choose CNC machine tool as the machine type, set **System** to Mock and **Model** to General (mock data is determined by the selected system — only the Mock system returns mock data), then set its local IP to 127.0.0.X. Its machine ID and slave ID default to X, where X can be set from 1 to 255 (see 3.3. Machine Configuration). Activate the machine and enable its auto-collection option. Restart the service on the home page, and you can then test the HTTP protocol. To test additional communication protocols, enable the corresponding communication method and configure its parameters in the **Communication Configuration** section of the gateway management page, then restart the service on the home page to test it. In actual use, please delete the mock machine to avoid slave ID conflicts.

11. Appendices

This chapter has five parts: a glossary, a command format reference, the optional cluster failover feature, the supported-device list and the per-interface support tables:

- 11.1. Glossary: definitions and calculation methods for concepts such as Machine OEE Data, Group OEE Data and Group Part Count.
- 11.2. Command Format: the format of commands such as PLC Data Task, Precondition, Task Interval, Post-processing (DA / $POST$ / $MOCK$), External PLC Data Task, External Machines, Protected API and Authorized APIs.
- 11.3. Cluster Failover (Optional): how the shared standby pool works, how to build it, failover and switch-back behaviour, forced takeover, and what is not replicated.
- 11.4. Supported Devices: the device types, systems and models the gateway can connect to.
- 11.5. Interface Support: which interfaces and which fields each system and model supports, for HTTP, MQTT RPC and task collection alike.

Enabling, authenticating and configuring clients for the gateway’ s built-in MCP server is covered in 9. MCP Service.

11.1. Glossary

11.1.1. Machine OEE Data

Machine OEE (Overall Equipment Efficiency) data includes figures such as the machine’s Off time, Wait time, Emergency time, Autorun time and Manual time, as well as its Availability.

Take machine autorun time as an example: it is the time the machine spent in automatic operation during the monitoring window. The monitoring window is set under “Monitor Mode” in 3.5.3. OEE Monitoring Settings.

```
Machine Availability = Autorun Time / Total Time
```

Here, Autorun Time is the machine’s autorun time and Total Time is the monitoring window.

OEE data requires local caching, so turn on the “Local Caching” switch under 3.12.4.1. Settings - Options.

11.1.2. Group OEE Data

Group OEE data includes figures such as the group’s Off time, Wait time, Emergency time, Autorun time and Manual time, as well as its Availability.

Take group autorun time as an example: it is the sum of the time the group’s activated machines spent in automatic operation during the monitoring window. The monitoring window is set under “Monitor Mode” in 3.5.3. OEE Monitoring Settings.

```
Group Availability = Autorun Time / Total Time
```

Here, Autorun Time is the group’s autorun time and Total Time is the total monitoring window, that is, the monitoring window multiplied by the number of activated machines in the group.

OEE data requires local caching, so turn on the “Local Caching” switch under 3.12.4.1. Settings - Options.

11.1.3. Group Part Count

The group part count is the group’s cumulative part count. It is calculated as follows:

```
Group Part Count = (Machine 1 Cumulative Part Count - Machine 1
Initial Value) * Machine 1 Count Multiplier
                 + (Machine 2 Cumulative Part Count - Machine 2 Initial
Value) * Machine 2 Count Multiplier
                 + ...
```

The initial value is that machine’s cumulative part count the first time the group part count was collected after the machine joined the group. The group part count is stored on the gateway and bound to the group number, and is never reset.

The group part count requires local caching, so turn on the “Local Caching” switch under 3.12.4.1. Settings - Options.

11.2. Command Format

11.2.1. PLC Data Task

A PLC data task has the format below. Separate multiple reads with “;” .

```
Area,Start Address[|Tag],Data Count,Data Type[,Precondition,Task
Interval,Post-processing];
```

Area, Start Address, Data Count and Data Type are mandatory — together they define where the data comes from and what shape it has. Tag, Precondition, Task Interval and Post-processing are optional.

- **Tag** identifies the data in MQTT, database and other communication channels. If you do not set one, the gateway assigns it automatically from the task order and the data length.
- **Precondition** is the precondition for this command. It is evaluated first, and the command runs only if the condition is met; if it is not set, the command always runs. It works exactly like 3.3.1.2.1. Precondition. For the syntax, see 4.2.2. Precondition.
- **Task Interval** is the interval for this command. If it is not set, the gateway uses the matching interval from 3.5.1. Machine Task Interval Settings and 3.5.2. Group Task Interval Settings. For the syntax, see 4.2.3. Task Interval.
- **Post-processing** may appear more than once. Each post-processing expression transforms data returned by this task or by another task and publishes the result under a new tag. See 4.2.4. Post-processing.

Taking [CNC]Fanuc [Oi-F] as an example:

```
R,0,9,Byte;
R,100|Data2,6,Byte,$COND$CNCStatus_cncStatus = "AUTO_RUN"$COND$;
R,200|rawData0,1,Byte,$INTERVAL$1000$INTERVAL$,$POST$___PLC_TrawData0___[0]>
```

- `R,0,9,Byte` reads 9 values of type Byte starting at PLC address R0 (R0 included), with no tag set.
- `R,100|Data2,6,Byte` reads 6 values of type Byte starting at PLC address R100 (R100 included), tagged “Data2” , with the precondition `CNCStatus_cncStatus = "AUTO_RUN"` — that is, only while the machine is in automatic operation. For more on preconditions, see 4.2.2. Precondition.

- `R,200|rawData0,1,Byte` reads one value of type Byte at PLC address R200, tagged "rawData0". `$INTERVAL$1000$INTERVAL$` sets the interval for this command to 1000 milliseconds. `$POST$___PLC_TrawData0___[0]>0$POST$|Heartbeat` is a post-processing expression and its tag: it tests whether the value read into rawData0 is greater than 0 and publishes the result under the tag "Heartbeat". For more on post-processing, see 4.2.4. Post-processing.

Taking [PLC]Simatic [S300] as an example:

```
DB1,1,3,Byte;DB10,100|FloatData,10,Float;
```

- `DB1,1,3,Byte` reads 3 values of type Byte starting at PLC address DB1.1 (DB1.1 included), with no tag.
- `DB10,100|FloatData,10,Float` reads 10 values of type Float starting at PLC address DB10.100 (DB10.100 included), tagged "FloatData".

Some devices support reading data by variable name. Taking [PLC]Allen-Bradley [1783] as an example:

```
$TAG$|name=AB.C[0],,8,Float;
```

`$TAG$|name=AB.C[0],,8,Float` reads elements 0 through 7 of the Float array held in the PLC variable named AB.C, addressed by tag. Note that the field corresponding to Start Address is left empty in this form.

For a fuller description of Area, Start Address, Data Count, Data Type and the other parameters, see the request-parameter section of 5.5.1.20. readPlcData (Read PLC Data).

Consult the device manual or contact the device manufacturer to obtain the area and address of the target PLC data and the type of the target data.

11.2.2. Precondition

A precondition sets the prerequisite for running a task: the collection task runs only when the condition you set is met. If it is left blank, there is no precondition and the task runs directly.

For non-PLC data tasks you set the precondition in 3.3.1.2.1. Precondition. For PLC data tasks you add a precondition command inside the command itself, in this format:

```
$COND$Precondition$COND$
```

A precondition is essentially a logical test. A single precondition command looks like this:

```
a > 100              // variable a is greater than 100
b < 50.0f            // variable b is less than 50.0f
c = "AUTO_RUN"       // variable c is the string "AUTO_RUN"
d <> 0               // variable d is not equal to 0
e >= 12.34f          // variable e is greater than or equal to 12.34f
f <= g               // variable f is less than or equal to variable g
not h in (1,2,3)     // variable h is not one of (1,2,3)
```

Several precondition commands can be combined with and, or and (), for example:

```
a > 100 and (b < 50.0f or c = "AUTO_RUN")
```

The variables currently available for logical tests are `<type>_<tag>` and `prev_<type>_<tag>`, where `<type>` is the data class, `<tag>` is the data tag, and prev refers to the previous value of the data. For details on `<type>` and `<tag>`, see 4.2. Data Description.

11.2.3. Task Interval

The task interval sets the interval for a specific task. If it is not set, the gateway uses the matching interval from 3.5.1. Machine Task Interval Settings and 3.5.2. Group Task Interval Settings. The format is:

```
$INTERVAL$Interval$INTERVAL$
```

Interval is a number, in milliseconds.

11.2.4. Post-processing

Post-processing commands let you transform the raw data you collect. Two kinds are currently available:

11.2.4.1. DA Direct Access

DA takes a single value at a given position out of the data read by the command to its left and publishes it under a given tag. It cannot perform calculations, but it lets you merge PLC tasks whose addresses are adjacent, reducing the number of PLC tasks and improving execution efficiency. For example:

```
R,2036|count,5,Int32,
$DA$0$DA$|Part Count,
$DA$3$DA$|Part Count-Good,
$DA$4$DA$|Part Count-Reject;
```

This reads 5 Int32 values starting at R2036, giving an array of length 5. DA then takes element 0 and tags it Part Count, takes element 3 and tags it Part Count-Good, takes element 4 and tags it Part Count-Reject, and uploads these three values.

If a DA tag is not specified, the default tag is `<raw data tag>_<number>`, where the number is the position given to DA. Had the example above set no tags, the three values would default to `count_0`, `count_3` and `count_4`.

11.2.4.2. $POST$ Post-processing

$POST$ performs calculations, comparisons and similar processing on one or more values that have already been collected, and uploads the result.

The gateway keeps the most recent task data as variables. Variables belonging to the same machine can be referenced inside $POST$ using the format `___VariableName___` — three underscores before and after the variable name to mark its start and end. Take `___PLC_TrawData2_data___`: PLC is the data type, T is the abbreviation for tag, rawData2 is the data’s tag, and data is a field of the PLC data class, which is an array. Together the variable name is `PLC_TrawData2_data`, meaning the PLC data class, with tag T set to rawData2 and field data — that array. Adding three underscores at each end per the variable format gives `___PLC_TrawData2_data___`. For details on data classes, data tags and fields, see 4.2. Data Description. This mechanism lets you use, inside $POST$, any data collected by any enabled task.

The gateway keeps at most the two most recent sets of task data. The most recent set is addressed as described above; for the set before it, add the prefix `prev_` to the variable name, for example `___prev_PLC_TrawData2_data___`.

Example 1:

```
R,2000|rawData2,1,Byte,
$INTERVAL$900$INTERVAL$,
$POST$(___PLC_TrawData2_data___[0] and 1>0$POST$|Heartbeat;
```

Every 900 milliseconds, one Byte is read starting at R2000, giving an array of length 1 stored under the tag rawData2. Inside $POST$, the value at position 0 of that rawData2 array is taken; if it is greater than 0 the result is True, otherwise False, and the result is uploaded under the tag Heartbeat.

You can use several $POST$ expressions in a single PLC data task to perform different calculations, as in Example 2 and Example 3.

Example 2:

```
R,3012|rawData3,3,Int32,
$POST$___PLC_TrawData3_data___[0]*0.1f$POST$|Tool-to-Tool Change Time,
$POST$___PLC_TrawData3_data___[1]*0.1f$POST$|Total Tool Change Time,
$POST$___PLC_TrawData3_data___[2]*0.1f$POST$|Tool Machining Time;
```

The raw values are Int32 in units of [0.1 s]; each is multiplied by 0.1f to convert it to a Float in units of [s].

Example 3:

```
R,2003|rawData4,1,Byte,
$POST$(___PLC_TrawData4_data___[0] and (1<<4))>0$POST$|Data Ready,
$POST$(___PLC_TrawData4_data___[0] and (1<<5))>0$POST$|Part Data Ready,
$POST$(___PLC_TrawData4_data___[0] and (1<<6))>0$POST$|Part OK,
$POST$(___PLC_TrawData4_data___[0] and (1<<7))>0$POST$|Part NG;
```

The raw value is a single Byte in which each bit has a different meaning. Bit 4 of the Byte is taken (bit 0 being the least significant bit) and compared with 0, giving True or False, tagged Data Ready; bit 5 is tested for greater than 0 and tagged Part Data Ready; bit 6 is tested and tagged Part OK; bit 7 is tested and tagged Part NG.

Besides the usual +, -, *, /, and, or, <, >, = operators, the gateway also supports nested if tests. Example 4 derives the current status from a PLC value.

Example 4:

```
D,0|rawData0,1,Int16,
$POST$if(___PLC_TrawData0_data___[0] = 1, "AUTO_RUN",
if(___PLC_TrawData0_data___[0] = 2,"MANUAL_HOLD",
if(___PLC_TrawData0_data___[0] = 4, "EMERGENCY", "WAIT")))$POST$|Machine
Status,
$POST$if(___PLC_TrawData0_data___[0] = 4, "ALARM",
if(___PLC_TrawData0_data___[0] = 5, "ALARM", "NO_ALARM"))$POST$|Alarm
Status;
```

The value of D0 represents the state of the machine. D0=1: automatic operation; D0=2: paused; D0=4: emergency stop; D0=5: alarm present; any other value of D0: Wait. D0=4 and D0=5 both count as an alarm state.

11.2.4.3. $POST$ Post-processing Functions

Inside a $POST$ command you can call methods on .NET static members such as Encoding, Enumberable, BitConverter and Math.

The following functions extract a sub-array from a source array; each name applies to arrays holding the corresponding element type:

```
PGetSubBytes,
PGetSubSBytes,
PGetSubChars,
PGetSubShorts,
PGetSubUShorts,
PGetSubInts,
PGetSubUInts,
PGetSubFloats,
PGetSubLongs,
PGetSubULongs,
PGetSubDoubles;
```

Input	Type	Description
source	Object[]	(Required) The source array
startIndex	Int32	(Required) The start position

Input	Type	Description
count	Int32	(Required) The number of elements
isReverse	Bool	Whether to reverse the order; False by default
isSwap	Bool	Whether to swap elements in pairs; False by default
step	Int32	The step, i.e. the gap between the elements taken; 0 by default

These functions build a sub-array out of the source array: they start at the element at the start position, skip the specified number of elements (0 by default) to reach the next one, and continue until the specified number of elements has been collected. The sub-array is then reversed if you asked for it (off by default), and finally its elements are swapped in pairs if you asked for it (off by default).

PGetSubBytes example:

```
source = {0x01, 0x02, 0x30, 0x40, 0x66, 0x88},
startIndex = 1,
count = 3,
isReverse = True,
isSwap = True,
step = 1,
```

Three elements are taken starting at position 1 with a gap of 1, giving `{0x02, 0x40, 0x88}`. Reversing them gives `{0x88, 0x40, 0x02}`. They are then swapped in pairs — note that the length 3 is odd, so the last element takes no part in the pairwise swap. The result is `{0x40, 0x88, 0x02}`.

Functions of this kind suit cases where a single PLC data task reads a large block of data from which each individual value is then parsed out and reported separately as a post-processing result. Handling it this way improves communication efficiency enormously. In the example below, 198 Bytes are read from a Siemens PLC in one go and converted, according to their address offsets, into String / Bool / UInt16 / Float / UInt32 and other types.

```
DB500,0|rawData0,198,Byte,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 0,
8))$POST$|Production Line,
```

```
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 8,
6))$POST$|Process No,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 14,
1))$POST$|Machine No,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 18,
20))$POST$|System Device No,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 38, 2, true),
0)$POST$|Device Status,
$POST$(___PLC_TrawData0_data___[40] and 1)>0$POST$|Device On,
$POST$(___PLC_TrawData0_data___[40] and (1<<1))>0$POST$|Device Off,
$POST$(___PLC_TrawData0_data___[40] and (1<<3))>0$POST$|Emergency,
$POST$(___PLC_TrawData0_data___[40] and (1<<4))>0$POST$|Device Fault Stop,
$POST$(___PLC_TrawData0_data___[40] and (1<<6))>0$POST$|Other Mode,
$POST$(___PLC_TrawData0_data___[40] and (1<<7))>0$POST$|Pass-Through Mode,
$POST$(___PLC_TrawData0_data___[41] and 1)>0$POST$|Auto Mode,
$POST$(___PLC_TrawData0_data___[41] and (1<<1))>0$POST$|Non-Auto Mode,
$POST$(___PLC_TrawData0_data___[41] and (1<<2))>0$POST$|Cycle Start,
$POST$(___PLC_TrawData0_data___[41] and (1<<3))>0$POST$|Cycle Stop,
$POST$(___PLC_TrawData0_data___[41] and (1<<4))>0$POST$|Special Mode,
$POST$(___PLC_TrawData0_data___[41] and (1<<5))>0$POST$|Dry Run,
$POST$(___PLC_TrawData0_data___[41] and (1<<6))>0$POST$|Material Jam,
$POST$(___PLC_TrawData0_data___[41] and (1<<7))>0$POST$|Waiting for
Material,
$POST$(___PLC_TrawData0_data___[42] and 1)>0$POST$|In Production,
$POST$(___PLC_TrawData0_data___[42] and (1<<1))>0$POST$|Device Alarm,
$POST$(___PLC_TrawData0_data___[42] and (1<<2))>0$POST$|Safety Door Open,
$POST$(___PLC_TrawData0_data___[42] and (1<<3))>0$POST$|Safety Door
Closed,
$POST$(___PLC_TrawData0_data___[44] and 1)>0$POST$|Station 1 Load Request,
$POST$(___PLC_TrawData0_data___[44] and (1<<1))>0$POST$|Station 1 Unload
Request,
$POST$(___PLC_TrawData0_data___[44] and (1<<2))>0$POST$|Station 1 Part
Present,
$POST$(___PLC_TrawData0_data___[46] and 1)>0$POST$|Station 2 Load Request,
$POST$(___PLC_TrawData0_data___[46] and (1<<1))>0$POST$|Station 2 Unload
Request,
$POST$(___PLC_TrawData0_data___[46] and (1<<2))>0$POST$|Station 2 Part
Present,
$POST$(___PLC_TrawData0_data___[48] and 1)>0$POST$|Station 3 Load Request,
```

```
$POST$(___PLC_TrawData0_data___[48] and (1<<1))>0$POST$|Station 3 Unload
Request,
$POST$(___PLC_TrawData0_data___[48] and (1<<2))>0$POST$|Station 3 Part
Present,
$POST$(___PLC_TrawData0_data___[50] and 1)>0$POST$|Station 4 Load Request,
$POST$(___PLC_TrawData0_data___[50] and (1<<1))>0$POST$|Station 4 Unload
Request,
$POST$(___PLC_TrawData0_data___[50] and (1<<2))>0$POST$|Station 4 Part
Present,
$POST$(___PLC_TrawData0_data___[52] and 1)>0$POST$|Station 5 Load Request,
$POST$(___PLC_TrawData0_data___[52] and (1<<1))>0$POST$|Station 5 Unload
Request,
$POST$(___PLC_TrawData0_data___[52] and (1<<2))>0$POST$|Station 5 Part
Present,
$POST$(___PLC_TrawData0_data___[54] and 1)>0$POST$|Load Station Load
Request,
$POST$(___PLC_TrawData0_data___[54] and (1<<1))>0$POST$|Load Station
Unload Request,
$POST$(___PLC_TrawData0_data___[54] and (1<<2))>0$POST$|Load Station Part
Present,
$POST$(___PLC_TrawData0_data___[56] and 1)>0$POST$|Unload Station Load
Request,
$POST$(___PLC_TrawData0_data___[56] and (1<<1))>0$POST$|Unload Station
Unload Request,
$POST$(___PLC_TrawData0_data___[56] and (1<<2))>0$POST$|Unload Station
Part Present,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 58, 2, true),
0)$POST$|Axis Position_TY,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 60, 2, true),
0)$POST$|Axis Position_TZ1,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 62, 2, true),
0)$POST$|Axis Position_TZ2,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 64, 2, true),
0)$POST$|Axis Position_TX1,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 66, 2, true),
0)$POST$|Axis Position_TX2,
$POST$(___PLC_TrawData0_data___[68] and 1)>0$POST$|Gripper 1 Clamped,
$POST$(___PLC_TrawData0_data___[68] and (1<<1))>0$POST$|Gripper 2 Clamped,
$POST$ToSingle(PGetSubBytes(___PLC_TrawData0_data___, 70, 4, true),
0)$POST$|Cycle Time (excluding wait),
```

```
$POST$(___PLC_TrawData0_data___[74] and 1)>0$POST$|Data Ready,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 76,
28))$POST$|Part QR Code,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 104,
10))$POST$|RFID_Process No,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 114,
2))$POST$|RFID_Spindle No,
$POST$ToUInt16(PGetSubBytes(___PLC_TrawData0_data___, 116, 2, true),
0)$POST$|RFID_Part Status,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 118,
2))$POST$|RFID_Inspection Info,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 120,
12))$POST$|Machining Time,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 130, 4, true),
0)$POST$|Alarm Code 1,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 134, 4, true),
0)$POST$|Alarm Code 2,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 138, 4, true),
0)$POST$|Alarm Code 3,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 142, 4, true),
0)$POST$|Alarm Code 4,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 146, 4, true),
0)$POST$|Alarm Code 5,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 150, 4, true),
0)$POST$|Alarm Code 6,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 154, 4, true),
0)$POST$|Alarm Code 7,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 158, 4, true),
0)$POST$|Alarm Code 8,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 162, 4, true),
0)$POST$|Alarm Code 9,
$POST$ToUInt32(PGetSubBytes(___PLC_TrawData0_data___, 166, 4, true),
0)$POST$|Alarm Code 10,
$POST$UTF8.GetString(PGetSubBytes(___PLC_TrawData0_data___, 170,
28))$POST$|Station 1 Part QR Code;
```

PToStringArray

```
PToStringArray(item)
```

Converts an input of any type into a string array: for an array or collection, the output holds the ToString() result of each element (a null element becomes an empty string); for a single value, the output is a one-element array holding that value’s ToString() result.

GetActiveBits

```
GetActiveBits(bytes, offset[, bytes2, offset2, ...])
```

Input	Type	Description
bytes	Byte[]	(Required) The Byte array to inspect
offset	Int32	(Required) Offset added to each index

Finds every Bit set to 1 in the Byte array and outputs a string array of “index + offset”. Bit j (counted from the least significant bit) of Byte i has index i×8+j, with both i and j starting at 0. Arguments must come in pairs; when several pairs are passed, their results are appended in order. This is typically used to turn a PLC alarm bit area into a list of alarm numbers.

For example, with bytes = `{0x05, 0x80}` and offset = 100, bits 0 and 2 of Byte 0 and bit 7 of Byte 1 are set, so the output is `{"100", "102", "115"}`.

11.2.4.4. $MOCK$ Mock Task

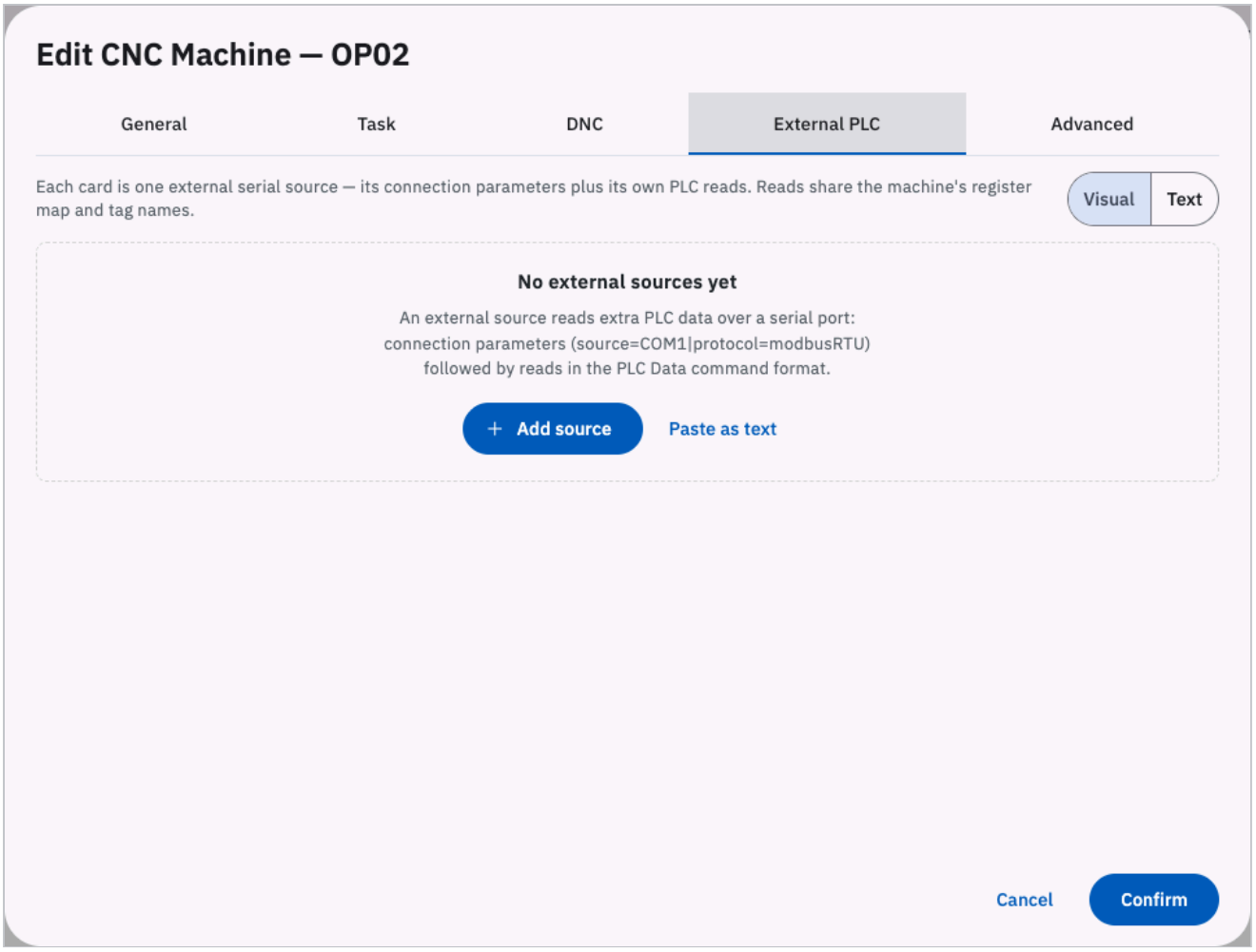
A $MOCK$ task returns the requested number of dummy values of the requested type. The returned values carry no real meaning; they exist only for a few special situations. Post-processing cannot stand on its own — it must follow the mandatory part of a PLC task. In some cases, however, the post-processing needs a different interval from the data read, or for other reasons has to be set up as a task of its own. A $MOCK$ task covers those cases.

```
$MACRO$,500|macro1_500_515,16,Double,
$DA$0$DA$,$DA$1$DA$,$DA$2$DA$,$DA$3$DA$,$DA$4$DA$,$DA$5$DA$,$DA$6$DA$,
$DA$7$DA$,$DA$8$DA$,$DA$9$DA$,$DA$10$DA$,$DA$11$DA$,$DA$12$DA$,$DA$13$DA$,
$DA$14$DA$,$DA$15$DA$;
$MOCK$,1,1,Byte,
$INTERVAL$60000$INTERVAL$,
$POST$(___PLC_Tmacro1_500_515_8_pDouble___ -
___PLC_Tmacro1_500_515_0_pDouble___)<=10$POST$|Cutoff Tool Life Low;
```

Here the first task reads machine macro variables 500 to 515 at the default interval of once every 5 seconds. The second task runs once a minute, subtracts macro variable 500 from macro variable 507 as read by the first task, and tests whether the difference is less than or equal to 10. The result, True or False, is uploaded under the tag “Cutoff Tool Life Low”.

11.2.5. External PLC Data Task

The gateway can collect data from equipment around the machine by adding an external PLC.



The command has three parts: the parameters of the external PLC, the parameters of the communication protocol, and the PLC read commands.

Parameters of the external PLC:

Parameter	Description
source	Data source. Allowed values: COM1, COM2, COM3 and so on, corresponding to serial ports 1-3 on the gateway.
protocol	Communication protocol. Allowed value: modbusRTU.

Parameter	Description
baudRate	Serial baud rate. Default 9600.
parity	Parity bit. Allowed values: none, odd, even. Default none.
dataBits	Data bits. Default 8.
stopBits	Stop bits. Allowed values: 1, 1.5, 2. Default 1.

Supported communication protocols:

modbusRTU

Parameter	Description
slaveID	Slave ID. Default 1.
plcAddresses	Whether the addresses are PLC addresses. Allowed values: false, true. Default false. When true, Modbus addresses start at 1; when false, Modbus addresses start at 0.
encoding	Encoding. Default GBK.
bit32TypeFormat	32-bit type format (Byte(4) Order). Allowed values: ABCD, CDAB, BADC, DCBA. Default DCBA.
bit64TypeFormat	64-bit type format (Byte(8) Order). Allowed values: ABCDEFGH, GHEFCDAB, BADCFEHG, HGFEDCBA. Default HGFEDCBA.

For the PLC read commands, see 4.2.1. PLC Data Task.

Example:

With a temperature sensor on the machine’s serial port 1 and a smart power meter on serial port 2, you can set up two external PLC task commands to collect the machine’s temperature and energy consumption.

```
source=COM1|protocal=modbusRTU;4x,0,1,Int16;
source=COM2|protocal=modbusRTU|baudRate=14400|encoding=ASCII;4x,0,1,Int16;4x
```

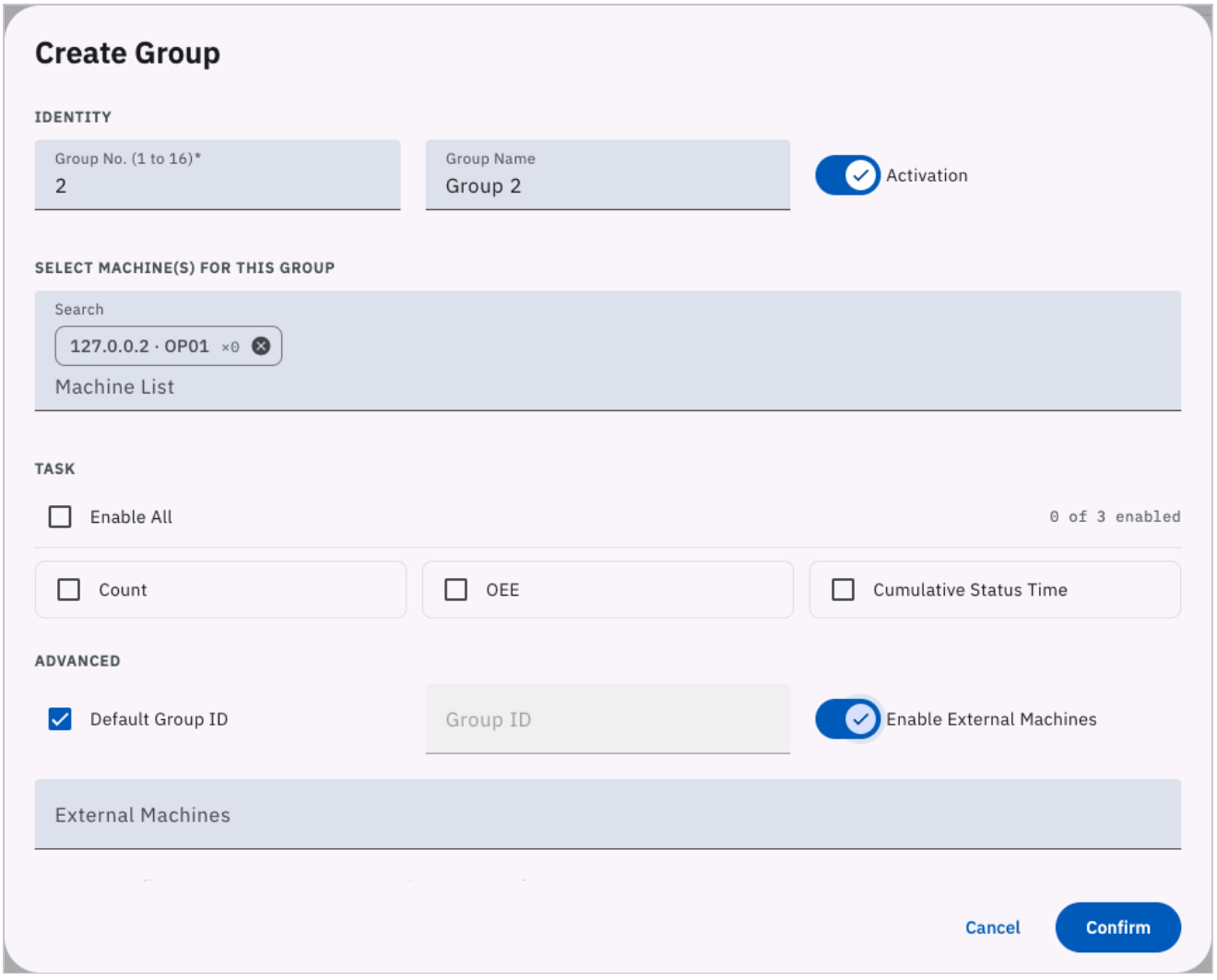
Separate different data sources with a line break.

- First line: over gateway serial port 1, using the modbusRTU protocol, read 1 Int16 from area 4x starting at address 0.
- Second line: over gateway serial port 2, using the modbusRTU protocol with the baud rate set to 14400 and the encoding set to ASCII, read 1 Int16 from area 4x starting at address 0,

and 2 Int32 starting at address 10.

11.2.6. External Machines

By adding external machines you can bring machines belonging to other gateways into a group on this gateway. You must first link the other gateway in Cloud Settings or Hub Settings.



The command format is:

```
UID,MachineID[|countMultiplier=COUNTMULTIPLIER|name=NAME,...];...
```

UID is the UID of the other, already linked gateway; MachineID is the machine number of the machine on that gateway; countMultiplier is the Count Multiplier, optional, 0 by default; name is the Machine Name, optional, the same as MachineID by default. Separate multiple external machines with “;” . For example:

```
iotgw1,1;iotgw2,100010|countMultiplier=1|name=Machining Center 10;
```

11.2.7. Protected API and Authorized APIs

API commands are used to configure protected APIs and authorized APIs.

The command format is:

```
Command Type,API address prefix or API address;
```

An API address is the part after “/api” of the address of each interface in 2.5. Data Read/Write Interfaces through 5.10. Gateway Function Interfaces. Separate multiple commands with “;”.

There are two command types: prefix match and exact match.

1. Prefix match covers every API address that begins with the given prefix:

```
prefix,/cnc;prefix,/analysis;prefix,/config;
```

Here prefix denotes a prefix match; /cnc covers every API whose address starts with /cnc, that is 5.5. Data Read/Write Interfaces and 5.6. File Management Interfaces; /analysis covers every API whose address starts with /analysis, that is 5.7. Data Analysis Interfaces; /config covers every API whose address starts with /config, that is 5.9. Gateway Configuration Interfaces.

> **Caution**
>
> The following command covers every API:
>
> ```
> prefix,/;
> ```
>
> Leaving it blank covers no API at all.

2. Exact match names the address of one specific API:

```
exact,/cnc/writeOffsetData;exact,/cnc/writePlcData;
```

Here exact denotes an exact match; /cnc/writeOffsetData is the address of the interface that writes tool offset data, see 2.5.1.14. writeOffsetData (Write Offset Data); /cnc/writePlcData is the

address of the interface that writes PLC data, see 2.5.1.19. writePlcData (Write PLC Data).

11.3. Cluster Failover (Optional)

Cluster failover is an optional gateway feature that shortens the collection outage caused by a whole-host gateway failure (power loss, hardware fault, system crash). All gateways under the same cloud platform share one pool of standby hosts: when any gateway fails as a whole, the cloud platform assigns an idle standby to take it over and run under that gateway’s identity. Once the original gateway is repaired, comes back online and stays stable, the system switches back automatically and the standby returns to the pool. The standby may sit at a different site, which makes this an off-site disaster-recovery setup.

> **Note**
>
> This feature requires the gateway to be connected to the Bivrost cloud platform. With no standby pool configured, a gateway behaves exactly as in a normal deployment.

11.3.1. How It Works

- **Shared standby pool**: a standby is not tied to one gateway. It is registered in the cloud platform’s standby pool and stands by for every gateway under that cloud platform (N+1).
- **Takeover means switching identity**: a standby does not collect while idle. When it receives a takeover command it imports the failed gateway’s configuration snapshot (machines, groups, tasks, communication settings and user accounts, plus that gateway’s cloud identity) and restarts under that gateway’s identity. The data source on the cloud platform and its dashboards follows the standby automatically — nothing upstream needs to be reconfigured.
- **One takeover at a time**: a standby covers only one gateway at any moment. If every standby is busy when another gateway fails, the cloud platform only records an event and pushes an alarm; it does not switch. As soon as a standby is released it automatically takes over the gateway that is still down. Register several standbys in the pool to cover more simultaneous failures.
- **Switchover time**: about 15 seconds (14–17 seconds measured) from the moment the gateway goes silent to the standby serving requests under that gateway’s identity with its collection engine started. Reconnecting each machine tool afterwards depends on how many machines there are and is not counted in that figure.

11.3.2. Prerequisites

1. The gateways must already be connected to the Bivrost cloud platform. The cloud platform is both the liveness arbiter and the channel used for configuration replication and takeover assignment.
2. A standby is a complete gateway host, with the same hardware and software specification as the gateways it protects.
3. The standby and the protected gateways should run the same software version. A takeover still works across versions, but it records a version-mismatch event; only run that way during a rolling-upgrade window.
4. The standby needs a licence bound to its own hardware, see 3.12.2.3. Upload License.
5. **Every gateway under the cloud platform is protected by default** — there is nothing to configure per gateway. Older gateways (which report neither heartbeats nor configuration snapshots) are not protected and are greyed out on the cluster admin page.

11.3.3. Building the Standby Pool

The pool is configured on the cloud platform's **Gateway Cluster** admin page. Contact Bivrost for the page address and the admin key.

1. Install the gateways as usual and connect them to the cloud platform.
2. Install each standby the way you install an **ordinary gateway**, and note its UID (see 3.2.3. Gateway Details).
3. Open the gateway cluster admin page on the cloud platform and enter the admin key.
4. Under "Standby Pool", click **Generate** to obtain the **pool token**, click + **Standby** to add each standby (UID and name), set the **stability window** (300 seconds by default), then click **Save**.
5. On each standby, enable the cloud platform connection, set the server address to the cloud platform and the AccessToken to the **pool token** generated in the previous step, then restart the service. Once the standby reports a heartbeat it enters the waiting state automatically: collection and cloud upload are suspended, and its own web page becomes read-only and shows a standby banner.
6. Back on the cluster admin page, confirm that the standby shows "Online · Waiting", that each gateway in the protected-gateway table shows "Protected", and that its **snapshot** timestamp refreshes after you change that gateway's configuration — this confirms the replication channel works. Each standby also caches the latest configuration snapshot of every protected gateway locally, for use in the double-fault case.
7. To leave a gateway unprotected, click **Exclude** on its row; click **Protect** to bring it back.

> **Caution**
>
> The pool token is what lets a standby join the pool. Keep it safe, and do not mix it up with a gateway’s own AccessToken.

11.3.4. Failover and Switch-back

Situation	Behaviour
Whole-host gateway failure (power loss, network loss, system crash)	After the heartbeat times out (15 seconds) the cloud platform assigns an idle standby. The standby imports that gateway’s configuration snapshot, restarts, and comes online under that gateway’s identity.
The gateway only loses its link to the cloud platform and keeps collecting locally	**No switchover.** The gateway keeps collecting and uploads the backlog once the link is restored.
Another gateway fails while every standby is busy	Only an event and an alarm are recorded; no switchover. The gateway that is still down is taken over automatically as soon as a standby is released.
The original gateway is repaired and comes back online	It is first put into read-only mode with collection stopped (to prevent dual collection). Once it has been online continuously for the **stability window**, the system switches back: the standby releases the identity and restarts back into the pool, and the original gateway pulls the latest configuration snapshot from the cloud platform (including changes made while it was covered) before resuming.
You need to postpone the switch-back (site maintenance still in progress)	Click **Hold** on that row of the cluster admin page. The original gateway will not switch back automatically even once stable; click **Resume auto switch-back** to cancel.
You need to switch back immediately	Click **Switch back now** (available only while the original gateway is online).

> **Caution**
>
> An off-site standby usually cannot reach the machine tools on the shop floor. In that case it only serves configuration, query and dashboard functions and collects no data, which is expected.

11.3.5. On-screen Notices

The cluster state is shown as a banner at the top of the gateway management page:

Banner	Meaning
This host is a shared standby, waiting in the pool (read-only)	This host is an idle standby in the pool and is not covering any gateway.
This shared standby is currently serving gateway *<name>*	This host is a standby currently running under that gateway’s identity.
This gateway is covered by a standby (read-only); it switches back automatically once stable	This host is the original gateway. It has just come back online and is waiting out the stability window.

A host that is waiting in the pool or is currently covered by a standby serves a **read-only** web page: you can sign in to inspect configuration and status, but every write is rejected, so the two hosts cannot conflict with each other.

In addition, when the browser cannot reach the current gateway the page offers the addresses of alternative nodes to jump to. Those addresses have to be configured in the gateway’s front-end configuration file beforehand — contact Bivrost to set this up.

11.3.6. Double Fault: Forced Takeover

If the site loses its link to the cloud platform **and** a gateway suffers a whole-host failure at the same time, the cloud platform cannot judge liveness or assign a standby, so no automatic takeover happens (without an arbiter, the system will not risk dual collection or a split brain). An administrator can take over manually instead:

1. Sign in to the **standby’s own** gateway management page with an administrator account.
2. Click **Force takeover** on the standby banner (visible to administrators only).
3. Pick the gateway to take over from the list that appears (the list comes from the configuration snapshots cached on the standby). After you confirm, the standby restarts and runs under that gateway’s identity.
4. Once the link to the cloud platform is restored, the cloud platform adopts this takeover automatically and folds it into the normal switch-back flow.

> **Caution**
>
> **Confirm that the target gateway really is down** before forcing a takeover. If that gateway is still collecting, a forced takeover leaves two hosts collecting from the same machines and produces duplicate data.

11.3.7. What Is Not Replicated

- **Historical and time-series data**: this lives on the cloud platform and is not replicated between hosts. A standby starts collecting from the moment it takes over and does not inherit the locally cached history.
- **The part-program directory** used by DNC (the gateway file server directory, e.g. `E:\dnc`): this is outside the product's replication scope and has to be synchronised at the operations level, for example with a scheduled task on each gateway that mirrors the directory to a share on the standby.
- **The local shop-floor entry point**: on-site users normally reach the gateway by its own IP address, which is unavailable after a whole-host failure. The entry point the product guarantees is the address served by the cloud platform, which always points at whichever host currently carries that gateway. If you need the local address to follow the switchover too, configure DNS or a virtual IP at the network layer.

11.3.8. Leaving the Standby Pool

After you **remove** a standby from the pool on the cluster admin page, that host stays exactly as it is; it does not turn itself back into an ordinary gateway (its local configuration may still hold some gateway's settings, and enabling it by hand would cause dual collection). To reuse the host as an ordinary gateway, reinstall the gateway software and configure it from scratch.

11.4. Supported Devices

This section lists the device types, systems and models the current gateway version can connect to. √ means supported and ready to configure; ◐ means in development — the protocol is covered by the communication library the gateway uses, but the integration is unfinished, so ask the vendor about the schedule before relying on it.

The machine type, system and model are chosen in 3.3.1.1. General Settings; the options there match the tables below.

Which data and file-transfer interfaces each system and model supports is listed interface by interface in 11.5. Interface Support.

11.4.1. CNC Machine Tools

System	Models	State
Bosunman	BSK Series Serial over Ethernet, BSK Series Ethernet, DF Series Serial over Ethernet, DF Series Ethernet	√
Brother	TC Series, S Series, A00 Series Serial over Ethernet	√
Citizen	Mitsubishi	√
Delta	General	√
Dmg Mori	730BM, 840D, OPC UA	√
Fagor	8035M, 8035T, 8040M, 8040T, 8055M, 8055T, 8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T	√
Fanuc	0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H, 0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A	√
Gsk	980, 988, 25i, Ethernet, Serial over Ethernet, 986 V4.15	√
Haas	General, Serial over Ethernet, MT-CONNECT	√
Heidenhain	TNC640, TNC640 DNC, iTNC530 DNC	√
Hnc	1.26.02, 1.24	√
IO Module	ADVANTECH ADAM-6000, USR IO424T	√
Jingdiao	JD50	√
Kede	GNC62	√
Knd	V4.3.00b, V5.1.00c	√
Lanhao	General	√

System	Models	State
Lnc	General	√
Lynuc	General	√
Makino	General	√
Mazak	MT-CONNECT, 640, Matrix, Smart, Smooth	√
Mitsubishi	C64, M70/M700 L Series, M70/M700 M Series, M80/M800 L Series, M80/M800 M Series	√
Okuma	P200L, P200M, P300L, P300M, P300S(LP), P300S(MP)	√
Rexroth	OPC UA	√
Siemens	802D, 808D, 828D, 840D sl, ONE, OPC UA, 810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)	√
Syntec	General	√
Mock	General	√

11.4.2. Laser Welders

System	Models	State
Hans	General, OPC UA	√
Mock	General	√

11.4.3. Robots

System	Models	State
ABB	RobotWare 6.0	√
Fanuc	General	√
Knd	General	√
Kuka	Controller-Side Proxy	√
Mock	General	√
Efort	ER7BC10	◐
Estun	General	◐
Hyundai	General	◐
Yamaha	RCX	◐

System	Models	State
Yaskawa	YRC1000, YRC High-speed Ethernet	◐

11.4.4. PLCs and Other Devices

System	Models	State
Standard Protocols	Modbus TCP, Modbus RTU, Modbus ASCII over TCP, Modbus RTU over TCP, OPC UA	√
IO Module	ADVANTECH ADAM-6000, USR IO424T	√
Jingtuo	JTE-800	√
Zhenhuaxing	VCTA-Z5ML	√
Allen-Bradley	1756, 1769, 1783, Micro800 Series, SLC Series, Others	√
Delta	AS Series, DVP Series	√
Keyence	KV700, KV300, KV Nano Series, Others	√
Melsec	Q Series, L Series, Ethernet Module QJ71E71, Ethernet Module QJ71EIP71, A Series, R Series, FX2N Series, FX3G Series, FX3S Series, FX3U Series, FX5U Series, Others	√
Omron	NJ Series, NX Series, NY Series, Others	√
Panasonic	General	√
Simatic	S7-200, S7-200 Smart, S7-300, S7-400, S7-1200, S7-1500, Others	√
Yongwei	General, General over TCP	√
Mock	General	√
Beckhoff	ADS (TwinCAT2, TwinCAT3)	◐
Cimon	All models	◐
Fatek	Programming port (serial, serial over Ethernet)	◐
Fuji	SPB, SPH	◐
GE	SRTP	◐
Inovance	Easy Series, H5U Series, H3U/S Series, H1U/S Series, EVO Series, AM Series, AC Series, Others	√
LSIS	XGB (Cnet, CPU, Fast Enet)	◐
MegMeet	MC80, MC100, MC200, MC200E, MC280	◐
Toyopuc	Computer Link (2PORT-EFR module)	◐
Vigor	VS Series	◐

System	Models	State
Xinje	XC, XD, XL Series	◐
Yaskawa	Memobus	◐
Yokogawa	Link	◐

11.4.5. Notes

- **Models of the same brand do not collect the same data.** Fagor [8060, 8065, 8070 Series], for example, supports neither file transfer nor tool data, while [8035, 8040, 8055 Series] supports both. Check the exact model against 11.5. Interface Support before choosing a system.
- **Some systems are rebadged controls.** Citizen uses a Mitsubishi control, and Makino and Dmg Mori [840D] use Fanuc and Siemens controls respectively, so they collect the same data as the underlying system.
- **IO Module** appears under both CNC machine tools and PLCs, for machines that offer no communication interface at all: an external IO module samples signal lamps, cycle-start signals and similar digital inputs, and the gateway derives machine status from them.
- **Mock** connects to no real device. It exists so gateway configuration and upstream integration can be verified without a machine — see 10. Mock Machine Testing.
- Whether data can actually be collected also depends on the optional functions enabled on the machine (Fanuc DNC permission, Heidenhain Option 18 and the like), on correct network and port settings, and on the tasks included in the gateway licence — see 3.3.1.2. Task Settings.

11.5. Interface Support

Controls differ widely in what they expose, so an interface is not available on every device. This section lists, interface by interface, which systems and models support it. It is the reference for deciding whether a given machine can supply a given value.

It applies to every transport: the HTTP APIs, the MQTT RPC commands and the collection tasks all use the same interfaces and have the same support.

Key:

Mark	Meaning
✓	**Supported**: the current version can read (or write) this data on this system and model.
◐	**In development**: the protocol is covered by the communication library the gateway uses, but the integration is unfinished. The marks show what will be available once it is.
✗	**Not supported**: the control itself does not offer the interface, so the gateway cannot support it.

> **Note**
> The tables describe the interface itself. Whether data can actually be read also depends on the optional functions enabled on the machine (Fanuc DNC permission, Heidenhain Option 18 and the like), on correct network and port settings, and on the tasks included in the gateway licence. For the full list of device types, systems and models, see 11.4. Supported Devices.

11.5.1. Interface Index

Interface	Data	Table	Section
`readCNCStatus`	Status	Basic Data	5.5.1.2
`readAlarm`	Alarms	Basic Data	5.5.1.1
`readCount`	Part Count	Basic Data	5.5.1.4
`readCurrentToolNumber`	Current Tool	Basic Data	5.5.1.5
`readTimeData`	Time Data	Basic Data	5.5.1.22

Interface	Data	Table	Section
`readProgramInfo`	Program Info	Basic Data	5.5.1.19
`readProgramBlock`	Program Block	Program and Position	5.5.1.18
`readPosition`	Position	Program and Position	5.5.1.17
`readCurrentProgram`	Running File	Program and Position	5.6.4
`readLog`	Machine Log	Program and Position	5.5.1.11
`readFeedAndSpindle`	Feed and Spindle	Feed, Load and Energy	5.5.1.8
`readFeed`	Feed	Feed, Load and Energy	5.5.1.7
`readLoad`	Load	Feed, Load and Energy	5.5.1.10
`readEnergyConsum`	Energy	Feed, Load and Energy	5.5.1.6
`readLaserPower`	Laser Power	Feed, Load and Energy	5.5.1.9
`readToolLife`	Tool Life	Tooling and PLC Data	5.5.1.23
`readOffsetData`	Read Offsets	Tooling and PLC Data	5.5.1.14
`writeOffsetData`	Write Offsets	Tooling and PLC Data	5.5.1.16
`readPlcData`	Read PLC Data	Tooling and PLC Data	5.5.1.20
`writePlcData`	Write PLC Data	Tooling and PLC Data	5.5.1.21
`readProgramList`	List Files	File Transfer	5.6.1
`receiveFileStream`	Receive File	File Transfer	5.6.2
`sendFileStream`	Send File	File Transfer	5.6.5
`deleteFile`	Delete File	File Transfer	5.6.8
`createDir`	Create Directory	Directories and Program Selection	5.6.11
`deleteDir`	Delete Directory	Directories and Program Selection	5.6.12
`selectProgram`	Select Program	Directories and Program Selection	5.6.14
`lockFileByRange`	Lock Editing	Directories and Program Selection	5.6.10

`readCNCStatusDetails` and `readCNCCommonStatus` derive from `readCNCStatus`, `readToolLifeDetails` derives from `readToolLife`, and each `batchRead*` interface derives from

its single-machine counterpart. Their support is identical to the interface they derive from, so they are not listed separately.

11.5.2. CNC Machine Tools

11.5.2.1. Basic Data

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Bosunman [BSK Series]	√	√	√	√	√	√
Bosunman [DF Series]	√	√	√	√	√	√
Brother [TC Series]	√	√	√	√	√	√
Brother [S Series]	√	√	√	√	√	√
Brother [A00 Series Serial over Ethernet]	√	√	√	√	√	√
Citizen [Mitsubishi]	√	√	√	√	√	√
Delta [General]	√	√	√	√	√	√
Dmg Mori [730BM]	√	√	√	√	✗	√
Dmg Mori [840D]	√	√	√	√	√	√
Dmg Mori [OPC UA]	√	√	√	√	√	√

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Fagor［8035M, 8035T, 8040M, 8040T］	√	√	√	√	√	√
Fagor［8055M, 8055T］	√	√	√	√	√	√
Fagor［8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T］	√	√	√	√	√	√
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	√	√	√	√	√
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	√	√	√	√	√
Gsk［980, 988］	√	√	√	√	√	√
Gsk［25i］	√	√	√	√	√	√

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Gsk [Ethernet, Serial over Ethernet]	√	√	√	√	√	√
Gsk [986 V4.15]	√	✗	√	√	√	√
Haas [General, Serial over Ethernet]	√	√	√	√	√	√
Haas [MT-CONNECT]	√	√	√	✗	√	√
Heidenhain [TNC640, TNC640 DNC]	√	√	√	√	√	√
Heidenhain [iTNC530 DNC]	√	√	√	√	√	√
Hnc [1.26.02]	√	√	√	√	✗	√
Hnc [1.24]	√	√	√	√	✗	√
IO Module [ADVANTECH ADAM-6000]	√	✗	✗	✗	✗	✗
IO Module [USR IO424T]	√	√	✗	✗	✗	✗
Jingdiao [JD50]	√	√	√	√	√	√
Kede [GNC62]	√	√	✗	√	√	√

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Knd［All models］	√	√	√	✗	√	√
Lanhao［General］	√	√	√	✗	√	√
Lnc［General］	√	✗	√	√	√	√
Lynuc［General］	√	√	√	√	√	√
Makino［General］	√	√	√	√	√	√
Mazak［MT-CONNECT］	√	√	√	√	√	√
Mazak［640, Matrix］	√	√	√	√	✗	√
Mazak［Smart, Smooth］	√	√	√	√	√	√
Mitsubishi［All models］	√	√	√	√	√	√
Okuma［All models］	√	√	√	√	√	√
Rexroth［OPC UA］	√	√	√	√	√	√
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	√	√	√	√	√

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Siemens ［810D(win XP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	√	√	√	√	√
Syntec ［General］	√	√	√	√	√	√
Mock ［General］	√	√	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.2.2. Program and Position

System [Model]	readProgram Block	readPosition	readCurrent Program	readLog
Bosunman［BSK Series］	✗	√	√	✗
Bosunman［DF Series］	✗	√	√	✗
Brother［TC Series］	✗	√	√	✗
Brother［S Series］	✗	√	√	✗
Brother［A00 Series Serial over Ethernet］	✗	√	√	✗
Citizen ［Mitsubishi］	√	√	√	✗
Delta［General］	√	√	√	✗

System [Model]	readProgram Block	readPosition	readCurrent Program	readLog
Dmg Mori ［730BM］	√	√	√	✗
Dmg Mori［840D］	√	√	√	✗
Dmg Mori［OPC UA］	✗	✗	✗	✗
Fagor［8035M, 8035T, 8040M, 8040T］	✗	√	√	✗
Fagor［8055M, 8055T］	✗	√	√	✗
Fagor［8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T］	✗	√	✗	✗
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	√	√	✗
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	√	√	✗
Gsk［980, 988］	✗	√	√	✗
Gsk［25i］	✗	√	√	✗
Gsk［Ethernet, Serial over Ethernet］	✗	√	✗	✗
Gsk［986 V4.15］	✗	√	✗	✗
Haas［General, Serial over Ethernet］	✗	√	✗	✗

System [Model]	readProgram Block	readPosition	readCurrent Program	readLog
Haas［MT-CONNECT］	✗	√	✗	✗
Heidenhain［TNC640, TNC640 DNC］	✗	√	√	✗
Heidenhain［iTNC530 DNC］	✗	√	√	✗
Hnc［1.26.02］	✗	√	√	✗
Hnc［1.24］	✗	√	√	✗
IO Module［ADVANTECH ADAM-6000］	✗	✗	✗	✗
IO Module［USR IO424T］	✗	✗	✗	✗
Jingdiao［JD50］	√	√	√	✗
Kede［GNC62］	√	√	✗	✗
Knd［All models］	✗	√	√	✗
Lanhao［General］	✗	√	√	✗
Lnc［General］	✗	√	✗	✗
Lynuc［General］	√	√	√	✗
Makino［General］	√	√	√	✗
Mazak［MT-CONNECT］	✗	√	✗	✗
Mazak［640, Matrix］	✗	✗	✗	✗
Mazak［Smart, Smooth］	✗	√	√	✗
Mitsubishi［All models］	√	√	√	✗

System [Model]	readProgram Block	readPosition	readCurrent Program	readLog
Okuma［All models］	✗	√	√	✗
Rexroth［OPC UA］	√	√	√	✗
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	√	√	✗
Siemens［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	√	√	✗
Syntec［General］	✗	√	√	✗
Mock［General］	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.2.3. Feed, Load and Energy

System [Model]	readFeedAnd Spindle	readFeed	readLoad	readEnergy Consum
Bosunman［BSK Series］	√	✗	√	✗
Bosunman［DF Series］	√	✗	√	✗
Brother［TC Series］	√	✗	✗	✗
Brother［S Series］	√	✗	✗	√
Brother［A00 Series Serial over Ethernet］	√	✗	✗	✗
Citizen［Mitsubishi］	√	✗	√	✗

System [Model]	readFeedAnd Spindle	readFeed	readLoad	readEnergy Consum
Delta［General］	√	✗	√	✗
Dmg Mori ［730BM］	√	✗	√	✗
Dmg Mori［840D］	√	✗	√	✗
Dmg Mori［OPC UA］	√	✗	✗	✗
Fagor［8035M, 8035T, 8040M, 8040T］	√	✗	✗	✗
Fagor［8055M, 8055T］	√	✗	✗	✗
Fagor［8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T］	√	✗	✗	✗
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	✗	√	✗
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	✗	√	✗
Gsk［980, 988］	√	✗	✗	✗
Gsk［25i］	√	✗	✗	✗
Gsk［Ethernet, Serial over Ethernet］	√	✗	✗	✗
Gsk［986 V4.15］	√	✗	✗	✗
Haas［General, Serial over Ethernet］	√	✗	√	✗

System [Model]	readFeedAnd Spindle	readFeed	readLoad	readEnergy Consum
Haas［MT-CONNECT］	√	✗	✗	✗
Heidenhain［TNC640, TNC640 DNC］	√	✗	√	✗
Heidenhain［iTNC530 DNC］	√	✗	✗	✗
Hnc［1.26.02］	√	✗	√	✗
Hnc［1.24］	√	✗	√	✗
IO Module［ADVANTECH ADAM-6000］	✗	✗	✗	✗
IO Module［USR IO424T］	✗	✗	✗	✗
Jingdiao［JD50］	√	✗	√	✗
Kede［GNC62］	√	√	√	✗
Knd［All models］	√	✗	√	✗
Lanhao［General］	✗	✗	✗	✗
Lnc［General］	√	✗	✗	✗
Lynuc［General］	√	✗	√	✗
Makino［General］	√	✗	√	✗
Mazak［MT-CONNECT］	√	✗	√	✗
Mazak［640, Matrix］	√	✗	√	✗
Mazak［Smart, Smooth］	√	✗	√	✗
Mitsubishi［All models］	√	✗	√	✗

System [Model]	readFeedAnd Spindle	readFeed	readLoad	readEnergy Consum
Okuma［All models］	√	✗	√	✗
Rexroth［OPC UA］	√	✗	√	✗
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	✗	√	✗
Siemens ［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	✗	√	✗
Syntec［General］	√	✗	√	✗
Mock［General］	√	✗	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.2.4. Tooling and PLC Data

System [Model]	readToolLife	readOffset Data	writeOffset Data	readPlcData	writePlcData
Bosunman ［BSK Series］	✗	√	√	√	√
Bosunman ［DF Series］	✗	√	√	√	√
Brother［TC Series］	✗	✗	✗	√	√
Brother［S Series］	√	√	√	√	√
Brother［A00 Series Serial over Ethernet］	√	√	✗	√	✗

System [Model]	readToolLife	readOffset Data	writeOffset Data	readPlcData	writePlcData
Citizen [Mitsubishi]	√	√	√	√	√
Delta [General]	✗	√	√	√	√
Dmg Mori [730BM]	✗	✗	✗	✗	✗
Dmg Mori [840D]	✗	✗	✗	√	✗
Dmg Mori [OPC UA]	✗	✗	✗	✗	✗
Fagor [8035M, 8035T, 8040M, 8040T]	✗	✗	✗	√	√
Fagor [8055M, 8055T]	✗	√	√	√	√
Fagor [8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T]	✗	✗	✗	√	√
Fanuc [0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H]	√	√	√	√	√
Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	√	√	√	√	√
Gsk [980, 988]	√	√	√	√	✗
Gsk [25i]	✗	√	√	√	✗

System [Model]	readToolLife	readOffset Data	writeOffset Data	readPlcData	writePlcData
Gsk [Ethernet, Serial over Ethernet]	√	√	√	√	✗
Gsk [986 V4.15]	✗	√	√	√	✗
Haas [General, Serial over Ethernet]	✗	√	✗	√	✗
Haas [MT-CONNECT]	✗	√	✗	✗	✗
Heidenhain [TNC640, TNC640 DNC]	√	√	✗	√	✗
Heidenhain [iTNC530 DNC]	√	√	✗	√	✗
Hnc [1.26.02]	✗	✗	✗	√	√
Hnc [1.24]	✗	✗	✗	√	√
IO Module [ADVANTECH ADAM-6000]	✗	✗	✗	√	√
IO Module [USR IO424T]	✗	✗	✗	✗	✗
Jingdiao [JD50]	√	√	√	√	√
Kede [GNC62]	√	√	✗	√	✗
Knd [All models]	✗	√	✗	√	✗
Lanhao [General]	✗	✗	✗	√	✗

System [Model]	readToolLife	readOffset Data	writeOffset Data	readPlcData	writePlcData
Lnc［General］	✗	✗	✗	√	✗
Lynuc［General］	√	√	√	√	√
Makino［General］	√	√	√	√	√
Mazak［MT-CONNECT］	✗	✗	✗	✗	✗
Mazak［640, Matrix］	✗	✗	✗	✗	✗
Mazak［Smart, Smooth］	√	√	√	√	√
Mitsubishi［All models］	✗	√	√	√	√
Okuma［All models］	√	√	√	✗	✗
Rexroth［OPC UA］	✗	✗	✗	√	✗
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	√	√	√	√
Siemens［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	√	✗	√	✗
Syntec［General］	√	√	√	√	√
Mock［General］	√	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

Besides being supported by the control, the tool-life and tool-offset interfaces need target-tool parameters whose form depends on the system. See 5.5.1.14. readOffsetData and 5.5.1.23. readToolLife.

11.5.2.5. File Transfer

System [Model]	readProgramList	receiveFileStream	sendFileStream	deleteFile
Bosunman［BSK Series］	√	√	√	√
Bosunman［DF Series］	√	√	√	√
Brother［TC Series］	√	√	√	√
Brother［S Series］	√	√	√	√
Brother［A00 Series Serial over Ethernet］	√	√	√	√
Citizen［Mitsubishi］	√	√	√	√
Delta［General］	√	√	√	√
Dmg Mori［730BM］	√	√	√	√
Dmg Mori［840D］	√	√	√	√
Dmg Mori［OPC UA］	✗	✗	✗	✗
Fagor［8035M, 8035T, 8040M, 8040T］	√	√	√	√
Fagor［8055M, 8055T］	√	√	√	√
Fagor［8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T］	✗	✗	✗	✗

System [Model]	readProgramList	receiveFileStream	sendFileStream	deleteFile
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	√	√	√
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	√	√	√
Gsk［980, 988］	√	√	√	√
Gsk［25i］	√	√	√	√
Gsk［Ethernet, Serial over Ethernet］	✗	✗	✗	✗
Gsk［986 V4.15］	✗	✗	✗	✗
Haas［General, Serial over Ethernet］	✗	✗	✗	✗
Haas［MT-CONNECT］	✗	✗	✗	✗
Heidenhain［TNC640, TNC640 DNC］	√	√	√	√
Heidenhain［iTNC530 DNC］	√	√	√	√
Hnc［1.26.02］	√	√	√	√
Hnc［1.24］	√	√	√	√
IO Module［ADVANTECH ADAM-6000］	✗	✗	✗	✗
IO Module［USR IO424T］	✗	✗	✗	✗
Jingdiao［JD50］	√	√	√	√
Kede［GNC62］	✗	✗	✗	✗

System [Model]	readProgramList	receiveFileStream	sendFileStream	deleteFile
Knd［All models］	√	√	√	√
Lanhao［General］	√	√	√	√
Lnc［General］	✗	✗	✗	✗
Lynuc［General］	√	√	√	√
Makino［General］	√	√	√	√
Mazak［MT-CONNECT］	✗	✗	✗	✗
Mazak［640, Matrix］	✗	✗	✗	✗
Mazak［Smart, Smooth］	√	√	√	√
Mitsubishi［All models］	√	√	√	√
Okuma［All models］	√	√	√	√
Rexroth［OPC UA］	√	√	√	√
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	√	√	√
Siemens［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	√	√	√
Syntec［General］	√	√	√	√
Mock［General］	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

The table above covers machine storage only. With an FTP server, a shared folder, a wireless transfer box or the gateway file server, every interface is available regardless of the machine’s

system and model. For each system’ s default root path, whether a target directory can be given, and whether subfolder listings are available, see 5.6. File Management APIs.

11.5.2.6. Directories and Program Selection

System [Model]	createDir	deleteDir	selectProgram	lockFileByRange
Bosunman［BSK Series］	✗	✗	√	✗
Bosunman［DF Series］	✗	✗	√	✗
Brother［TC Series］	√	√	√	✗
Brother［S Series］	√	√	√	✗
Brother［A00 Series Serial over Ethernet］	√	√	√	✗
Citizen［Mitsubishi］	√	√	✗	✗
Delta［General］	√	√	√	✗
Dmg Mori［730BM］	✗	✗	✗	✗
Dmg Mori［840D］	√	√	✗	✗
Dmg Mori［OPC UA］	✗	✗	✗	✗
Fagor［8035M, 8035T, 8040M, 8040T］	✗	✗	✗	✗
Fagor［8055M, 8055T］	✗	✗	✗	✗
Fagor［8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T］	✗	✗	✗	✗

System [Model]	createDir	deleteDir	selectProgram	lockFileByRange
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	✗	✗	✗	√
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	√	√	√
Gsk［980, 988］	✗	✗	√	✗
Gsk［25i］	✗	✗	√	✗
Gsk［Ethernet, Serial over Ethernet］	✗	✗	✗	✗
Gsk［986 V4.15］	✗	✗	✗	✗
Haas［General, Serial over Ethernet］	✗	✗	✗	✗
Haas［MT-CONNECT］	✗	✗	✗	✗
Heidenhain［TNC640, TNC640 DNC］	√	√	✗	✗
Heidenhain［iTNC530 DNC］	√	√	✗	✗
Hnc［1.26.02］	✗	✗	√	✗
Hnc［1.24］	✗	✗	✗	✗
IO Module［ADVANTECH ADAM-6000］	✗	✗	✗	✗
IO Module［USR IO424T］	✗	✗	✗	✗
Jingdiao［JD50］	√	√	√	✗
Kede［GNC62］	✗	✗	✗	✗

System [Model]	createDir	deleteDir	selectProgram	lockFileByRange
Knd [All models]	✗	✗	√	✗
Lanhao [General]	√	√	✗	✗
Lnc [General]	✗	✗	✗	✗
Lynuc [General]	✗	✗	√	✗
Makino [General]	√	√	√	√
Mazak [MT-CONNECT]	✗	✗	✗	✗
Mazak [640, Matrix]	✗	✗	✗	✗
Mazak [Smart, Smooth]	✗	✗	✗	✗
Mitsubishi [All models]	√	√	✗	✗
Okuma [All models]	√	√	√	✗
Rexroth [OPC UA]	√	√	✗	✗
Siemens [802D, 808D, 828D, 840D sl, ONE, OPC UA]	√	√	✗	✗
Siemens [810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)]	√	√	✗	✗
Syntec [General]	✗	✗	√	✗
Mock [General]	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.3. Laser Welders

11.5.3.1. Basic Data

System [Model]	read CNCStatus	readAlarm	readCount	readCurrent ToolNumber	readTime Data	read Program Info
Hans [General]	√	√	✗	✗	✗	√
Hans [OPC UA]	√	√	✗	✗	✗	√
Mock [General]	√	√	√	✗	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.3.2. Program and Position

System [Model]	readProgram Block	readPosition	readCurrent Program	readLog
Hans [General]	✗	√	✗	✗
Hans [OPC UA]	√	√	√	✗
Mock [General]	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.3.3. Feed, Power and PLC Data

System [Model]	readFeed	readEnergy Consum	readLaser Power	readPlcData	writePlcData
Hans [General]	√	✗	√	√	✗
Hans [OPC UA]	√	✗	✗	√	✗
Mock [General]	√	✗	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.3.4. File Transfer

System [Model]	readProgramList	receiveFileStream	sendFileStream	deleteFile
Hans［General］	✗	✗	✗	✗
Hans［OPC UA］	√	√	√	√
Mock［General］	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

11.5.3.5. Directories and Program Selection

System [Model]	createDir	deleteDir	selectProgram	lockFileByRange
Hans［General］	✗	✗	✗	✗
Hans［OPC UA］	√	√	✗	✗
Mock［General］	✗	✗	✗	√

√ Supported; ◐ In development; ✗ Not supported

11.5.4. Robots

11.5.4.1. Basic Data

System [Model]	readCNCStatus	readAlarm	readCount	readProgramInfo
ABB［RobotWare 6.0］	√	√	✗	√
Fanuc［General］	√	√	✗	√
Knd［General］	√	√	✗	√
Kuka［Controller-Side Proxy］	√	√	✗	√
Mock［General］	√	√	✗	√
Efort［ER7BC10］	◐	◐	✗	◐
Estun［General］	◐	◐	✗	◐

System [Model]	readCNCStatus	readAlarm	readCount	readProgramInfo
Hyundai [General]	◐	◐	✗	◐
Yamaha [RCX]	◐	◐	✗	◐
Yaskawa [YRC1000, YRC High-speed Ethernet]	◐	◐	✗	◐

√ Supported; ◐ In development; ✗ Not supported

11.5.4.2. Position, Logs and PLC Data

System [Model]	readPosition	readLog	readPlcData	writePlcData
ABB [RobotWare 6.0]	√	√	√	✗
Fanuc [General]	√	√	√	✗
Knd [General]	√	✗	√	√
Kuka [Controller-Side Proxy]	√	✗	√	√
Mock [General]	√	√	√	√
Efort [ER7BC10]	◐	✗	✗	✗
Estun [General]	◐	✗	✗	✗
Hyundai [General]	◐	✗	✗	✗
Yamaha [RCX]	◐	✗	✗	✗
Yaskawa [YRC1000, YRC High-speed Ethernet]	◐	✗	✗	✗

√ Supported; ◐ In development; ✗ Not supported

11.5.5. PLCs and Other Devices

System [Model]	read CNCStatus	readAlarm	readCount	readPlcData	writePlcData
Standard Protocols［All models］	✗	✗	✗	√	√
IO Module ［ADVANTECH ADAM-6000］	√	✗	✗	√	√
IO Module ［USR IO424T］	√	√	✗	✗	✗
Jingtuo［JTE-800］	√	✗	✗	√	✗
Zhenhuaxing ［VCTA-Z5ML］	✗	✗	✗	√	✗
Allen-Bradley ［All models］	✗	✗	✗	√	√
Delta［All models］	✗	✗	✗	√	√
Keyence［All models］	✗	✗	✗	√	√
Melsec［All models］	✗	✗	✗	√	√
Omron［All models］	✗	✗	✗	√	√
Panasonic ［General］	✗	✗	✗	√	√
Simatic［All models］	✗	✗	✗	√	√
Yongwei［All models］	✗	✗	✗	√	✗
Mock ［General］	√	✗	✗	√	√

System [Model]	read CNCStatus	readAlarm	readCount	readPlcData	writePlcData
Beckhoff ［ADS (TwinCAT2, TwinCAT3)］	✗	✗	✗	◐	◐
Cimon ［All models］	✗	✗	✗	◐	◐
Fatek ［Programming port (serial, serial over Ethernet)］	✗	✗	✗	◐	◐
Fuji ［SPB, SPH］	✗	✗	✗	◐	◐
GE ［SRTP］	✗	✗	✗	◐	◐
Inovance ［Easy Series, H5U Series, H3U/S Series, H1U/S Series, EVO Series, AM Series, AC Series, Others］	✗	✗	✗	✓	✓
LSIS ［XGB (Cnet, CPU, Fast Enet)］	✗	✗	✗	◐	◐
MegMeet ［MC80, MC100, MC200, MC200E, MC280］	✗	✗	✗	◐	◐
Toyopuc ［Computer Link (2PORT-EFR module)］	✗	✗	✗	◐	◐

System [Model]	read CNCStatus	readAlarm	readCount	readPlcData	writePlcData
Vigor［VS Series］	✗	✗	✗	◐	◐
Xinje［XC, XD, XL Series］	✗	✗	✗	◐	◐
Yaskawa［Memobus］	✗	✗	✗	◐	◐
Yokogawa［Link］	✗	✗	✗	◐	◐

√ Supported; ◐ In development; ✗ Not supported

PLCs and other devices are read mainly through the PLC data interfaces. Common values such as machine status, part count and current alarms can be taken from the result of a PLC task instead of the native interface; for the command format see 11.2.1. PLC Data Task.

11.5.6. Field Support

The same interface does not return the same fields on every system. The tables below list, per data class, **the fields that differ between systems**: a field not listed here is returned by every system that supports the interface, and 4.2. Data Description explains what it means. A system that does not support the interface is absent from these tables; for that question see 11.5.2. CNC Machine Tools.

A row of ✗ throughout means the system returns only the common fields of that data class. As with the interfaces, the tables describe the field itself; whether data can actually be read also depends on the optional functions and settings of the machine.

11.5.6.1. Machine Status

The fields below are returned by 5.5.1.12. readCNCCommonStatus - Read Common Machine Status. They appear only on the systems marked here and are not returned on any other machine.

System [Model]	Manual Mode Selection	Ready Status	Alarm Only Status	Emergency Set Status	Signal Lights
Dmg Mori [OPC UA]	✗	✗	✗	✗	√
Fanuc [0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H]	√[1]	✗	✗	✗	✗
Knd [All models]	✗	√	✗	✗	✗
Lanhao [General]	✗	✗	✗	√	✗
Lnc [General]	✗	✗	√	√	✗
Mazak [640, Matrix]	✗	✗	✗	√	✗
Rexroth [OPC UA]	√[2]	✗	✗	✗	✗

√ Supported; ◐ In development; ✗ Not supported Systems not listed return none of these fields.

Fields: Manual Mode Selection `mode2`; Ready Status `readyStatus`; Alarm Only Status `alarmOnlyStatus`; Emergency Set Status `emergencySetStatus`; Signal Lights `lightColor1`, `lightStatus1`, `lightColor2`, `lightStatus2`, `lightColor3`, `lightStatus3`, `lightColor4`, `lightStatus4`

1. 15 and 15i only
2. secondary mode

11.5.6.2. Machine Time Data

Each time value is stored as a Min and a Ms field; for the conversion, for how the values are reset and for the Siemens systems that always report 0, see 4.2.24. TimeData: Machine Time Data. There are too many fields for one table, so they run on into a second.

System [Model]	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Cutting Time
Bosunman [BSK Series]	√	√	√	√	✗
Bosunman [DF Series]	✗	✗	✗	√	✗
Brother [TC Series]	√	✗	√	√	✗
Brother [S Series]	√	✗	√	√	✗
Brother [A00 Series Serial over Ethernet]	√	✗	√	√	✗
Citizen [Mitsubishi]	√	✗	√	✗	√
Delta [General]	✗	✗	√	√	✗
Dmg Mori [840D]	√	√	✗	√	√
Dmg Mori [OPC UA]	√	✗	√	✗	✗
Fagor [8035M, 8035T, 8040M, 8040T]	✗	✗	✗	✗	✗
Fagor [8055M, 8055T]	✗	✗	✗	✗	✗
Fagor [8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T]	✗	√	✗	✗	✗

System [Model]	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Cutting Time
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	✗	√	✗	√
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	✗	√	✗	√
Gsk［980, 988］	✗	✗	✗	√	✗
Gsk［25i］	✗	✗	✗	√	✗
Gsk［Ethernet, Serial over Ethernet］	✗	✗	✗	√	✗
Gsk［986 V4.15］	✗	✗	✗	√	✗
Haas［General, Serial over Ethernet］	√	✗	√	✗	✗
Haas［MT-CONNECT］	✗	✗	✗	✗	✗
Heidenhain［TNC640, TNC640 DNC］	√	✗	√	✗	✗
Heidenhain［iTNC530 DNC］	√	✗	√	✗	✗
Jingdiao［JD50］	✗	✗	√	√	✗

System [Model]	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Cutting Time
Kede［GNC62］	✗	√	✗	✗	✗
Knd［All models］	✗	✗	√	√	✗
Lanhao［General］	√	✗	√	✗	✗
Lnc［General］	✗	✗	✗	✗	√
Lynuc［General］	✗	✗	✗	√	✗
Makino［General］	√	✗	√	✗	√
Mazak［MT-CONNECT］	√	✗	√	✗	√
Mazak［Smart, Smooth］	√	√	√	√	√
Mitsubishi［All models］	√	✗	√	✗	√
Okuma［All models］	√	✗	√	✗	√
Rexroth［OPC UA］	✗	✗	✗	√	✗
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	√	√	✗	√	√
Siemens［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	√	√	✗	√	√
Syntec［General］	√	√	✗	√[1]	√

System [Model]	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Cutting Time
Mock [General]	√	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

Fields: Cumulative Power-On Time `cumulativePowerOnTimeMin`, `cumulativePowerOnTimeMs`; Current Power-On Time `powerOnTimeMin`, `powerOnTimeMs`; Cumulative Operating Time `cumulativeOperatingTimeMin`, `cumulativeOperatingTimeMs`; Current Operating Time `operatingTimeMin`, `operatingTimeMs`; Cumulative Cutting Time `cumulativeCuttingTimeMin`, `cumulativeCuttingTimeMs`

1. auto run time of the current program

System [Model]	Current Cutting Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Bosunman [BSK Series]	✗	✗	√	✗	✗
Bosunman [DF Series]	✗	✗	√	✗	✗
Brother [TC Series]	✗	✗	✗	✗	✗
Brother [S Series]	✗	✗	√	√	✗
Brother [A00 Series Serial over Ethernet]	✗	✗	✗	✗	✗
Citizen [Mitsubishi]	✗	✗	√	✗	✗
Delta [General]	✗	✗	✗	✗	✗
Dmg Mori [840D]	✗	✗	√	✗	✗

System [Model]	Current Cutting Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Dmg Mori [OPC UA]	✗	✗	√	✗	✗
Fagor [8035M, 8035T, 8040M, 8040T]	✗	✗	√	✗	✗
Fagor [8055M, 8055T]	✗	✗	√	✗	✗
Fagor [8060L, 8060M, 8060T, 8065M, 8065T, 8070L, 8070M, 8070T]	✗	✗	√	✗	✗
Fanuc [0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H]	✗	✗	√[1]	✗	✗
Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A]	✗	✗	√[1]	✗	✗
Gsk [980, 988]	√	✗	✗	✗	✗
Gsk [25i]	√	✗	✗	✗	✗
Gsk [Ethernet, Serial over Ethernet]	√	✗	✗	✗	✗
Gsk [986 V4.15]	√	✗	✗	✗	✗

System [Model]	Current Cutting Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Haas [General, Serial over Ethernet]	✗	√	√	✗	✗
Haas [MT-CONNECT]	✗	√	√	✗	√
Heidenhain [TNC640, TNC640 DNC]	✗	✗	✗	✗	✗
Heidenhain [iTNC530 DNC]	✗	✗	✗	✗	✗
Jingdiao [JD50]	√	✗	√	✗	✗
Kede [GNC62]	✗	✗	√	✗	√
Knd [All models]	✗	✗	✗	✗	✗
Lanhao [General]	✗	✗	✗	✗	✗
Lnc [General]	✗	✗	√	✗	✗
Lynuc [General]	√	✗	√	✗	✗
Makino [General]	✗	✗	√[1]	✗	✗
Mazak [MT-CONNECT]	✗	✗	✗	✗	✗
Mazak [Smart, Smooth]	✗	✗	✗	✗	✗
Mitsubishi [All models]	✗	✗	√	✗	✗

System [Model]	Current Cutting Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Okuma［All models］	✗	✗	✗	✗	✗
Rexroth［OPC UA］	✗	✗	√	✗	✗
Siemens［802D, 808D, 828D, 840D sl, ONE, OPC UA］	✗	✗	√	✗	✗
Siemens［810D(winXP), 810D(winNT), 840D(winXP), 840D(winNT)］	✗	✗	√	✗	✗
Syntec［General］	✗	√	√	✗	✗
Mock［General］	√	√	√	√	√

√ Supported; ◐ In development; ✗ Not supported

Fields: Current Cutting Time `cuttingTimeMin`, `cuttingTimeMs`; Last Cycle Time `lastCycleTimeMin`, `lastCycleTimeMs`; Current Cycle Time `currentCycleTimeMin`, `currentCycleTimeMs`; Current Workpiece Cutting Time `currentCuttingTimeMin`, `currentCuttingTimeMs`; Remaining Cycle Time `remainingCycleTimeMin`, `remainingCycleTimeMs`

1. must be reset at the end of the program to clear

11.5.6.3. Tool Life Data

Count, time and wear are the three tool life types; the rest are raw data fields returned only by 5.5.1.25. readToolLifeDetails - Read Tool Life Details. Besides being supported by the system, reading tool life needs target-tool parameters whose form depends on the system, see 4.2.25. ToolLife: Tool Life Data.

System [Model]	Count Life	Time Life	Wear Life	Tool ID and Overrun	Life Exceeded [count]	Prewarning Life [time]
Brother［S Series］	√	√	✗	✗	✗	√
Brother ［A00 Series Serial over Ethernet］	√	√	✗	✗	✗	✗
Citizen ［Mitsubishi］	√	√	✗	✗	✗	✗
Fanuc［0i-B, 0i-C, 15i, 16i, 16i-W, 18i, 18i-W, 21i, Power Mate i-D, Power Mate i-H］	√	√	✗	✗	✗	✗
Fanuc［0i-D, 0i-F, 30i, 31i, 32i, 35i, Power Motion i-A］	√	√	✗	✗	✗	✗
Gsk［980, 988］	√	√	✗	✗	✗	✗
Gsk ［Ethernet, Serial over Ethernet］	√	√	✗	✗	✗	✗
Heidenhain ［TNC640, TNC640 DNC］	✗	√	✗	√	✗	✗
Heidenhain ［iTNC530 DNC］	✗	√	✗	√	✗	✗

System [Model]	Count Life	Time Life	Wear Life	Tool ID and Overrun	Life Exceeded [count]	Prewarning Life [time]
Jingdiao [JD50]	√	√	✗	✗	✗	✗
Kede [GNC62]	√	√	√	✗	✗	✗
Lynuc [General]	√	√	✗	✗	✗	✗
Makino [General]	√	√	✗	✗	✗	✗
Mazak [Smart, Smooth]	√	√	✗	✗	✗	✗
Okuma [All models]	√	√	√	✗	✗	✗
Siemens [802D, 808D, 828D, 840D sl, ONE, OPC UA]	√	√	√	✗	√[1]	✗
Siemens [810D(win XP), 810D(winNT), 840D(winXP), 840D(winNT)]	√	√	√	✗	√[1]	✗
Syntec [General]	√	√	✗	✗	✗	✗
Mock [General]	√[2]	√[3]	√[4]	✗	✗	✗

√ Supported; ◐ In development; ✗ Not supported

Fields: Count Life `countLimit`, `currentCount`, `prewarningCount`; Time Life `timeLimit`, `currentTime`, `prewarningTime`; Wear Life `wearLimit`, `currentWear`, `prewarningWear`; Tool ID and Overrun `inventoryNum`, `rawTimeLimit1`, `rawTimeLimit2`, `rawOverTime`; Life Exceeded [count] `rawOverCount`; Prewarning Life [time] `rawPrewarningTime`

1. user-defined module
2. ToolNum 1-10
3. ToolNum 11-20
4. ToolNum 21-30

11.5.7. Notes

- **Models whose support is identical share one row**; a model that differs in any single interface is listed separately.
- **Lock program editing** (`lockFileByRange`) is available on the whole Fanuc range and on Makino only.
- **Energy data** (`readEnergyConsum`) is currently supported only by Brother [S Series]; **machine log** (`readLog`) and **feed** (`readFeed`) only by Mock and Kede [GNC62] respectively. On other systems, read these values through the PLC data interfaces using addresses supplied by the machine builder.
- Interfaces not listed here are gateway functions of their own (analysis, gateway configuration, history, file-server management) and do not depend on the machine’ s system and model.

12. FAQ

12.1. Cannot Open the Gateway Management Page

Use an Ethernet cable to connect the gateway’s LAN1 or LAN2 port to the LAN port of your computer, either directly or through a switch.

Check the computer’s network settings. Its IP address must be on the same subnet as the gateway’s port, and the subnet mask must be identical. The gateway’s LAN1 port defaults to IP address 192.168.100.1 with subnet mask 255.255.0.0; the LAN2 port obtains both its IP address and its subnet mask automatically. If LAN1’s IP address and subnet mask have been changed and you no longer remember the current values, connect through LAN2 instead.

Check whether the gateway’s power LED is lit steadily and whether the network port LED is blinking. If the power LED is off, the gateway is not powered on. If the port LED is not blinking, check both ends of the Ethernet cable and make sure the computer and the gateway are properly connected.

If the power LED is on and the port LEDs at both ends are blinking, a network connection between the computer and the gateway has been detected. Open the gateway management page on the computer. If it does not open, enter one of the following commands in the computer’s command prompt, depending on the operating system:

```
ping IOTGW -4       (Windows)
ping IOTGW.local    (Linux or Mac)
```

Note

On some early gateway models, if IOTGW does not work, replace IOTGW with BIV-[first part of the gateway UID]. The gateway UID is printed on the label on the bottom of the gateway. For example, if the gateway UID is “152KUCG-1TOSJH4-1WXYSON-JY531Q”, the first part of the UID is “152KUCG”, so on a Windows computer the command is:

```
ping BIV-152KUCG -4
```

If the ping succeeds and the reply shows the IP address of the gateway’s current port, communication between the computer and the gateway is working. If the gateway management page still will not open, contact customer support.

If the ping fails, disconnect all other devices, make sure the computer is connected to a single gateway only, and run the command again.

If the command now succeeds, either the gateway or the computer had an IP address conflict with a device that has since been disconnected from the LAN. Change the conflicting IP address.

If the ping fails again, there is a network fault between the computer and the gateway, usually caused by the network port or the cable.

12.2. The Gateway Page Is Blank or Incomplete in the Browser

The browser you are using does not support the page framework. Use Google Chrome, Microsoft Edge or Apple Safari. If none of these is available, download and install the latest version of Chrome.

12.3. Changing the Gateway’s LAN1 IP Address

Open the gateway management page and click **Network** in the navigation bar. LAN1’s IP address can be changed in the network settings; see 3.7.1. Wired Network.

12.4. Finding the Gateway’s Current LAN2 IP Address

Use an Ethernet cable to connect the gateway’s LAN2 port to the LAN port of your computer, either directly or through a switch. Follow 5.1. Cannot Open the Gateway Management Page and run the command to obtain the address of the gateway port you are connected to.

> **Note**
>
> LAN2 obtains its IP address automatically by default, so the address may differ every time you connect through LAN2.

12.5. Recovering the LAN1 IP Address

Use an Ethernet cable to connect the gateway’s LAN2 port to the LAN port of your computer, either directly or through a switch. Sign in from the computer using the domain name (see 3.1. Signing In); alternatively, find LAN2’s current IP address first (see 5.4. Finding the Gateway’s Current LAN2 IP Address) and sign in with that IP address.

Go to **Network** and change the LAN1 address (see 3.7.1. Wired Network).

12.6. Resetting the Gateway’s LAN1 Settings

Gateway software version 1.5.6.8 and later supports resetting LAN1. Rename any USB flash drive that is not a system boot disk (used for installing the operating system) to “resetlan1” (case does not matter) and plug it into any USB port on the gateway. After the gateway is shut down and restarted, LAN1’s IP address and subnet mask are automatically reset to the factory settings: IP address 192.168.100.1, subnet mask 255.255.0.0. Remove the USB drive once the reset is complete, otherwise the reset will run again on the next restart.

12.7. Cannot Read Machine Data from API Test

If you cannot read data from the target machine on the **API Test** page, follow the returned error code and its remedy. The main API Test error codes are listed below.

Error Code: 3, Machine off or offline

Remedy: Unable to communicate with the machine. Please check: 1. the machine is turned on; 2. the machine and the gateway are on the same subnet of the same LAN; 3. the machine’s network is properly set ①; 4. the gateway is properly configured ②; 5. network hardware such as ports, cables and switches is working correctly.

Error Code: 102, Machine network error, unable to read data

Remedy: Please ensure the machine’s communication service is started. Check whether another device on the LAN conflicts with this machine’s IP address ③, and try changing the machine’s IP address.

Error Code: 10003, machineID does not exist

Remedy: The machineID is wrong, the corresponding machine is not activated, or the gateway service was not restarted after the machine’s IP address or activation status was edited.

Error Code: 10006, API not authorized

Remedy: Contact Bivrost if you need to use an unauthorized API.

The remaining error codes are described in section 5.2. Error Handling. If the error code does not let you resolve the problem, contact customer support for remote assistance; see 5.10. Getting Remote Assistance.

Note

① On machines with certain control systems, setting the IP address is not enough: you also have to enable the communication options, configure the firewall and so on before data can be collected. On some models, after the IP address is set, the machine must be powered off for 3 seconds and then powered on again for the setting to take effect.

② On machines with certain control systems, besides setting the system model and the IP address on the gateway, you also have to set a user name and password. When you are finished, you must click **Restart Service** on the home page.

③ A machine IP conflict usually means the IP address configured on the machine is already used by another device on the same LAN. Some devices cannot be pinged because of their firewall settings, so their IP addresses are often mistaken for unused ones. In this case, change the machine's IP address to one that is currently free and power-cycle the machine; at the same time, change the machine's IP address to the new one on the gateway's **Machines** page, click **Restart Service**, and run the collection test again.

The other form of address conflict is a subnet conflict. This is common when the acquisition network and another internal network are connected to the same switch and both use the same subnet. When this happens, move either the conflicting internal devices or the gateway and machines to a subnet that is not in use.

To keep IP conflicts from recurring, see 5.11. The Gateway or a Machine Has an IP Conflict with Another Device on the LAN.

12.8. Cannot Get Automatically Collected Machine Data over MODBUS, MQTT or a Database

On the gateway's **API Test** page, check whether the machine's common data can be read; see 3.8. API Test.

If it cannot, follow 5.7. Cannot Read Machine Data from API Test.

If API Test does return machine data, open the **Machines** page, then the "Edit Machine" dialog and its "Task" tab, and check whether the machine's automatic collection tasks are enabled; see 3.3.1.2. Task Settings. Task results are only uploaded automatically over the protocols once the corresponding automatic collection tasks are enabled.

Check the settings for the protocol you are using on the gateway's **Communication** page; see 3.6. Communication.

Once all of the gateway settings above are confirmed, click **Restart Service** on the gateway’s home page.

If the automatically collected data still does not arrive over MODBUS, MQTT or the database, check the settings of the receiving system and the network connection between it and the gateway.

12.9. Cannot Get Automatically Collected Group Data over MODBUS, MQTT or a Database

First confirm that the target group has been added successfully on the gateway’s **Groups** page and is activated; see 3.4. Groups.

Check the group’s settings in the “Edit Group” dialog on the gateway’s **Groups** page, and make sure the group contains at least one machine; see 3.4.2. Editing a Group.

In the “Group Task Settings” dialog for that group on the gateway’s **Groups** page, check whether the automatic collection tasks are enabled; see 3.4.1.2. Group Task Settings. Task results are only uploaded automatically over the protocols once automatic collection is enabled.

On the gateway’s **API Test** page, check whether the common data of the machines in the group can be read; see 3.8. API Test. If it cannot, follow 5.7. Cannot Read Machine Data from API Test.

Once everything is confirmed, click **Restart Service** on the gateway’s home page and run the collection test again.

12.10. Getting Remote Assistance

Scan the QR code on the cover to contact customer support and describe the problem; support will assign an engineer to work with you.

Once remote assistance has been agreed, connect the gateway’s LAN2 port to the internet (if LAN2’s settings have been changed, make sure the current settings still allow internet access), or connect the gateway to the internet over WiFi (see 3.7.2. WiFi Settings). Then sign in, open **Home**, confirm that “Network” under “Gateway Status” reads “Connected” and that the “Remote Assistance” switch in the “Settings” dialog is turned on, and notify the engineer you are working with.

12.11. The Gateway or a Machine Has an IP Conflict with Another Device on the LAN

An IP address or a whole subnet already being in use is a common problem when connecting the gateway or a machine to a network. It can be resolved as follows:

1. Put the gateway and the machines on their own router, isolated from other devices.
2. When connecting, make sure the gateway and the machines do not use an IP address or subnet mask that is already taken.
3. In the router’s management page, bind the static IP addresses of the gateway and the machines to their MAC addresses so that those IP addresses are not handed out to other devices automatically.

If IP addresses cannot be bound to MAC addresses, apply either of the following settings in the router’s management page:

1. Disable the router’s DHCP server so that IP addresses are no longer assigned automatically and devices can only be connected with manually configured addresses.
2. Restrict the router’s DHCP address pool so that it does not overlap the addresses configured on the gateway and the machines.

For the router settings themselves, refer to the router’s own manual.

12.12. The DNC Page Can List, Receive, Send and Delete Files but the Matching API Calls Fail

The path or file name contains unsafe characters such as special symbols or spaces, which must be converted to URL encoding. For example, the path `TNC:\nc_prog` becomes `TNC%3A%5Cnc_prog`, and `Sinumerik/FileSystem/Part Program` becomes `Sinumerik/FileSystem/Part%20Program`. Most browsers and test tools apply URL encoding automatically, but some software still requires you to convert the path by hand when calling the API.

12.13. HTTP communication returns “This site can’t be reached (ERR_CONNECTION_REFUSED)”

Please check whether the computer you are using can access the gateway management page. If not, check whether the gateway is powered on and whether the network connection between the current computer and the gateway is normal.

12.14. HTTP communication returns `"errorCode":3`

Please use the **Interface Test** feature on the gateway management page to check the communication between the gateway and the machine tool. (See 3.8. Interface Test.) If you cannot find the target device in the "Select Machine" dropdown on the **Interface Test** page, it means the device has not been added or activated. Please refer to 3.3. Machine Configuration to add and activate the machine. After completing the setup, you must click **Restart Service** on the **Home** page.

If you found the target device on the **Interface Test** page under "Select Machine" but clicking **Read Device** returns "Read Failed", please follow the "Troubleshooting" guidance shown below "Read Failed".

12.15. No data is received during MODBUS or MQTT communication

Please check the machine's automatic data collection settings (see 3.3. Machine Configuration). After completing the setup, you must click **Restart Service** on the **Home** page.

Please check the task configuration (see 3.5. Task Configuration). After completing the setup, you must click **Restart Service** on the **Home** page.

If you still cannot receive any data, please follow the steps described in 7.2.

12.16. Valid data cannot be obtained at a given address during MODBUS communication

A common cause is a misconfigured MODBUS client — it should be configured as 0-based rather than 1-based. The gateway's MODBUS server is configured as 0-based; if the client is 1-based, the address must be incremented by 1. For testing, it is recommended to use the Modbus Poll software to read address data — when setting the address range, **do not check "(Base 1)"**, i.e. configure it as 0-based.

Some CNC models with older system software versions do not support certain data collection items; the address data corresponding to these items remains 0.

12.17. Local InfluxDB Installation and Startup Directories on Windows Gateways

For a new installation, the standalone InfluxDB installer accepts only the complete installation directory. For example:

```
installInflux.bat "D:\Program Files\InfluxData\influxdb"
```

Data is automatically stored in the installation directory's `data` subdirectory, which is `D:\Program Files\InfluxData\influxdb\data` in this example. No separate data argument is needed. A factory reset clears this subdirectory and preserves the program files. The installer does not overwrite an existing service or migrate existing data; existing services retain their original data location.

If an existing service logs `AppDirectory is unset` and falls back to `C:\Windows`, maintenance personnel should set NSSM's `Application` to the full path of `influxd.exe`, set `AppDirectory` to its containing directory, and restart the InfluxDB service. Keep the existing data location when repairing the startup directory. A gateway firmware upgrade does not automatically change the independently installed InfluxDB service configuration.

A service shown as "Paused" may be waiting for NSSM to retry startup. Check the InfluxDB logs for errors such as a port conflict rather than assuming the directory warning is the cause. A manual run under an administrator account may use different data from the LocalSystem service; identify and back up the original service data before migration.

13. Known Issues

13.1. The LAN Port IP Address Shown on the Gateway’s Network Page Does Not Match the Actual Setting

Prior to version 1.12.2.0, if a LAN port was not connected the gateway could not determine that port’s address and assigned it a temporary one instead. This does not affect normal operation: once the port has a working connection again, the actual IP address is displayed.

13.2. A Brief Power Trip Causes the Gateway to Fail to Restart Automatically

If power is restored within 15 seconds of a momentary trip, the gateway may fail to restart automatically, in which case you have to start it manually by pressing the gateway’s power button. To avoid frequent power interruptions, install the gateway where the supply is stable, and fit it with a UPS if possible.

13.3. The Gateway Is Connected to Several Networks and One Is Restricted, So It Cannot Reach the Internet

When the gateway has a wired network, a WiFi network and an external USB LAN adapter connected at the same time, and all of them can reach the internet, the gateway uses whichever network connected first. If that network is restricted and can only reach a limited set of external hosts, internet access fails even though the other networks are unrestricted.

The correct way to connect to the internet is to disconnect every network first, connect the gateway to the unrestricted network, and only then connect the others. For example, suppose LAN1 is connected to the company intranet (behind a firewall) and an external USB LAN adapter is used to reach the internet for remote assistance. To get remote assistance working, unplug the LAN1 cable, plug in the external adapter, wait until the remote assistance connection is established, and then plug the LAN1 cable back in.

13.4. LAN1 and LAN2 on the Same Subnet Cause a Subnet Conflict and Communication Failure

If the gateway’s LAN1 and LAN2 ports are on the same subnet, the resulting subnet conflict can cause communication failures. Do not put LAN1 and LAN2 on the same subnet.

Changelog

v1.19.7.60 (2026-10-16)

Release date: 2026-10-16

1. Added data reading and writing for Inovance Easy, H5U, H3U/S, H1U/S, EVO, AM and AC series and Others, with Ethernet, Serial over Ethernet, EasyNet, Connected CIP and OPC UA connections depending on the model.

v1.19.7.57 (2026-10-13)

Release date: 2026-10-13

1. The Haas model previously displayed as “Serial” is now “Serial over Ethernet”. Existing machine configurations remain compatible.
2. Corrected the required-field message in the Chinese machine configuration dialog to use Simplified Chinese.

v1.19.7.56 (2026-10-12)

Release date: 2026-10-12

1. 3.12.6.3. Factory Reset: moved to the Backup & Restore card. Both the local gateway page and Hub require the gateway administrator username and password. Wait a few minutes for the local database to initialize after completion.
2. 3.12. Settings: follows the account menu and settings card order, with new and updated screenshots in both languages and corrected bold formatting.
3. Fixed missing results when searching the online manual for some Chinese phrases.

v1.19.7.55 (2026-10-12)

Release date: 2026-10-12

1. 3.12.6.3. Factory Reset: administrators can clear business settings, accounts, local collected data, logs, queued data and program files from Settings while keeping the license, network and installation settings. Reset can also be requested through Hub. Cluster members must leave the cluster first. Sign in with `admin/admin` after completion.

v1.19.7.54 (2026-10-08)

Release date: 2026-10-08

1. Improved the management interface display.

v1.19.7.53 (2026-09-29)

Release date: 2026-09-29

1. 3.9.3. Sending a Local File to a Machine: a send to a Syntec machine that drops partway now returns the new error code 132 (CNCWRAPPER_SEND_FILE_INTERRUPTED; the machine may keep a truncated program) instead of a network error; the web page then refreshes that machine’s directory to show the leftover, and retrying overwrites it.

v1.19.7.51 (2026-09-28)

Release date: 2026-09-28

1. 5.9.3.7. duplicate-machine now respects the licensed machine count: when the template machine is activated, copies beyond the license are created deactivated and statusMsg returns `More activated machines than licensed.`.

v1.19.7.48 (2026-09-23)

Release date: 2026-09-23

1. 3.6.3. MQTT Settings adds China Unicom Gewu Cloud (`GeWu`): the gateway uses per-device credentials, while machines and groups are weak-validation subdevices whose products allow dynamic registration. 7.1. Data Upload Message Format documents the topic and payload.
2. [breaking] In 7.1. IoTDA Mode, `service_id` is now the `fieldID` and keys in `properties` are the original field names. Consumers that parsed tag-prefixed property names must adjust. The 5.9.6. MQTT settings API now describes the `GeWu` fields.

v1.19.7.47 (2026-09-23)

Release date: 2026-09-23

1. 5.7.2.3. count gained the request parameter `sumByGroup`: when true, each grouping interval returns a single group record whose count is the sum of each machine’s count × its count multiplier; “3.10.2. Count Analysis” gained the “Group total by count multiplier” toggle, and `analyze` in “9. MCP Service” takes the same parameter.

v1.19.7.46 (2026-09-21)

Release date: 2026-09-21

1. 5.5.1.20. readPlcData and 5.5.1.21. writePlcData: the R area of Mazak [Smart, Smooth] now reads and writes Int16, UInt16, Int32 and UInt32 (each R address is 16 bits); the PLC Data Types table is updated accordingly.

v1.19.7.45 (2026-09-21)

Release date: 2026-09-21

1. “3.11.1. Machine Monitor”: the “Current Count” card at the top moved into the “Production Count” card, next to its title.

v1.19.7.44 (2026-09-18)

Release date: 2026-09-18

1. 5.10.1.7. settings gained the watchdog settings `watchdogEnabled`, `watchdogUnhealthyMinutes`, `watchdogCooldownMinutes` and `watchdogProbeTimeoutMs`: Core and Gateway probe each other every minute and start / force-restart the peer service once it has been stopped or unresponsive for 3 minutes. `serviceQueueCapacity` is now a fixed cap (used when disk overflow is off) and no longer grows with memory usage.
2. “3.2. Home”: the local cache (database) stays in “Waiting for Connection” instead of “Running” until it has connected; the Hub broadcasting pipeline’s backlog is now buffered on disk as well and can be discarded from its row.

v1.19.7.43 (2026-09-17)

Release date: 2026-09-17

1. Rewrote “9. MCP Service”: the 33 tools are consolidated into 13, adding `get_fleet_status` (plant-wide cached status snapshot), `read_machine` (several live items

in one call via items), `analyze` / `query_history` (machine and group analysis and history in one tool each, with `newestFirst` for the most recent rows), `read_task_data` (cached sample of any data class) and `get_system_info`; the old tool names such as `read_cnc_status`, `analyze_oee` and `batch_read_errors` are gone.

2. “9.4. Tool List” documents the accepted time parameter forms (Unix seconds, ISO-8601, dates, relative offsets, now/today/yesterday); “9.5. Results and Errors” now distinguishes passthrough from composite results and the error message format gained a `Hint:` suffix.

v1.19.7.42 (2026-09-16)

Release date: 2026-09-16

1. “3.11.1. Machine Monitor” now says the “Current Count” card is labelled “Current period”, and drops the note about the “Monitor” permission it used to need; the Monitor pages and earlier changelog entries now call these cells cards throughout.

v1.19.7.41 (2026-09-15)

Release date: 2026-09-15

1. Documented the “Current Count” card and its counting period in “3.11.1. Machine Monitor”.

v1.19.7.40 (2026-09-14)

Release date: 2026-09-14

1. Added the CNC system Makino [general] (Fanuc-based). It behaves like Fanuc 30i series; Makino has been added to the tag-combination, time data, tool life, file management, and PLC special region/data type tables.
2. Added the CNC system Makino to the supported machine list.

v1.19.7.38 (2026-09-15)

Release date: 2026-09-15

1. Documented the PToStringArray and GetActiveBits functions in “11.2.4.3. $POST$ Post-processing Functions”.

v1.19.7.37 (2026-09-09)

Release date: 2026-09-09

1. [breaking] `isOnline` returned by 5.10.2.7. internet-connection is now nullable: it is null until the gateway’s first internet probe round has a result (a few seconds after startup) and whenever probing is disabled in the configuration. Callers that treat `isOnline == true` as online are unaffected; callers that treat false as offline must handle null. The `checkedAtUnix` and `lastOkUnix` fields were added.
2. The interface now reads the cached result of the gateway’s background internet probe instead of connecting on each request. 5.10.1.7. settings gained the probe settings `internetProbeTargets`, `internetProbeIntervalMs`, `internetProbeTimeoutMs` and `internetProbeFailuresBeforeOffline`.
3. Renamed the home page’s “Network” row to “Internet” and documented how its states are decided, in “3.2. Home”.

v1.19.7.36 (2026-09-08)

Release date: 2026-09-08

1. 5.10.2.17. wifi and 5.10.2.19. search-wifi return error code 10051 (GENERAL_WIFI_NOT_AVAILABLE) when the gateway has no usable wireless adapter, with the cause in `statusMsg`; previously they returned 1 (Unexpected Error.).

v1.19.7.31 (2026-09-15)

Release date: 2026-09-15

1. “3.12.2.2. Upgrade Gateway” now notes that an upload can be cancelled while in progress.

v1.19.7.30 (2026-09-02)

Release date: 2026-09-02

1. 5.10.1.7. settings gained `processMetricsIntervalMs` (the output interval of the process metrics log line).

v1.19.7.29 (2026-09-01)

Release date: 2026-09-01

1. Added 5.10.1.8. drop-service-queue, which permanently discards a service’s queued data in memory and on disk; the command is also available over MQTT RPC.
2. Added “3.2.2.1. Queue Backlog and Discarding It”, covering the data-pipeline backlog display, the “Recovering” status, and the new button that discards queued data.

v1.19.7.24 (2026-08-27)

Release date: 2026-08-27

1. [breaking] The MCP service now has a switch and is **disabled by default**; while it is off, `/mcp` returns HTTP 404. Enable it on the gateway from the Communication page (effective immediately). Existing MCP clients must have the switch turned on after upgrading. See 9. MCP Service.
2. The MCP service now has a switch and is disabled by default; it must be enabled under 3.6.8. MCP Settings before it can be reached. Updated “9. MCP Service” accordingly.

v1.19.7.22 (2026-08-20)

Release date: 2026-08-20

1. Added the MCP service. The gateway and the cloud platform now provide an MCP server at `/mcp`, exposing machine configuration, live status, data analysis and gateway system information to AI clients as 33 read-only tools, using the same authentication as the HTTP interfaces. See 9. MCP Service.
2. Added “11.3. Cluster Failover (Optional)”, covering how the shared standby pool works, how to build it, failover and switch-back behaviour, forced takeover, and what is not replicated.

v1.19.7.18 Subsequent Updates (2026-08-05)

Release date: 2026-08-06

1. 【breaking】CNC-Siemens model names changed (v1.19.7.16). The Siemens model names returned by the machine configuration interfaces have changed; existing machine configurations are unaffected. Old model => new model (* marks the model an existing configuration maps to after the upgrade):
    - General => 828D, *840D sl

    - 810D/840D (winXP) => 810D(winXP), *840D(winXP)
    - 810D/840D (winNT) => 810D(winNT), *840D(winNT)
    - SINUMERIK ONE => *ONE
2. Added CNC-Siemens models 802D and 808D (v1.19.7.16).
3. IP white list behaviour changed (v1.19.7.17). The login endpoint `/api/auth/login` and the product information endpoint `/api/misc/product-details` are no longer subject to the IP white list; requests authenticated with the JWT method skip the IP white list check, while requests authenticated with the secret key method are still subject to it. See 5.3.3. IP White List for details.
4. Machine reconnection wait changed (v1.19.7.18). Previously a fixed wait; now the first connection attempt does not wait, consecutive failures back off as 2s (after 1 failure) → 4s → 8s → 16s → 32s → 60s (capped from the 6th failure), and the backoff state resets after a successful connection.
5. This documentation has been fully revised against the gateway source code, correcting errors in endpoint paths, field names, field types, enum values and supported machine models, and adding previously missing field and endpoint descriptions.
6. Reworked the Siemens CNC model list: the generic model was split into 828D and 840D sl; 810D/840D (winXP) was split into 810D(winXP) and 840D(winXP); 810D/840D (winNT) was split into 810D(winNT) and 840D(winNT); SINUMERIK ONE was renamed ONE; added the 802D and 808D models. Existing machine configurations are unaffected.
7. Revised the “Enable IP Whitelist” description: the login endpoint is not restricted by the whitelist, JWT-authenticated requests skip the whitelist check, and SecretKey-authenticated requests must still pass it.

v1.19.7 (2026-07-19)

Release date: 2026-07-19

1. Added parameter NetworkErrorAsOffline (treat network errors as offline) to machine configuration.
2. Added support for the GSK 25i machine model.
3. Added function GetSetBitIndices to task post-processing.
4. Added gateway function interface hardware-resources to retrieve gateway hardware resources.
5. Added Alarm Mapping.
6. Added the “Network Error as Offline” (NetworkErrorAsOffline) option to the machine configuration.

7. Files downloaded from DNC no longer carry a file extension.
8. Added compatibility with older Chromium-based browsers.

v1.19.6 (2026-07-02)

Release date: 2026-07-02

1. Removed deprecated fields from the AlarmHistory (alarm history) and AlarmLog (current alarms) data classes.
2. Added new detailed waiting sub-statuses to the CNCStatus running status, and added a Description column.
3. Revised the Monitor documentation.

v1.19.5 (2026-06-15)

Release date: 2026-06-15

1. Modified the selectProgram interface; added parameter mode (execution mode).
2. Added Toolpath viewing to the “DNC” page.

v1.19.4 (2026-02-11)

Release date: 2026-06-05

1. Changed the authentication method; the original API Key method was replaced with the Secret Key method.
2. The original Cumulative Status Time machine task was merged into the Machine Status task; cumulative time is now tracked automatically when retrieving machine status.
3. Added variable LastTotalTime (previous cycle total time) to Cycle data.
4. Added output variable StartTime (cycle start time) to the `/api/analysis/cycle` machine cycle data analysis interface and the `/api/group-analysis/cycle` group cycle data analysis interface.
5. Expanded the group number range from 1-8 to 1-16.
6. Revised the table of supported RPC interfaces.
7. Added field CmdFeed to the Feed data class. Added fields CmdFeed and CmdSpindle to the FeedAndSpindle data class.
8. Updated corresponding reference locations due to chapter changes in the Bivrost Gateway Manual.

9. Updated the description of slave ID in MODBUS communication.
10. Moved to the new UI and updated the content accordingly.
11. Added the Monitor page.
12. Added documentation for PLC task post-processing.
13. Added documentation for custom tasks.

v1.19.3 (2025-12-22)

Release date: 2025-12-22

1. Fixed an incorrect description. For the readCount (read machining count) interface, corrected the type of the count field in the response to Int32.
2. Removed the deprecated data classes AxialLoad (servo axis load) and SpindleLoad (spindle load). Use the Load data class for this data instead.
3. Merged the ToolLife (tool life) and ToolLifeDetails (tool life details) data classes.
4. Added description of the authentication method and authentication interface.
5. Revised the interface categorization; renamed the file transfer interfaces to file management interfaces.
6. Added interfaces: readErrorSummary (read status summary) and searchFile (search machine files).
7. Added data type LogHistory (log history) and the corresponding machine task.
8. Removed deprecated parameters from the MQTT settings: UseGatewayAlias (use gateway alias) and KeepAliveInterval (keep-alive period).
9. Added parameters to machine configuration: numberOfTasks (number of enabled tasks), custom status parameters, custom alarm parameters, and custom machining count parameters.
10. Updated the tool offset data tag combination table.
11. Updated the tool life data tag combination table.
12. Added interfaces remote-access (get remote access settings) and update-remote-access (update remote access settings).
13. Added support for request parameter Channel (channel number) to the ReadPlcData and WritePlcData interfaces.
14. Modified the following interfaces:

- `/api/gateway/wifi`
- `/api/gateway/update-wifi`
- `/api/gateway/search-wifi`

- `/api/config/gateway-info`
- `/api/config/cloud-settings`
- `/api/config/update-cloud-settings`

15. Added interface `/core/license-info`.

v1.19.2 (2025-06-03)

Release date: 2025-06-03

1. Added the following interfaces
    - `/config/batch-delete-machines`
    - `/config/batch-delete-groups`
    - `/analysis/cycle`
    - `/group-analysis/cycle`
2. Split the original alarm machine task (alarm information) into alarm (current alarms) and alarmHistory (alarm history).
3. Adjusted variable names and order for some machine configuration interfaces.
4. Removed the variable monitorOverloadInterval from overload monitoring settings; added the variable readOverloadInteral to the machine task interval settings as a replacement.

v1.19.1 (2025-05-14)

Release date: 2025-05-14

1. Renamed Fanuc model PMi to Power Motion i-A.
2. Fixed broken hyperlinks.
3. Updated the tool life data tag combination table.
4. Added the following interfaces
    - `/core/restart`
    - `/config/settings`
    - `/config/update-settings`
    - `/config/update-security`
    - `/config/remote-access`
    - `/config/update-remote-access`
    - `/gateway/alias`

- `/gateway/update-alias`
- `/gateway/time`
- `/gateway/sync-time`
- `/gateway/time-zone`
- `/gateway/update-time-zone`
- `/gateway/system-locale`
- `/gateway/update-system-locale`
- `/gateway/lan`
- `/gateway/update-lan`
- `/gateway/wifi`
- `/gateway/update-wifi`
- `/gateway/search-wifi`
- `/gateway/connect-wifi`
- `/gateway/disconnect-wifi`
- `/gateway/static-routing`
- `/gateway/update-static-routing`
- `/gateway/connect-remote-host`
- `/gateway/disconnect-remote-host`
- `/gateway/list-file-server-items`
- `/gateway/delete-file-server-item`
- `/gateway/network-adapters`
- `/api/cnc/readLog`

5. Adjusted variable names and order for some machine configuration interfaces.

v1.18.8 (2024-11-08)

Release date: 2024-11-08

1. Added a large number of fields to the machine configuration interface.
2. Fixed a spelling error for taskEnergy in the machine configuration interface.
3. Added some fields to the get machine status monitoring settings interface.

v1.18.1 (2024-06-14)

Release date: 2024-06-14

1. Added gateway function interfaces and configuration interfaces.
2. Added description of the ID database identifier.
3. Fixed content of the batchDeleteFile (batch delete machine files) interface.

v1.18.0 (2024-04-29)

Release date: 2024-04-29

1. Adjusted some documentation content.
2. Reworked the “Settings” dialog: the former “Time Settings” is now “Time And Region Setup”. Added the “Auto Sync Time Interval” and “System Locale” settings. Reworked the “Time Server Address” setting so that several addresses can be configured.
3. On the “Communication” page, the enable toggles for the individual communication protocols were moved.

v1.17.6 (2024-04-16)

Release date: 2024-04-16

1. Added HTTP interface and RPC interface: readAllPrograms (browse all files). This interface retrieves all program names and their paths under a specified path, including those in subfolders (if the machine supports directory switching).
2. Added description of tool offset and tool life for MAZAK machines.
3. Revised the description of DNC (Professional Edition).

v1.17.5 (2024-04-01)

Release date: 2024-04-01

1. Removed the deprecated field runningPrgName from Cycle (cycle data).
2. Added field adjustedStatus to CNCStatus (machine status data).
3. On the “Tasks” page, added Machine Status Monitoring Settings.

v1.17.4 (2024-03-28)

Release date: 2024-03-28

1. On the “DNC” page, added DNC (Professional Edition).

v1.17.3 (2024-03-29)

Release date: 2024-03-29

1. Removed the deprecated field runningPrgName from Cycle (cycle data).
2. For the readPlcData interface, if a retrieved Double value is double.NaN, it was previously converted to double.Min for output; it can now be output directly as double.NaN.

v1.17.2 (2024-03-19)

Release date: 2024-03-19

1. Added HTTP interface and RPC interface: selectProgram (select program).

v1.17.1 (2024-03-19)

Release date: 2024-03-19

1. Added HTTP interface and RPC interface: batchReadErrors (batch read machine connection status).

v1.17.0 (2024-03-15)

Release date: 2024-03-15

1. Modified the response body of the batch processing interfaces batchReadOffsetData, batchReadToolLife, and batchReadToolLifeDetails: if an individual request fails, the corresponding position in the response array now contains its error information.

v1.16.5 (2024-03-08)

Release date: 2024-03-08

1. Added interface batchDeleteFile (batch delete machine files) to the file transfer interface category of the HTTP and RPC interfaces.

v1.16.4 (2024-03-08)

Release date: 2024-03-08

1. Added interface batchSendFile (batch send files to machine) to the file transfer interface category of the HTTP and RPC interfaces.

v1.16.3 (2024-03-08)

Release date: 2024-03-08

1. Added interface backupFiles (back up machine files) to the file transfer interface category of the HTTP and RPC interfaces.

v1.16.2 (2024-03-04)

Release date: 2024-03-04

1. Added a new interface category to the HTTP and RPC interfaces: historical data interfaces, including machine (machine historical data) and group-machine (historical data for all machines in a group).
2. Added response parameter time (data retrieval time) to the data read/write interfaces readTaskData (read machine task data), batchReadTaskData (batch read machine task data), readGroupTaskData (read group task data), and batchReadGroupTaskData (batch read group task data).
3. Added description of the IoTDA MQTT mode to MQTT communication.
4. Adjusted content related to the readPlcData (read PLC data) and writePlcData (write PLC data) interfaces; added descriptions for some supported CNC system models.
5. Revised parts of the documentation.

v1.16.1 (2024-02-21)

Release date: 2024-02-21

1. Restructured the document content; added the chapter “1. Important Notes” and moved descriptions common to all communication methods into the new chapter. Other chapter numbers were adjusted accordingly.
2. Added data classes AxialLoad (servo axis load), LaserPower (laser power), Offset (tool offset data), and SpindleLoad (spindle load).
3. Added new tag channel (channel number) to some data classes.
4. Updated the section numbers used when referring to the *Communication Protocol*.

v1.16.0 (2024-01-30)

Release date: 2024-02-05

1. Replaced groupNum (group number) with groupID (group identifier) in the HTTP interfaces, RPC interfaces, MQTT, and database.
2. On the “Groups” page, “Create Group” / “Edit Group” gained an “Advanced” tab.

v1.15.8 (2024-02-20)

Release date: 2024-02-20

1. On the “Network” page, added WiFi scanning and configuration.
2. Gateway management page - Network: added support for WiFi networks with Chinese SSIDs, added support for WPA3SAE networks, and fixed connection problems with common WiFi networks.

v1.15.7 (2024-01-20)

Release date: 2024-02-04

1. Changed the default root directory path for Fanuc [0i-D, 0i-F, 30i, 31i, 32i, 35i, PMi] to `//CNC_MEM/`.
2. On the “Machines” page, “Create” / “Edit Machine” - “DNC” tab, “File Server Type” gained a “Wireless Disk” option.

v1.15.5 (2024-01-22)

Release date: 2024-02-04

1. Updated the table of system models supported by the file transfer interfaces.
2. On the “Communication” page - “Gateway File Server Settings”, added the “Enable SMB” toggle.
3. On the “DNC” page, added support for sending several files at once.

v1.15.4 (2024-02-20)

Release date: 2024-02-20

1. On the “Communication” page - “Gateway File Server Settings”, added the “Enable SMB” toggle.
2. On the “DNC” page, added support for sending several files at once.

v1.15.3 (2023-12-26)

Release date: 2024-02-04

1. Added support for the subDir parameter to all file transfer interfaces.
2. Changed the types of rawRemainingCount and rawPrewarningRemainingCount in the HTTP and RPC `/ToolLife` interface from Double to Int32.
3. On the “Machines” page, “Create” / “Edit Machine” - “DNC” tab, added the “Gateway File Server” settings.
4. On the “Communication” page, added the “Gateway File Server Settings” section.

v1.15.2 (2023-12-18)

Release date: 2024-02-04

1. Added HTTP interfaces `/sendFileStream` and `/receiveFileStream` to support large file transfers.
2. On the “Communication” page, the default value of “Database Settings” - “Storage Mode” changed from “Log” to “User Defined”. This change does not affect your current settings.

v1.15.1 (2023-12-06)

Release date: 2024-02-02

1. Revised some descriptive content.
2. On the “Communication” page, “Database Settings” gained an “Enable Primary Key” toggle.
3. On the “Communication” page, “Database Settings” - “Storage Mode” gained the “User Defined” option, which lets you choose Log mode or Real Time mode for each task individually.
4. On the “Analysis” page, the order of the analyzer options changed.
5. On the “Tasks” page, “Monitor Mode” under “OEE Monitoring Settings” and “Count Monitoring Settings” now defaults to “Scheduled Reset”. This change does not

affect your current settings.

6. Revised parts of the documentation.

v1.15.0 (2023-11-28)

Release date: 2023-11-28

1. Added HTTP and RPC interface `/bathReadGroupTaskData`.
2. Added description of S7 tag support for Siemens [generic] to the `/readPlcData` interface.
3. Added response parameters remainingCycleTimeMin and remainingCycleTimeMs to the `/readTimeData` interface.
4. Added description related to Kede machines.
5. Added support for request parameter channel to the `/readProgramInfo` and `/readProgramBlock` interface descriptions.
6. Corrected a spelling error for Position in the type field of the data reporting message.
7. Fixed capitalization of keywords in the file transfer interfaces; keyword initials are now all lowercase across all interfaces.
8. Changed the address length of axis names in Modbus communication to 2, to support axis names up to 4 characters long.
9. Added content related to external PLC task data to MODBUS communication and MQTT communication.
10. On the “API Test” page, the “Program Info” and “Program Block” interfaces gained a channel option.
11. Home - “Gateway Status” panel: removed “License Description” and added “License Status”.

v1.14.0 (2023-11-01)

Release date: 2023-11-01

1. Added HTTP and RPC interfaces `/readTaskData`, `/bathReadTaskData`, and `/readGroupTaskData`.
2. Fixed an incorrect description of the data obtainable for GSK machines in the `/readTimeData` interface. Added descriptions for newly supported device system models.
3. Changed the format of the request parameter area in `/readPlcData` and `/writePlcData`.
4. Changed the format of the interface descriptions in Chapter I, HTTP Communication.
5. Changed content in section 3.1. Data Reporting Message Format.

6. On the “Machines” page, “Create” / “Edit Machine” gained an “Advanced” tab where you can set a custom machine ID and Modbus slave ID.
7. The PLC Data task format changed.

v1.13.0 (2023-10-23)

Release date: 2023-10-23

1. Added HTTP and RPC interfaces `/batchReadOffsetData`, `/bathReadToolLife`, and `/bathReadToolLifeDetails`.
2. The licensing policy changed: a single gateway can now import licenses for several different device types. When upgrading from an older version to 1.13.0 or later, the license file must be regenerated.
3. Machine types other than CNC can now join a group.
4. On the “Machines” page, the “Task” tab of the “Create PLC Machine” / “Edit PLC Machine” dialog gained the “Machine Status”, “OEE Monitoring” and “Cumulative Status Time” tasks.
5. On the “Machines” page, the “IO Module” option of the System field moved from “Create CNC Machine” / “Edit CNC Machine” to the System field of “Create PLC Machine” / “Edit PLC Machine”. Machines added as “IO Module” in an older version must be configured again.

v1.12.0 (2023-09-26)

Release date: 2023-09-26

1. Added HTTP and RPC interface `/readTaskData`, which retrieves the most recent data for the corresponding task from the gateway cache.
2. Added support for grouping intervals to the data analysis interfaces. Added output variables StartTime and EndTime to all analysis interfaces.
3. On the “DNC” page, added the “New folder” and “Delete folder” buttons.
4. On the “Network” page, “LAN1 Settings” and “LAN2 Settings” gained DNS server settings; a new status indicator shows whether the LAN port is connected (“Connected” means the LAN adapter has detected a device at the other end of the cable; it does not guarantee that the device can be pinged).
5. On the “Machines” page, the “Task” tab of the “Create CNC Machine” / “Edit CNC Machine” dialog gained the “Tool Offset” task.
6. On the “Tasks” page, “Machine Task Interval Settings” gained “Tool Offset (ms)”.

v1.11.0 (2023-09-06)

Release date: 2023-09-06

1. Adjusted some RPC data analysis interfaces to align with the corresponding HTTP interfaces.
2. Added support for binary files to the `/receiveFile` and `/sendFile` interfaces.
3. On the “Machines” page, the layout of the “DNC” tab in the “Create” / “Edit Machine” dialog was reworked.

v1.10.0 (2023-08-22)

Release date: 2023-08-22

1. Added HTTP and RPC interface `/readFeed`, applicable to CNC machines and laser cutting machines, to retrieve feed data.
2. Added HTTP and RPC interface `/readEnergyConsumption`, applicable to CNC machines, to retrieve energy consumption data.
3. Added HTTP and RPC interface `/readLaserPower`, applicable to laser cutting machines, to retrieve preset power and actual power data for laser cutting machines.
4. Added support for laser cutting machine laser power data to MODBUS, MQTT, database, and other communication methods.
5. Fixed an address error for group data in MODBUS; the address offset between groups is 100, not 1000.
6. Added support for multi-channel systems to some HTTP and RPC interfaces.
7. Changed Modbus communication: address positions previously at 3000 and above were adjusted to 1000 and above.
8. On the “Machines” page, “Create” / “Edit Machine” gained an “External PLC” tab for setting up serial data collection.
9. Added support for laser cutters. The “Machines” page gained a “Create Laser Cutter” button. The “Task” tab of the “Create Laser Cutter” / “Edit Laser Cutter” dialog gained the “Laser Power” task. “Machine Task Interval Settings” on the “Tasks” page gained “Laser Power (ms)”. The “API Test” page gained a “Laser Power” interface option for laser cutters.
10. On the “Machines” page, for laser cutters, robots and PLCs, the System field on the “General” tab of the “Create” / “Edit Machine” dialog gained a “Mock” option.
11. On the “Network” page, WiFi Settings, the “Status” section now shows the current WiFi IP address.

12. On the “API Test” page, some interfaces gained a “Channel” option, which defaults to 0, the main channel.

v1.9.3 (2023-07-10)

Release date: 2023-07-10

1. Removed content related to per-machine write mode in database communication. In the new version, the old per-machine write mode operates as log mode. (Bivrost Gateway Manual)
2. Added the ToolLife and ToolLifeDetails data classes under MQTT communication.
3. Added interfaces `/createDir` and `/deleteDir`.
4. Added machine status MANUAL_SINGLE_BLOCK (setup status: single block execution).
5. Removed the per-machine write option from database communication. Setups that used per-machine write in an older version now run in Log mode.

v1.9.2 (2023-06-28)

Release date: 2023-06-28

1. Added support for the current tool offset number to HTTP, MODBUS, MQTT, database, and other communication methods.

v1.9.1 (2023-06-25)

Release date: 2023-06-25

1. Changed the MQTT communication protocol data class Plc to PLC. This change is not backward compatible with the old Plc data class.

v1.9.0 (2023-06-21)

Release date: 2023-06-21

1. Changed the MQTT communication data reporting message format. After this change, all data classes except OverLoadS and Plc remain backward compatible with the old version.
2. Added data class AxialOverLoad to the MQTT communication protocol.
3. Changed the MQTT communication protocol data class OverLoadS to SpindleOverLoad. This change is not backward compatible with the old OverLoadS data class.

4. Added address positions to Modbus, including rapid feed override and cumulative overload time for axes 1 through 18.
5. Adjusted address positions in Modbus, including spindle load and spindle cumulative overload time.

v1.8.2 (2023-06-20)

Release date: 2023-06-20

1. Removed alarmTime from alarmHistory in MODBUS, database, and MQTT.

v1.8.1 (2023-06-19)

Release date: 2023-06-19

1. Removed alarmTime from the `/readAlarm` interface. Removed alarmTime from alarmLog in MODBUS, database, and MQTT.
2. To obtain a machine’s alarm start time, use the HTTP or RPC interface analysis/alarm, or calculate the time from the alarm activation/clearance records in alarmHistory.

v1.8.0 (2023-05-12)

Release date: 2023-05-12

1. Fixed the default root directory path for Mitsubishi system CNC.
2. Added HTTP and RPC interfaces `/readToolLife` and `/readToolLifeDetails`; tool-life-related variables from the original `/readToolOffset` interface were moved to the new interfaces.
3. Fixed keyword content to Content in `/sendFile` and `/receiveFile`.
4. Added interfaces group-analysis/oee, group-analysis/alarm, group-analysis/count, group-analysis/overall, and analysis/overall.
5. Added variable programStack to the `/readProgramInfo` interface.
6. Reordered the HTTP communication interfaces, dividing HTTP interfaces into three categories: status read/write, file transfer, and data analysis.
7. On Home, the data upload services in “Gateway Status” gained a “Delayed” state.
8. The top toolbar gained a “Diagnosis” button.
9. On the “Machines” page, the “Task” tab of the “Create CNC Machine” / “Edit CNC Machine” dialog gained the “Tool Life” task.

10. On the “Tasks” page, added “Tool Life Monitoring Settings”.
11. On the “Tasks” page, “Machine Task Interval Settings” gained “Tool Life (ms)”.
12. On the “Tasks” page, “OEE Monitoring Settings” gained the break time settings.
13. On the “Network” page, “WiFi Settings” now shows signal strength.
14. The “API Test” page gained the “Tool Life” and “Tool Life Details” interfaces.
15. The “Analysis” page gained the Overall Analysis analyzer.

v1.7.5 (2023-03-10)

Release date: 2023-03-10

1. Modified the HTTP and RPC interface `/readAlarm`, adding output field alarmLevel (alarm level).
2. Added an alarm level variable to the Alarm data in MODBUS, MQTT, and the database.
3. Added an alarm level variable to AlarmHistory in MQTT and the database.
4. Added machine cumulative alarm time to MODBUS, MQTT, and the database; the machine alarm information task must be activated before use.
5. Added machine cycle data to MODBUS, MQTT, and the database; the machine cycle data task must be activated before use.
6. Modified the HTTP and RPC interfaces `/readPlcData` and `/writePlcData`, adding support for reading and writing by variable name on some machine models.
7. Added HTTP interface `/api/analysis/count`.
8. Added more RPC interface support; now every HTTP interface has a corresponding RPC interface.
9. The “Tasks” page gained an “Alarm Monitoring Settings” section.
10. The “Machines” page gained a “Create Robot” button, adding support for collecting data from robots.
11. Reworked the “Settings” dialog: added a “More” tab, moved the “Default Config” button into it, and added the “Backup Config” and “Restore Config” buttons.
12. Added a “Restart Service” reminder.
13. The shutdown button on Home was merged into the new “Gateway Shutdown/Restart” button.
14. The “Analysis” page gained “Count Analysis”.
15. Added content to the chapter *7. FAQ*.
16. On the “Machines” page, the “Task” tab of the “Create” and “Edit Machine” dialogs gained “Advanced” settings.

v1.7.0 (2022-12-12)

Release date: 2022-12-12

1. Changed the request body structure of HTTP interfaces `/writePlcData`, `/writeOffsetData`, and others.
2. Changed the response structure of the HTTP interface `/readOffsetData`.
3. Changed the request message structure of RPC interfaces `/writePlcData`, `/writeOffsetData`, `/sendFile`, and others.
4. On the "Machines" page, "Count" on the "Task" tab of the "Create CNC Machine" / "Edit CNC Machine" dialog gained the "Overload" task.
5. The "Tasks" page gained "Count Monitoring Settings" and "Overload Monitoring Settings".
6. On the "API Test" page, the "Read" button became "Send".
7. On the "API Test" page, the "PLC Data" interface option became "Read PLC Data".
8. The "API Test" page gained the "Write PLC Data", "Read Offset Data" and "Write Offset Data" options.
9. "Advanced Options" in the "Settings" dialog gained the permission toggles "Allow Write PLC Data" and "Allow Write Offset Data".

v1.6.2 (2022-12-09)

Release date: 2022-12-09

1. Added HTTP interface `/writePlcData`.
2. Added RPC interfaces `/readPlcData` and `/writePlcData`.
3. Added upload data variables CurrentCount (current machining count) and spindle overload time `OverLoadS<i>` to Modbus, MQTT, and database communication.
4. Added support in MQTT communication for PLC data to use tag names configured in the task command.

v1.6.1 (2022-11-11)

Release date: 2022-11-11

1. Modified the HTTP interfaces `/readOffsetData` and `/writeOffsetData`, adding optional input ToolNum to support tool offset read/write on Siemens machines.
2. Revised the RPC interface descriptions.

3. The “Settings” dialog gained a “Manage Users” button for adding, editing and deleting gateway users and assigning their permissions.

v1.6.0 (2022-10-31)

Release date: 2022-10-31

1. Changed the return result of the HTTP interface `/readPlcData`: previously all data types returned a UInt16 array; now the data is returned according to the data type specified by the user.
2. Removed support for the Char data type from the HTTP interface `/readPlcData`. To obtain Char data, users can retrieve Byte data and convert it themselves.
3. Changed support for the String type in the HTTP interface `/readPlcData`. When the type is String, Count represents the maximum length (in bytes) of the data to read.
4. Integrated support for retrieving machine macro variables (local variables, common variables, etc.) into the HTTP interface `/readPlcData`.
5. Added machine cumulative status time and group cumulative status time to MODBUS, MQTT, and database communication.
6. Adjusted the address length of all MODBUS communication mappings to expand capacity. The PLC data mapping address was changed from 4000 to 20000, and its length from 1000 to 10000.
7. Added support for 64-bit format data to MODBUS communication.
8. Adjusted the PLC data tags and content in MQTT communication.
9. Adjusted the RPC interface request and response formats.
10. Added RPC interfaces `/analysisOEE` and `/analysisAlarm`.
11. Added machine status MANUAL_RAPID.
12. On the “Machines” page, the “Create” button was split into “Create CNC” and “Create PLC”. “Create CNC” keeps the behaviour of the original “Create” button; “Create PLC” opens the PLC configuration dialog, adding support for collecting data from PLC devices.
13. Added offline gateway upgrade.
14. On the “Communication” page, “MODBUS Settings” gained a “Byte(8) Order” option.
15. On the “Communication” page, “Database Settings” gained a “Storage Mode” option; the former per-machine write option was folded into “Storage Mode”.

v1.5.9 (2022-09-21)

Release date: 2022-09-21

1. Added cumulative alarm count to MODBUS, MQTT, and database communication.

v1.5.8 (2022-08-17)

Release date: 2022-09-15

1. Added HTTP interface `/readProgramBlock`. Added corresponding content for this interface to MODBUS, MQTT, and database communication.
2. Added HTTP interface `/api/analysis/alarm`.
3. On the “Machines” page, the “Task” tab of the “Create” and “Edit Machine” dialogs gained the “Program Block” option.
4. On the “Tasks” page, “Machine Task Interval Settings” gained the “Current Program Block (ms)” setting.
5. The “API Test” page gained the “Program Block” option.
6. The “Analysis” page gained the “Alarm Analysis” option.
7. On the “Communication” page, “Database Settings” gained a “Use Local Time” toggle.

v1.5.7 (2022-08-03)

Release date: 2022-08-08

1. Added description of String type data support to the HTTP interface `/readPlcData`.
2. Added RPC interface `/readCurrentProgram`.
3. Added content related to database communication.
4. Fixed an issue with incorrect axis name output in MODBUS.
5. Moved the FAQ and known issues content from the *User Commissioning Manual* into this document.
6. The “Communication” page gained “Database Settings”.
7. The “Settings” dialog gained the “LAN2 Settings” toggle.
8. The default gateway alias changed to “iotgw”.
9. On the “Machines” page, the “General” tab of the “Create” and “Edit Machine” dialogs gained the “Encoding” setting.

v1.5.6 (2022-07-13)

Release date: 2022-07-27

1. Updated references to the Bivrost Gateway Manual due to changes in the manual.
2. Added description of the file server type for program transfer. Changed the default root directory path for Fanuc [0i-D, 0i-F, 30i, 31i, 32i, PMi].
3. Updated the description of the HTTP interface `/readPlcData`.
4. Rebuilt the table on the “Machines” page: added sorting, search and pagination; the “Status” column became the “Activate” column, which shows the activation state as an icon; added a “Connection” column, which shows the connection state as an icon and refreshes every ten seconds; added a “Task Count” column showing the number of tasks on the machine; the former “Edit” and “Delete” buttons are now icons; hovering over an icon shows a tooltip; the former “Machine Task Settings” dialog moved into the “Task” tab of the “Edit Machine” dialog; the “Edit Machine” dialog gained a “DNC” tab.
5. Rebuilt the table on the “Groups” page: added sorting, search and pagination; the “Status” column became the “Activate” column, which shows the activation state as an icon; added a “Machine Count” column showing the number of machines in the group; added a “Task Count” column showing the number of tasks on the group; the former “Edit” and “Delete” buttons are now icons; hovering over an icon shows a tooltip; the former “Group Task Settings” dialog moved into the “Task” tab of the “Edit Group” dialog.
6. “DNC” gained support for Shared Folder, Shared Folder (Win XP) and FTP Server.
7. On the “API Test” page, “Data Type” for reading “PLC Data” changed from a text field to a dropdown.
8. On the “DNC” page, “Root Directory” is read-only and only shows the “Root Directory” configured on the “Machines” page - “Edit Machine” dialog - “DNC” tab.
9. Updated *3. Specifications*.

v1.5.5 (2022-06-19)

Release date: 2022-06-24

1. Updated references to the Bivrost Gateway Manual due to changes in the manual.
2. Revised the description of file transfer interfaces in Chapter I, HTTP Communication.
3. Added content on RPC interfaces for MQTT communication, section 3.3. RPC Interface.
4. Reworked the layout of the top toolbar and Home: “Change Username & Password”, “Change Gateway Alias”, “Create Default Config” and “Advanced Options” moved into the new “Settings” dialog.

5. The “Settings” dialog gained “Time Settings”.
6. On the “Communication” page, “MQTT Settings” gained the RPC interface settings.

v1.5.4 (2022-06-17)

Release date: 2022-06-17

1. The following HTTP interface names changed while their addresses remained the same:
    - 1.5. CNC System Time Data renamed to Machine Time Data;
    - 1.6. CNC Status renamed to Machine Status;
    - 1.14. CNC Local Program List renamed to Machine Local Program List;
    - 1.15. Receive CNC File renamed to Receive Machine File;
    - 1.16. Send File to CNC renamed to Send File to Machine;
    - 1.17. Delete CNC Local File renamed to Delete Machine Local File;
    - 1.18. Lock/Unlock CNC Local Program Editing renamed to Lock/Unlock Machine Local Program Editing
2. Revised the description of machine OEE and group OEE.
3. Added English language support.
4. Revised the Chinese content. The “Collection” page became the “Tasks” page. On the “Machines” and “Groups” pages, the “Settings” button in the Operations column became the “Task” button.
5. Machine OEE was split out from group OEE and is now enabled and disabled in “Machine Task Settings”.
6. On the DNC page, the “Use” button on a folder in the file list became “Open folder”; clicking it lists the subfolder directly.
7. Revised *4. Networking*.
8. Added *6. Appendices* and moved the glossary into that chapter, adjusting the rest of the document accordingly.

v1.5.3 (2022-05-31)

Release date: 2022-05-31

1. Changed the input parameters of the HTTP interface `/readPlcData`.
2. Added CNC status “Setup: Trial Run”.
3. Revised the support list and added “PLC Data”.
4. Revised *5.1. Signing In*.

5. Adjusted the “PLC Data” documentation to reflect the changes to the “PLC Data” interface.

v1.5.2 (2022-05-09)

Release date: 2022-05-09

1. Changed the type of load data axialLoad and spindleLoad from Int32 to Float.
2. Added HTTP interface `/readPlcData`; added PLC data output to MODBUS and MQTT.
3. Added HTTP interface `api/analysis/oee`.
4. Added content for referencing the gateway name and custom group name to MQTT communication.
5. Removed content related to multi-channel systems.
6. On the “Machines” page, the “Auto Collection Settings” dialog gained the “PLC Data” option.
7. On the “Collection” page, “Machine Collection Interval Settings” gained the “PLC Data (ms)” setting. “OEE Monitoring Settings” gained the “Availability Mode” option.
8. On the “Communication” page, “MQTT Settings” gained the “Use Gateway Alias” toggle.
9. The “API Test” page gained the “PLC Data” option.
10. Added the “Analysis” page.

v1.5.1 (2022-04-26)

Release date: 2022-04-26

1. Changed the description of OEE monitoring settings.
2. On the “Collection” page, “Data Statistics Settings” became “OEE Monitoring Settings”. Added “Monitoring Time Settings”, with the options “Time Window” and “Scheduled Reset”.

v1.5.0 (2022-04-20)

Release date: 2022-04-20

1. Changed the tool-offset-related interfaces in the HTTP protocol.
2. Changed the message content structure of the MQTT protocol.
3. Split the MQTT protocol data type AxisData into FeedAndSpindle, Load, and Position.

4. MQTT protocol: renamed AlarmHistory to Alarm History.
5. MQTT protocol: changed keyword CurrentToolNumber.
6. Added error codes related to file transfer.
7. Revised some text content.
8. On the “Machines” page, in the “Create” and “Edit Machine” dialogs, the “System” field was split into “System” and “Model”.
9. Reworked the “Auto Collection Settings” dialog on the “Machines” page and added more collection options.
10. On the “Collection” page, “Machine Collection Interval Settings” gained interval settings for the additional collection options.
11. On the “Communication” page, MQTT Settings gained the “Publish on Value Change” toggle.
12. On the Network page, “WiFi Settings” gained a “Disconnect” button.

v1.4.3 (2022-03-28)

Release date: 2022-03-28

1. Revised some text content.
2. Reworked the Home layout: moved the “Restart Service”, “Gateway Shutdown” and “Create Default Config” buttons to the first row of Home.
3. Reworked the content and position of “Network Status” on Home.
4. “Advanced Operations” on Home became “Advanced Options”.
5. The “Local Caching” and “Remote Assistance” toggles moved from “Network Status” on Home to “Advanced Options”.
6. In the navigation bar, the “Collection Test” page became the “API Test” page, with more interfaces available to call.
7. Added the “DNC” page to the navigation bar.

v1.4.2 (2022-03-15)

Release date: 2022-02-19

1. Fixed the data types of some output values in the MODBUS and MQTT protocols.
2. Fixed the wording of CNC status, CNC running status, and alarm status.
3. Added a new chapter, Frequently Asked Questions.
4. Fixed some descriptions in Common Errors.

5. Extended and revised parts of the “Collection” content.

v1.4.1 (2022-02-15)

Release date: 2022-02-15

1. Added output values GroupCount, GroupOEE, OEE, and others to the MODBUS and MQTT protocols.
2. Divided MODBUS protocol output values into machine-related data and group-related data.
3. Added supplementary description to the MQTT protocol.
4. Fixed incorrect information in the AxisData section of the MQTT protocol.
5. Added Chapter V, Virtual Machine Tool Testing.
6. Updated the list of supported machines.
7. “Gateway Status” on Home gained an “Enable/Disable Local Caching” toggle.
8. “Gateway Status” on Home gained an “Enable/Disable Remote Assistance” toggle.
9. Added the “Groups” configuration page.
10. Reworked the “Collection” page: it is now split into “Machine Collection Interval Settings”, “Group Collection Interval Settings” and “Data Statistics Settings”, with several new settings.
11. The “Communication” page gained the “Cloud Settings” option.

v1.3.5 (2021-12-27)

Release date: 2021-12-27

1. Changed all alarm time results from second precision to millisecond precision.
2. Removed decPoint from the HTTP interface ReadPosition.
3. Modified the HTTP interface ReadProgramList, adding response values dirAtCNC and subDirs, while keeping the original programs response structure unchanged to maintain backward compatibility with the old interface.
4. Added Modbus protocol output address 40260, running program block sequence number.
5. Added Modbus protocol output address 40500, cumulative production count.
6. Home gained a network status panel.
7. Unified the names and the order of the collected data items across all pages.
8. On the Collection page, “Frequency” was renamed “Interval”.

9. On the Communication page, the MQTT protocol now supports changing the encoding, and the default encoding changed from gb2312 to utf8.