Bivrost IoT Gateway

彼络物联网关

Communication Protocol

Document Version V1.19.7

Bivrost Technologies Company Limited

深圳市彼络科技有限公司

Scan for support on WeChat

Contents

Introduction

This document is the communication protocol reference for the Bivrost gateway, version v1.19.7, intended for developers who need to integrate with the gateway programmatically. For instructions on installing and configuring the gateway and using its web management interface, please refer to the Bivrost Gateway Manual (referred to below as the “Manual”).

Communication Methods

HTTP Communication

Actively read/write machine data, transfer files, analyze data, and configure the gateway via HTTP interfaces. All features have corresponding interfaces.

MODBUS Communication

The gateway acts as a MODBUS server (port 502), storing automatic data collection task results in the corresponding address locations.

MQTT Communication

The gateway publishes automatic data collection task results as JSON messages to the specified MQTT server, and supports an RPC interface.

Database Communication

The gateway writes automatic data collection task results to the specified database (InfluxDB, MySQL, SQL Server, PostgreSQL, etc.).

MCP Service

The gateway's built-in MCP server exposes machine data, data analysis, and gateway status to AI clients as read-only tools.

> **Note**
>
> MODBUS, MQTT, and database communication all output the results of **automatic data collection tasks**. Before use, you need to enable the automatic data collection task for the target machine/group on the gateway management page (see the Manual, 5.3.1.2. Task Settings and 5.4.1.2. Group Task Settings) and restart the service.

Reading Guide

1. Start with 1. Important Notes for the common identifiers, data classes, and variable definitions.
2. Then consult the section for the communication method you use.
3. If you don’ t have an actual machine tool available, use the mock machine to quickly test the communication protocol.

More Resources

FAQ →

Common errors and troubleshooting steps for each communication method.

Version Change History →

Protocol and documentation changes in each gateway version.

Bivrost Gateway Manual →

Instructions for installing and configuring the gateway and using its web management interface.

1. Important Notes

1.1. Identifiers

While reading this documentation, some explanations may refer to the Bivrost Gateway Manual (referred to hereafter as the “Manual”).

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).

1.1.1. 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 the Bivrost Gateway Manual 5.3.1.5. 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.

1.1.2. 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 the Bivrost Gateway Manual 5.4.1.3. Advanced Settings). By default, the groupID consists of the letter “g” followed by the group number, e.g. “g2”.

1.1.3. 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 the Bivrost Gateway Manual 5.3.1.5. Advanced Settings). By default, the slaveID is the fourth octet of the machine’s IP address.

1.1.4. 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.

1.1.5. 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 2.9.1.1. 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.

1.2. 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.

1.2.1. 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.

1.2.2. AlarmLog: Current Alarm

Current alarm `<field>` data fields:

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

1.2.3. 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

1.2.4. 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 the Bivrost Gateway Manual 5.5.8. 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 2.5.1.3. readCNCStatusDetails - Read Machine Status Details, 2.5.2.1. readTaskData - Read Machine Task Data, or 2.5.2.2. 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

1.2.5. 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.

1.2.6. 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

1.2.7. 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.

1.2.8. 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.

1.2.9. 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

1.2.10. 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

1.2.11. 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 the Bivrost Gateway Manual 6.1.3. Group Machining Count for details.

1.2.12. GroupCumulativeTime: Group Cumulative Status Time

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

Field	Type	Description
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 sums of each status time, accumulated since the gateway began automatically collecting data from this group, across all devices in the group.

1.2.13. 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 the Bivrost Gateway Manual 6.1.2. Group OEE Data for details.

1.2.14. LaserPower: Laser Power

Laser power `<field>` data fields:

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

1.2.15. 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

1.2.16. LogHistory: Log History

Log history `<field>` data fields:

Field	Type	Description
msg	String	Log content

1.2.17. 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 the Bivrost Gateway Manual 6.1.1. Machine OEE Data for details.

1.2.18. 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

Field	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 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

System Model	toolNum (tool number)	offsetNum (offset number)
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
Haas 哈斯	X	O
Kede 科德	O	X
Knd 凯恩帝	X	O
Lynuc 铼纳克	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.

1.2.19. 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 the Bivrost Gateway Manual 6.2.1. 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.

1.2.20. 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

1.2.21. 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

1.2.22. 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; supported on Siemens 西门子 only ([OPC UA], [General] (840D sl / 828D, S7 protocol), [810D/840D])

Current program `<tag>` data tags:

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

1.2.23. 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

1.2.24. 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]

The time data supported by each system model is shown in the table below, where:

O (shown as ✅ in the table below): time data supported by the machine. 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.

Time data supported by each system model (✅ = supported, blank = not supported):

System Model	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Machining (Cutting) Time	Current Machining (Cutting) Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Description	Total accumulated power-on time	Power-on time counted from this power-on	Total accumulated auto run time	Time in the current auto run state	Total accumulated cutting time	Cutting time in the current auto run state	Cycle time of the previous workpiece	Elapsed cycle time of the current workpiece	Elapsed cutting time of the current workpiece	Remaining time of the current cycle
Bosunman 博尚 [DF Series]				✅				✅		
Bosunman 博尚 [BSK Series]	✅	✅	✅	✅				✅		
Brother 兄弟 [General]	✅		✅	✅						
Brother 兄弟 [S Series]	✅		✅	✅				✅	✅	
Delta 台达			✅	✅						
Dmg Mori 德玛吉森精机 [730BM]										
Dmg Mori 德玛吉森精机 [OPC UA]	✅		✅					✅		
Fagor 法格 [8035, 8040, 8055]								✅		
Fagor 法格 [8060, 8065, 8070]		✅						✅		
Fanuc 发那科	✅		✅		✅			✅ (must be reset at the end of the program to clear)		
Gsk 广州数控				✅		✅				
Haas 哈斯 [General, Serial]	✅		✅				✅	✅		
Haas 哈斯 [MT-CONNECT]							✅	✅		✅
Heidenhain 海德汉	✅		✅							
Hnc 华中数控										

System Model	Cumulative Power-On Time	Current Power-On Time	Cumulative Operating Time	Current Operating Time	Cumulative Machining (Cutting) Time	Current Machining (Cutting) Time	Last Cycle Time	Current Cycle Time	Current Workpiece Cutting Time	Remaining Cycle Time
Jingdiao 北京精雕			✅	✅		✅		✅		
Kede 科德		✅						✅		✅
Knd 凯恩帝			✅	✅						
Lnc 宝元					✅			✅		
Lynuc 铼纳克				✅		✅		✅		
Mazak 马扎克 [MT-CONNECT]	✅		✅		✅					
Mazak 马扎克 [Smart, Smooth]	✅	✅	✅	✅	✅					
Mitsubishi 三菱	✅		✅		✅			✅		
Mock 模拟机台	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅
Okuma 大隈	✅		✅		✅					
Rexroth 力士乐 [OPC UA]				✅				✅		
Siemens 西门子	✅	✅		✅	✅			✅		
Syntec 新代	✅	✅		✅ (auto run time of the current program)	✅		✅	✅		

1.2.25. 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

Field	Type	Description
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], Heidenhain only
rawTimeLimit2	Double	Maximum life of the called tool [time], Heidenhain only
rawCurrentTime	Double	Current life [time]
rawOverTime	Double	Tool life exceeded [time], Heidenhain only

Field	Type	Description
rawOverCount	Int32	Tool life exceeded [count], Siemens only (user-defined module)
rawPrewarningTime	Double	Prewarning life [time], Brother only
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, Heidenhain only

For the meaning of each field, see the examples in 2.5.1.21. readToolLife - Read Tool Life and 2.5.1.22. readToolLifeDetails - Read Tool Life Details.

Raw data is returned in the following cases:

1. Using the 2.5.1.22. readToolLifeDetails - Read Tool Life Details, 2.5.1.24. batchReadToolLifeDetails - Batch Read Tool Life Details, 2.5.2.1. readTaskData - Read Machine Task Data, or 2.5.2.2. 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 铼纳克	X	X	O	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.

Life types supported by each system model (tool life types):

System Model	Count	Time	Wear
Brother 兄弟	O	O	X

System Model	Count	Time	Wear
Fanuc 发那科	O	O	X
Gsk 广数	O	O	X
Heidenhain 海德汉	X	O	X
Kede 科德	O	O	O
Lynuc 铼纳克	O	O	X
Mock 模拟机台	O; ToolNum 1~10	O; ToolNum 11~20	O; ToolNum 21~30
Okuma 大隈	O	O	O
Siemens 西门子	O	O	O
Syntec 新代	O	O	X

O: supported;

X: not supported.

1.3. Variable Definitions

1.3.1. Machine Status

1.3.1.1. 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.

1.3.1.2. 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)

1.3.1.3. 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)

1.3.1.4. DryRunStatus: Dry Run Status

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

1.3.2. Alarm Information

1.3.2.1. AlarmLevel: Alarm Level

Alarm levels, in descending order of priority, are Error (ERR), Warning (WRN), and Message (INF). Users can refer to the Bivrost Gateway Manual 5.5.7. 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

1.3.3. Service Information

1.3.3.1. 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

2. HTTP Communication

2.1. 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 1.1.1. 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.

2.2. 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:

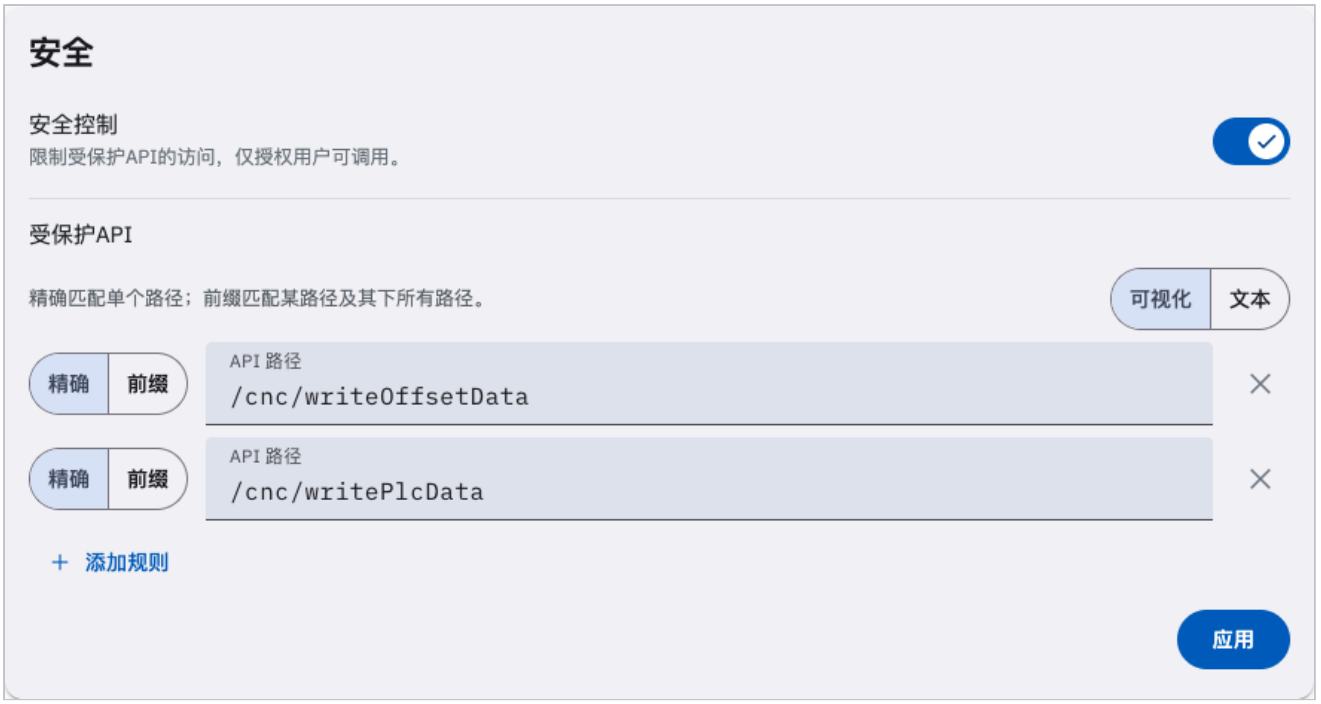
Common Error Codes

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

Error Code	Message	Description
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
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

2.3. 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.

2.3.1. 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.

2.3.2. 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.

2.3.3. 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 2.9.2. Communication Settings.

2.4. Authentication Endpoints

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

2.4.1. 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.

2.4.2. 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

2.5. 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.

2.5.1. Direct Read/Write

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

2.5.1.1. readAlarm - Read Alarm Information

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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 2.7.1.2. alarm - Machine Alarm Data Analysis.

2.5.1.2. readCNCStatus - Read Machine Status

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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 2.5.2.1. readTaskData - Read Machine Task Data or 2.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

2.5.1.3. 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 1.1.1. 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 2.5.2.1. readTaskData - Read Machine Task Data or 2.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

2.5.1.4. readCount - Read Machining Count

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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 2.5.2.1. readTaskData - Read Machine Task Data or 2.5.2.2. batchReadTaskData - Batch Read Machine Task Data, or via MODBUS, MQTT, database, or other means.

2.5.1.5. readCurrentToolNumber - Read Current Tool Number

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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

2.5.1.6. 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 1.1.1. 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]

2.5.1.7. 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 1.1.1. 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

2.5.1.8. 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 1.1.1. 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.

2.5.1.9. 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 1.1.1. machineID Machine Identifier

Response Example

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

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

2.5.1.10. 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 1.1.1. 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 2.5.1.15. readPosition - Read Position Data.

2.5.1.11. 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 1.1.1. 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.

2.5.1.25. 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 1.1.1. 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 on Fanuc [15, 15i]; secondary mode on Rexroth [OPC UA]
readyStatus	String	Ready status, KND only
alarmOnlyStatus	String	Alarm status (excluding warnings), LNC only
emergencySetStatus	String	Emergency-stop setting status; LNC, Lanhao and Mazak [640] only
lightColor1	String	Color of signal light 1, Dmg Mori only
lightStatus1	String	Status of signal light 1, Dmg Mori only
lightColor2	String	Color of signal light 2, Dmg Mori only
lightStatus2	String	Status of signal light 2, Dmg Mori only
lightColor3	String	Color of signal light 3, Dmg Mori only
lightStatus3	String	Status of signal light 3, Dmg Mori only
lightColor4	String	Color of signal light 4, Dmg Mori only
lightStatus4	String	Status of signal light 4, Dmg Mori only

> **Note**
>
> The system-specific parameters in the table are returned only on machines running the corresponding control system; other machines do not return them.

2.5.1.26. 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 1.1.1. 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.

2.5.1. Direct Read/Write: Tool Offsets, Coordinates, Program & PLC

This page continues from 2.5.1. Direct Read/Write and covers endpoints 2.5.1.12–2.5.1.20: tool offset data, coordinate data, program block/program information, PLC data read/write, and machine time data.

2.5.1.12. 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 1.1.1. 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.

2.5.1.13. 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 1.1.1. 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.

2.5.1.14. 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 1.1.1. 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

2.5.1.15. readPosition - Read Coordinate Data

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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.

2.5.1.16. 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 1.1.1. 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

2.5.1.17. readProgramInfo - Read Machining Program Information

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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; currently only supported on Siemens and the simulated machine
channel	Int32	Machine channel number; appears only when channel is supplied in the request and is not 0

For Fanuc, 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.

2.5.1.18. 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 1.1.1. 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`
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`
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	`$MACRO$`	

System Model	Area	area Example	Notes
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									*	
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			*	

Data Type	Bit	Byte	SByte	Int16	UInt16	Float	Int32	UInt32	Int64	UInt64	Double	String
Mazak [Smart, Smooth]							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.

Response Parameter	Type	Description
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.

2.5.1.19. 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 1.1.1. 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

2.5.1.20. readTimeData - Read Machine Time Data

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

Request Parameter	Type	Description
machineID	String	(Required) Target machine identifier, see 1.1.1. 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 1.2.24. TimeData: Machine Time Data for the calculation method.

2.5.1. Direct Read/Write: Tool Life

This page continues 2.5.1. Direct Read/Write, covering tool-life-related endpoints (2.5.1.21–2.5.1.24).

2.5.1.21. 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 1.1.1. 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 1.2.25. 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 1.2.25. 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 1.2.25. ToolLife: Tool Life Data.

2.5.1.22. 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 1.1.1. 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 1.2.25. 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 1.2.25. 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 1.2.25. ToolLife: Tool Life Data.

2.5.1.23. 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 1.1.1. 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 1.2.25. 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 1.2.25. 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 1.2.25. ToolLife: Tool Life Data.

2.5.1.24. 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 1.1.1. 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 1.2.25. 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 1.2.25. 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 1.2.25. ToolLife: Tool Life Data.

2.5.2. 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.

2.5.2.1. 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 1.1.1. machineID Machine Identifier
type	String	(Required) Data class, see 1.2. Data Description
tag	String	Field form of the tag, see 1.2. 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 1.2. Data Description.

2.5.2.2. 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 1.1.1. machineID Machine Identifier
Type	String	(Required) Data class, see 1.2. Data Description
Tag	String	Field form of the tag, see 1.2. 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 1.2. Data Description.

2.5.2.3. 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 1.1.2. groupID Machine Group Identifier
type	String	(Required) Data class, currently only 1.2.11. GroupCount: Machine Group Processing Count Data, 1.2.12. GroupCumulativeTime: Machine Group Cumulative Status Time, and 1.2.13. GroupOEE: Machine Group OEE are supported
tag	String	Field form of the tag, see 1.2. 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 1.2. Data Description.

2.5.2.4. 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 1.1.2. groupID Machine Group Identifier
type	String	(Required) Data class, currently only 1.2.11. GroupCount: Machine Group Processing Count Data, 1.2.12. GroupCumulativeTime: Machine Group Cumulative Status Time, and 1.2.13. GroupOEE: Machine Group OEE are supported
tag	String	Field form of the tag, see 1.2. 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 1.2. Data Description.

2.5.2.5. 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 1.1.1. 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

2.5.2.6. 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

2.6. 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 the Bivrost Gateway Manual 5.3.1.3. 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 the Bivrost Gateway Manual 5.3.1.3. 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`

File Server Type	System [Model]	Target Directory Path Supported	Default Root Directory Path
Machine storage	Haas [general, MT-CONNECT]	X	None
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	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], 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.

2.6.1. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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	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
Shared folder (Win XP)		O	O

File Server Type	System [Model]	Retrieve File List	Retrieve Subfolder List
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] returns 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.

2.6.2. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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.

2.6.3. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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.

2.6.4. 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 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 the Bivrost Gateway Manual 5.3.1.3. 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

Response Parameter	Type	Description
errorCode	Int32	Error code, 0 indicates success.
errorMsg	String	Error message
statusMsg	String	Error details

2.6.5. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.6.6. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.6.7. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.6.8. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.6.9. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.6.10. lockFileByRange — Lock/Unlock Machine Program Editing

Currently, this only supports Fanuc 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 1.1.1. machineID Machine Identifier
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

2.6.11. 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 1.1.1. machineID Machine Identifier
dirName	String	(Required) Name of the directory to create
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 the Bivrost Gateway Manual 5.3.1.3. 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	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

2.6.12. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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.

2.6.13. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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)"
}
```

2.6.14. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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.

Response Parameter	Type	Description
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]	
Mock machine	Always returns SUCCESS, but does not change the current program.
Okuma [all models]	The mode execution mode parameter must be supplied.
Syntec	

2.6.15. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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.

2.6.16. 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 1.1.1. 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 the Bivrost Gateway Manual 5.3.1.3. 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

2.7. 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 the Bivrost Gateway Manual 5.12.2.3. 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.

2.7.1. 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`.

2.7.1.1. 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 1.1.1. 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 the Bivrost Gateway Manual 6.1.1. 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.

2.7.1.2. 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 1.1.1. 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 the Bivrost Gateway Manual 5.5.7. 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.

2.7.1.3. 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 1.1.1. 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"
```

```
  }
]
```

2.7.1.4. 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 1.1.1. 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
  }
]
```

2.7.1.5. 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 1.1.1. 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

2.7.2. 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.

2.7.2.1. 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 1.1.2. 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 1.1.1. 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]

2.7.2.2. 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 1.1.2. 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 1.1.1. 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 the Bivrost Gateway Manual 5.5.7. 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.

2.7.2.3. 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 1.1.2. 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.

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 1.1.1. 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

Response Parameter	Type	Description
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.

2.7.2.4. 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 1.1.2. 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 1.1.1. 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]

2.7.2.5. 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 1.1.2. 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 1.1.1. 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

2.8. 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 the Bivrost Gateway Manual 5.12.2.3. 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.

2.8.1. 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 1.1.1. machineID Machine Identifier
type	String	(Required) Data class, see 1.2. Data Description

Request Parameter	Type	Description
startUnix	Int64	(Required) Start timestamp [seconds]
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 1.1. 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 1.2. Data Description.

2.8.2. 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 1.1.2. groupID Group Identifier
type	String	(Required) Data class, see 1.2. 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 1.1. 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 1.2. Data Description.

2.8.3. 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 1.2. 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 1.1. 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 1.2. Data Description.

2.9. Gateway Configuration Interface

The gateway function interface is used to command the gateway to perform specific functions, with base address `/api/config`.

2.9.1. Global Configuration Interface

2.9.1.1. 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

2.9.1.2. 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

2.9.1.3. 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

2.9.1.4. 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 `;`.

2.9.1.5. 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.

2.9.1.6. 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.

2.9.1.7. 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.

2.9.1.8. 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.

2.9.2. User Configuration API

2.9.2.1. user — Get user settings

```
GET /api/config/user
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 1.1.4. ID Database Identifier

Response example

```
{
  "id": 0,
  "username": "admin",
  "isAdmin": true
}
```

Response Parameter	Type	Description
id	Int32	(Required) Database identifier, see 1.1.4. 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.

2.9.2.2. 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 1.1.4. 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.

2.9.2.3. 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 1.1.4. 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.

2.9.2.4. 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 1.1.4. 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 1.1.4. 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.

2.9.2.5. delete-user — Delete a user

```
GET /api/config/delete-user
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 1.1.4. 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

2.9.2.6. user-security — Get user security settings

```
GET /api/config/user-security
```

Request Parameter	Type	Description
id	Int32	(Required) Database identifier, see 1.1.4. 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 1.1.4. 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.

2.9.2.7. 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 1.1.4. 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 1.1.4. 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.

2.9.3. Machine Configuration Interface

The machine configuration interface is used to retrieve, add, modify, or delete machine configurations. For more information on machine configuration, see the Bivrost Gateway Manual 5.3. 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 1.1.4. 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 1.1.1. machineID Machine Identifier
slaveID	Int32	Slave identifier. See 1.1.3. 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

Parameter	Type	Description
customAttributes	String	Custom attributes. Example: `{"attribute": value,…}`
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 address. Required for MODBUS communication. true = enabled, false = disabled. When enabled, the target address is offset by +1.
modbusReverseString	Bool	Reverse string. Required for MODBUS communication. true = enabled, false = disabled.
plcRack	Int32	Rack number. Required for some devices.
plcSlot	Int32	Slot number. Required for some devices.
plcStation	Int32	Station number. Required for some devices.

Parameter	Type	Description
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).

2.9.3.1. machine Get Machine Configuration

```
GET /api/config/machine?ID=ID&machineID=MACHINEID
```

Request Parameter	Type	Description
id	Int32	Database identifier. See 1.1.4. ID Database Identifier
machineID	String	Target machine identifier. See 1.1.1. 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.

2.9.3.2. 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.

2.9.3.3. 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.

2.9.3.4. 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.

2.9.3.5. delete-machine Delete Machine Configuration

```
GET /api/config/delete-machine?ID=ID&machineID=MACHINEID
```

Request Parameter	Type	Description
id	Int32	Database identifier. See 1.1.4. ID Database Identifier
machineID	String	Target machine identifier. See 1.1.1. 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

2.9.3.6. 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 1.1.4. 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

2.9.3.7. 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.

```
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 1.1.4. ID Database Identifier
machineID	String	Identifier of the template machine. See 1.1.1. 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

2.9.3.8. 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

2.9.4. Group Configuration API

The group configuration API is used to retrieve, add, modify, or delete group configurations. For details on group configuration, see the Bivrost Gateway Manual 5.4. 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 1.1.4. 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 1.1.2. 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 1.1.4. 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 1.1.1. 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 1.1.3. 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

2.9.4.1. group Get group configuration

```
GET /api/config/group?ID=ID&groupID=GROUPID
```

Request Parameter	Type	Description
id	Int32	Database identifier, see 1.1.4. ID Database Identifier
groupID	String	Target group identifier, see 1.1.2. 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.

2.9.4.2. 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.

2.9.4.3. 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 2.2. 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.

2.9.4.4. 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.

2.9.4.5. delete-group Delete group configuration

```
GET /api/config/delete-group?ID=ID&groupID=GROUPID
```

Request Parameter	Type	Description
ID	Int32	Database identifier, see 1.1.4. ID Database Identifier
groupID	String	Target group identifier, see 1.1.2. 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

2.9.4.6. 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 1.1.4. 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 2.2. Error Handling) rather than deleted=0.

Response Parameter	Type	Description
deleted	Int32	(Required) Number of groups deleted

2.9.5. Task Configuration Interface

The task configuration interface is used to retrieve or modify task configuration. For details on task configuration, see the Bivrost Gateway Manual 5.5. Task Configuration.

2.9.5.1. 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)

2.9.5.2. 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.

2.9.5.3. 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)

2.9.5.4. 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.

2.9.5.5. 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)

2.9.5.6. 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.

2.9.5.7. 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.

2.9.5.8. 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.

2.9.5.9. 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 ;.

2.9.5.10. 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.

2.9.5.11. 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 (%)

2.9.5.12. 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.

2.9.5.13. 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

2.9.5.14. 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.

2.9.5.15. 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.

2.9.5.16. 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.

2.9.5.17. 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

2.9.5.18. 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.

2.9.5.19. 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

2.9.5.20. 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.

2.9.5.21. 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

2.9.6. Communication Configuration Interface

The communication configuration interface is used to get or modify the task configuration. For details on communication configuration, see the Bivrost Gateway Manual 5.6. Communication Configuration.

2.9.6.1. 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.

2.9.6.2. 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.

2.9.6.3. 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.

2.9.6.4. 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.

2.9.6.5. 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
}
```

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
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.

Parameter	Type	Description
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.

MQTT mode

Option
Default
TB2
Brm
IoTDA
WisIoT
MKT
TB

2.9.6.6. 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": "IoTDA",
  "brokerAddress": "brokerAddress2",
  "port": 2222,
  "clientID": "client2",
  "username": "username2",
  "password": "password2",
  "dataReportTopic": "topic2",
  "rpcRequestTopic": "rpcReq",
  "rpcResponseTopic": "rpcRes",
  "encoding": "UTF-8",
  "allowAnonymous": true,
  "arrayToString": true,
  "publishOnValueChange": true,
  "publishAllOnValueChange": false,
  "timeoutReportingInterval": 0
}
```


For request parameters, see MQTT Settings Parameters.

Response example


```
{
  "enable": true,
  "mode": "IoTDA",
  "brokerAddress": "brokerAddress2",
  "port": 2222,
  "clientID": "client2",
  "username": "username2",
  "password": "password2",
  "dataReportTopic": "topic2",
  "rpcRequestTopic": "rpcReq",
  "rpcResponseTopic": "rpcRes",
  "encoding": "UTF-8",
  "allowAnonymous": true,
  "arrayToString": true,
  "publishOnValueChange": true,
```

```
  "publishAllOnValueChange": false,
  "timeoutReportingInterval": 0
}
```

For response parameters, see MQTT Settings Parameters.

2.9.6.7. 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.
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 1.2. Data Description for details.
noActionModeTypes	String[]	Effective when the storage mode is User Defined. See the `<type>` data classes in 1.2. Data Description for details.

Parameter	Type	Description
realTimeModeTypes	String[]	Effective when the storage mode is User Defined. See the `<type>` data classes in 1.2. 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
SQL Server
PostgreSQL

Database Storage Modes

Option	Description
Logging	Logging
Real Time	Real time
User Defined	User defined

2.9.6.8. 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.

2.9.6.9. 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.
username	String	Username. Read-only, cannot be modified.
password	String	Password

2.9.6.10. 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.

2.9.6.11. 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.

2.9.6.12. 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.

2.9.6.13. 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.
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.

2.9.6.14. 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.

2.9.6.15. 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.
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).

2.9.6.16. 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.

2.10. 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.

2.10.1. Core Service Function Interfaces

Base address **/api/core**.

2.10.1.1. 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

2.10.1.2. 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.

2.10.1.3. 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

2.10.1.4. 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

2.10.1.5. 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

2.10.1.6. 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

2.10.1.7. 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,
  "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.
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.

Response Parameter	Type	Description
enableMqttDefaultCustomAttributes	Bool	Whether custom attributes are included in MQTT output by default.
maxAllowedWrapperInstanceCount	Int32	Maximum number of acquisition process instances.
serviceQueueCapacity	Int32	In-memory capacity limit of a service queue.
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).
taskManagerLaunchingOffset	Int32	Interval between task launches (milliseconds).
fanucRobotAlarmLogCount	Int32	Number of historical alarm records read from FANUC robots.
fanucRobotCurAlarmCount	Int32	Number of current alarm records read from FANUC robots.
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.

Response Parameter	Type	Description
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.

2.10.2. Gateway Service Function Interfaces

Base URL `/api/gateway`.

2.10.2.1. 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

2.10.2.2. 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

2.10.2.3. 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

2.10.2.4. 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

2.10.2.5. 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

2.10.2.6. 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

2.10.2.7. internet-connection - Get gateway network status

Gets the gateway’s connection status to the internet. This interface takes no request parameters.

```
GET /api/gateway/internet-connection
```

Response example

```
{
  "isOnline": true
}
```

Response Parameter	Type	Description
isOnline	Bool	(Required) Whether connected to the internet, true = connected, false = not connected.

2.10.2.8. 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).
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.

2.10.2.9. 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).

2.10.2.10. 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

2.10.2.11. 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)

2.10.2.12. 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.

2.10.2.13. 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

2.10.2.14. 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

2.10.2.15. 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.
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

2.10.2.16. 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
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

2.10.2.17. 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
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.

2.10.2.18. 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.

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

> **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.

2.10.2.19. 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.

2.10.2.20. 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

2.10.2.21. 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

2.10.2.22. 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”.

2.10.2.23. 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”.

2.10.2.24. 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

2.10.2.25. 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

2.10.2.26. 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.

2.10.2.27. 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

2.10.2.28. 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.

2.10.2.29. 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.

2.10.2.30. 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.
rttMs	Int64	Round-trip time of this hop (milliseconds); null when the hop did not respond.

2.10.2.31. 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.

Request Parameter	Type	Description
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

2.10.2.32. 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.

3. MODBUS Communication

After enabling MODBUS communication and turning on the automatic data collection task for the target machine/group (see the Bivrost Gateway Manual 5.3.1.2. Task Settings and 5.4.1.2. 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 the Bivrost Gateway Manual 5.3.1.5. 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 the Bivrost Gateway Manual 5.6.2. 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.

3.1. 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 1.2.24. 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 the Bivrost Gateway Manual 6.1.1. Machine OEE Data for details.

> **Note 3**
>
> `String(200)` means this string is converted to a 200-byte array using the configured encoding (see the Bivrost Gateway Manual 5.6.2. 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.

3.1.1. 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

3.2. 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 the Bivrost Gateway Manual 6.1.3. 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 the Bivrost Gateway Manual 6.1.2. Group OEE Data for details.

4. MQTT Communication

After enabling MQTT communication and turning on the automatic acquisition task for the target machine/group (see the Bivrost Gateway Manual 5.3.1.2. Task Settings and 5.4.1.2. Group Task Settings), the gateway converts the automatic acquisition task results into JSON-formatted messages based on the MQTT protocol and publishes them to the specified MQTT server. The default topic for data upload messages is “r”. The gateway also supports the RPC interface. Users can configure the MQTT service by following the instructions in the Bivrost Gateway Manual 5.6.3. MQTT Configuration.

4.1. Data Upload Message Format

The gateway’s MQTT protocol data upload messages support multiple modes, including Default, MKT, TB, TB2, Brm, IoTDA, WisIoT, and others.

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": <type>,
                    "properties": {
                        "<tagField…>_<field>": <value>
                    },
                    "event_time": <time>
                }
            ]
        }
    ]
}
```

> **Note**
>
> In IoTDA mode the prefix of the property name is determined by the **tags** of the data class: if the data class contains no **tags**, the property name is just `<field>`; if the data class contains several **tags**, all **tag field forms** are joined by underscores before being used as the prefix, in the order of the tag’ s sequence number in the data class’ s **tag** table (the same rule as `<fieldID>` in key concept 5 below).

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 1.1. Basic Description for machineID and groupID.
2. `<alias>` Machine or group name: the user-defined machine name or group name (see the Bivrost Gateway Manual 5.3.1. Adding a Machine and 5.4.1. 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. `<tagField>` The field form of a tag, composed of the **tag** abbreviation and the **tag value**. For example, S1 is the field form of SpindleNo=1.
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 data from only a single data class <type> of the same machine or group <entityID>. Modes such as TB, TB2, IoTDA, Brm, and WisIoT support combining data from different data classes <type> of different machines or groups <entityID> into a single 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 1.2. Data Description, which leads to 1.2.4. 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 1.2.4. 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 1.1. 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 1.2. Data Description, which leads to 1.2.11. 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 1.2.11. 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 1.2. Data Description, which leads to 1.2.27. 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 1.2.27. 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 1.2.27. 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 1.2. Data Description, which leads to 1.2.27. 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 1.2.27. 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 1.2.27. 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 1.2. Data Description, which leads to 1.2.27. 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 1.2.27. 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 1.2.27. 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",
          "properties": {
            "T9_O2_timeLimit": 1800,
            "T9_O2_currentTime": 1620,
            "T9_O2_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",`

Its format is `"service_id": <type>`. Using “ToolLife” as the keyword, you can find it in the `<type>` data class table in 1.2. Data Description, which leads to 1.2.27. ToolLife: Tool Life Data, identifying the data in this message as tool life data.

Lines 8 to 11,

```
"properties": {
```

```
    "T9_O2_timeLimit": 1800,
    "T9_O2_currentTime": 1620,
    "T9_O2_prewarningTime": 1680},
```

T9_O2 is the `<tagField>` field form of the tags; T9 and O2 are combinations of `<tag>` data tag abbreviations and tag values. Looking up the `<tag>` data tags in 1.2.27. 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 1.2.27. 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.

4.2. 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 the Bivrost Gateway Manual 5.6.3. 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

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
lan	Get wired network settings

RPC Command	Description
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 2. HTTP Communication; input parameters and return results can be found in the interface descriptions in 2. 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 2.6.6. 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 2.5.1.14. 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"
  }
}
```

5. Database Communication

After enabling database communication and turning on the automatic data collection task for the target machine/group (see the Bivrost Gateway Manual 5.3.1.2. Task Settings, and 5.4.1.2. 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 the Bivrost Gateway Manual 5.6.4. 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 the Bivrost Gateway Manual 5.6.4. Database Configuration); the table name is the same as the `type` of the data class used in MQTT communication (see 1.2. 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 1.2. 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.

6. 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, so **the data returned through MCP is identical to that of the corresponding HTTP interface**. For field definitions, see the sections under 2. HTTP Communication.

6.1. 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. On the gateway, turning the MCP switch on from the Communication page takes effect immediately with no restart.

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.

33 **read-only** tools are currently provided, covering machine configuration, live status, data analysis, and gateway system information; see 6.4. 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.

> **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 6.2.2. Cloud Platform Authentication.

6.2. Authentication

6.2.1. Gateway Authentication

The MCP server uses the same authentication as the HTTP interfaces; see 2.3. Authentication Methods. Send `Authorization: Bearer <token>` in the request headers, where the token is either a token obtained via 2.3.1. JWT Method or a key generated via 2.3.2. Secret Key Method. 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, the source IP of a `/mcp` request must be in the whitelist, following the same rules as `/api`; see 2.3.3. IP Whitelist.
- **Per-user authorized APIs**: when security control is enabled, administrators may call every tool, while other users may only call the tools covered by their authorized API list (matching is done on the interface path behind each tool, using the same configuration as HTTP requests). Without permission, the tool call returns an error whose message starts with `Forbidden: ...`.
- With security control disabled, tools can be called without user authentication (the IP whitelist still applies).

6.2.2. 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 token in the cloud platform settings; see 2.9.6.1. 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`.

6.3. 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"}'
```

6.4. Tool List

Parameters marked "optional" below may be omitted, in which case the corresponding interface's default is used. All time parameters are Unix timestamps in seconds. For the meaning of machineID and groupID see 1.1. Basics; they can be obtained with `list_machines` and `list_groups`.

6.4.1. Configuration

Tool	Description	Parameters	Interface
list_machines	List all machine configurations (brand, model, IP, port, etc.)	none	machines
get_machine	Get the configuration of a single machine	machineID	machine
list_groups	List all machine groups and their member machines	none	groups
get_gateway_info	Get gateway identity and version information	none	gateway-info

6.4.2. Machine Status

Tool	Description	Parameters	Interface
read_cnc_status	Read machine run status	machineID, channel (optional)	readCNCStatus
read_cnc_status_details	Read machine status details (status, mode, program, feed, spindle, alarms)	machineID, channel (optional)	readCNCStatusDetails
read_alarm	Read active alarms	machineID, channel (optional)	readAlarm
read_position	Read axis positions	machineID, channel (optional)	readPosition
read_load	Read axis and spindle load	machineID, channel (optional)	readLoad
read_feed_and_spindle	Read feed rate and spindle speed	machineID, channel (optional)	readFeedAndSpindle
read_current_tool_number	Read the current tool number	machineID, channel (optional)	readCurrentToolNumber

Tool	Description	Parameters	Interface
read_time_data	Read machine time data (power-on, operating, cutting, cycle time)	machineID, channel (optional)	readTimeData
read_count	Read machining counts	machineID, channel (optional)	readCount
read_tool_life	Read tool life data	machineID, toolNum / offsetNum / groupNum (all optional; omit to read all tools)	readToolLife
read_tool_life_details	Read tool life details	machineID, toolNum / groupNum (both optional)	readToolLifeDetails
read_program_list	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)	machineID, dirAtCNC / subDir (both optional)	readCurrentProgram
batch_read_errors	Read the connection status of multiple machines at once; error code 0 means communication is healthy	machineIDs (array of machine identifiers)	batchReadErrors

6.4.3. Data Analysis

Tool	Description	Parameters	Interface
analyze_oee	Machine OEE analysis	machineID, startUnix, endUnix (optional), interval (optional)	oee
analyze_alarm	Machine alarm analysis	machineID, startUnix, endUnix (optional), interval (optional)	alarm

Tool	Description	Parameters	Interface
analyze_count	Machine count analysis	machineID, startUnix, endUnix (optional), interval (optional), enableCountPerProgram (optional)	count
analyze_cycle	Machine cycle time analysis	machineID, startUnix, endUnix (optional), interval (optional)	cycle
analyze_overall	Machine overall analysis (running/idle/alarm/offline time breakdown)	machineID, startUnix, endUnix (optional), interval (optional)	overall
analyze_group_oee	Machine group OEE analysis	groupID, startUnix, endUnix (optional), interval (optional)	oee
analyze_group_overall	Machine group overall analysis	groupID, startUnix, endUnix (optional), interval (optional)	overall
query_machine_history	Query historical machine data	machineID, type (data type), startUnix, endUnix (optional), limit (optional)	machine

6.4.4. System Information

Tool	Description	Parameters	Interface
get_core_info	Get Core service information	none	info
get_core_license_info	Get license information (licensed machine count, expiry)	none	license-info
get_core_service_status	Get the Core service running status	none	service-status
get_hardware_resources	Get gateway hardware resource usage (CPU, memory, disk)	none	hardware-resources
get_gateway_time	Get the gateway’s current time and time zone	none	time

Tool	Description	Parameters	Interface
get_network_adapters	List the gateway’s network adapters	none	network-adapters
get_error_log	Query the gateway’s interface error log	startUnix (optional), endUnix (optional)	`/api/log/error`

6.5. Results and Errors

On success, a tool returns the response body of the corresponding HTTP interface (a JSON string); see each interface’s section for field descriptions.

On failure, a tool returns an MCP tool error (`isError`) whose message has the format:

```
{interface path} failed: {error code name}({errorCode}) {error description}
```

For example, querying a machine that does not exist:

```
/cnc/readCNCStatus failed: GENERAL_MACHINE_ID_NOT_EXISTED(10003)
```

The errorCode is exactly the same as that of the HTTP interfaces; see 2.2. Error Handling.

6.6. 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), the free-form historical data query interface, and interfaces carrying sensitive information such as user configuration and security settings.
- 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.

7. 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 the Bivrost Gateway Manual 5.3. 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.

8. FAQ

8.1. 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.

8.2. 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 the Bivrost Gateway Manual 5.8. 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 the Bivrost Gateway Manual 5.3. 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”.

8.3. No data is received during MODBUS or MQTT communication

Please check the machine’s automatic data collection settings (see the Bivrost Gateway Manual 5.3. Machine Configuration). After completing the setup, you must click **Restart Service** on the **Home** page.

Please check the task configuration (see the Bivrost Gateway Manual 5.5. 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.

8.4. 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.

8.5. Machine data is received using any communication protocol, but no group data is received

Please check whether the group is correctly activated and whether group task settings are enabled (see the Bivrost Gateway Manual 5.4. Group Configuration). After completing the setup, you must click **Restart Service** on the **Home** page.

Please check the task configuration (see the Bivrost Gateway Manual 5.5. Task Configuration). After completing the setup, you must click **Restart Service** on the **Home** page.

8.6. Using the interface to list files, or to receive/send/delete files, fails, even though the gateway program’s Transfer page works normally

If the path or file name contains special symbols or unsafe characters such as spaces, they must be converted to URL encoding. For example: the path `TNC:\nc_prog` should be converted to `TNC%3A%5Cnc_prog`; `Sinumerik/FileSystem/Part Program` should be converted to `Sinumerik%2FFileSystem%2FPart%20Program`. In general, most browsers and testing tools perform URL encoding automatically, but when calling the interface from some software, manual conversion is still required.

Version Change History

> **Note**
>
> This page records **gateway firmware** version changes. The version of this document itself is shown in the sidebar badge and on the PDF cover as “Document Version”; despite the similar numbering the two are not the same thing.

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 6. MCP Service.

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 6. MCP Service.

v1.19.7.18 Subsequent Updates (2026-08-05)

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 2.3.3. 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.

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.

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.

v1.19.5 (2026-06-15)

Release date: 2026-06-15

1. Modified the selectProgram interface; added parameter mode (execution mode).

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.

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.

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.

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).

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.

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.

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.

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/`.

v1.15.5 (2024-01-22)

Release date: 2024-02-04

1. Updated the table of system models supported by the file transfer interfaces.

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.

v1.15.2 (2023-12-18)

Release date: 2024-02-04

1. Added HTTP interfaces `/sendFileStream` and `/receiveFileStream` to support large file transfers.

v1.15.1 (2023-12-06)

Release date: 2024-02-02

1. Revised some descriptive content.

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.

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.

v1.13.0 (2023-10-23)

Release date: 2023-10-23

1. Added HTTP and RPC interfaces `/batchReadOffsetData`, `/bathReadToolLife`, and `/bathReadToolLifeDetails`.

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.

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.

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.

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).

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.

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.

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.

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.

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.

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`.

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.

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`.

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.

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.

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”.

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.

v1.5.1 (2022-04-26)

Release date: 2022-04-26

1. Changed the description of OEE monitoring settings.

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.

v1.4.3 (2022-03-28)

Release date: 2022-03-28

1. Revised some text content.

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.

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.

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.