

ROBOTICS

Application manual

IoT Gateway



Trace back information:
Workspace Main version a646
Checked in 2025-02-13
Skribenta version 5.6.018

Application manual

IoT Gateway

Document ID: 3HAC078375-001

Revision: G

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damage to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

© Copyright 2020-2025 ABB. All rights reserved.
Specifications subject to change without notice.

Table of contents

Overview of this manual	7
Product documentation	9
1 Getting started	11
2 IoT Gateway configuration application	17
2.1 Introduction	17
2.2 About the IoT Gateway configuration application	18
2.2.1 Main screen components	18
2.2.2 Aliases	21
2.2.3 Add new Alias dialog screen components	23
2.2.4 Edit Alias	27
2.2.5 Add remote controller	29
2.2.6 The IoT Gateway configuration file	31
2.2.7 Configure IoT Gateway plug-in	32
2.2.8 IoT Gateway username/password	33
2.2.9 Server control	35
2.2.10 Logs	36
2.3 How to add/edit IRC5 or OmniCore robot alias	37
2.3.1 How to add new IRC5 or OmniCore robot aliases	37
2.3.2 How to edit a robot alias	40
2.4 Certificate management	42
2.4.1 Client certificates	42
2.4.2 Server Application instance certificates	44
3 OPC UA Server	49
3.1 Address space	49
3.2 Events implementation	52
3.3 Security	54
3.3.1 Transport Protocols	54
3.3.2 Security configuration	55
3.3.3 Restricting access to application folder	57
3.4 How to connect to OPC UA Server	59
4 MQTT Publisher	63
4.1 Introduction to MQTT Publisher	63
4.2 Configuring MQTT Publisher	65
4.2.1 Introduction	65
4.2.2 MQTT client configuration	66
4.2.3 MQTT Engineering Tool	69
4.2.3.1 Introduction	69
4.2.3.2 Main Screen Components	70
4.2.4 Robot configuration	77
4.2.5 Trigger configuration	79
4.2.5.1 Overview	79
4.2.5.2 Trigger Type	81
4.2.5.3 Guard condition	92
4.2.5.4 Topic	93
4.2.6 Payload configuration	94
4.2.6.1 Overview	94
4.2.6.2 Robot model	96
4.2.6.3 Payload template examples	127
4.3 Handlebars	132
4.3.1 Introduction	132
4.3.2 Language features	133

Table of contents

5	Troubleshooting	137
6	Appendix	139
6.1	Appendix A - Robotics companion specification	139
6.2	Appendix B - Performance tests summary	148
6.3	Appendix C - IoT Gateway configuration file	149
6.4	Appendix D - ABB Robotics OPC UA proprietary information model	151
6.4.1	Overview	151
6.4.2	OPC Unified Architecture for ABB Robotics Controller	152
Index		161

Overview of this manual

About this manual

This manual contains instructions for daily operation of IoT Gateway.

Usage

This manual should be used during operation, installation and configuration of IoT Gateway.

Who should read this manual?

This manual is intended for:

- Users of the product IoT Gateway.

Prerequisites

The reader should.

- use the manual as an online help and
- have IoT Gateway installed.

References

Reference	Document ID
<i>Technical reference manual - System parameters</i>	3HAC065041-001
<i>Technical reference manual - System parameters</i>	3HAC050948-001
<i>Technical reference manual - RAPID kernel</i>	3HAC050946-001
<i>Operating manual - Integrator's guide OmniCore</i>	3HAC065037-001
<i>Operating manual - IRC5 Integrator's guide</i>	3HAC050940-001
<i>Technical reference manual - Event logs for RobotWare 7</i>	3HAC066553-001
<i>Operating manual - Troubleshooting IRC5</i>	3HAC020738-001

Revisions

Version	Description
A	First edition. Released with IoT Gateway v1.0
B	Released with IoT Gateway v1.1
C	Released with IoT Gateway v1.2 Following are the updates: <ul style="list-style-type: none"> • Added the section MQTT Engineering Tool on page 69. • Added the section ElogMessageQ on page 107. • Added the section Default configuration on page 77. • Added the section ElogMessageQClear on page 123. • Added the section ElogMessageQPop on page 123. • Updated the section ElogMessage on page 105. • Updated the section EventLog Message trigger on page 89.

Continues on next page

Version	Description
D	Released with IoT Gateway v1.2.1 Following are the updates: <ul style="list-style-type: none">• Added the section Mechanical unit service info on page 114.• Updated the section Getting started on page 11.• Updated the section Device status icons on page 20.• Updated the section IoT Gateway username/password on page 33.
E	Released with IoT Gateway v1.3 Following are the updates: <ul style="list-style-type: none">• Added the section Changing the location of configuration files on page 16.• Updated the section How to add/edit IRC5 or OmniCore robot alias on page 37.• Updated the chapter Getting started on page 11.• Updated the section Appendix D - ABB Robotics OPC UA proprietary information model on page 151.
F	Released with IoT Gateway v1.4 Following are the updates: <ul style="list-style-type: none">• Updated various sections in the chapter IoT Gateway configuration application on page 17.• Updated the section Firewall settings on page 12.• Updated the section Create application certificate on page 45.• Updated the section Security policies on page 55.• Updated the section Top Level on page 97.
G	Released with IoT Gateway Following are the updates: <ul style="list-style-type: none">• Updated the section RobotWare software requirements on page 11• Updated the section Device status icons on page 20• Updated the section Add New Alias dialog icons on page 26• Updated the section Event log attributes on page 53.• Updated the section Subscription for data changes on page 51.• Updated the section Troubleshooting on page 137.

Product documentation

Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional.



Tip

All documents can be found via myABB Business Portal, www.abb.com/myABB.

Product manuals

Manipulators, controllers, DressPack, and most other hardware is delivered with a **Product manual** that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected life time of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Troubleshooting.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with corresponding figures (or references to separate spare parts lists).
- References to circuit diagrams.

Technical reference manuals

The technical reference manuals describe reference information for robotics products, for example lubrication, the RAPID language, and system parameters.

Application manuals

Specific applications (for example software or hardware options) are described in **Application manuals**. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, software).
- How to install included or required hardware.
- How to use the application.

Continues on next page

- Examples of how to use the application.

Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and troubleshooters.

1 Getting started

Overview

IoT Gateway is an extendable communication gateway that provides OPC UA Server capabilities for the IRC5 (with RobotWare 6) and OmniCore generation of robot controllers.

ABB can easily add new customer specific communication gateways without the need for a new release of the IoT Gateway product itself. A typical example is a customer that requires a MQTT client that publish data from the robot controller in a specific, customer defined format.

The product is implemented as a Windows service that runs the communication gateway(s) and a Windows application used for configuration and commissioning. The product is especially suitable for retrofit solutions as it does not require an upgrade of RobotWare.

Product requirements

Overview

Before installing the IoT Gateway, ensure that the computer meets the following hardware and software requirements.

Hardware

Medium to high-performance industrial or desktop PC, with the following requirements:

Part	Requirement
CPU	2.0GHz or faster processor, multiple cores recommended
Memory	2 GB minimum. 4GB or more recommended
Disk	2+ GB free space, solid state drive (SSD) recommended.Disk

Software

Following are the software requirements:

- Operating system: Windows 10 Anniversary update or later (64-bit edition)
- MicroSoft.NET version 7.00 or later

It is recommended to run Windows updates to get the latest updates to Windows before installing and running IoT Gateway. It is also strongly recommended to keep the Windows operating system updated with the latest security updates according to Microsoft recommendation or company policy.

Windows Firewall can block certain features that are necessary to run IoT Gateway, which must be unblocked as required. For more information on Windows Firewall, visit www.microsoft.com

RobotWare software requirements

IoT Gateway supports RobotWare 6 and RobotWare 7.2 and newer.

- For Omnicore controllers from RobotWare 7.2 and newer, “3154-1 IoT Data Gateway” option is required.

Continues on next page

1 Getting started

Continued

- For all RobotWare 6 versions starting from version 6.12.03, the “1582-1 IoT Data Gateway” RobotWare options is required.
- For Robotware version starting from 6.11 to 6.12.02, the “1582-1 OPC UA Server” RobotWare options is required.
- The IoT Gateway may also be used with RobotWare versions prior to 6.11. Please contact ABB for information on licensing.

Firewall settings

The firewall settings are applicable to real and virtual controllers. The following table describes the necessary firewall configurations:

Name	Action	Direction	Protocol	Remote Address	Local Service	Remote Service	Application
RobNetscan-Host	Allow	Out	UDP/IP	Any	Any	5512,5514	robnetscan-host.exe
Robot Controller	Allow	IN	UDP/IP	Any	5513	Any	robnetscan-host.exe
RobComCtrlServer	Allow	Out	TCP/IP	Any	Any	5515	robcomctrlserver.exe
IoT Gateway	Allow	IN	TCP/IP	Any	61510 (configurable)	Any	ABB.Robotics.IoTGateway

Product installation

Prerequisites

Following is the prerequisites to install IoT Gateway:

- Administrative rights are required in order to install and configure IoT Gateway.

Procedure

Follow these steps to install IoT Gateway:

- 1 Click Setup.exe from <https://developercenter.robotstudio.com>. The IoT Gateway Setup window opens.
- 2 Proceed to install IoT Gateway. IoT Gateway is installed.

How to set up alias

Following are the prerequisite for creating aliases:

- Connect the robot to the same computer network on which the IoT Gateway is running.
- OPC UA Server displays data only for IRC5 and OmniCore generation of robots with configured aliases (a descriptor that identifies a particular robot; see [Aliases on page 21](#)). Configuration entries are stored in the [IoTGateway.config on page 149](#).
- For RobotWare options, see [RobotWare software requirements on page 11](#).

Continues on next page

**Note**

For details about accessing the OPC UA Server from a client over the network, see [Security on page 54](#).

Use the following procedure to set up and view data from an IRC5 and OmniCore robot controller:

Step	Action
1	Create aliases for those robots you want to communicate with. For more details, see How to add new IRC5 or OmniCore robot aliases on page 37 .
2	OPC UA clients can use the discovery process to find OPC UA Server over the network for the session connection. Or Can use direct Endpoint URL to establish a connection. For more details, see How to connect to OPC UA Server on page 59 .
3	Install any OPC UA client to browse the address space, create subscriptions and monitored items, to read and write data. <div data-bbox="564 891 627 949" data-label="Image"></div> <div data-bbox="649 904 715 934" data-label="Section-Header">Note</div> <div data-bbox="560 958 1428 1019" data-label="Text"> <p>There are number of free OPC UA Client test tools available for download on Internet, e.g. Softing's dataFEED OPC UA Client.</p> </div>

Product features

IoT Gateway provide OPC UA Server and MQTT publisher capabilities for the IRC5 and Omnicore generation of robot controllers.

The OPC UA Server implements the functionality of the UA Address Space Model 1.04 Specification. It is a UA server which enables UA clients to browse the address space, create subscriptions and monitor items, and read and write data. The MQTT publisher is configurable using the MQTT Engineering Tool. Both message content and format can be configured using template files. Triggers define when to publish messages.

Certificates

Certificates are used to establish secure communication between the OPC UA Client and OPC UA Server.

Endpoints

OPC UA Clients can connect to the OPC UA Server using a URL with the following format:

```
opc.tcp://HOSTNAME:PORT NUMBER/SERVERNAME
```

The actual PORTNUMBER and SERVERNAME should match the Port Number and Server name fields in the 'Server Control' tab in the IoT Gateway Config tool. For more details, see [Server control on page 35](#).

The HOSTNAME can be found using the hostname command in Windows command prompt. To get the actual server endpoint URL from the log, see [How to connect to OPC UA Server on page 59](#).

Continues on next page

1 Getting started

Continued

User authentication

OPC UA Server supports the following user authentication modes:

- **Anonymous:** No user identity is provided.
- **UserName:** A user identified by user name and password.

Client authentication

The OPC UA Server authenticates (Identifies) OPC UA client using **Client application instance certificate**.

Client certificates will be stored in certificate store: <C:\ProgramData\ABB\IoT Gateway\CertificateStores>.

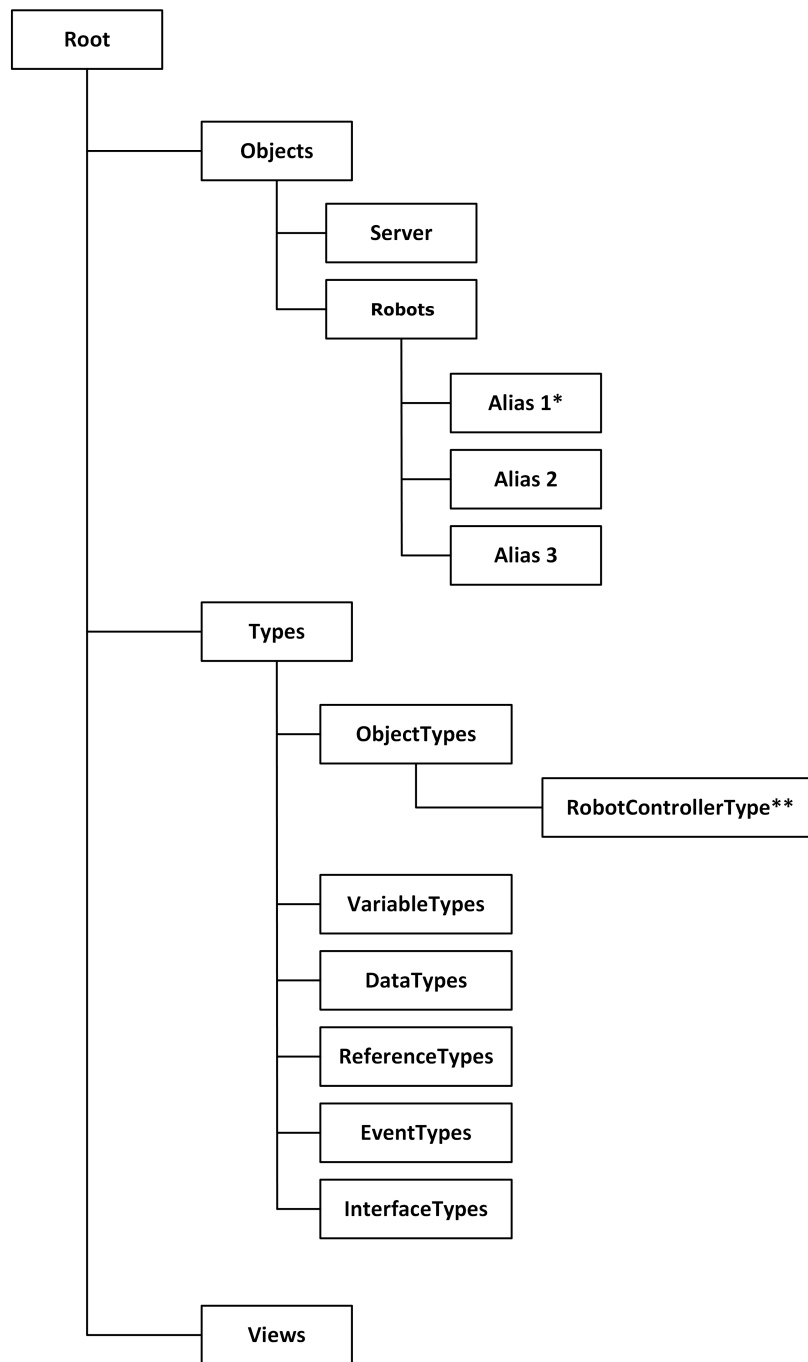
Security

OPC UA Server supports OPC UA standard security modes and policies.

Address Space

The server's address space represents its contents as a set of **Nodes** connected by **References**. The address space begins with the top node **Root**, which Organizes **Objects, Types and Views**.

Continues on next page



xx210000352

* A robot controller is identified by its alias name that must be unique.

** Top level object type for an ABB Robotics controller.



Note

For more information on address space, see section [Address space on page 49](#).

Cybersecurity

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damage and/or loss related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

For more information, see the section Cyber security in *Operating manual - IRC5 Integrator's guide* (for IRC5 robot networks) and *Operating manual - Integrator's guide OmniCore* (for OmniCore robot networks).

Changing the location of configuration files

All IoT Gateway configuration files created and edited by the IoT Gateway Config tool and the MQTT Engineering Tool are by default stored under the Program Data folder, normally "C:\ProgramData\ABB\IoT Gateway". It is recommended to continue using this location. But if another location for the configuration files is required, use the following procedure to configure it:

- 1 Open command prompt as administrator
- 2 Change directory to the IoT Gateway installation folder. The default installation folder is "C:\Program Files (x86)\ABB\IoT Gateway", but this may differ if another folder was specified during installation.
- 3 Run the following command from the command line:

```
MoveUserFiles <path>
```

Where <path> is the full path to new location of IoT Gateway configuration files.

2 IoT Gateway configuration application

2.1 Introduction

Overview

The IoT Gateway configuration application is used to create and manage aliases for ABB robot controllers. An Alias is a user-friendly descriptor that represents a communications interface with ABB robot controller. You need to create an alias for each robot controller that will be accessed by the OPC UA Server.

2 IoT Gateway configuration application

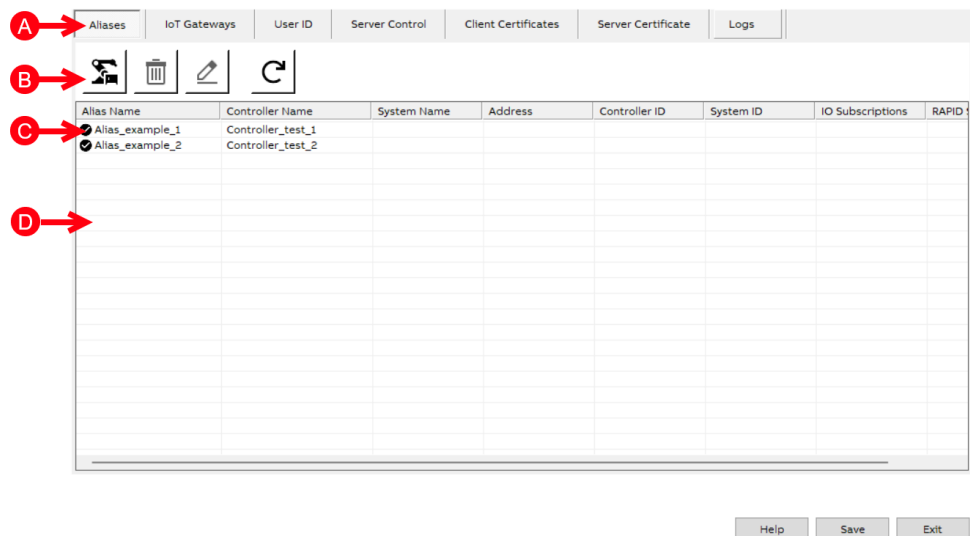
2.2.1 Main screen components

2.2 About the IoT Gateway configuration application

2.2.1 Main screen components

Server configuration

The IoT Gateway Configuration application main screen shows a list of Aliases that you have created. The main screen of IoT Gateway Configuration application displays important information about the created Aliases, such as assigned name, Controller Name, System Name, Address, and so on as shown in the following figure:



xx2100000141

A	Function tabs
B	Toolbar buttons
C	Device status icons
D	Device pane

Function tabs

The following table provides information about the function tabs in IoT Gateway Configuration application:

Component	Function
Alias	The IoT Gateway Configuration application "main screen".
IoT Gateways	IoT Gateway configuration application allows enabling/disabling of individual information models (gateways).
User ID	Type the user name and password for OPC UA Server access to the robot. For more information, see IoT Gateway username/password on page 33 .
Server Control	Select the Start or Stop button to start /stop the IoT Gateway.
Client Certificates	Provides an interface for managing the configuration of the security certificates for the application.

Continues on next page

2 IoT Gateway configuration application





2.2.1 Main screen components

Continued

Component	Function
Server Certificates	Provides an interface to create a new application certificate, add the application to Local Discovery Server (LDS) trust list, Import application certificate, & Export application certificate.
Logs	Lists all the information/error/warning messages related to IoT Gateway.

Toolbar buttons

The following table provides information about the tool bar buttons in the main screen:

Button	Function
 Add Alias	Opens the Add New Alias dialog see Add new Alias dialog screen components on page 23 .
 Delete Alias	Deletes the highlighted alias from the device pane.
 Edit Alias	Opens the Edit Alias dialog see Edit Alias on page 27 .
 Refresh Main Screen	Click to refresh Alias connectivity status.

Device pane

The device pane displays a list of robot aliases and their associated attributes.

Component	Function
Alias Name	Displays the name of the alias.
Controller Name	The name of the IRC5 or OmniCore controller.
System Name	The RobotWare system name running on the IRC5 or OmniCore controller.
Address	The IP-address of the IRC5 or OmniCore controller.
Controller ID	The ID of the IRC5 or OmniCore controller.
System ID	The ID of the RobotWare system running on the IRC5 or OmniCore controller.
IO Subscriptions	The number of I/O signal "change-events" subscribed to the IRC5 or OmniCore controller (currently subscribed / maximum number of subscriptions allowed). Note: This value is only updated when a Refresh button is pressed.

Continues on next page

2 IoT Gateway configuration application




2.2.1 Main screen components

Continued

Component	Function
RAPID Subscriptions	The number of RAPID variable "change-events" subscribed to the IRC5 or OmniCore controller (currently subscribed / maximum number of subscriptions allowed). Note: This value is only updated when a Refresh button is pressed.

Device status icons

The following table provides information about the device status icons used in the main screen:

Icon	Description
 Connectable Alias	Alias is available on the network.
 Disconnected Alias	Alias is disconnected from the network.
 Inaccessible Alias	Alias is connected to the network. The OPC UA Server option is missing. For more information, refer RobotWare software requirements on page 11 .

2.2.2 Aliases

Overview

Aliases are used by IoT Gateway to map between a user defined name for a robot controller and one or more controller parameters. When creating or editing an Alias definition, the user may select which controller parameter(s) the IoT Gateway should use to associate the correct robot controller with a particular Alias.

Aliases

An Alias is a user friendly name used by the IoT Gateway to identify a robot controller.

Parameters

When a robot controller is connected to a network, it identifies itself by broadcasting certain information onto the network. This information includes the following parameters:

- Controller Name
- Controller ID
- System Name
- System ID
- IP address

Multiple controllers matching the same Alias definition

If the Alias reference to the robot controller cannot be resolved, then the application will not be able to communicate with the robot. You must define the type of association for an Alias carefully, as various associations behave differently.

Example

Imagine that an Alias is defined that uses the Controller Name as the only connection criteria. If there are two or more controllers on the network that have the same Controller Name, the IoT Gateway has no way of distinguishing between them. The robot controller that is discovered first will be associated with the Alias. If another controller comes on line with the same Controller Name, an error will be logged, and no connection is established.

Address

The following table provides information on IP address parameters for alias:

If...	Then...
you associate an Alias with only the parameter IP address	you should make sure that the IP address is statically assigned to the robot controller.
you use DHCP	your DHCP server must be carefully configured in order to guarantee repeatable assignments of the IP address.
the IP addresses are re-assigned	the Alias might resolve to a different controller.

Continues on next page

2 IoT Gateway configuration application

2.2.2 Aliases

Continued

System name

The following table provides information on system name parameters for alias:

If...	Then...
you associate an Alias with only the parameter System Name	you must ensure that the System Name is unique for each robot controller. This can be inconvenient if you have several robots that could otherwise be loaded with the same RobotWare system (and thus be given the same System Name). The same resolution difficulties exist as in the case of duplicate Controller Names.

Controller ID

The following table provides information on controller ID parameters for alias:

If...	Then...
you associate an Alias with only the parameter Controller ID	you can be sure that the Controller ID is unique. A Controller ID is permanently assigned to a specific robot controller.
you replace the robot controller hardware	the Controller ID will change, and you must then redefine the Alias to associate it with the new controller.

System ID

The following table provides information on system ID parameters for alias:

If...	Then...
you associate an Alias with only the parameter System ID, and you are certain that the System ID is unique	you must remember that the System ID will change when RobotWare is upgraded or in re-installed. If you need to make any modifications to the RobotWare system configuration that would result in a subsequent reload, the System ID will change and you must redefine the Alias.

Recommended associations

ABB recommends that you define Aliases to ensure stability of the association of the Alias to a particular robot.

If you use static IP address or your DHCP server which is configured in such a way that its IP address assignment is repeatable, you should associate Alias definitions with both Controller Name and IP address. Using this approach, you can download new RobotWare systems, or replace an entire controller without the need to redefine the Alias.

If you cannot guarantee stable IP addresses, then you should define Aliases using both the Controller Name and Controller ID Connection Criteria. With this approach, you can change the IP address, or you can download new BaseWare systems without the need to redefine the Alias. However, if you replace controller hardware, the Controller ID will change and you must redefine the Alias.

Either approach reliably ensures that an Alias will always resolve to a unique, well-known robot controller.

2.2.3 Add new Alias dialog screen components


Add new Alias dialog components

On the main screen, click **Add Alias**.

The **Add New Alias** window is displayed.

Controller Name	System Name	Address	Controller ID	System ID
Controller_test_1	Controller_test_1	127.0.0.1	VIRTUAL_USE	5eec988d-dea8-425d-bf...
Controller_test_2	Controller_test_2	127.0.0.1	VIRTUAL_USE	72a27fc9-f88c-4c72-99a...

xx210000138

Component	Description
Alias Name field	<p>This field allows you to enter the Alias name.</p> <p> Note</p> <p>Controllers that are offline can be added as an alias using its alias name. For more details, see Add offline controller on page 24.</p>
Create button	<p>Click to create a new Alias. This button becomes active after selecting any connection criterion.</p>
Connection Criteria check boxes	<ul style="list-style-type: none"> Controller Name - This is the name assigned to the robot controller. System Name - This is the name assigned to the currently active BaseWare system in the robot controller. Address - This is the robot controller's IP address. If the controller is a Virtual Controller (VC), this is the path to the VC system directory. Controller ID- This is the robot controller's unique hardware ID. System ID - This is the ID assigned to the currently active BaseWare system in the robot controller.

Continues on next page

2 IoT Gateway configuration application

2.2.3 Add new Alias dialog screen components

Continued

Component	Description
Locate Remote Controller	Allows you to search for the remote controllers. Type IP address of the remote controller and click the Locate button. If the controller is available, it appears in the scan results page. If it is not available, an information message is displayed. For more details, see Add remote controller on page 29 .
Rescan button	Refreshes the list of available controllers in the network.
Scan results	Displays the list of available controllers. Following are the two options available for filtering the results: <ul style="list-style-type: none">• Show only robots with no assigned Alias: Refreshes the scan results list and displays only those robots that has no assigned alias.• Show only robots that match connection criteria: Refreshes the scan results list and displays only those robots that matches the connection criteria.

Add offline controller

The offline controllers can be added as an alias so that you can configure the known controllers which are offline.

Use the following procedure to add the offline controllers:

- 1 On the main screen, click Add Alias. The Add New Alias window is displayed.

xx240000721

- 2 Type the name of the alias in the **Alias Name** field.
- 3 In the **Connection Criteria** section, type the known fields.
- 4 Click **Create**.

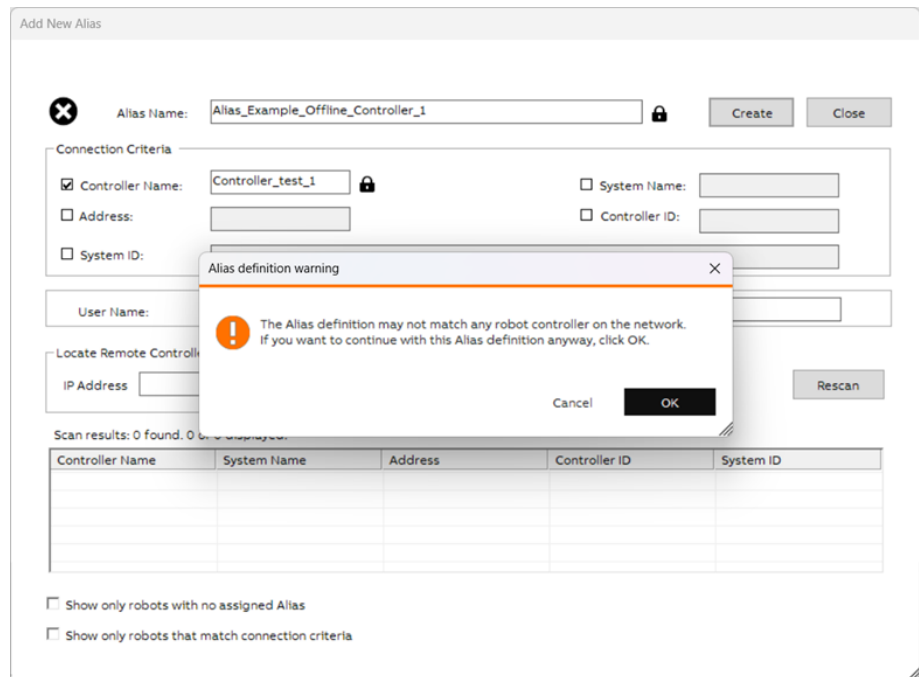
The following confirmation window is displayed.

Continues on next page

2 IoT Gateway configuration application

2.2.3 Add new Alias dialog screen components

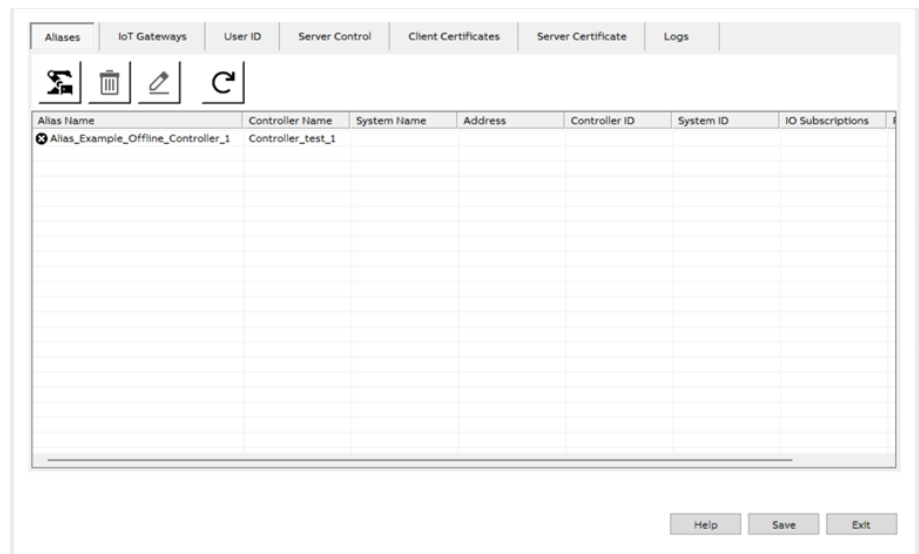
Continued



xx2400000722

5 Click OK.

The new alias with offline controller is created.



xx2400000723

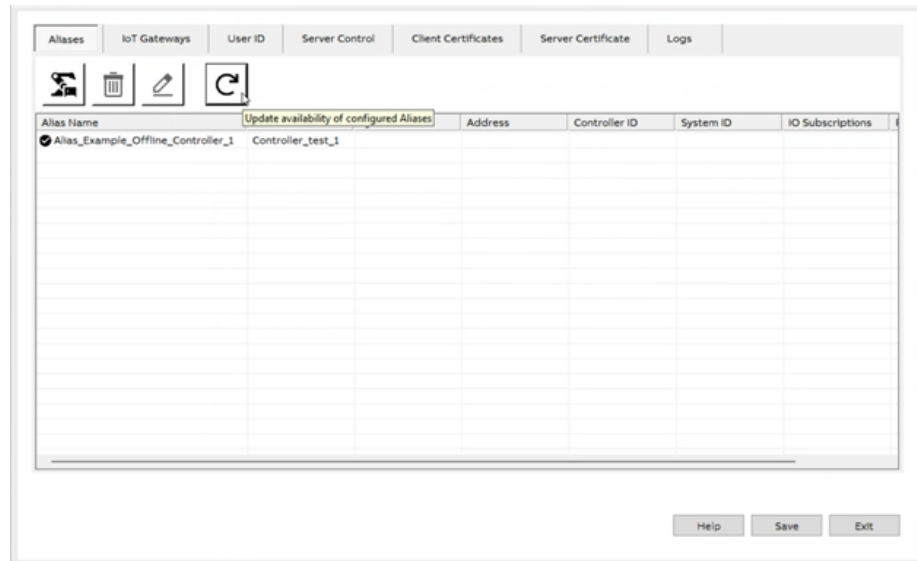
6 When controller comes live, click on the refresh icon button to update the alias status.

Continues on next page

2 IoT Gateway configuration application

2.2.3 Add new Alias dialog screen components




Continued



xx240000724

Add New Alias dialog icons

The following icons are displayed in the **Controller Name** list of the **Scan results** section:

Icon	Description
 Inaccessible Alias	Alias is connected to the network. The OPC UA Server option is missing. For more information, refer RobotWare software requirements on page 11 .
 Field locked	When one of the fields is modified manually, a small lock will appear next to it. This indicates that from now on, the configuration tool will not modify these fields automatically, but let the user enter the values manually.
 Accessible Alias	The alias is connected to the network. Required RobotWare options are present.

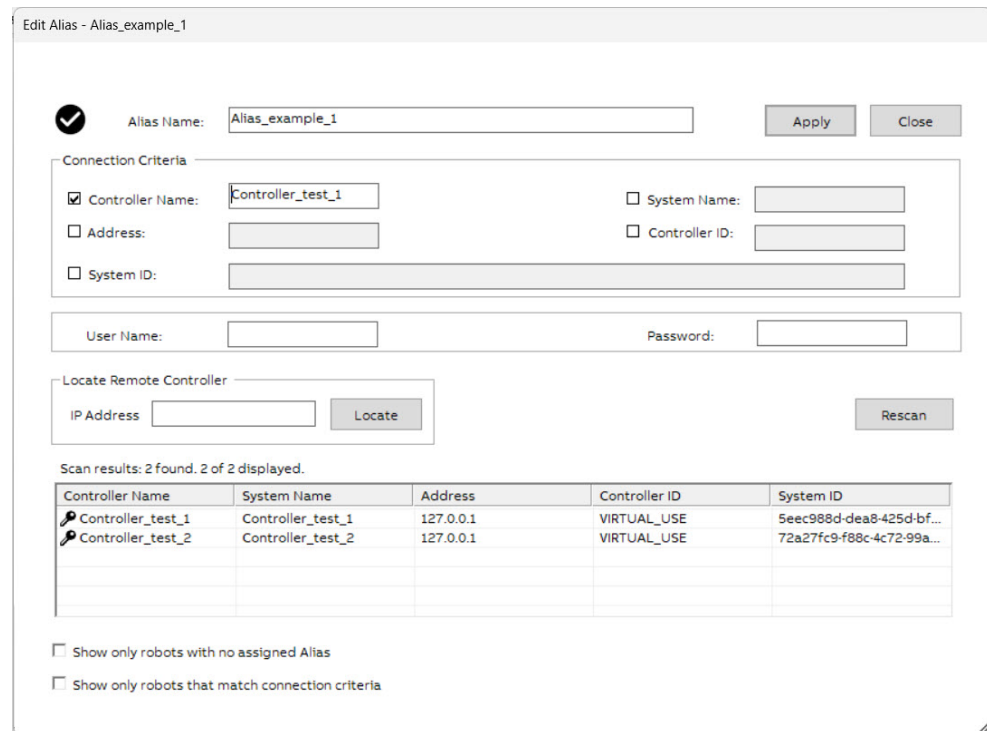
2.2.4 Edit Alias

Overview

This section describes the procedure of editing an alias and provides a description of components displayed in the **Edit Alias** window.

Edit alias screen components

The following figure and table provide details of the components in the **Edit Alias** window:



xx210000140

Component	Function
Alias Name field	This field is disabled by default.
Connection Criteria check boxes	<p>Criteria:</p> <ul style="list-style-type: none"> Controller Name - This is the name assigned to the robot controller. System Name - This is the name assigned to the currently active RobotWare system in the robot controller. Address - This is the robot controller's IP address. If the controller is a Virtual Controller (VC), this is the path to the VC system directory. Controller ID - This is the robot controller's unique hardware ID. System ID - This is the ID assigned to the currently active RobotWare system in the robot controller.
Close button	Closes the display.
Rescan button	Refreshes the list of available controllers in the network.
Scan Results list box	Shows a list of all of the robots detected on the network.

Continues on next page

2 IoT Gateway configuration application

2.2.4 Edit Alias

Continued

Component	Function
Show only robots with no assigned Alias check box	Click to show only those robots that do not have an Alias assigned.
Show only robots that match connection criteria check box	Click to show only those robots whose criteria match the criteria selected in the Connection Criteria fields.

Editing the alias

Use the following procedure to edit an alias:

- 1 On the main screen, select an alias from the list.
- 2 Click the edit alias icon.
The **Edit Alias** window for the selected alias is displayed.
- 3 Select the required controller from the **Scan results** section.



Note

For remote controller, type the IP address in the **IP Address** field and click the **Locate** button. Then, select the remote controller from the **Scan results** section.

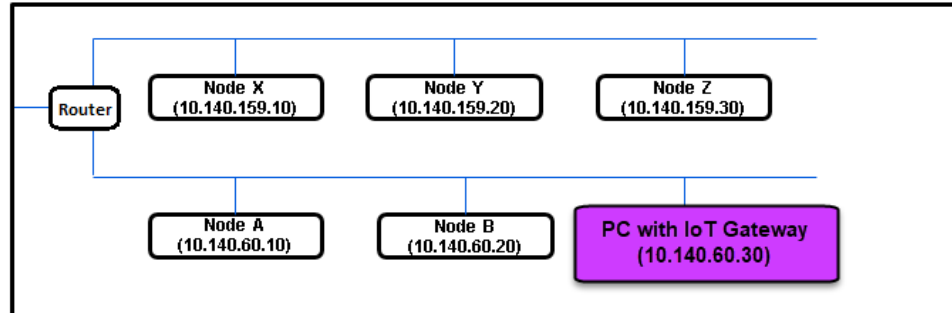
- 4 In the **Connection Criteria** section, edit the fields according to your requirement.
- 5 Click **Apply**.
- 6 Click **Save** on the main screen.
The changes to the selected alias are saved.

2.2.5 Add remote controller

Adding remote controller

You can add a remote controller from multiple subnets.

The following image is a representation of remote controllers in different subnets:



xx210000277

Node	Description
A,B	Controllers in same subnet.
X,Y,Z	Controllers in another subnet.

Add New Alias

✕ Alias Name: Create Close

Connection Criteria

Controller Name: System Name:

Address: Controller ID:

System ID:

User Name: Password:

Locate Remote Controller

IP Address Locate Rescan

Scan results: 2 found. 2 of 2 displayed.

Controller Name	System Name	Address	Controller ID	System ID
🔑 Controller_test_1	Controller_test_1	127.0.0.1	VIRTUAL_USE	5eec988d-dea8-425d-bf...
🔑 Controller_test_2	Controller_test_2	127.0.0.1	VIRTUAL_USE	72a27fc9-f88c-4c72-99a...

Show only robots with no assigned Alias

Show only robots that match connection criteria

xx210000138

Use the following steps to add a remote controller:

- 1 On the main screen, click **Add Alias**.
The **Add New Alias** window is displayed.
- 2 Type the IP address of the remote controller in the **IP Address** text box.

Continues on next page

2 IoT Gateway configuration application

2.2.5 Add remote controller

Continued

3 Click **Locate**.

If located, the available remote controllers are listed in the scan results.

4 You can select the listed remote controller and can add it as an Alias.

2.2.6 The IoT Gateway configuration file

Overview

As discussed in the [Getting started on page 11](#), each IRC5 and OmniCore robot you wish to communicate to the IoT Gateway must contain an Alias definition in the IoT Gateway.

During installation of the IoT Gateway, the installation program places a file called `IoTGateway.config` in a subdirectory of this location, `C:\ProgramData\ABB\IoT Gateway`.

Use the IoT Gateway Config tool to make changes to this configuration file. For a detailed description of the configuration file, see [Appendix C - IoT Gateway configuration file on page 149](#).



Note

It is not recommended to manually edit the `IoTGateway.config` file.

2 IoT Gateway configuration application

2.2.7 Configure IoT Gateway plug-in

2.2.7 Configure IoT Gateway plug-in

Overview

IoT GateWay tab displays the all the supported Gateways. Select or clear the checkbox next to an IoT GateWay to enable or disable it.

IoT GateWay parameters shall be configured with Parameters lane as shown in the following image.

Name	Value	Unit
Broker	localhost	
Topic	abb.com/example	
Port	1883	
Double	3.14	
Boolean	False	
GUID	00000000-0000-0000-0000-000000000000	

xx2100000350

2.2.8 IoT Gateway username/password

Overview

The user credentials entered in the **UserID** tab have the following two purposes:

- To validate connection requests from OPC UA clients.
- As default credentials when connecting to configured robot controllers, unless the Alias configuration for a given robot controller defines specific user credentials.

For information regarding adding or editing user credentials for a specific robot controller, see the sections [How to add new IRC5 or OmniCore robot aliases on page 37](#) and [How to edit a robot alias on page 40](#).



Note

It is recommended to add a dedicated user for the IoT Gateway to all configured robot controllers. This will allow better control of what the IoT Gateway may or may not do, e.g. limiting write access, and it provides for improved auditing.

The dedicated user may have the same username and password on all robot controllers, which simplifies the administration of user credentials.

Username settings

On the main screen, click the **User ID** tab. The **User ID** page is displayed.

Aliases | IoT Gateways | **User ID** | Server Control | Client Certificates | Server Certificate | Logs

Enter the Username and Password that the IoT Gateway uses to obtain privileges to read and write data to the robot controllers.

User name:

Password:

Help Save Exit

xx210000145

Use the following procedure save the user settings:


Step	Action
1	Type a user name in the User Name field.

Continues on next page

2 IoT Gateway configuration application

2.2.8 IoT Gateway username/password

Continued

Step	Action
2	<p>Type a password in the Password field.</p> <p> Note</p> <p>To view the password you typed, select the check box next to the Password field.</p>
3	Click Save to save the changes.

2.2.9 Server control

Overview

In this section, you can control start and stop of the IoT Gateway after you have made changes to its configuration.



Note

You can also automatically restart the server using this option.

The screenshot shows the 'Server Control' tab of the IoT Gateway configuration application. The interface includes a navigation bar with tabs for Aliases, IoT Gateways, User ID, Server Control (selected), Client Certificates, Server Certificate, and Logs. Below the navigation bar, there is a section with the text: 'Use these controls to start and stop the IoT Gateway after you have made changes to its configuration.' This section contains two buttons: 'Restart' and 'Stop'. Below this, there is a section for configuring endpoints with the text: 'Enter the Port number and Server name to configure the Endpoints.' This section contains two input fields: 'Port number' with the value '61510' and 'Server name' with the value 'ABB.loTGateway'. At the bottom right of the application window, there are three buttons: 'Help', 'Save', and 'Exit'.

xx210000146

2 IoT Gateway configuration application

2.2.10 Logs

2.2.10 Logs

Overview

In this section, all the log informations on the operation done are listed.

The screenshot shows the 'Logs' tab in the IoT Gateway configuration application. The log entries are as follows:

```
2024-03-25 15:00:16.998 +05:30 [INF] Gateway enabled - OPC UA for Robotics Companion Specification (OPC 40010-1)
2024-03-25 15:00:16.998 +05:30 [INF] Starting all enabled gateways
2024-03-25 15:00:17.737 +05:30 [INF] [OPC UA] Imported the PFX private key for [A8FE56A4413E8B9CF094F651A8DDA8CD3CA74233].
2024-03-25 15:00:17.798 +05:30 [INF] [OPC UA] Server - Start application IoT Gateway.
2024-03-25 15:00:17.808 +05:30 [INF] [OPC UA] Server - CreateResourceManager.
2024-03-25 15:00:17.891 +05:30 [INF] [OPC UA] Server - CreateRequestManager.
2024-03-25 15:00:17.891 +05:30 [INF] [OPC UA] Server - CreateMasterNodeManager.
2024-03-25 15:00:17.909 +05:30 [INF] [OPC UA] MasterNodeManager.Startup - NodeManagers=6
2024-03-25 15:00:20.154 +05:30 [INF] [OPC UA] Server - CreateEventManager.
2024-03-25 15:00:20.174 +05:30 [INF] [OPC UA] Server - CreateAggregateManager.
2024-03-25 15:00:20.178 +05:30 [INF] [OPC UA] Server - CreateSessionManager.
2024-03-25 15:00:20.181 +05:30 [INF] [OPC UA] Server - CreateSubscriptionManager.
2024-03-25 15:00:20.189 +05:30 [INF] [OPC UA] Server - Enter Running state.
2024-03-25 15:00:20.190 +05:30 [INF] [OPC UA] Server - Started.
2024-03-25 15:00:20.230 +05:30 [INF] [OPC UA] Server - Configuration watcher started.
2024-03-25 15:00:20.233 +05:30 [INF] OPC UA Server started
2024-03-25 15:00:20.234 +05:30 [INF] opc.tcp://in-7506439:61510/ABB.IoTGateway
2024-03-25 15:00:22.330 +05:30 [ERR] [SEC] Invalid user name or password with Alias: Alias_example_1
2024-03-25 15:00:22.361 +05:30 [INF] [Manager] Alias_example_1: Controller is not connected yet
2024-03-25 15:00:22.361 +05:30 [INF] [Manager] Alias_example_1: Controller found but unable to connect
2024-03-25 15:00:22.410 +05:30 [ERR] [SEC] Invalid user name or password with Alias: Alias_example_2
2024-03-25 15:00:22.417 +05:30 [INF] [Manager] Alias_example_2: Controller is not connected yet
2024-03-25 15:00:22.417 +05:30 [INF] [Manager] Alias_example_2: Controller found but unable to connect
2024-03-25 15:00:22.855 +05:30 [INF] Up and running
```

xx210000147

- Latest logs can be seen in Logs tab in IoT Gateway configuration tool.
- IoT Gateway log files can be found at location `C:\ProgramData\ABB\IoT Gateway`.
- IoT Gateway configured to store only latest 10 log files.

2.3 How to add/edit IRC5 or OmniCore robot alias

2.3.1 How to add new IRC5 or OmniCore robot aliases

Adding information manually

In this procedure, you add an Alias name and the Connection Criteria necessary to reliably identify the controller for which you are creating the Alias.





Tip

See ABB's Recommended Associations in section [Aliases on page 21](#) to reliably identify controllers.

Adding new alias

Use the following procedure to add a new robot alias to the IoT Gateway configuration application in the main screen.

Step	Action	Information
1	Click the Add New Alias icon  from the main screen.	The Add New Alias window is displayed.
2	Type an Alias name in the Alias Name field.	
3	Select the connection criteria that you wish to use to identify the robot and enter correct values. You can select more than one criterion.	 Note See ABB's recommended associations to reliably identify controllers in the section Aliases on page 21 .
4	Type user name and password specific to this robot controller. If left empty, the username and password defined in the User ID tab will be used instead.	
5	Click Create .	The Alias you created now appears in the Aliases tab.

Continues on next page

2 IoT Gateway configuration application

2.3.1 How to add new IRC5 or OmniCore robot aliases

Continued

Add New Alias

✕
Alias Name:

Connection Criteria

Controller Name:

System Name:

Address:

Controller ID:

System ID:

User Name:
Password:

Locate Remote Controller

IP Address

Scan results: 2 found. 2 of 2 displayed.

Controller Name	System Name	Address	Controller ID	System ID
Controller_test_1	Controller_test_1	127.0.0.1	VIRTUAL_USE	5eec988d-dea8-425d-bf...
Controller_test_2	Controller_test_2	127.0.0.1	VIRTUAL_USE	72a27fc9-f88c-4c72-99a...

Show only robots with no assigned Alias
 Show only robots that match connection criteria

xx210000148

Using the refresh feature

Use the following procedure to use the Robot Scan feature to detect IRC5 or OmniCore robots connected to the network:

Step	Action	Information/Illustration
1	Click the Add New Alias icon from the main screen.	The Add New Alias window is displayed.
2	Select the controller that you wish to create an Alias for from the list. The IoT Gateway Configuration application displays a default name for the Alias in the Alias Name field, based on the Controller Name and System Name.	
3	Select the Connection Criteria that you wish to use to identify the robot. You can select more than one criterion.	<div style="display: flex; align-items: center;"> <div> <p>Note</p> <p>Note: See ABB's Recommended Associations to reliably identify controllers in Aliases on page 21.</p> </div> </div>
4	You can change the Alias Name as required.	
5	Type user name and password specific to the selected robot controller. If left empty, the username and password defined in the User ID tab will be used instead.	
6	Click Create .	The Alias you created now appears on the IoT Gateway Configuration application main screen.

Continues on next page

2 IoT Gateway configuration application

2.3.1 How to add new IRC5 or OmniCore robot aliases

Continued

Step	Action	Information/Illustration
7	Repeat steps 1-6 for the further alias addition.	
8	Click Cancel to close the window.	
9	Click Save to save the changes.	



2 IoT Gateway configuration application

2.3.2 How to edit a robot alias

2.3.2 How to edit a robot alias

Editing an alias

Use the following procedure to change the association parameters for the selected robot Alias.

Step	Action	Information
1	Click the Edit Alias button  from the main screen.	The Edit Alias dialog box appears.
2	Edit the Alias Name and Connection Criteria that you want to use to identify the robot. You can select more than one criteria.	 Note You can click the Rescan button to identify the robot in the network.
3	Edit user name and password specific to the selected robot controller. If left empty, the username and password defined in the User ID tab will be used instead.	
4	Click Apply .	
5	Click the Save button in main screen.	
6	Restart IoT Gateway.	

Edit Alias - Alias_example_1

Alias Name:

Connection Criteria

Controller Name: System Name:

Address: Controller ID:



System ID:

User Name: Password:

Locate Remote Controller

IP Address

Scan results: 2 found. 2 of 2 displayed.

Controller Name	System Name	Address	Controller ID	System ID
 Controller_test_1	Controller_test_1	127.0.0.1	VIRTUAL_USE	5eec988d-dea8-425d-bf...
 Controller_test_2	Controller_test_2	127.0.0.1	VIRTUAL_USE	72a27fc9-f88c-4c72-99a...

Show only robots with no assigned Alias

Show only robots that match connection criteria

xx210000149

Using the scan feature



Use the following procedure to use the Robot Scan feature to detect IoT Gateway Configuration application connected to the network.

Continues on next page



Note

For information on scan feature components, refer [Edit alias screen components on page 27](#) section.

Step	Action	Information/Illustration
1	Click the Edit Alias icon  from the main screen.	The Edit Alias dialog will appear.
2	Select the controller that you want to create an Alias for from the list. The IoT Gateway Configuration application displays a default name for the Alias in the Alias Name field, based on the Controller Name and System Name.	
3	Select the Connection Criteria that you wish to use to identify the robot. You may select more than one criterion.	 Note Note: See ABB's Recommended Associations to reliably identify controllers in Aliases on page 21 .
4	You can change the Alias Name as required.	
5	Edit user name and password specific to this robot controller. If left empty, the username and password defined in the User ID tab will be used instead."	
6	Click Apply .	The edited Alias appears on the IoT Gateway Configuration application main screen.

2 IoT Gateway configuration application

2.4.1 Client certificates

2.4 Certificate management

2.4.1 Client certificates

Overview

The Client Certificates tab provides an interface for managing the configuration of the security certificates for the application. It allows the user view trusted client certificate list and rejected certificate list.

Name	Valid From	Valid Until
CN=SoftingOpcUaClient	9/26/2023	9/26/2024
CN=UaExpert@acos, OU=dummy, O=dummy, L=dummy, S=dummy, C=...	9/27/2023	9/25/2028

xx2100000150

Trusted client certificates list

Displays the list of certificate that the application trusts. This list includes any Certificate Authority (CA) certificates. Administrator can use this list to check the expiration dates of certificates and renew any certificate prior to their expiration. The administrator can also use the list to ensure that only application that are authorized applications are in the trust list.

- Delete
- Import
- Reject

Delete Trusted Certificate(s)

Delete option allows an administrator to delete the certificate from trusted list.

Import Certificate(s)

Import option allows an administrator to select a certificate and add it to the list of trusted certificates. The certificate can be on any location including flash drives or network share locations. It is the administrator's responsibility to review the certificate and ensure that it is a certificate that belongs to an application that is to be trusted.

Continues on next page

Import option allows an administrator to import an entire list of certificates or to move a list from one store to another. This feature can be used along with a network share to build a list of certificates that are to be stored and then import the list for a new installation.

Reject client certificate(s)

Reject option allows an administrator to reject the certificate(s) from trusted list. so that rejected certificate(s) shall be moved from trusted list to rejected list.

Rejected client certificates list

A Rejected list of Certificates shall contain certificates which are rejected by IoT Gateway Config tool (or) which are rejected by administrator.

- Trust
- Delete

Trust the rejected client certificate(s)

Trust option allows an administrator to trust an entire list of certificates (or) to trust a client certificate. Which will move the client certificate(s) from rejected list(s) to trusted list.

Delete Rejected Certificate(s)

Delete option allows an administrator to delete the certificate(s) from rejected list.

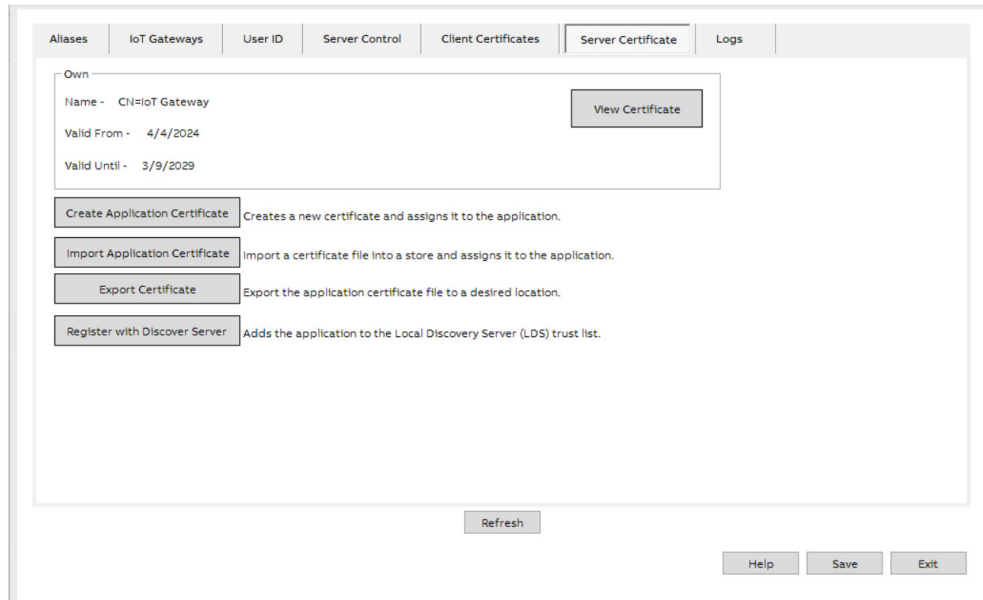
2 IoT Gateway configuration application

2.4.2 Server Application instance certificates




2.4.2 Server Application instance certificates

Overview

You can add the IoT Gateway to Local Discovery Server (LDS) to trust list and also create a new certificate and assign it to the application.



xx210000151

Options	Description
View Certificate button	<p>Click the View Certificate button to display the details of the Application certificate.</p> <p> Note</p> <p>If there are no certificates in the certificate store, error messages will be displayed in a new window and the View Certificate button is disabled.</p>
Create Application Certificate	<p>Creates a new certificate and assigns it to the application.</p> <p> Note</p> <p>The existing certificate will be moved to the backup folder: %ProgramData%\ABB\IoT Gateway\CertificateStores\ApplicationCertificate\backup</p>
Import Application Certificate	<p>Imports a certificate file into a store and assigns it to the application.</p> <p> Note</p> <p>The existing certificate will be moved to the backup folder: %ProgramData%\ABB\IoT Gateway\CertificateStores\ApplicationCertificate\backup</p>

Continues on next page

2 IoT Gateway configuration application

2.4.2 Server Application instance certificates


Continued

Options	Description
Export Certificate	Exports application certificate file to the desired location.
Register with Discover Server	Adds the application to the Local Discover Server (LDS) trust list.

Create application certificate

This creates a new application certificate.

xx210000152

Options	Description
Store Path	Specifies where the application certificate will be placed after it is created.
CA Key File	It is a .pfx file containing the Certificate Authority private key. If left blank a self-signed certificate is created. If provided then the CA Password is required.
CA Password	Displays the password required for CA Key File.
Application Name	Displays the name of the application.
Organization	Displays the name of the organization.
Application URL	Displays the name of the application URL.  Note If checkbox is unchecked then the tool will generate information automatically from the other information provided
Subject Name	Displays the subject name. If checkbox is not selected then the tool will generate information automatically from the other information provided

Continues on next page

2 IoT Gateway configuration application

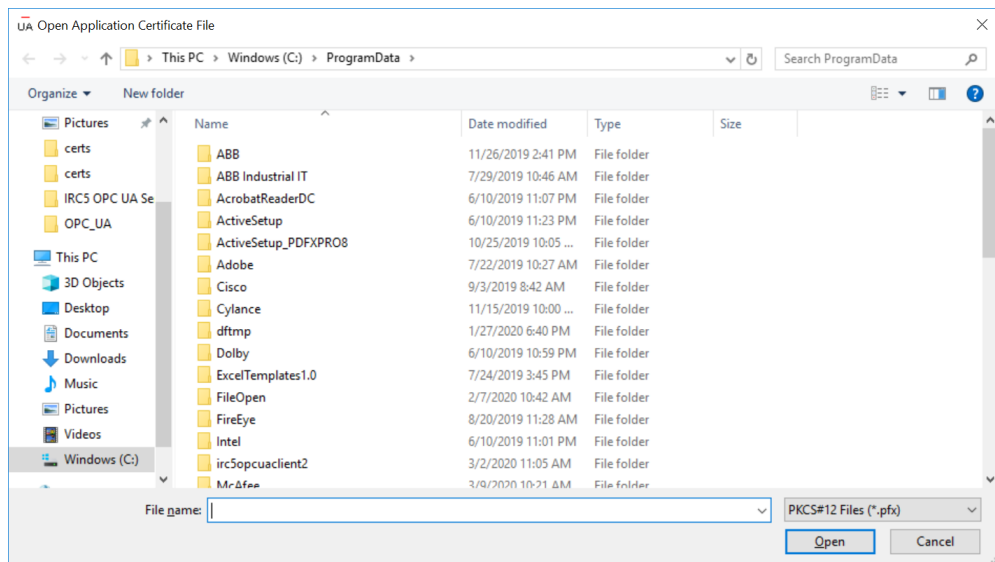
2.4.2 Server Application instance certificates

Continued

Options	Description
Domains	Displays the name of the domain. By default, only the hostname of the computer running the IoT Gateway is listed, but multiple domains may be listed separated by commas. This is especially helpful if a client must use the IP-address instead of the hostname to connect to the IoT Gateway. If the hostame is "MyHost" and the IP-address is 192.168.1.100, then type MyHost, 192.168.1.100 in this field.
Key Size	Displays the size of the key.
Lifetime	Displays the validity of the key.
Key Format	Displays the format of the key.

Import application certificate

This imports an application certificate from a .PFX file stored on disk. It prompts the user to enter a password if one is required. The imported Certificate will replace any Application Instance certificate that may already be assigned to the application.



xx200000294

Application instance certificate can be created by open source tools like using [UA Configuration Tool](#) and export the certificate using IoT Gateway Configuration tool.

Export certificate

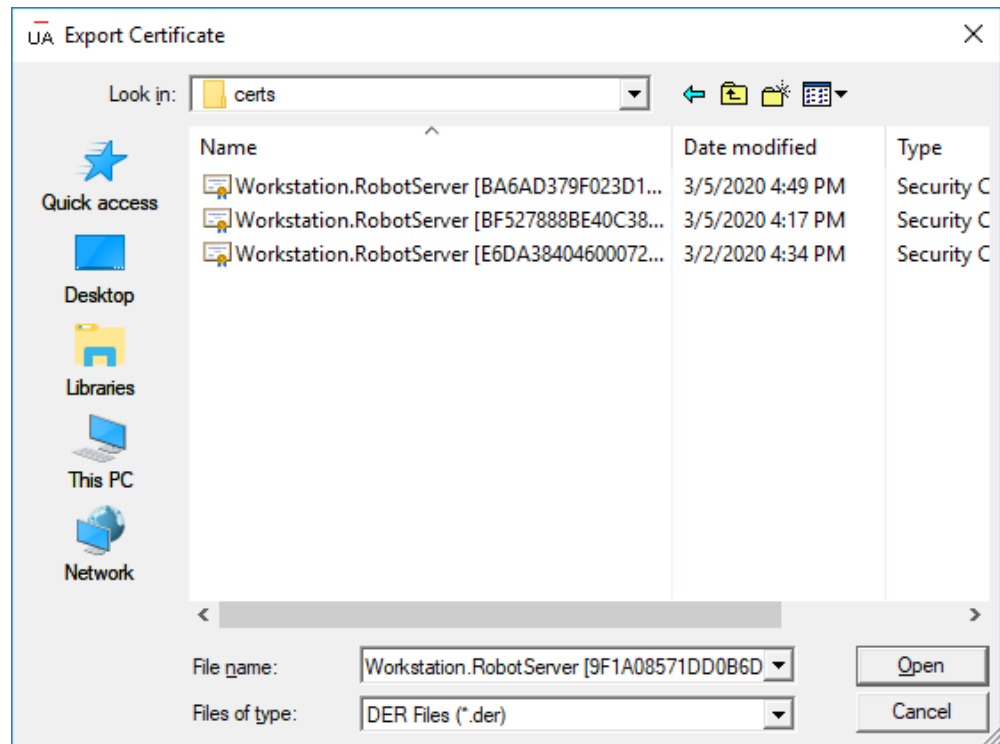
This allows an administrator to generate a file that contains the certificate. This file can then be used to import the certificate onto another machine. The resulting file is a .DER file and does not contain the Private Key information.

Continues on next page

2 IoT Gateway configuration application

2.4.2 Server Application instance certificates

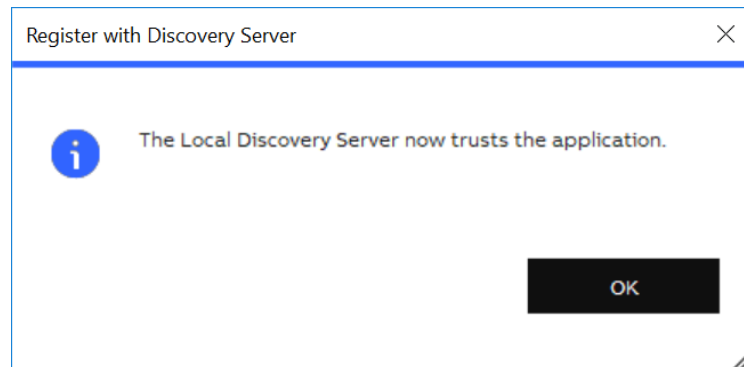
Continued



xx200000292

Register with Discover Server

This allows an administrator to register the selected application's certificate with the Discovery server as a trusted certificate.



xx200000379

The IoT Gateway Configuration tool will give the warning message **The Local Discovery Server is not installed** when Local Discovery Server (LDS) is not installed in the system.

For more information on **Local Discovery Server**, see

<https://opcfoundation.org/developer-tools/samples-and-tools-unified-architecture/local-discovery-server-lds/>

This page is intentionally left blank

3 OPC UA Server

3.1 Address space

Introduction

The OPC UA Data Access function of the OPC UA Server is to read and write data managed by the ABB robot controller.

Data items in OPC UA Server are referred by their node names.

The OPC UA Server presents various predefined nodes that provide information concerning to the robot controller's current state. In addition to these predefined nodes, the OPC UA Server presents up to 1000 additional nodes that contain the values of the I/O signals, as well as up to 200 nodes that contain the values of the RAPID data values for each IRC5 or OmniCore controller configured in the OPC UA Server.

ABB information model

The ABB information model is an ABB Robotics Proprietary OPC UA Information Model for robot controllers.

The tags exposed by the OPC UA Server follow the hierarchical structure of the IRC5 or OmniCore Controller object model.

Tags in the Controller domain of the Controller object model diagram	Unsupported tags in the Controller domain of the IRC5 Controller object model diagram	Tag updated when OPC UA client application requests an update from the server
All of the RAPID and IOSYSTEM tags	CollisionDetectState	SystemClock
OperatingMode	RapidProgramFreememory	
ControllerState	RapidProgramUsedMemory	
ControllerExecutionState		
SpeedRatio		
MasterRAPID		
MasterCFG		
InterfaceState		



Note

All other items are updated only when the controller restarts.

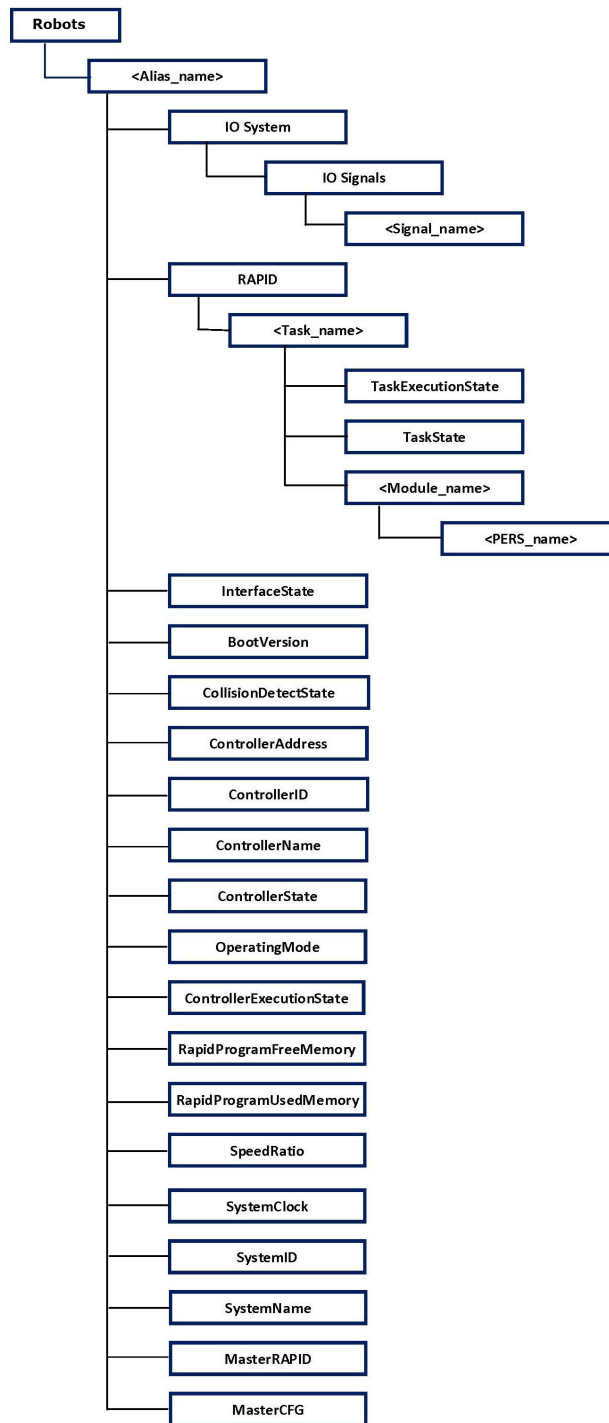
The following image shows the Objects and Variables :

Continues on next page

3 OPC UA Server

3.1 Address space

Continued



xx210000353



Note

For detailed description of the OPC UA Information Model for IRC5 or OmniCore robot controllers, see [Appendix D - ABB Robotics OPC UA proprietary information model on page 151](#).

Continues on next page

OPC UA Robotics Companion Specification

The OPC UA Companion Specification for Robotics specifies an OPC UA Information Model for the representation of a complete motion device system as an interface for higher-level control and evaluation systems. A motion device system consists out of one or more motion devices, which can be any existing or future robot type (e.g. industrial robots, mobile robots), kinematics or manipulator as well as their control units and other peripheral components.

The OPC UA Server supports the mandatory parts and some of the optional parts of the OPC UA Companion Specification for Robotics.

For more information, see [OPC 40010-1 - Robotics Part 1: Vertical Integration](#).

Subscription for data changes

An OPC UA client can subscribe to multiple nodes monitored by OPC UA server which notifies the OPC UA client about the value changes.

The OPC UA server supports subscription to all variables in the address space. Notification of changes to variables like I/O signals and RAPID variables are sent to a subscribing OPC UA client based on change events from the robot controller. Other subscribed variables that typically do not change that often are polled at a two second interval.

Examples of event based variables are:

- I/O signals
- Persistent RAPID variables
- OperatingMode
- ControllerState

Examples of polled variables are:

- SpeedRatio
- TaskProgramName
- TotalPowerOnTime

Robotware supports subscription to a maximum of 200 RAPID variables and 1000 IO signals.



Note

For some Versions of RobotWare, there will be no error message if maximum number of subscriptions exceeded.

3 OPC UA Server

3.2 Events implementation

3.2 Events implementation

Overview

An event monitored item is a special type of a monitored item designed for receiving event notifications from the UA Server. For creating this kind of object, the user needs to create a monitored item for the EventNotifier attribute of an object node. The object node needs to have the SubscribeToEvents bit mask set in the EventNotifier attribute in order to allow the creation of event monitored items.

Event log event

All OPC UA event logs are OPC UA generated events. Some of the parameters included in the event structure that may have special meaning in the context are described below. See OPC UA Alarms and Events specification for the complete list.

- **Source**– The alias name of the controller that generated the event.
- **Message**– The title or brief explanation of the event.
- **Event Category** – The Event Log category.
- **Severity**– The severity of the event.

Following parameters contain event data if attribute values are requested by the client.

- **Number of Event Attributes**– The length of the event attribute array.
- **Event Attributes** – A pointer to the ABB specific event attributes as requested by the client according to the OPC specification.

Source Name	Time	Message	Severity
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:17.000 AM	Automatic mode confirmed	100
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:16.000 AM	Motors OFF state	100
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:15.000 AM	Speed adjusted	100
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:15.000 AM	Automatic mode requested	100
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:15.000 AM	Manual mode selected	100
IN-L-BTGIS15033_OpcUa_VC_Vera	10:24:15.000 AM	Safety guard stop state	100

xx2000000178

Event severity level

The OPC UA Alarms & Events Server automatically translates ABB IRC5 or OmniCore event log types to specific severity levels as shown in the table below.

Event log type	Severity
State Change	100
Warning	300
Error	600

Continues on next page

Event log attributes

In addition to the standard attributes required by the OPC UA Alarms and Events specification, the IoT Gateway Server Alarms and Events Server can provide the attributes defined in the table below.

Attribute ID	Attribute	Description
1	CategoryId	The category to which the event message belongs.
2	Number	The number of the message.
3	Type	The type of the event message.

Model change event

If there is any change in OPC UA Server address space after the server comes online, the server sends a `BaseModelChangeEvent`. This event does not contain information about the changes but indicates only the changes occurred. Therefore the client shall assume that any or all of the nodes may have changed. For example, changes in I/O Signal configuration and Persistent RAPID variables.

The `SourceNode` property identifies the Alias node that the event originated from.

3 OPC UA Server

3.3.1 Transport Protocols

3.3 Security

3.3.1 Transport Protocols

Overview

The OPC UA specification currently defines two data encoding, multiple security protocols and two transport protocols.

Data encodings

Data encodings are available for XML and UA Binary. IoT Gateway uses UA Binary encoding.

UA Binary: This message format encodes the data serialized into a byte array. UA Binary offers reduced computational cost in terms of encoding and decoding but can only be interpreted by OPC-UA compliant clients. UA Binary is more likely to be used in device level communications where processing power is limited and performance is a high priority.

Security protocols

A security protocol ensures the integrity and privacy of UA messages that are exchanged between OPC UA applications.

There are two security protocols defined for OPC UA:

- WS Secure Conversation
- UA Secure Conversation

IoT Gateway supports UA Secure Conversation.

Transport protocols

OPC UA supports the following transport protocols:

- OPC UA TCP
- SOAP/HTTP
- HTTPS

IoT Gateway supports OPC UA TCP transport protocol.

OPC UA TCP: This is a TCP (sockets) based protocol providing a full duplex channel between client and server. Messages are packaged into a structure specified by the OPC UA TCP binary protocol and the structure is transmitted using a socket or secure socket (depending on the endpoints security requirements). As OPC UA TCP is specific to the OPC UA specification only OPC UA Clients and Servers are capable of receiving data transmitted with OPC UA TCP.

3.3.2 Security configuration

Overview

When securing the communication with the OPC UA protocol, the following settings are required:

- Security Policies
- User Token Policies

Security policies

Security policy and SecurityMode (message mode) parameters specify the security algorithms that the UA server supports.

- **Security policy:**

Selection of cryptographic algorithms. Any existing client and server which needs to interact should support this policy. Weaker security policies use outdated algorithms and should not be used. At a minimum, the Security Policy 'Basic256Sha256' should be chosen.

IoT Gateway uses following security policies:

- Basic256Sha256
- Aes128_Sha256_RsaOaep
- Aes256_Sha256_RsaPss

- **SecurityMode:**

The SecurityMode should be 'Sign' or 'SignAndEncrypt'. This ensures that, authentication at the application level is enforced. The SecurityMode 'None' does not provide any protection. SecurityMode 'SignAndEncrypt' provides integrity and confidentiality for the data.

OPC UA supports the following security modes:

- **None** - no encryption, security is turned off. Messages can be read by a 3rd party and tampered with.
- **Sign** - messages are signed to ensure data integrity but the message body is unencrypted. Messages can be read by a 3rd party.
- **Sign and encrypt** - as above but with the message body encrypted. Secure, messages are private and their integrity is assured.

IoT Gateway by default supports only 'Sign' and 'Sign and encrypt'. 'None' security mode is disabled.



Note

'None' security mode can be enabled in IoT Gateway configuration file. But it is not recommended for normal use due to the risk of unintentional or malicious activity.

Continues on next page

3 OPC UA Server

3.3.2 Security configuration

Continued



Note

To configure the security mode as 'None', navigate to the `<SecurityPolicies>` section of the `ABB.Robotics.OPCUA.Server.Config.xml` file and uncomment the four lines defining `<SecurityMode>None_1</SecurityMode>`. `ABB.Robotics.OPCUA.Server.Config.xml` is in the installation folder of IoT Gateway (normally `C:\Program Files (x86)\ABB\IoT Gateway`).

This file is not intended to be edited by the user. Be aware that editing this file will make the IoT Gateway OPC UA server vulnerable to cyber attacks.

User token policies

OPC UA Applications supports authentication of users by providing the necessary authentication credentials to the other entities. OPC UA Applications accept tokens in any of the following forms:

- **Anonymous:** No user information is available
- **UserName:** A user identified by user name and password
- **X509v3:** A user identified by an X509v3 Certificate
- **WSS:** A user identified by a WS-SecurityToken. (e.g. SAML, Kerberos-Ticket)

IoT Gateway supports only Anonymous, UserName user token policies.



CAUTION

The identifier 'anonymous' should be used "only for accessing non-critical UA server resources as it does not provide any protection. It is not possible to trace who has changed the data or configuration on the server side when this generic identifier is used. Also, an attacker could use this identifier to read or write data in an unauthorized manner if no adequate restriction of the rights of the identifier 'anonymous' was configured

Server application instance certificate(X.509 certificate)

When the UA application starts, it attempts to locate and retrieve the certificate configured in the `<ApplicationCertificate>` section of the application configuration. In case the OPC UA stack cannot find the certificate specified by the configuration, it will attempt to create a self-signed certificate using some internal methods. The newly created certificate will be saved at the location specified by configuration in order to be found at the next run of the OPC UA Server. Authentication of clients and servers achieved by using application instance certificates (X.509 certificates). See [Create application certificate](#) and [Import application certificate](#) using IoT Gateway Configuration tool.

3.3.3 Restricting access to application folder

Overview

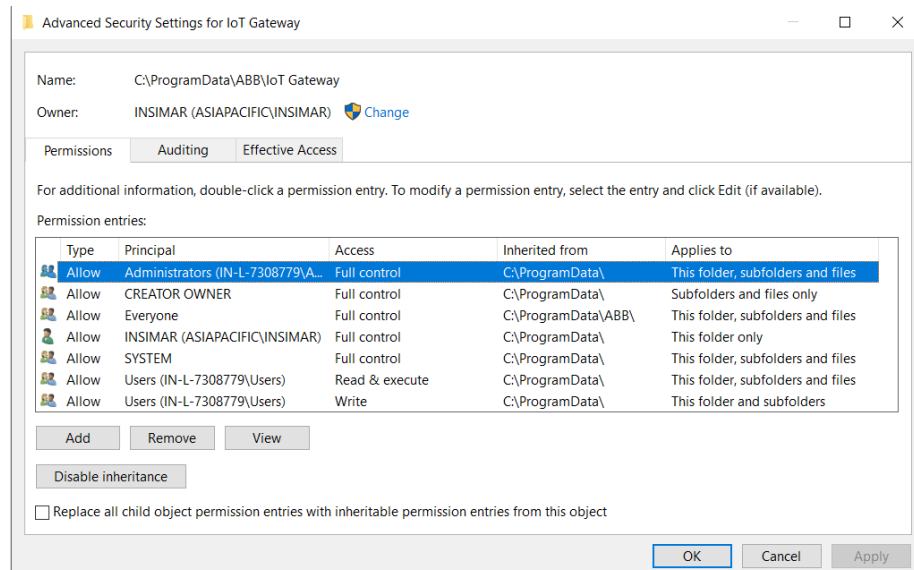
The IoT Gateway keeps several files in a folder called “IoT Gateway” under %ProgramData%\ABB¹. These files include configuration, log and certificate files, and should be protected from unintentional or malicious access. During initial installation, this folder is created, and security settings are configured to allow access by users belonging to the Administrators group and the SYSTEM user only. It is possible to change these security settings manually to provide access to other users. Any changes to the security settings are preserved during an upgrade of the IoT Gateway.

Viewing security settings

Follow these steps to view the security settings :

- 1 Browse to %ProgramData%\ABB folder from Windows Explorer.
- 2 Right-click IoT Gateway folder and select Properties.
- 3 In the Properties dialog, select Security tab, and press Advanced.
The Advanced Security Settings dialog may display Continue.
- 4 Click Continue and enter administrative credentials, if prompted.

The following screen is displayed.



xx2100000163

Follow these steps to restore the security settings to recommended default values :

- 1 Press **Disable inheritance** and turn off the inheritance.
- 2 Modify the list of permission entries to include SYSTEM and Administrators only.
- 3 Select the **Replace all child object permissions entries from this object** check box.

¹ %ProgramData% is by default C:\ProgramData, but may be different depending on the Windows installation.

Continues on next page

3 OPC UA Server

3.3.3 Restricting access to application folder

Continued

- 4 Click **Apply**.

3.4 How to connect to OPC UA Server

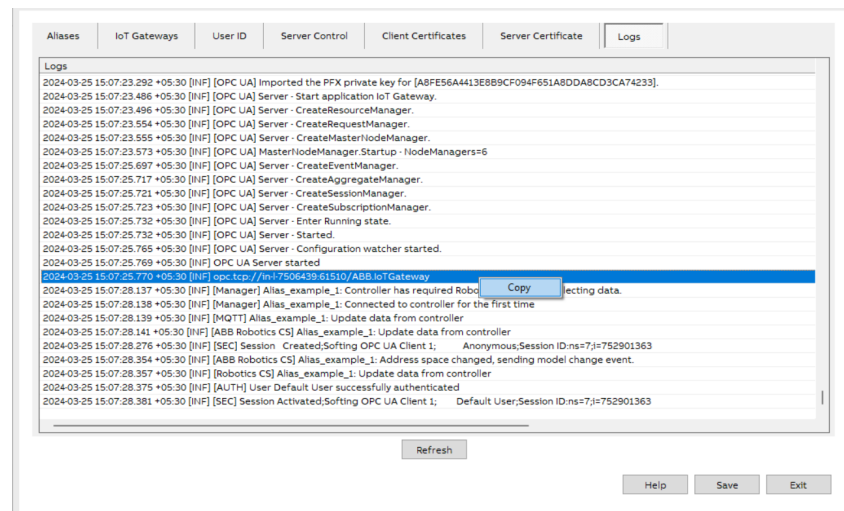
Overview

In a secure connection, the OPC UA server and OPC UA client must trust each other to protect the data exchange between the OPC UA server and OPC UA client. To establish a secure connection between an OPC UA server and OPC UA client, perform the following tasks:

1 Endpoint Information

- Endpoint URL:

The easiest way to get the correct endpoint URL is to go to the Log tab in the IoT Gateway Configuration Tool, find the line containing the server's endpoint URL and right-click it to copy. The server's endpoint URL will not change unless the name of the computer the server is running on is changed, or the port number is changed using the IoT Gateway Configuration Tool.



xx2100000162

- Security Settings

Security Policy: Select one of the following OPC UA Server supported security policies.

- Basic256Sha256
- Aes128_Sha256_RsaOaep
- Aes256_Sha256_RsaPss

Message Security Mode (or) Security Policy: Select one of the following OPC UA Server supported security mode

- Sign
- Sign and encrypt

Security Message Encoding: Select **Binary** as it is supported by OPC UA Server.

2 Authentication Settings

Continues on next page

3 OPC UA Server

3.4 How to connect to OPC UA Server

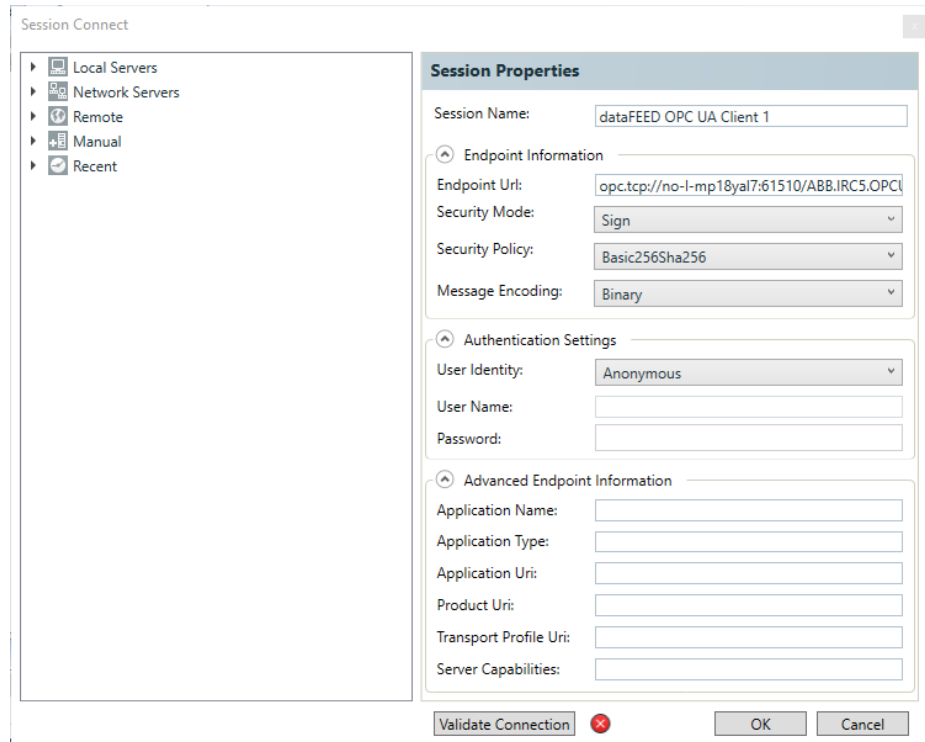
Continued

Select one of the OPC UA Server supported user token type (User Identity)

- **Anonymous:** No user information is available.
- **User Name:** A user is identified by username and password.

OPC UA Clients should provide RobotWare username and password for authentication.

For example, we are using Softing's dataFEED OPC UA test client as an example



xx2000001099

- 3 The first time a connection attempt made from a new OPC UA client, it will fail.

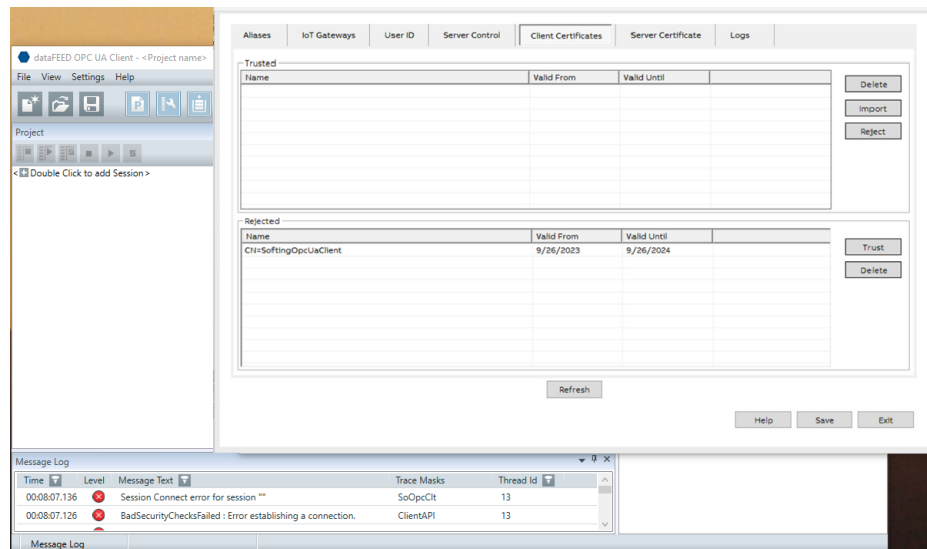
This is for security reasons; OPC UA clients and servers may use certificates to make sure they communicate with an approved server or client, and the default security settings for the OPC UA Server is to require a trusted client certificate. In addition, firewall settings may also cause the connection to fail.

- 4 Trust OPC UA Client Certificate:

OPC UA Client may give a **BadSecurityChecksFailed** error message. Go to the **Client Certificates** tab in the OPC UA Config Tool and check for new certificate in the **Rejected** list. Select this certificate and press the **Trust** button. This moves the certificate to the **Trusted** list.

When a client tries to connect for the first time, it will be rejected and its certificate will be placed in the **Rejected** list. This allows the administrator to review clients before allowing them to connect.

Continues on next page



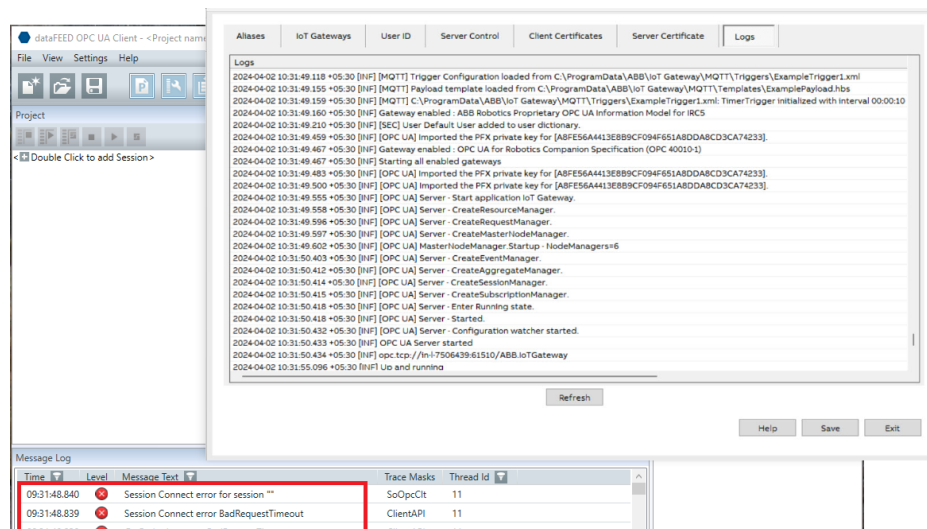
xx2000001097

If this does not help, check for firewall log and firewall settings and consider adding a rule for allowing incoming connections to the port number specified in the endpoint URL.

- 5 Trust OPC UA Server Certificate: OPC UA Clients should also trust the OPC UA Server certificate to establish secure communication.
- 6 Firewall issues :

If the OPC UA client complains about timeout connection errors, and there is nothing in the OPC UA Server log about failed connection attempts, it is most likely that the firewall settings needs to be changed.

How to change the firewall settings depends on the firewall solution. Consult IS-support to create a firewall rule that allows incoming connections to the IoT Gateway. The port number used by the OPC UA Server can be seen in the Server Control tab of the configuration tool.



xx2000001098

Continues on next page

3 OPC UA Server

3.4 How to connect to OPC UA Server

Continued



Note

First, check that the OPC UA Server is running. This is easy to do by opening the Windows **Task Manager** and select the **Services** tab. The **ABB.Robotics.IoTGateway** should have a **Running** status. If it is stopped, right click and select **Start** to start it again.

4 MQTT Publisher

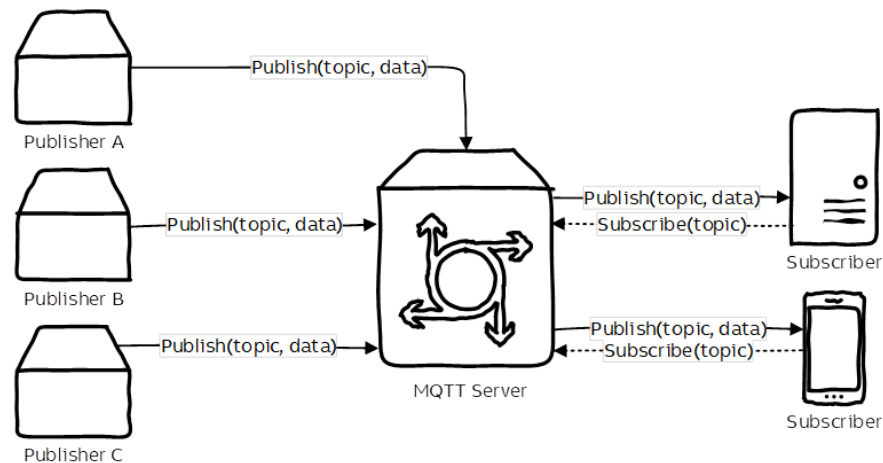
4.1 Introduction to MQTT Publisher

Introduction

MQTT is a light weight publish and subscribe based protocol that is widely used to collect data from industrial devices. MQTT is managed by Oasis and is an open standard and an ISO recommendation (ISO/IEC 20922).

Working of MQTT

In MQTT there are clients that may publish messages, or subscribe to messages, or both. All messages are transferred via an MQTT server. The server will receive a message from a publisher and forward it to all clients that subscribe to that message. Messages are published under a topic. This allows the server and subscriber clients to differentiate between messages.



Terminology

The table below define terms that are important to understand how the Configurable MQTT Publisher works.

Term	Description
Network Connection	(OASIS, 2019): A construct provided by the underlying transport protocol that is being used by MQTT. <ul style="list-style-type: none"> It connects the Client to the Server. It provides the means to send an ordered, lossless, stream of bytes in both directions.
Application Message	(OASIS, 2019): The data carried by the MQTT protocol across the network for the application. When an Application Message is transported by MQTT it contains payload data, a Quality of Service (QoS), a collection of Properties, and a Topic Name.

Continues on next page

4 MQTT Publisher

4.1 Introduction to MQTT Publisher

Continued

Term	Description
Client	(OASIS, 2019): A program or device that uses MQTT. A Client: <ul style="list-style-type: none">• opens the Network Connection to the Server• publishes Application Messages that other Clients might be interested in.• subscribes to request Application Messages that it is interested in receiving.• unsubscribes to remove a request for Application Messages.• closes the Network Connection to the Server.
Server	(OASIS, 2019): A program or device that acts as an intermediary between Clients which publish Application Messages and Clients which have made Subscriptions. A Server: <ul style="list-style-type: none">• accepts Network Connections from Clients.• accepts Application Messages published by Clients.• processes Subscribe and Unsubscribe requests from Clients.• forwards Application Messages that match Client Subscriptions.• closes the Network Connection from the Client.
Subscription	(OASIS, 2019): A Subscription comprises a Topic Filter and a maximum QoS. A Subscription is associated with a single Session. A Session can contain more than one Subscription. Each Subscription within a Session has a different Topic Filter.
Topic Name	(OASIS, 2019): The label attached to an Application Message which is matched against the Subscriptions known to the Server.
Will Message	(OASIS, 2019): An Application Message which is published by the Server after the Network Connection is closed in cases where the Network Connection is not closed normally.

4.2 Configuring MQTT Publisher

4.2.1 Introduction

Overview

The configuration of the MQTT Publisher is split into four separate parts:

Term	Description
Client Configuration	Configuration of the MQTT publisher client. This includes name or address of MQTT server, authentication, and other security settings. For more details, see MQTT client configuration on page 66 .
Robot Configuration	The main purpose of the Robot Configuration is to configure which Triggers apply to each robot controller. Robot controllers are identified by their IoT Gateway Alias name. For more details, see Robot configuration on page 77 .
Trigger Configuration	Used to define when and what to publish. For more details, see Trigger configuration on page 79 .
Payload Configuration	Defines templates used to generate the payload of MQTT messages. For more details, see Payload configuration on page 94 .

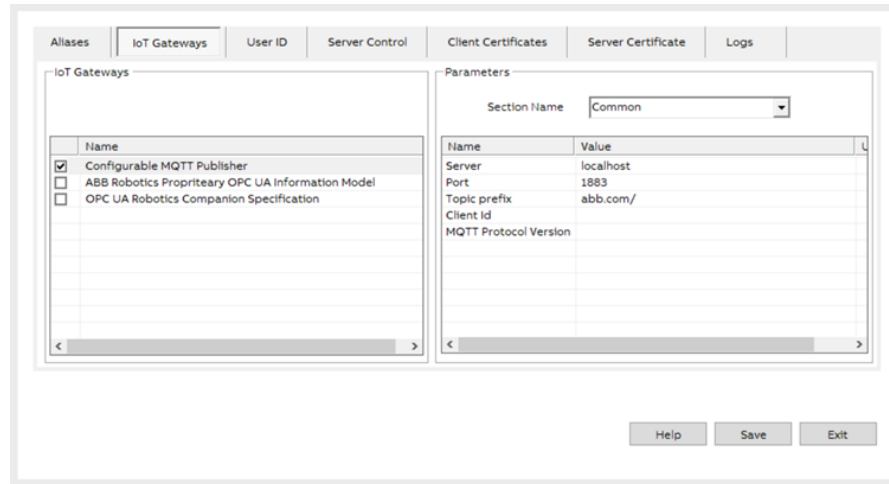
4 MQTT Publisher

4.2.2 MQTT client configuration

4.2.2 MQTT client configuration

Introduction

To use the Configurable MQTT Publisher, it must be enabled using the IoT Gateway Config tool as shown in the following image.



xx2100001468

Also, the following parameters need to be configured:

- Common parameters
 - **Server** – the name or IP address of the MQTT server.
 - **Port** – the port number, default value is 1883.
- Security parameters
 - Username and Password if required by MQTT server, leave empty otherwise.
 - "Use TLS" set to 'false' to disable secure communication.



Note

If the value of Use TLS parameter is set to 'true', be sure to have a matching TLS configuration on MQTT Server and Clients.

Common Parameters

Overview

The details of the Common Parameters are documented in this section.

Server

The hostname or IP address of the MQTT Server.

All MQTT messages published by the Configurable MQTT Publisher are sent to the MQTT Server identified by this hostname or IPC address. See also the Port parameter.

Port

The port number of the MQTT Server. Default value is 1883.

Continues on next page

All MQTT messages published by the Configurable MQTT Publisher are sent to the MQTT Server listening to this port number. See also the Server parameter.

Topic prefix

A string that is used as the start of all MQTT topic names in published messages. The Topic parameter of Each Trigger Configuration is appended to this Topic prefix to form the final MQTT topic name. Leave empty if there is no common topic prefix.

MQTT Client ID

Client ID to use with MQTT server. Leave empty if not used.

See also <https://www.cloudmqtt.com/blog/mqtt-what-is-client-id.html>

MQTT Protocol Version

Allows the protocol version to be specified. This parameter should normally be left empty. This allows the MQTT client to adapt to the protocol version used by the MQTT server it connects to.

Security parameters

Overview

A good understanding on how MQTT security works is necessary to correctly configure these parameters. The Security chapter in the MQTT specification found at <https://mqtt.org/mqtt-specification/> is one source of information, but for a more readable introduction, the blog posts at <https://www.hivemq.com/tags/mqtt-security-fundamentals/> are highly recommended.

The Security section contains the following parameters.

Username

Username to use with MQTT server. Leave empty if username and password are not used.

Password

Password to use with MQTT server. Leave empty if not used.

Use TLS

Set to true to enable Transport Layer Security for MQTT.

If set to false, the remaining TLS related parameters are ignored.

CA Certificate

File name of CA certificate to use with TLS.

Optional, but if defined, used to validate the server certificate.

Client Certificate

File name of .pfx file to use with TLS for client authentication.

Optional, but if defined, used to authenticate the client at the server.

Ignore Chain Errors

Set to true to ignore certificate chain errors.

Must be true when using self-signed certificates. Not recommended for public networks.

Continues on next page

4 MQTT Publisher

4.2.2 MQTT client configuration

Continued

Allow Untrusted

Set to true to allow untrusted certificates. Not recommended for public networks.

Ignore Revocation Error

Set to true to ignore certificate revocation errors.

Must be true if there are no revocation list or server.

Will Message parameters

Overview

An MQTT client may send a "Will Message" when connecting to the MQTT Server. If the client or client connection "dies" unexpectedly, for example, due to network issues, the Server will send the Will Message to all MQTT subscribers subscribing to the "Will Topic".

The parameters related to the Will Message are described below and can be found in the section named "Will" in the configuration tool.

Topic

Topic of the Will Message. An empty string disables use of Will Message. Note that the Topic prefix parameter is not prepended to the Will Topic. The Will Topic must be a UTF-8 Encoded string.

Payload

Text to use as payload for Will Message.

QoS

QoS level to be used when publishing the Will Message. Must be one of:

- 0 - At most once delivery
- 1 - At least once delivery
- 2 - Exactly once delivery

Retain

Set to true if the Will Message is to be retained when it is published.

Format Parameters

The Format section contains parameters related to formatting data.

Language tag

The language tag is used to select the culture to be used when rendering payload templates. This includes which character to use as decimal separator in numeric values, date and time formats, etc.

Language tag is set to:

- 'Empty' to use the Invariant Culture (default).
- The desired language tag, e.g., 'de-CH' for German-Switzerland. A list of supported language tags in Windows can be found here: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-licid/.
- 'installed' to use the culture installed with the operating system

4.2.3 MQTT Engineering Tool

4.2.3.1 Introduction

Overview

MQTT Engineering Tool is an application specialized in editing and validating configuration files (Robot Configuration, Trigger Configuration, Payload Configuration) for the configurable MQTT Publisher in IoT Gateway.

4 MQTT Publisher

4.2.3.2 Main Screen Components

4.2.3.2 Main Screen Components

Overview

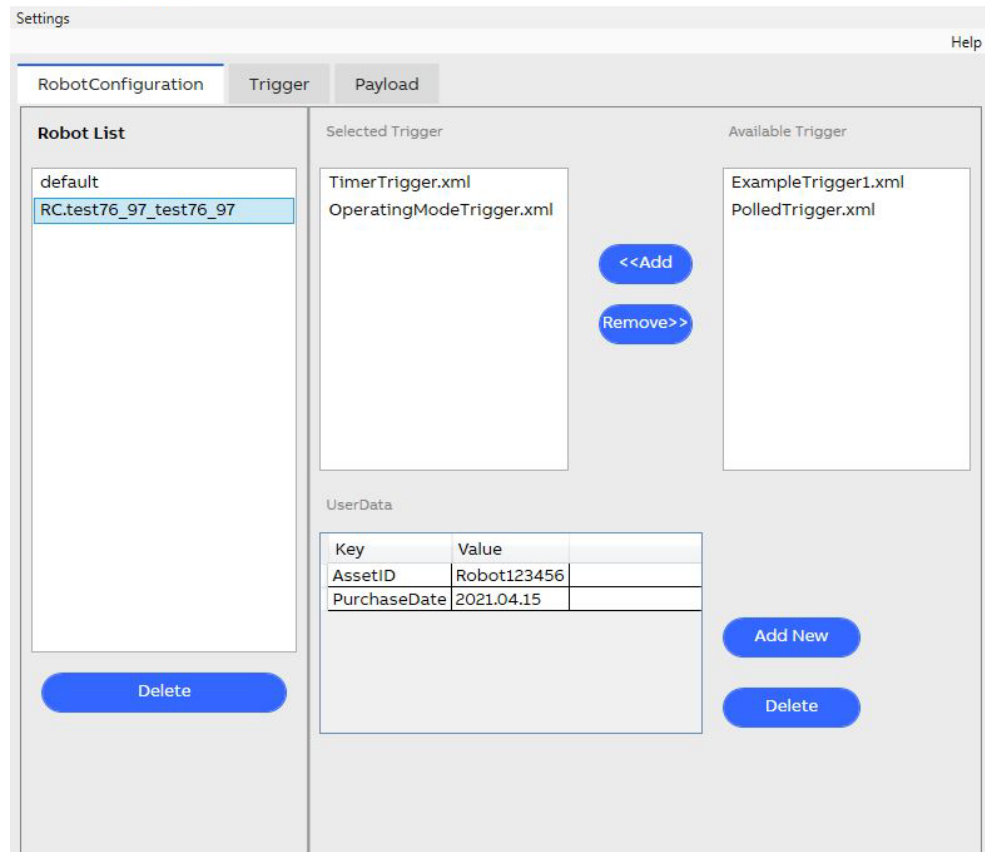
The main screen of MQTT Engineering Tool consists of the following:

- 3 tabs for each configuration file (Robot Configuration, Trigger, and Payload)
- Settings menu at top left corner
- Help option at top right corner

Robot configuration tab

The **Robot Configuration** tab displays a list of robot configurations, selected triggers, available triggers, and user data. For more information about Robot Configuration, see [Robot configuration on page 77](#).

The following figure and table provides a description of the fields available in the **Robot Configuration** tab.



xx2200000452

Field	Description
Robot List	Displays the list of available robot configurations.
Delete	Deletes a selected robot configuration from the system.
Selected Trigger	Displays the list of triggers that are selected in the robot configuration file.

Continues on next page

Field	Description
Available Trigger	Displays the list of triggers that are available for the selected robot configuration file.
Add	Adds a selected trigger from the Available Trigger list to the Selected Trigger list.
Remove	Allows you to remove a selected trigger from the Selected Trigger list to the Available Trigger list.
UserData	Displays the userdata parameters and its values. <ul style="list-style-type: none"> • Key: The unique name for the UserData item. • Value: The value of the UserData item.
Add New	Adds a new UserData Key Value pair.
Delete	Deletes a selected UserData from the list.

Trigger tab

The Trigger tab allows you to manage the triggers. For more information about Trigger Configuration, see [Trigger configuration on page 79](#).

The following figure and table provides a description of the fields available in the Trigger tab.

The screenshot shows the 'Trigger' configuration tab in the MQTT Publisher settings. On the left, there is a 'Trigger List' containing several trigger names, with 'PolledTrigger' selected. Below the list are 'New', 'Copy', and 'Delete' buttons. The main configuration area for the selected trigger includes:

- Name:** PolledTrigger
- Type:** Polled (dropdown menu)
- Parameters:** 10
- Guard Condition:** true = true
- Topic:** {{Alias}}
- Payload:** MaxPayload.hbs (dropdown menu)
- QoS:** 1 (dropdown menu)
- Retain flag:** Checked

xx220000453



Field	Description
Trigger List	Displays a list of available triggers.
New	Creates a new trigger.

Continues on next page

4 MQTT Publisher

4.2.3.2 Main Screen Components

Continued

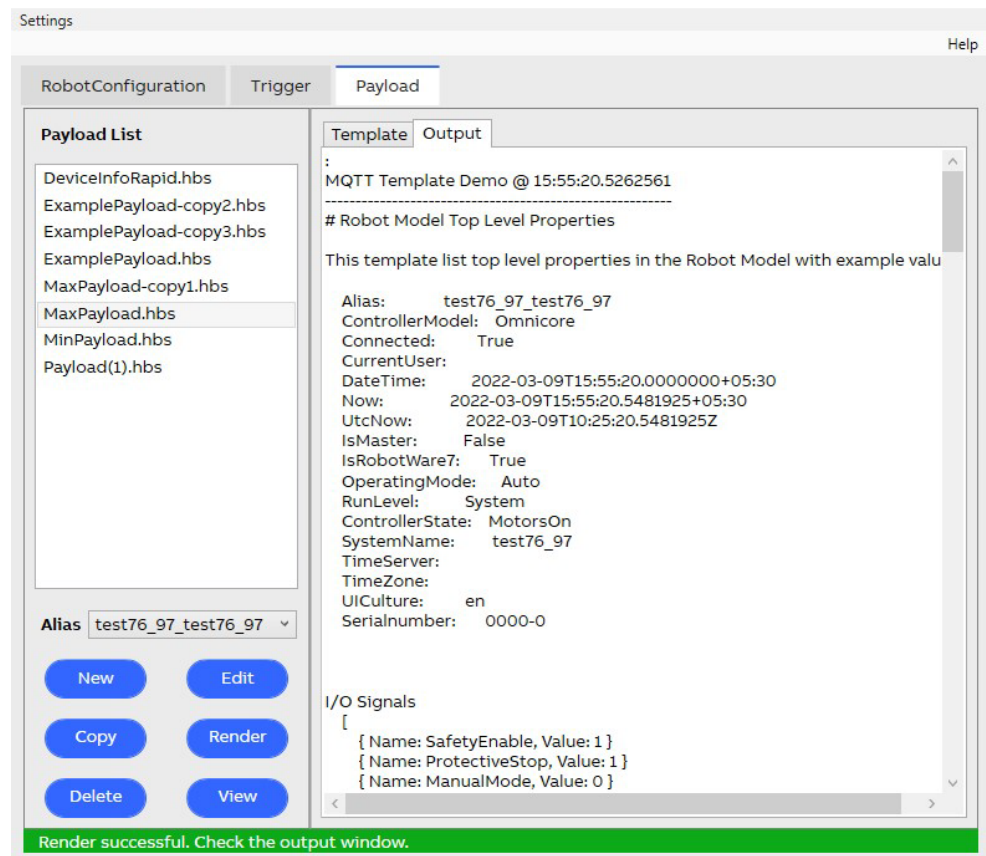
Field	Description
Copy	Makes a copy of the current selected trigger.
Delete	Deletes a selected trigger.
Name	Allows you to edit the name of the trigger.
Type	Allows you to select a type for the trigger. For more information, see Trigger configuration on page 79 .
Parameters	<p>Allows you to provide a value for the selected trigger type. For more information, see Trigger configuration on page 79.</p> <p> Note Validation is not done for the values.</p>
Guard Condition	<p>The optional logical expression that must be true for publishing message. For more information, see Trigger configuration on page 79.</p> <p> Note Validation is not done for the values.</p>
Topic	MQTT topic for the published message. For more information, see Trigger configuration on page 79 .
Payload	The Payload configuration file name. For more information, see Trigger configuration on page 79 .
QoS	MQTT Quality of Service. For more information, see Trigger configuration on page 79 .
Retain flag	Sets or unsets the MQTT retain flag. For more information, see Trigger configuration on page 79 .

Payload tab

The **Payload** tab displays a list of payload templates, their content, and rendered output. For more information about payload configuration, see [Payload configuration on page 94](#).

The following figure and table provides a description of the fields available in the **Payload** tab.

Continues on next page



xx220000454

Field	Description
Payload	Displays a list of available payload templates.
Alias	Displays a list of configured controllers.
New	Creates a new payload template.
Copy	Creates a copy of the selected payload template.
Delete	Deletes the selected payload template.
Edit	Opens the selected payload template for editing. By default, it opens the notepad application if no editor is configured in the settings.
Render	Renders the selected payload template. The rendered output is available in the Output tab.
View	Displays the rendered output of the selected payload template. By default, it opens the notepad application if no editor is configured in the settings.
Template	Displays content of the selected payload template.
Output	Displays rendered output of the selected payload template.

Settings

The Settings tab contains the options to set editor and viewer application path. The following figure and table provides a description of the fields available in the Settings tab.

Continues on next page

4 MQTT Publisher

4.2.3.2 Main Screen Components

Continued

The screenshot shows a configuration dialog with two main sections: 'Editor' and 'Viewer'. Each section contains a text input field for the path and a 'Browse' button. Below each path field is a 'Command-line Arguments' text input field. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

xx2200000455

Field	Description
Editor	Allows you to configure an editor for editing the templates. Browse button: Browse and select a text editor. Command-line Arguments: Text Editor related command-line parameters can be used . This field is optional.
Viewer	Allows you to configure a viewer for viewing the templates. Browse button: Browse and select a text editor. Command-line Arguments: Text editor related command-line parameters can be used. This field is optional.
OK	Saves the changes made to the settings.
Cancel	Cancels the changes made to the settings.

Example: Editor configuration

In the following example, editor path is set to

“C:\Users\AppData\Local\Programs\Microsoft VS Code\Code.exe” with the Command-line Argument “-n”. Once this settings are saved, the **Edit** button in the **Payload** tab opens the selected payload in MS VS Code in a new window.

Continues on next page

The screenshot shows a configuration dialog with two main sections: 'Editor' and 'Viewer'.
The 'Editor' section contains a text field with the path 'C:\Users\AppData\Local\Programs\Microsoft VS Code\Code.exe' and a blue 'Browse' button to its right. Below this is a 'Command-line Arguments' field containing '-n'.
The 'Viewer' section contains an empty text field and a blue 'Browse' button to its right. Below this is another empty 'Command-line Arguments' field.
At the bottom right of the dialog are two blue buttons: 'OK' and 'Cancel'.

xx220000456

Example: Viewer configuration

In the following example, the Viewer path is set to “C:\Users\AppData\Local\Programs\Microsoft VS Code\Code.exe” with the Command-line Argument “-n”. Once this settings are saved, the View button in the Payload tab opens the selected payload output in MS VS Code in a new window.

Continues on next page

4 MQTT Publisher

4.2.3.2 Main Screen Components

Continued

Editor

Command-line Arguments

Viewer

Command-line Arguments

xx2200000457

4.2.4 Robot configuration

Purpose

The purpose of the Robot Configuration is twofold:

- Define a list of Triggers that applies to the robot.
- Define a table of user defined data items that is specific to each robot controller.

Files

For each robot controller Alias configured in the IoT Gateway Config tool, there is a Robot Configuration file. The file is named according to the template:

```
RC.<Alias>.xml
```

Where <Alias> is replaced by the Alias name of the robot controller.

The Robot Configuration files are stored in: %ProgramData%\ABB\IoT Gateway\MQTT\Controllers

Any missing Robot Configuration files will be automatically created when the IoT Gateway is (re-)started.

Default configuration

When a new Robot configuration file is created automatically after adding a new Alias to the IoT Gateway, the default configuration is taken from the `default.xml` file found in the %ProgramData%\ABB\IoT Gateway\MQTT\Controllers folder. The `default.xml` file may be edited to contain a list of trigger files and user data items most suitable for installation or project.

Content

List of Triggers

The list of Triggers is a list of file names identifying MQTT message triggers that applies to this robot controller. The same Trigger Configuration file may be referenced by multiple Robot Configuration files.

User Data

A table of user defined data items that is specific to each robot controller stored as key-value pairs. The value of a user data item can be rendered in an MQTT message by referencing it in a Handlebars placeholder. For details see [User data on page 126](#).

Example

Example of a Robot Configuration file listing two Trigger Files that applies to this Robot and two items in the UserData table:

```
<?xml version="1.0" encoding="utf-8"?>
<RobotController
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <TriggerFiles>
    <string>ControllerStateTrigger.xml</string>
```

Continues on next page

4 MQTT Publisher

4.2.4 Robot configuration

Continued

```
    <string>PolledTrigger.xml</string>
  </TriggerFiles>
  <UserData>
    <item>
      <key>AssetID</key>
      <value>Robot123456</value>
    </item>
    <item>
      <key>PurchaseDate</key>
      <value>2021.04.15</value>
    </item>
  </UserData>
</RobotController>
```

4.2.5 Trigger configuration

4.2.5.1 Overview

Purpose

A Trigger configuration defines when to publish an MQTT message, which topic to publish it under, and the template to use for the payload.



Note

Trigger events will be lost if the time between trigger events are shorter than the time it takes to retrieve required data from the robot controller. When this happens, a "Lost trigger" warning will be logged. After logging a lost trigger event, no new warning will be logged for a period of 10 seconds. To avoid or reduce the number of lost trigger events, consider reducing payload size or trigger frequency.

Files

Each Trigger configuration is stored in its own file. It is recommended to use XML as file name extension, and the files must be stored in: %ProgramData%\ABB\IoT Gateway\MQTT\Triggers

If a Robot configuration file references a Trigger configuration file that does not exist, a default Trigger Configuration file with the correct name will be created when the IoT Gateway is restarted.

Content

The elements of a Trigger configuration are listed below.

Type

Trigger type. There are several different trigger types described in the following sections. Examples are triggers for changes in an I/O signal, a robot controller state variable, or a Rapid variable.

Parameters

Parameters according to the trigger type, for example, the name of the I/O signal to monitor the changes. See the sections on each trigger type for details.

Guard Condition

An optional logical expression that must evaluate to TRUE before the MQTT message is published. See the section on the Guard Condition for details.

Topic

The Topic under which the MQTT message is published. The configured Topic is treated as a template, and hence it may contain Handlebar placeholders. See the section on the Topic for details.

Payload

The name of a file containing the Handlebars template to use for the MQTT message payload. See the section on the Payload Configuration for details

Continues on next page

4 MQTT Publisher

4.2.5.1 Overview

Continued

QoS

MQTT defines three levels of Quality of Service (QoS). The QoS defines how hard the server/client will try to ensure that a message is received. Following are the three levels:

- 0: The server/client will deliver the message once, with no confirmation.
- 1: The server/client will deliver the message at least once, with confirmation required.
- 2: The server/client will deliver the message exactly once by using a four-step handshake.



Note

Higher levels of QoS are more reliable, but involve higher latency and have higher bandwidth requirements.

Retain

All the messages may be set to be retained. This means that the server will keep the messages even after sending it to all the current subscribers. If a new subscription is made that matches the topic of the retained message, then the message will be send to the client. This is useful as a "last known good" mechanism. If a topic is not updated frequently, then without a retained message, a newly subscribed client may have to wait a long time to receive an update. With the retained messages, client will receive an instant update.

Example

Below is an example of a Trigger Configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Timer</Type>
  <Parameters>10</Parameters>
  <GuardCondition>true</GuardCondition>
  <Topic>{{Alias}}/Example</Topic>
  <Payload>ExamplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```


4.2.5.2 Trigger Type

4.2.5.2.1 Introduction

Overview

There are several different Trigger Types to choose from. This section describes each of them in detail.

4 MQTT Publisher

4.2.5.2.2 ControllerState trigger

4.2.5.2.2 ControllerState trigger

Purpose

The ControllerState trigger publishes an MQTT message when the Controller State changes.

Example

Example of a ControlleState Trigger:

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>ControllerState</Type>
  <Parameters></Parameters>
  <GuardCondition>true</GuardCondition>
  <Topic>{{Alias}}/ControllerState</Topic>
  <Payload>ControllerStateChange.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4.2.5.2.3 Polled trigger

Purpose

If none of the other trigger types provide the desired trigger functionality, the Polled trigger can be used. It will trigger on a change in value of any data item that is available in the Robot Model.

How it works

As the name suggests, a Polled trigger does not depend on a (subscribe-able) event like the other trigger types. Instead, the trigger condition specified in the parameters is evaluated at regular intervals. If there is a change in the evaluated value from the previous poll, and the Guard Condition evaluates to TRUE, the MQTT message is published.

Example

Example of Polled trigger configured to publish an MQTT message when the SpeedRatio changes:

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Polled</Type>
  <Parameters>{{SpeedRatio}}</Parameters>
  <GuardCondition>true</GuardCondition>
  <Topic>{{Alias}}/SpeedRatio</Topic>
  <Payload>SpeedRatioChange.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4 MQTT Publisher

4.2.5.2.4 Timer trigger

4.2.5.2.4 Timer trigger

Purpose

The Timer trigger is used to publish a message at regular intervals.

How it works

A timer is set up to expire at regular intervals defined by the Parameters section. When the timer expires, a message is published provided the Guard Condition evaluates to TRUE.

Parameters

A string defining the interval of the timer. It is expected to contain a string that conforms to the form:

```
{ s[.ff] | d.hh:mm[:ss[.ff]] | hh:mm[:ss[.ff]] }
```

Elements in square brackets ([and]) are optional. One selection from the list of alternatives enclosed in braces ({ and }) and separated by vertical bars (|) is required. The following table describes each element.

Element	Description
s	Seconds, ranging from 0 to 2 ³² .
ff	Optional fractional seconds, consisting of one to seven decimal digits.
d	Days, ranging from 0 to 10675199.
hh	Hours, ranging from 0 to 23.
mm	Minutes, ranging from 0 to 59.
ss	Optional seconds, ranging from 0 to 59.

Example

Example of Timer trigger configured to publish an MQTT message every 5 s:

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Timer</Type>
  <Parameters>5</Parameters>
  <GuardCondition>>true</GuardCondition>
  <Topic>{{Alias}}/RegularUpdate</Topic>
  <Payload>ExamplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>>true</Retain>
</Trigger>
```

4.2.5.2.5 Program Execution trigger

Purpose

The Program Execution trigger is used to send a message when the Rapid Execution Status changes.

How it works

Whenever the Rapid ExecutionStatus changes, a message will be send provided that the Guard Condition evaluates to TRUE.

Example

There are two Execution Status need to be provided as parameter: start, stop
Example of Program Execution Trigger with “start”.

```
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>ProgramState</Type>
  <Parameters>start</Parameters>
  <GuardCondition></GuardCondition>
  <Topic>/ProgramState</Topic>
  <Payload>examplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

Example of Program Execution Trigger with “stop”.

```
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>ProgramState</Type>
  <Parameters>stop</Parameters>
  <GuardCondition></GuardCondition>
  <Topic>{{Alias}}/ProgramState</Topic>
  <Payload>examplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4 MQTT Publisher

4.2.5.2.6 Operating Mode trigger

4.2.5.2.6 Operating Mode trigger

Purpose

The Operating Mode trigger is used to send a message when the Operating Mode of the robot controller changes.

How it works

Whenever the RAPID OperatingMode changes, a message will be send provided that the Guard Condition evaluates to TRUE.

Example

Example of Operating Mode Trigger

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>OperatingMode</Type>
  <Parameters></Parameters>
  <GuardCondition>true</GuardCondition>
  <Topic>/{{Alias}}/ControllerOperatingMode</Topic>
  <Payload>examplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4.2.5.2.7 Controller Connect trigger

Purpose

The Controller Connect trigger is used to send a message when the IoT Gateway (re-)connects to a robot controller.

How it works

Whenever the IoT Gateway (re-)connects to a robot controller, a message will be send provided that the Guard Condition evaluates to TRUE.

Example

Example of Controller Connect Trigger.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>ControllerConnected</Type>
  <Parameters></Parameters>
  <GuardCondition>>true</GuardCondition>
  <Topic>{{Alias}}/ControlleConnected</Topic>
  <Payload>DeviceInfo.hbs</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4 MQTT Publisher

4.2.5.2.8 Controller I/O Signal trigger

4.2.5.2.8 Controller I/O Signal trigger

Purpose

The Controller I/O Signal trigger publishes a message when an I/O signal changes.

How it works

The I/O Signal specified in the Parameters section is evaluated. If there is a change in the IO Signal value and the GuardCondition evaluates to TRUE, the MQTT message is published.

Example

Example of Controller I/O Signal Trigger.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>IOSignal</Type>
  <Parameters>do1_1</Parameters>
  <GuardCondition></GuardCondition>
  <Topic>{{Alias}}/IOSignal</Topic>
  <Payload>examplePayload_iosignal.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>

</Trigger>
```


4.2.5.2.9 EventLog Message trigger

Purpose

The EventLog Message trigger publishes a message when one or more event logs occur.

How it works

The EventLog trigger works in two different modes, unbuffered and queued.

In unbuffered mode, any new Event Log message triggers an evaluation of the GuardCondition and, if it evaluates to TRUE, the configured payload is rendered and published. To use unbuffered mode, leave the Parameters field empty.

In queued mode, the Parameters field contains a floating-point number indicating how many seconds to wait after receiving the first EventLog Message in a series. After the wait time, the GuardCondition is evaluated and, if TRUE, the configured payload is rendered and published. In this mode, the payload template may send all outstanding EventLog messages in the same MQTT message. For details, See [ElogMessageQ on page 107](#).

When the IoT Gateway reconnects with a robot controller after being disconnected for a while, the Event Log Message trigger will try to retrieve all event log messages from the robot controller that has occurred in the meantime, up to a maximum of 1000 messages. These messages are subsequently processed as defined by the configuration of the Event Log Message trigger.

Parameters

A string defining how long to wait after receiving an event log message from the robot controller until processing it and any additional event log messages received in the meantime.

A string conforming to the following format is expected:

[s[.ff]]

Element	Description
s	Seconds, ranging from 0 to 2 ³² .
ff	Optional fractional seconds, consisting of one to seven decimal digits.

Elements in square brackets ('[' and ']') are optional.

If the string is empty, then the trigger operates in unbuffered mode.

If the string contains a valid, positive floating-point number, trigger operates in queued mode.

In all other cases, an error is reported, and the trigger operates in unbuffered mode.

Example

Example of EventLog Message Trigger with a four second delay before rendering and sending the MQTT message.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Type>EventLog</Type>
```

Continues on next page

4 MQTT Publisher

4.2.5.2.9 EventLog Message trigger

Continued

```
<Parameters>4.0</Parameters>  
<GuardCondition>>true</GuardCondition>  
<Topic>/EventLog</Topic>  
<Payload>EventLogs.hbs</Payload>  
<QoS>1</QoS>  
<Retain>true</Retain>  
</Trigger>
```

4.2.5.2.10 Rapid Variable trigger

Purpose

The Rapid Variable trigger publishes a message when a persistent RAPID variable changes.

How it works

If there is a change in the Rapid Variable value and the GuardCondition evaluates to TRUE, the MQTT message is published.

The Parameters element refers a persistent RAPID variable by providing task name, module name and variable name separated by space or '.' (dot). For example:
"T_ROB1.user.MyVar"

Example

Example of Rapid Variable Trigger.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>RapidVariable</Type>
  <Parameters>T_ROB1.MainModule.valBool</Parameters>
  <GuardCondition></GuardCondition>
  <Topic>{{Alias}}/RapidVariable</Topic>
  <Payload>examplePayload_Rapid.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4 MQTT Publisher

4.2.5.3 Guard condition

4.2.5.3 Guard condition

Purpose

Guard condition is used to further limit when a message is published after a trigger has been activated. The Guard Condition is a logical expression that is evaluated before publishing an MQTT message. If the Guard Condition evaluates to TRUE, the message is published.

How it works

When a Guard Condition is evaluated, it is first rendered as a Handlebars template. This will replace any Handlebars placeholders with the current value of whatever the placeholder represents. All robot data available for the Payload template is also available for the Guard Condition.

Next, the resulting logical expression is evaluated using a mathematical expression solver. The solver supports the following logical operators:

- ()
- AND
- OR
- =
- <
- <=
- >
- >=
- !=

In addition, common mathematical operators may be applied to numbers:

- ()
- *
- /
- +
- -

Example

Here is an example of a trigger with a Guard Condition making sure the MQTT message is published only if the robot controller is in Automatic mode.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Timer</Type>
  <Parameters>5</Parameters>
  <GuardCondition>'{{OperatingMode}}'='Auto'</GuardCondition>
  <Topic>{{Alias}}</Topic>
  <Payload>ExamplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>>true</Retain>
</Trigger>
```

4.2.5.4 Topic

Purpose

A trigger configuration includes a Topic that will be used as the MQTT topic in messages published by the Trigger.

How it works

The Topic is treated as a Handlebars template. This means that any Handlebars placeholders are replaced by the current value of whatever the placeholder represents. All robot data available for the Payload template is also available for the Topic.

The resulting string is appended to the **Topic prefix** parameter value as described in [Topic prefix on page 67](#).

Example

Here is an example of a trigger with a Topic that use the `{{Alias}}` placeholder to make the alias name of the robot controller part of the MQTT topic.

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Timer</Type>
  <Parameters>5</Parameters>
  <GuardCondition>'{{OperatingMode}}'='Auto'</GuardCondition>
  <Topic>{{Alias}}</Topic>
  <Payload>ExamplePayload.txt</Payload>
  <QoS>1</QoS>
  <Retain>true</Retain>
</Trigger>
```

4 MQTT Publisher

4.2.6.1 Overview

4.2.6 Payload configuration

4.2.6.1 Overview

Purpose

The Payload Configuration defines the structure and data content of published MQTT messages.

Files

Each Payload Configuration is stored in its own file. It is recommended to use HBS as file name extension, and the files must be stored in: %ProgramData%\ABB\IoT Gateway\MQTT\Templates

A Payload Configuration file may be referenced by several Trigger Configuration files. If a Trigger configuration file references a Payload configuration file that does not exist, a default Payload Configuration file with the correct name is created when the IoT Gateway is restarted.

Content

A Payload Configuration file is a text file that is used as a template for generating an MQTT message payload. The MQTT Publisher looks for placeholders in the template and replace the placeholders with the robot controller data that the placeholder references. Everything else in the template is copied as is to the resulting message payload.

A placeholder is recognized by double curly brackets surrounding some text that identifies which data to replace it with, for example, {{Alias}}. The template language is called Handlebars. See the Handlebars chapter for a detailed description of the language.

Apart from the placeholders, there is no restriction on the syntax of the payload template. This means that payload templates may be written in any text format required.

Example

In this example, let us assume that the robot controller has the following tags and values:

Tag	Value
Alias	Welder1
OperatingMode	Auto
ControllerExecutionState	Running

JSON

If the MQTT messages needs to be published in JSON format, a template using the example tags may be written as:

```
{ "StatusMessage" :  
  {  
    "Name" : " {{Alias}} ",
```

Continues on next page

```
"Mode": "{{OperatingMode}}",
"ProgramStatus": "{{ControllerExecutionState}}"
}
}
```

When rendered, this template will generate the following output:

```
{"StatusMessage":
{
"Name": "Welder1",
"Mode": "Auto",
"ProgramStatus": "Running"
}
}
```

XML

If the MQTT messages need to be published in XML format, a template using the example tags may be written as:

```
<?xml version="1.0" encoding="utf-8"?>
<StatusMessage>
<Name>{{Alias}}</Name>
<Mode>{{OperatingMode}}</Mode>
<ProgramStatus>{{ControllerExecutionState}}</ProgramStatus>
</StatusMessage>
```

When rendered, this template will generate the following output:

```
<?xml version="1.0" encoding="utf-8"?>
<StatusMessage>
<Name>Welder1</Name>
<Mode>Auto</Mode>
<ProgramStatus>Running</ProgramStatus>
</StatusMessage>
```

Markdown

If the MQTT messages needs to be published in Markdown format, a template using the example tags may be written as:

```
# Robot Status for {{Alias}}

Operating mode: {{OperatingMode}}

Program status: {{ControllerExecutionState}}
```

When rendered, this template will generate the following output:

```
# Robot Status for Welder1

Operating mode: Auto

Program status: Running
```

4 MQTT Publisher

4.2.6.2.1 Introduction

4.2.6.2 Robot model

4.2.6.2.1 Introduction

Overview

The Robot Model provides all the data of a robot controller that may be accessed from a Handlebars template.

4.2.6.2.2 Top Level

Introduction

At the top level of the Robot Model, there are several properties and state variables that may be accessed from a Handlebars template.

Alias

Gets the alias name of the robot controller as defined in the IoT Gateway configuration.

Syntax: `{{Alias}}`

Connected

Flag telling if there is currently a connection with the robot controller.

Syntax: `{{Connected}}`

Possible values:

- True – if currently connected
 - False – if not connected
-

Controller model

Gets the model name of the controller.

Syntax: `{{ControllerModel}}`

Possible values:

- True – if currently connected
 - False – if not connected
-

Controller state

Gets the current state of the controller.

Syntax: `{{ControllerState}}`

Possible values:

- Init
 - MotorsOff
 - MotorsOn
 - GuardStop
 - EmergencyStop
 - EmergencyStopReset
 - SystemFailure
 - Unknown
-

DateTime

Gets the current local time of the controller.

Syntax: `{{DateTime}}`

Continues on next page

4 MQTT Publisher

4.2.6.2.2 Top Level

Continued

UtcDateTime

Gets the current UTC time of the robot controller.

Syntax: `{{UtcDateTime}}`

Now

Gets the current time of the IoT Gateway.

Syntax: `{{Now}}`

UtcNow

Gets the current universal time of the IoT Gateway.

Syntax: `{{UtcNow}}`

IsMaster

Flag telling if the IoT Gateway currently has mastership of the robot controller.

Syntax: `{{IsMaster}}`

Possible values:

- True – if master
- False – if not master

IsRobotWare7

Flag telling if the robot controller runs RobotWare 7.

Syntax: `{{IsRobotWare7}}`

Possible values:

- True – if RobotWare 7
- False – if not RobotWare 7

Operating mode

Gets the current operating mode of the controller.

Syntax: `{{OperatingMode}}`

Possible values:

- Auto
- Init
- ManualReducedSpeed
- ManualFullSpeed
- AutoChange
- ManualFullSpeedChange
- NotApplicable

RunLevel

Gets the current run level of the robot controller.

Syntax: `{{RunLevel}}`

Possible values:

- Unknown

Continues on next page

- System
- Boot

Serialnumber

Gets the serial number of the first mechanical unit that has a serial number.

Syntax: {{Serialnumber}}

StartTime

Gets the system clock at the last startup of the controller.

Syntax: {{StartTime}}

SystemId

Gets the id of the current system of the controller.

Syntax: {{SystemId}}

SystemName

Gets the name of the current system of the controller.

Syntax: {{SystemName}}

TimeServer

Gets the NTP time server of the controller.

Syntax: {{TimeServer}}

TimeZone

Gets the time zone of the controller.

Syntax: {{TimeZone}}

UICulture

Gets the UI Culture.

Syntax: {{UICulture}}

4 MQTT Publisher

4.2.6.2.3 Configuration

4.2.6.2.3 Configuration

Introduction

Provides access to the CFG domain.

The configuration data is organized as a database inside RobotWare. For MQTT it is exposed as a multi-level dictionary, starting with the CFG Domains at the top.

To get the value of a configuration parameter, the syntax is as follows:

```
{{Cfg.<domain>.<type>.<instance>.Value}}
```

Where:

<domain> is the name of the cfg domain

<type> is the name of a cfg type within the cfg domain

<instance> is the name of a cfg instance within the cfg type.

Cfg Domain

Gets a single CFG domain.

Syntax: {{Cfg.<domain>}}

Where <domain> is the name of the domain, e.g. SYS.

To list all available domains:

```
{{#each Cfg}}
```

```
{{@key}}
```

```
{{/each}}
```

Cfg Domain Type

Gets a cfg type.

Syntax: {{Cfg.<domain>.<type>}}

Where:

<domain> is the name of the domain, e.g. SYS

<type> is the name of a Cfg type within the domain

To list all available types within a domain:

```
{{#each Cfg.<domain>}}
```

```
{{@key}}
```

```
{{/each}}
```

Cfg Instance

Gets a cfg instance.

Syntax: {{Cfg.<domain>.<type>.<instance>}}

Where:

<domain> is the name of the cfg domain, e.g. SYS

<type> is the name of a cfg type within the cfg domain

<instance> is the name of a cfg instance within the cfg type.

To list all available instances of a cfg type:

Continues on next page

```
{{#each Cfg.<domain>.<type>}}  
  {{@key}}  
{{/each}}
```

Cfg Instance Attribute

Gets an attribute of a cfg instance.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>}}

Where:

<domain> is the name of the cfg domain, e.g. SYS

<type> is the name of a cfg type within the cfg domain

<instance> is the name of a cfg instance within the cfg type.

<attribute> is the name of the cfg attribute

To list all the attributes with values of an instance:

```
{{#each Cfg.<domain>.<type>.<instance>}}  
  {{@key}}: {{Value}}  
{{/each}}
```

Cfg Instance Attribute Properties

Each attribute has several properties. To list all the properties of an attribute:

```
{{#each Cfg.<domain>.<type>.<instance>.<attribute>}}  
  {{@key}}  
{{/each}}
```

InitialValue

Gets the initial value of the attribute.

Syntax:

```
{{Cfg.<domain>.<type>.<instance>.<attribute>.InitialValue}}
```

Mandatory

Flag that tells if the attribute is mandatory or not.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>.Mandatory}}

MaxValue

Gets the maximum value of the attribute.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>.MaxValue}}

MinValue

Gets the minimum value of the attribute.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>.MinValue}}

Size

Gets the size of the attribute. 1 is a single value, all others are arrays.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>.Size}}

Continues on next page

4 MQTT Publisher

4.2.6.2.3 Configuration

Continued

Unit

Gets the unit of the attribute.

Syntax: {{Cfg.<domain>.<type>.<instance>.<attribute>.Unit}}

4.2.6.2.4 ControllerInfo

Introduction

ControllerInfo contains a set of controller properties that do not change unless the robot controller reconfigured, or a new RobotWare version is installed.

Alias

Gets the alias name of the robot controller as defined in the IoT Gateway configuration. This is the same as the Alias property at the top level.

Syntax: `{{ControllerInfo.Alias}}`

RemoteController

Gets a flag indicating whether the robot controller is in another subnet.

Syntax: `{{ControllerInfo.RemoteController}}`

Possible values:

- True – if controller is in another subnet
- False – if not

ControllerConnectionStatus

Gets current controller connection status.

Syntax: `{{ControllerInfo.ControllerConnectionStatus}}`

Possible values:

- NotYetConnected - No connection has been made to the robot controller yet.
- Connected - There is an active connection to the robot controller.
- LostConnection - The connection to the robot controller was lost.
- MissingOption - A proper connection cannot be made as the robot controller is missing a mandatory option.

ControllerName

Gets the robot controller name.

Syntax: `{{ControllerInfo.ControllerName}}`

HostName

Gets the host name of the controller.

Syntax: `{{ControllerInfo.HostName}}`

ControllerID

Gets the controller ID.

Syntax: `{{ControllerInfo.ControllerID}}`

IPAddress

Gets the robot controller IP address.

Syntax: `{{ControllerInfo.IPAddress}}`

Continues on next page

4 MQTT Publisher

4.2.6.2.4 ControllerInfo

Continued

IsVirtual

Gets a flag to indicate if the controller is virtual or real.

Syntax: `{{ControllerInfo.IsVirtual}}`

Possible values:

- True – if controller is a virtual controller
- False – if not

MacAddress

Gets the mac address of the controller.

Syntax: `{{ControllerInfo.MacAddress}}`

SystemId

Gets the system ID (GUID)

Syntax: `{{ControllerInfo.SystemId}}`

SystemName

Gets the system name.

Syntax: `{{ControllerInfo.SystemName}}`

Version

Gets the RobotWare version as a Version object. This allows the separate fields of the version number to be accessed individually as shown in the syntax description below.

Syntax:

`{{ControllerInfo.Version}}`

`{{ControllerInfo.Version.Major}}`

`{{ControllerInfo.Version.Minor}}`

`{{ControllerInfo.Version.Build}}`

`{{ControllerInfo.Version.Revision}}`

VersionName

Gets the RobotWare version as a string.

Syntax: `{{ControllerInfo.VersionName}}`

4.2.6.2.5 ElogMessage

Introduction

Depending on the configured mode of the EventLog Message Trigger, the ElogMessage behaves as follows:

In **unbuffered mode**, the ElogMessage is updated by the EventLog Message Trigger when a new event log message is received from the robot controller.

In **queued mode**, the ElogMessage contains the next event log message in the queue waiting to be processed during rendering of the template associated with the EventLog Message Trigger.



Note

The data in the ElogMessage is valid only during execution of the EventLog Message Trigger.

SequenceNumber

Gets the sequence number for the event

Syntax: `{{ElogMessage.SequenceNumber}}`

Timestamp

Gets the local time when the event happened.

Syntax: `{{ElogMessage.Timestamp}}`

UtcTimestamp

Gets the UTC time when the event happened.

Syntax: `{{ElogMessage.UtcTimestamp}}`

CategoryId

Gets the category to which the event belongs.

Syntax: `{{ElogMessage.CategoryId}}`

Number

Gets the number of the event.

Syntax: `{{ElogMessage.Number}}`

Code

Gets the code number of the event.

Calculated as $(CategoryId * 10000 + Number)$.

Syntax: `{{ElogMessage.Code}}`

Type

Gets the type of the event.

Syntax: `{{ElogMessage.Type}}`

Continues on next page

4 MQTT Publisher

4.2.6.2.5 ElogMessage

Continued

Possible values:

Error	An error event.
Information	An information event.
Warning	A warning event.

Title

Gets the title of the event.

Syntax: {{ElogMessage.Title}}

Description

Gets the description of the event.

Syntax: {{ElogMessage.Description}}

Hint

Get proposed user actions to fix the cause of the event.

Syntax: {{ElogMessage.Hint}}

Consequences

Gets a description of the consequences of the event.

Syntax: {{ElogMessage.Consequences}}

Causes

Gets likely causes for the event.

Syntax: {{ElogMessage.Causes}}

Body

Gets the XML encoded body of the event.

Syntax: {{ElogMessage.Body}}

4.2.6.2.6 ElogMessageQ

Introduction

The ElogMessageQ is only used when the EventLog Message Trigger is configured for queued mode. When an EventLog Message Trigger event occurs, the ElogMessageQ contains an ordered list of event log messages that can be processed when rendering the associated Handlebars template.

ElogMessageQ

Gets the queue of unprocessed event log messages.

Syntax:

```
{{#each ElogMessageQ}}  
  {{<elog message field>}} {{<elog message field>}} ...  
{{/each}} {{ElogMessageQClear}}
```

Where <elog message field> is any field as described in the section [ElogMessage on page 105](#).

ElogMessageQClear is a helper that removes all event log messages from the queue. For more details, see [Robotics handlebars helpers on page 123](#).

Count

Gets the current number of event log messages in the queue.

Syntax: {{ElogMessageQ.Count}}

LastDequeued

Gets the last ElogMessage popped from the queue. This is primarily intended for internal use to keep track of the last processed event log message. This allows the IoT Gateway to know from which message to start reporting after a robot controller and the IoT Gateway have been disconnected for a while.

Syntax: {{ElogMessageQ.LastDequeued.<elog message field>}}

Where <elog message field> is any field as described in the section [ElogMessage on page 105](#).

4 MQTT Publisher

4.2.6.2.7 I/O signal

4.2.6.2.7 I/O signal

Introduction

Individual I/O signals can be accessed by name as shown below.

Syntax: `{{IO.Signals.<name>}}`

Where < name > is the name of the I/O signal.

Name

Gets the name of the I/O signal.

Syntax: `{{IO.Signals.<name>.Name}}`

Value

Gets the current logical value of the I/O signal.

Syntax: `{{IO.Signals.<name>.Value}}`

IsSet

Valid for Digital signals only.

Gets flag telling if the digital signal is currently set or not.

Syntax: `{{IO.Signals.<name>.IsSet}}`

Possible values:

- True - The digital signal is not set (value = 0).
 - False - The digital signal is set (value = 1).
-

GroupValue

Valid for Group signals only.

Gets the current value of the group signal as an integer value.

Syntax: `{{IO.Signals.<name>.GroupValue}}`

MinValue

Gets the logical minimum value of the I/O signal.

Syntax: `{{IO.Signals.<name>.MinValue}}`

GroupMinValue

Valid for Group signals only.

Gets the minimum value of the group signal as an integer value.

Syntax: `{{IO.Signals.<name>.GroupMinValue}}`

MaxValue

Gets the logical maximum value of the I/O signal.

Syntax: `{{IO.Signals.<name>.MaxValue}}`

GroupMaxValue

Valid for Group signals only.

Gets the maximum value of the group signal as an integer value.

Continues on next page

Syntax: `{{IO.Signals.<name>.GroupMaxValue}}`

Quality

Gets the current quality of the I/O signal.

Syntax: `{{IO.Signals.<name>.Quality}}`

Possible values:

- **Bad:** The signal value is not useful.
 - **Good:** The quality of the value is good.
 - **Unknown:** The quality of the signal value is unknown.
-

Type

Gets the type of the I/O signal.

Syntax: `{{IO.Signals.<name>.Type}}`

Possible values:

- **AnalogInput**
 - **AnalogOutput**
 - **DigitalInput**
 - **DigitalOutput**
 - **GroupInput**
 - **GroupOutput**
-

IsSimulated

Gets a flag telling if the I/O signal is simulated.

Syntax: `{{IO.Signals.<name>.IsSimulated}}`

Possible values:

- **False:** The I/O signal is not a simulated signal.
- **True:** The I/O signal is a simulated signal.

4 MQTT Publisher

4.2.6.2.8 I/O system

4.2.6.2.8 I/O system

Introduction

Provides access to the I/O system.

Signals

Gets a dictionary of the I/O signals in the controller.

Syntax: `{{IO.Signals}}`

4.2.6.2.9 Main computer service info

Introduction

Provides access to the service information for the main computer.

BoardType

Gets the board type.

Syntax: `{{MainComputerServiceInfo.BoardType}}`

CpuInfo

Gets the Cpu info.

Syntax: `{{MainComputerServiceInfo.CpuInfo}}`

RamSize

Gets the size of RAM in Mega Bytes.

Syntax: `{{MainComputerServiceInfo.CpuInfo.RamSize}}`

Temperature

Gets the temperature in degrees Celsius.

Syntax: `{{MainComputerServiceInfo.CpuInfo.Temperature}}`

4 MQTT Publisher

4.2.6.2.10 MechanicalUnit

4.2.6.2.10 MechanicalUnit

Introduction

Gets value and other properties of a mechanical unit.

Syntax: `{{Motion.MechanicalUnits.<name>}}`

Where <name> is the name of the mechanical unit.

Name

Gets the name of the Mechanical Unit.

Syntax:

`{{Motion.ActiveMechanicalUnit.Name}}`

`{{Motion.MechanicalUnits.<name>.Name}}`

Model

Gets the robot model of the Mechanical Unit, if available.

Syntax: `{{Motion.MechanicalUnits.<name>.Model}}`

NumberOfAxes

Gets the robot model of the Mechanical Unit.

Syntax: `{{Motion.MechanicalUnits.<name>.NumberOfAxes}}`

SerialNumber

Gets the serial number of the Mechanical Unit, if available.

Syntax: `{{Motion.MechanicalUnits.<name>.SerialNumber}}`

Type

Gets the type of the Mechanical Unit.

Syntax: `{{Motion.MechanicalUnits.<name>.Type}}`

Possible values:

- **MultiAxes:** Type is multiple axes manipulator.
 - **None:** Type not defined.
 - **SingleAxis:** Type is a single axis manipulator.
 - **TcpRobot:** Type is TCP robot.
-

PayLoad

Gets the current payload of the Mechanical Unit as a Rapid loaddata structure.

Syntax: `{{Motion.MechanicalUnits.<name>.PayLoad}}`

Examples of accessing the details of the loaddata structure:

`{{#with Motion.MechanicalUnits.<name>.PayLoad}}`

`{{Mass}}`

`{{Ix}}`

`{{Iy}}`

`{{Iz}}`

Continues on next page


```

{{Cog.X}}
{{Cog.Y}}
{{Cog.Z}}
{{Aom.Q1}}
{{Aom.Q2}}
{{Aom.Q3}}
{{Aom.Q4}}
{{/with}}

```

See the RAPID reference manual for details.

Tool

Gets the current tool name of the Mechanical Unit.

Syntax:

```

{{Motion.MechanicalUnits.<name>.Tool}}
{{Motion.MechanicalUnits.<name>.Tool.Name}}

```

Gets the current tool data of the Mechanical Unit.

Syntax: {{Motion.MechanicalUnits.<name>.Tool.Data}}

Task

Gets the Rapid Task that controls the Mechanical Unit.

Syntax: {{Motion.MechanicalUnits.<name>.Task}}

ServiceInfo

Gets service info for this Mechanical Unit. This property is not valid for a Virtual Controller.

Syntax: {{Motion.MechanicalUnits.<name>.ServiceInfo}}

WorkObject

Gets the current work object name of the Mechanical Unit.

Syntax:

```

{{Motion.MechanicalUnits.<name>.WorkObject}}
{{Motion.MechanicalUnits.<name>.WorkObject.Name}}

```

Gets the current tool data of the Mechanical Unit.

Syntax: {{Motion.MechanicalUnits.<name>.WorkObject.Data}}

DriveModule

Gets the current drive module number of the Mechanical Unit.

Syntax: {{Motion.MechanicalUnits.<name>.DriveModule}}

IsCalibrated

Gets a flag telling if the Mechanical Unit is calibrated or not.

Syntax: {{Motion.MechanicalUnits.<name>.IsCalibrated}}

4 MQTT Publisher

4.2.6.2.11 Mechanical unit service info

4.2.6.2.11 Mechanical unit service info

Introduction

Provides access to the service information for a mechanical unit.

Description

Syntax: `{{Motion.MechanicalUnits.<name>.ServiceInfo.<property>}}`

Where `<name>` is the name of the mechanical unit and `<property>` is one of the properties listed below:

Property	Description	Data Type
<code>ElapsedCalenderTimeSinceLastService</code>	Elapsed calendar time since last service.	TimeSpan
<code>ElapsedProductionTime</code>	Elapsed production time since last SIS reset.	TimeSpan
<code>ElapsedProductionTimeSinceLastService</code>	Elapsed production time since last service.	TimeSpan
<code>LastStart</code>	Time of the last start.	DateTime
<code>ServiceInterval.CalendarTime</code>	Service interval in calendar time.	TimeSpan
<code>ServiceInterval.ProductionTime</code>	Service interval in production time.	TimeSpan
<code>WarningLevel.CalendarService</code>	Warning level for periodic (calendar) service.	Integer (percent)
<code>WarningLevel.GearboxService</code>	Warning level for gear box service.	Integer (percent)
<code>WarningLevel.ProductionService</code>	Warning level for production time service.	Integer (percent)

TimeSpan Data Type

The TimeSpan data type has several properties allowing a TimeSpan to be rendered in different ways. The table below lists the possibilities:

Property	Description	Data Type
<code>Days</code>	The day component of the TimeSpan	Integer
<code>Hours</code>	The hour component of the TimeSpan	Integer
<code>Minutes</code>	The minute component of the TimeSpan	Integer
<code>Seconds</code>	The second component of the TimeSpan	Integer
<code>TotalDays</code>	The TimeSpan expressed in whole and fractional days.	Floating point
<code>TotalHours</code>	The TimeSpan expressed in whole and fractional hours.	Floating point
<code>TotalMinutes</code>	The TimeSpan expressed in whole and fractional minutes.	Floating point
<code>TotalSeconds</code>	The TimeSpan expressed in whole and fractional seconds.	Floating point

Example:

Continues on next page

Given the template below:

```
Production Time Service Interval:
{{#with Motion.MechanicalUnits.ROB_1.ServiceInfo.ServiceInterval}}
Default: {{ProductionTime}}
Days: {{ProductionTime.Days}}
TotalDays: {{ProductionTime.TotalDays}}
TotalHours: {{ProductionTime.TotalHours}}
{{/with}}
```

Assuming Production Time Service Interval is 20000 hours, then the above template will render:

```
Production Time Service Interval:
Default: 833.08:00:00
Days: 833
TotalDays: 833.3333
TotalHours: 20000
```

4 MQTT Publisher

4.2.6.2.12 Motion

4.2.6.2.12 Motion

Introduction

Provides access to the Motion domain.

Speed Ratio

Gets the current speed ratio of the controller.

Syntax: `{{Motion.SpeedRatio}}`

Range: 0 - 100

ActiveMechanicalUnit

Gets the mechanical unit that is currently active.

MechanicalUnits

Gets a dictionary of persistent variables in a module.

Syntax: `{{Motion.MechanicalUnits}}`

4.2.6.2.13 Network settings

Introduction

Provides access to the network settings for the controller.

Address

Gets the IP address.

Syntax: `{{NetworkSettings.Address}}`

Gateway

Gets the gateway settings.

Syntax: `{{NetworkSettings.Gateway}}`

IsDhcp

Flag indicating that Dhcp is used to configure the network of the controller.

Syntax: `{{NetworkSettings.IsDhcp}}`

MacAddress

Gets the Mac address.

Syntax: `{{NetworkSettings.MacAddress}}`

SubnetMask

Gets the subnet mask.

Syntax: `{{NetworkSettings.SubnetMask}}`

4 MQTT Publisher

4.2.6.2.14 Rapid

4.2.6.2.14 Rapid

Introduction

Provides access to the Rapid domain.

Cycle

Gets the current execution cycle setting.

Syntax: `{{Rapid.Cycle}}`

Possible values:

- **AsIs:** Leaves the execution cycle counter as is.
 - **Forever:** Executes forever.
 - **Max:** The maximum number of cycles, except for Forever.
 - **None:** No more execution cycles.
 - **Once:** Executes a single cycle.
 - **Undefined:** Two tasks have different remaining cycles.
-

ExecutionStatus

Gets the current execution status of the controller.

Syntax: `{{Rapid.ExecutionStatus}}`

Possible values:

- **Running:** At least one normal RAPID task is executing or performing regain.
 - **Stopped:** No normal RAPID task is executing or performing regain.
 - **Unknown:** Status is unknown.
-

RemainingCycles

Gets the remaining cycle counter.

Syntax: `{{Rapid.RemainingCycles}}`

Tasks

Gets a dictionary of the RAPID tasks in the controller.

Syntax: `{{Rapid.Tasks}}`

4.2.6.2.15 Rapid module

Introduction

Individual Rapid modules can be accessed by name as shown below.

Syntax: `{{Rapid.Tasks.<taskname>.Modules.<modulename>}}`

Name

Gets name of Rapid module.

Syntax: `{{Rapid.Tasks.<taskname>.Modules.<modulename>.Name}}`

Vars

Gets a dictionary of persistent variables in a module.

Syntax: `{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars}}`

4 MQTT Publisher

4.2.6.2.16 Rapid task

4.2.6.2.16 Rapid task

Introduction

Individual tasks can be accessed by name as shown below.

Syntax: `{{Rapid.Tasks.<taskname>}}`

Where `<taskname>` is the name of the RAPID task.

Name

Gets the name of the task.

Syntax: `{{Rapid.Tasks.<taskname>.Name}}`

Cycle

Gets the current execution cycle setting.

Syntax: `Rapid.Tasks.<taskname>.Cycle`

Possible values:

- **AsIs:** Leave execution cycle counter as is.
 - **Forever:** Execute forever.
 - **Max:** The maximum number of cycles, except for Forever.
 - **None:** No more execution cycles.
 - **Once:** Execute a single cycle.
-

Enabled

Gets the Enabled state of the Task. 'Enabled' is also referred to as activated/deactivated in the user interface. Returns true if the task is activated in the task selection panel and will react to start requests. Returns false if the task is deactivated and will not react to start requests.

Syntax: `{{Rapid.Tasks.<taskname>.Enabled}}`

ExecutionStatus

Gets the execution status of a task.

Syntax: `{{Rapid.Tasks.<taskname>.ExecutionStatus}}`

Possible values:

- **Ready:** The task has no PCP or execution context.
 - **Running:** Task is executing or performing regain.
 - **Stopped:** Task is not executing or not performing regain. PCP and execution context is defined in task.
 - **UnInitiated:** The program server is not initialized. State only assumed during startup.
 - **Unknown:** Status is unknown.
-

ExecutionType

Gets the current execution type.

Syntax: `{{Rapid.Tasks.<taskname>.ExecutionType}}`

Continues on next page

Possible values:

- **EventRoutine:** An event routine. For example, RESET, START, STOP, POWER_ON and so on. EventRoutine can be executed at any execution level.
- **ExternalInterrupt:** An external interrupt, via the system input Interrupt. Can be executed at any execution level.
- **Interrupt:** A programmatic interrupt
- **None:** No execution context, i.e. no ProgramPointer is set
- **Normal:** Normal program execution
- **UserRoutine:** A service routine.

IsMotionTask

Returns true if task is a motion task, false otherwise.

Syntax: `{{Rapid.Tasks.<taskname>.IsMotionTask}}`

TaskType

Gets the task type, i.e. Normal, Static or Semi-static.

Syntax: `{{Rapid.Tasks.<taskname>.TaskType}}`

Possible values:

- **Normal:** Normal task type. Possible to enable/disable in task selection panel.
- **SemiStatic:** Semi-static task type.
- **Static:** Static task type.

Modules

Gets a dictionary of modules in the task.

Syntax: `{{Rapid.Tasks.<taskname>.Modules}}`

4 MQTT Publisher

4.2.6.2.17 Rapid variable

4.2.6.2.17 Rapid variable

Introduction

Gets value and other properties of persistent Rapid variables in a Module.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>}}
```

Where <taskname> is the name of the Rapid task, <modulename> is the name of the Rapid module, and <name> is the name of the Rapid.

Name

Gets the name of the Rapid variable.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>.Name}}
```

Dim

Gets the dimension of the Rapid variable.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>.Dim}}
```

Possible values:

- -1: Not an array
- 1: One-dimensional array
- 2: Two-dimensional array
- 3: Three-dimensional array

IsArray

Returns True if the Rapid variable is an array, False otherwise.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>.IsArray}}
```

RapidType

Gets the type of the Rapid variable.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>.RapidType}}
```

Value

Gets the value of the Rapid variable.

Syntax:

```
{{Rapid.Tasks.<taskname>.Modules.<modulename>.Vars.<name>.Value}}
```

Note that for complex variable types, it is possible to address individual fields, e.g. to access the x translation value of a robot target named HomePos, write:

```
{{Rapid.Tasks.T_ROB1.Modules.MyModule.Vars.HomePos.Value.trans.x}}
```

See Ch. 1.5.2 Payload Template Examples for more examples.

4.2.6.2.18 Robotics handbars helpers

Introduction

This chapter describes ABB Robotics specific Handbars helpers.

ElogMessageQClear

Removes all event log messages from the EventLogMessageQ.

This helper is only used with an EventLog Message Trigger configured for queued mode.

Depending on how event log messages should be rendered, either ElogMessageQClear or ElogMessageQPop must be used to take processed event log messages away from the ElogMessageQ.

Syntax: `{{ElogMessageQClear}}`

Example:

In this example, all queued event log messages are processed, and then the queue is cleared in the end.

```

{{#each ElogMessageQ}}
  {{UtcTimestamp}} {{Code}} {{Type}} {{Title}}
{{/each}} {{ElogMessageQClear}}

```

ElogMessageQPop

Removes the first event log message from the EventLogMessageQ.

This helper is only used with an EventLog Message Trigger configured for queued mode.

Depending on how event log messages should be rendered, either ElogMessageQClear or ElogMessageQPop must be used to take processed event log messages away from the ElogMessageQ.

Syntax: `{{ElogMessageQPop}}`

Example:

In this example, a maximum of two event log messages are processed at the time. If the ElogMessageQ contains more messages, the process is repeated so that additional MQTT messages can be published until the queue is empty.

```

Elog Queue Length: {{ElogMessageQ.Count}}
{{#with ElogMessage}}
[1] {{UtcTimestamp}} {{Code}} {{Type}} {{Title}}
{{/with}} {{ElogMessageQPop}}
{{#if ElogMessageQ.Count}}
{{#with ElogMessage}}
[2] {{UtcTimestamp}} {{Code}} {{Type}} {{Title}}
{{/with}} {{ElogMessageQPop}}
{{/if}}

```

JointTarget

Gets the current position as a jointtarget. Default task name is "T_ROB1"

Syntax: `{{CJointT}}` or `{{CJointT Rapid.Tasks.<taskname>}}` or `{{CJointT <taskname>}}`

Continues on next page

4 MQTT Publisher

4.2.6.2.18 Robotics handlbars helpers

Continued

Map

Maps from an enumerated value to custom text.

Syntax:

```
{{Map <enumerated-value> "<key>:<value>" + [ "(default):<value>" ]
}}
```

Where:

<code><enumerated-value></code>	Any Robot model variable that yields an enumerated value, for example, <code>OperatingMode</code>
<code>"<key>:<value>" +</code>	One or more key - value pairs. Each key-value pair must be enclosed in quotes, and the key and the value separated by a colon.
<code>["(default):<value>"]</code>	An optional default value to use if the enumerated value does not match any of the keys.

Example:

```
{{Map OperatingMode "Auto:AUTO" "ManualReducedSpeed:MAN1"
"ManualFullSpeed:MAN2" "(default):UNKNOWN"}}
```

RobTarget

Gets the current position as a robtarget. Default task name is "T_ROB1"

Syntax: `{{CRobT}}` or `{{CRobT Rapid.Tasks.<taskname>}}` or `{{CRobT <taskname>}}`

Tool Name

Gets the current tool name. Default mechanical unit name is "ROB_1"

Syntax:

- `{{CToolName}}`
 - `{{CToolName Motion.MechanicalUnits.<unitname>}}`
 - `{{CToolName <unitname>}}`
-

Tool

Gets the current tool data. Default mechanical unit name is "ROB_1"

Syntax: `{{CTool}}` or `{{CTool Motion.MechanicalUnits.<unitname>}}`
or `{{CTool <unitname>}}`

WorkObject Name

Gets the current work object name. Default mechanical unit name is "ROB_1"

Syntax:

- `{{CWorkObjName}}`
 - `{{CWorkObjName Motion.MechanicalUnits.<unitname>}}`
 - `{{CWorkObjName <unitname>}}`
-

WorkObject

Gets the current work object data. Default mechanical unit name is "ROB_1"

Continues on next page

Syntax: There are multiple ways to get the current work object.

- `{{CWOBJ}}`
- `{{CWOBJ Motion.MechanicalUnits.<unitname>}}`
- `{{CWOBJ <unitname>}}`

4 MQTT Publisher

4.2.6.2.19 User data

4.2.6.2.19 User data

Overview

Gets the value of a key-value pair item from the UserData table in the Robot Configuration file. For details on entering data in the UserData table, see [Robot configuration on page 77](#).

Syntax: `{{UserData.<key>}}`

4.2.6.3 Payload template examples

Introduction

This chapter contains many examples of payload templates illustrating both simple and more advanced ways to use Handlebars to render data from the robot. Most of the examples will render as plain text, but there are some that shows how to write a template that renders as JSON and XML.

Controller Properties

```
SystemID: {{SystemId}}
SystemName: {{SystemName}}
ControllerID: {{ControllerID}}
ControllerName: {{ControllerName}}
RobotWareVersion: {{RobotWareVersion}}
ControllerAddress: {{Address}}
```

Controller States

```
Operating Mode: {{OperatingMode}}
Controller State: {{ControllerState}}
Controller Execution State: {{ControllerExecutionState}}
Speed Ratio: {{SpeedRatio}}
System Clock: {{SystemClock}}
Rapid Mastership: {{Rapid.MasterRapid}}
Cfg Mastership: {{Cfg.MasterCFG}}
Interface State: {{InterfaceState}}
```

IOSignals (XML)

```
<Signals>
  {{#each IO.Signals}}
  <Signal>
    <name>{{@key}}</name>
    <value>{{@value.Value}}</value>
    <type>{{@value.Type}}</type>
  </Signal>
  {{/each}}
</Signals>
```

Rapid Scalar Variable

```
{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varNum}}
Array: {{isArray}}
Dim: {{Dim}}
RapidType: {{RapidType}}
Value: {{Value}}
{{/with}}
```

Rapid Pre-Defined Record Variable

```
{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varRobTarget}}
Array: {{isArray}}
Dim: {{Dim}}
RapidType: {{RapidType}}
```

Continues on next page

4 MQTT Publisher

4.2.6.3 Payload template examples

Continued

```
Value: {{Value}}
Trans
{{#with Value.Trans}}
X: {{X}}
Y: {{Y}}
Z: {{Z}}
{{/with}}
Rot
{{#with Value.Rot}}
Q1: {{Q1}}
Q2: {{Q2}}
Q3: {{Q3}}
Q4: {{Q4}}
{{/with}}
{{/with}}
```

Rapid User-Defined Record Variable

```
{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varRPoint}}
Array: {{isArray}}
Dim: {{Dim}}
{{! RECORD Num X; Num Y; ENDRECORD}}
RapidType: {{RapidType}}
Value: {{Value}}

X: {{Value.[0]}}{{!Properties can be accessed by index in
user-defined types}}
Y: {{Value.[1]}}
{{/with}}
```

Rapid One-Dimensional Array Variable

```
{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varSingleArr}}
Array: {{isArray}}
Dim: {{Dim}}
RapidType: {{RapidType}}
Value: {{Value}}

{{! Accessing one element in one dimensional array:}}
Value[1]: {{Value.[1]}}

{{! Accessing all elements in sequence in one dimensional array:}}
{{#each Value}}
Value[{{@index}}]: {{this}}
{{/each}}
{{/with}}
```

Rapid Two-Dimensional Array Variable (Markdown)

```
{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varDoubleArr}}
Array: {{isArray}}
Dim: {{Dim}}
RapidType: {{RapidType}}
```

Continues on next page


```

Value: {{Value}}
Value[2][1]: {{Value.[2].[1]}} {{! Accessing one element in two
dimensional array }}
{{#each Value}} {{! Accessing all elements in sequence in two
dimensional array }}
{{#each this}}
Value[{{@../index}}][{{@index}}]: {{this}}
{{/each}}
{{/each}}
{{/with}}

```

Rapid Three-Dimensional Array Variable

```

{{#with Rapid.Tasks.T_ROB1.Modules.MainModule.Vars.varTrippleArr}}
Array: {{isArray}}
Dim: {{Dim}}
RapidType: {{RapidType}}
Value: {{Value}}

{{!Accessing one element in three dimensional array}}
Value[0][2][1]:{{Value.[0].[2].[1]}}

{{! Accessing all elements in sequence in three dimensional array
}}
{{#each Value}}
{{#each this}}
{{#each this}}
Value[{{@../../index}}][{{@../index}}][{{@index}}]: {{this}}
{{/each}}
{{/each}}
{{/each}}
{{/with}}

```

Tool Data

```

{{#with (CTool Motion.MechanicalUnits.ROB_1)}}
Tframe
X: {{Tframe.Trans.X}}
Y: {{Tframe.Trans.Y}}
Z: {{Tframe.Trans.Z}}
Q1: {{Tframe.Rot.Q1}}
Q2: {{Tframe.Rot.Q2}}
Q3: {{Tframe.Rot.Q3}}
Q4: {{Tframe.Rot.Q4}}
Tload
Cog
X: {{Tload.Cog.X}}
Y: {{Tload.Cog.Y}}
Z: {{Tload.Cog.Z}}
Aom
Q1: {{Tload.Aom.Q1}}
Q2: {{Tload.Aom.Q2}}
Q3: {{Tload.Aom.Q3}}

```

Continues on next page

4 MQTT Publisher

4.2.6.3 Payload template examples

Continued

```
Q4: {{Tload.Aom.Q4}}
Mass: {{Tload.Mass}}
Ix: {{Tload.Ix}}
Iy: {{Tload.Iy}}
Iz: {{Tload.Iz}}
{{/with}}
```

WorkObject Data

```
{{#with (CWObj)}}
Uframe
X: {{Uframe.Trans.X}}
Y: {{Uframe.Trans.Y}}
Z: {{Uframe.Trans.Z}}
Q1: {{Uframe.Rot.Q1}}
Q2: {{Uframe.Rot.Q2}}
Q3: {{Uframe.Rot.Q3}}
Q4: {{Uframe.Rot.Q4}}
Oframe
{{#with Oframe.Trans}}
X: {{X}}
Y: {{Y}}
Z: {{Z}}
{{/with}}
{{#with Oframe.Rot}}
Q1: {{Q1}}
Q2: {{Q2}}
Q3: {{Q3}}
Q4: {{Q4}}
{{/with}}
{{/with}}
```

RobTarget (JSON)

```
{{#with (CRobT `T_ROB1')}}
{
  "RobTarget": {
    "Trans": {
      "X": {{Trans.X}},
      "Y": {{Trans.Y}},
      "Z": {{Trans.Z}}
    },
    "Rot": {
      "Q1": {{Rot.Q1}},
      "Q2": {{Rot.Q2}},
      "Q3": {{Rot.Q3}},
      "Q4": {{Rot.Q4}}
    }
  }
}
{{/with}}
```

Continues on next page

JointTarget

```
{{#with (CJointT Rapid.Tasks.T_ROB1)}}  
RobAx  
Rax_1: {{RobAx.Rax_1}}  
Rax_2: {{RobAx.Rax_2}}  
Rax_3: {{RobAx.Rax_3}}  
Rax_4: {{RobAx.Rax_4}}  
Rax_5: {{RobAx.Rax_5}}  
Rax_6: {{RobAx.Rax_6}}  
  
ExtAx  
Eax_a: {{ExtAx.Eax_a}}  
Eax_b: {{ExtAx.Eax_b}}  
Eax_c: {{ExtAx.Eax_c}}  
Eax_d: {{ExtAx.Eax_d}}  
Eax_e: {{ExtAx.Eax_e}}  
Eax_f: {{ExtAx.Eax_f}}  
{{/with}}
```

EventLog Properties

```
Title: {{ElogMessage.Title}}  
Description: {{ElogMessage.Description}}  
Code: {{ElogMessage.Code}}  
Hint: {{ElogMessage.Hint}}  
Consequences: {{ElogMessage.Consequences}}  
Causes: {{ElogMessage.Causes}}  
Body: {{ElogMessage.Body}}  
CategoryId: {{ElogMessage.CategoryId}}  
Number: {{ElogMessage.Number}}  
SequenceNumber: {{ElogMessage.SequenceNumber}}  
Timestamp: {{ElogMessage.Timestamp}}  
Type : {{ElogMessage.Type}}
```

4 MQTT Publisher

4.3.1 Introduction

4.3 Handlebars

4.3.1 Introduction

Overview

The template language supported by the IoT Gateway's Configurable MQTT Publisher is called Handlebars. Details about the language can be found at <https://handlebarsjs.com/>

4.3.2 Language features

Simple expressions

The basic concept of a Handlebars template is to include references to data in double curly brackets like the simple template below illustrates.

```
Operating mode: {{OperatingMode}}
```

Depending on the current operating mode of the robot controller, the above template may result in the following output when rendered:

```
Operating mode: ManualReducedSpeed
```

Nested input objects

A dot-notation is used to address parts of more complex objects in the data model like arrays, dictionaries, and records. As an example, the ControllerInfo object in the robot controller data model contains several sub-objects or values, and the system Id of the running system is one of them. The template below illustrates how to get the system Id.

```
System ID: {{ControllerInfo.SystemId}}
```

This will render as something like this:

```
System ID: 3ac20b09-6766-4583-a528-9694e4d8a6b4
```

The dot-notation can be used to access data at an arbitrary level.

Evaluation context

Using the dot-notation may be very verbose if a lot of properties needs to be accessed in a sub-object of the data model. There are some so called block-helpers that makes it possible to change the current context from the top-level of the data model to a sub-object at any level. The most common and built-in block helpers that does this are each and with.

The with-helper changes the evaluation context from the top-level to the properties of the given object. Using the ControllerInfo object as an example:

```
{{#with ControllerInfo}}  
System name: {{SystemName}}  
System ID : {{SystemId}}  
{{/with}}
```

Both SystemName and SystemId are properties of ControllerInfo.

Template comments

Template comments can be used for documentation purposes or to just disable parts of a template for test purposes.

Syntax:

```
{{!-- <comment> --}}
```

Note that the comment may contain multiple lines and what would normally be interpreted as placeholders, i.e., '{{' and '}}'. Everything will just be ignored and left out of the resulting output.

Continues on next page

4 MQTT Publisher

4.3.2 Language features

Continued

Escaping HTML special characters

Handlebars was originally designed to generate html, hence Handlebars by default replaces characters that have special meaning in html with the corresponding html-escape character, for example, the '<' character in the rendered output will be replaced by '<'. Use triple curly brackets instead of double curly brackets to disable this feature:

Data that can contain special html characters: {{{UserData.CustomData}}}

Built-in helpers

Handlebars helpers are special functions that extends the possibilities beyond just referencing data in templates. The most common helpers are described below.

For a full list of build in helpers, see <https://handlebarsjs.com/guide/>

#if

The 'if' helper conditionally renders a block.

Syntax:

```
{{#if <condition>}}  
<block>  
{{/if}}
```

The <block> is only rendered if <condition> does not evaluate to a false value.

False values include 'false', '0', null, "" (empty string), and empty container.

An optional 'else' section may be used to render a block if <condition> evaluates to a false value.

Syntax:

```
{{#if <condition>}}  
<block rendered if <condition> evaluates to a true value  
{{else}}  
<block rendered if <condition> evaluates to a false value  
{{/if}}
```

#unless

The inverse of #if.

Syntax:

```
{{#unless <condition>}}  
<block>  
{{/unless}}
```

The <block> is rendered only if <condition> evaluates to a false value.

#each

Iterate over all elements of a container, e.g., a list or dictionary.

Syntax:

```
{{#each <container>}}  
<block rendered for each element in container >  
{{/each}}
```

The <block> is rendered once for each element in the <container>.

The keyword 'this' can be used inside the <block> to reference the current element of the <container>.

Continues on next page

An optional 'else' section may be used to render a block if the `<container>` is empty.

Syntax:

```

{{#each <container>}}
<block rendered for each element in container>
{{else}}
<block rendered if container is empty>
{{/each}}

```

Inside the loop, the following special variables may be used:

`{{@index}}` – The current loop index

`{{@key}}` – The current key name, e.g., when iteration through a dictionary.

`@first` – Identifies the first step of the iteration. Typical use: `{{#if @first}}`
`<block>` `{{/if}}`

`@last` – Identifies the last step of the iteration.

#with

Change evaluation context when rendering a block.

Syntax:

```

{{#with <variable>}}
<block>
{{/with}}

```

This changes the evaluation context to `<variable>` inside the `<block>`.

An optional 'else' section may be used to render a block if `<variable>` evaluates to an empty value.

Syntax:

```

{{#with <variable>}}
<block rendered if <variable> evaluates to a non-empty value
{{else}}
<block rendered if <variable> evaluates to an empty value
{{/with}}

```

Where `<variable>` is any variable in the Robot model that contains sub-elements.

Data variables

Data variables starts with an `@` and are dynamically updated by Handlebars during rendering. They may be used the same way as any normal variable from the Robot model as shown in the following example:

```

{{#each Cfg}}
{{@key}}
{{/each}}

```

Will output a list of all the Cfg domains.

@root

Evaluates to the initial context of the template regardless of how many nested `#with` statements there might be, i.e., it always evaluates to the top of the Robot model.

Continues on next page

4 MQTT Publisher

4.3.2 Language features

Continued

@first

Set to 'true' by the #each helper in the first iteration.

@index

Contains a zero-based index for the current iteration of the #each helper.

@key

Contains the key name for the current iteration of the #each helper. This is useful when, for example, iterating over a dictionary.

@last

Set to 'true' by the #each helper in the last iteration.

5 Troubleshooting

Overview

The following table provides some helpful information for troubleshooting:

Behavior	Notes
An OPC UA client cannot connect to the OPC UA server although the server is running as windows service. The error message Host unreachable (or similar) appears.	Check whether firewall settings prevent communication with the OPC UA server. The server port must be open for incoming TCP communication so that a client can connect. The server port is configurable, and the current setting can be found in the Server Control tab of the IoT Gateway Configuration Tool.
An OPC UA client sees the server's endpoints, but a connection with them fails with the error message Host unreachable .	<ol style="list-style-type: none"> 1 Check that the name resolution in your network is working properly and that the server is accessible under its host name. Even if the OPC UA Client apparently connects to the IP address of the server (e.g. <code>opc.tcp://192.168.0.1:65150</code>) to access the OPC UA server's endpoints, the server always returns its own host name in its endpoints. If the client connects directly to one of the endpoints, it will use the host name of the server again. If the name resolution does not work, the connection fails. 2 Try using hostname in endpoint URL Example: <code>opc.tcp://MyOPCUAServer-Host:61510/ABB.IRC5.OPCUA.Server</code>
An OPC UA client sees the endpoints of the server, but a connection to a secure endpoint fails. The error message BadSecurityChecksFailed appears.	Check whether the server trusts the client certificate. The required configuration steps can be found in section Trust the rejected client certificate(s).
An OPC UA client sees the endpoints of the server, but a connection to a secure endpoint fails. The error message BadSecurityPolicyRejected appears.	OPC UA Server rejects the client connection because of unsupported security policy provided by client. OPC UA Client should select the security policies that is supported by OPC UA Server. For more information, see Security configuration on page 55 or How to connect to OPC UA Server on page 59 .
OPC UA Server fails to start due to <i>"Thumbprint was explicitly specified in the configuration. Cannot generate a new certificate"</i> .	Check for certificate store folder. Check for own (server application) certificate in <code>%ProgramData%\ABB\IoT Gateway\CertificateStores\ApplicationCertificate</code> folder. Make sure the certificate is valid. If the problem persists, try to create a new application certificate using the IoT Gateway configuration tool. For more details, see the section Server Application instance certificates on page 44 .
OPC UA Server fails to start due to <i>"Failed to establish TCP listener sockets for Ipv4 and IPv6. Check TCP port is already in use."</i>	Make sure no process in the system is using the same TCP port as IoT Gateway (OPC UA Server). Change the port number using the IoT Gateway configuration tool. For more details, see the section Server control on page 35 .

Continues on next page

5 Troubleshooting

Continued

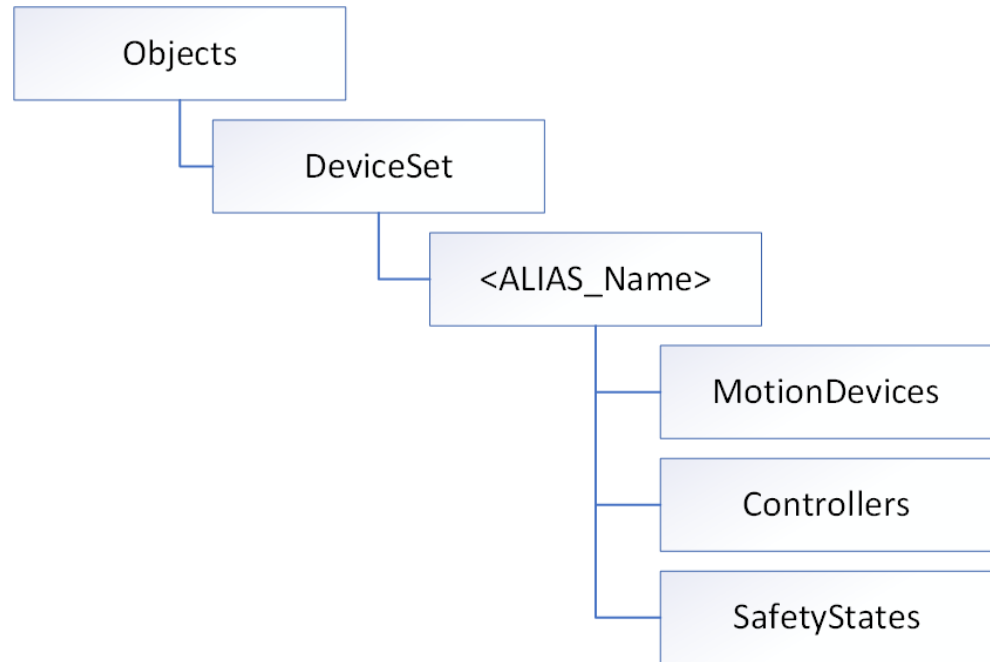
Behavior	Notes
IoT Gateway performance goes down or appears to freeze.	<p>If there are too many subscription updates due to many and frequent subscriptions to IO Signals & Rapid variables the IoT Gateway might seem to freeze. Exactly how many subscriptions updates it takes to cause this depends on PC hardware in use. Lab test has shown that more than 500 subscription items which change every second can cause this behavior. When this happens large increase in thread count for the IoT Gateway service can be observed. Normal thread count lies between 25 to 150.</p> <p>Find <code>ABB.Robotics.IoTGateway.exe</code> in Resource Monitor and look for the Threads column.</p> <p>Possible actions to resolve this issue</p> <ul style="list-style-type: none">• Reduce the number of subscriptions.• Reduce the frequency of changes to subscribed items.• Run IoT Gateway on a PC with higher performance.
While adding alias in IoT Gateway config tool the information message unknown status of required RobotWare options appears.	<p>Check the firewall settings or Network configuration. If any security control application like Iboss (used for security controls including allow/block list, all policies, configuration and controls) running in background results in unknown status of required RobotWare option information message.</p>

6 Appendix

6.1 Appendix A - Robotics companion specification

Introduction

The IRC5 OPC UA Server supports all mandatory and some of the optional parts of OPC 40010-1 OPC UA for Robotics, Part 1: Vertical Integration. This chapter describes the supported features and how they map to RobotWare.



xx2000002342

Supported features

DeviceSet

The DeviceSet is a container for all instances of ComponentType defined in OPC Unified Architecture for Devices (DI). One of the subtypes of the ComponentType is the MotionDeviceSystemType as described below.

Feature	Browse Name	Description
MotionDeviceSystem	<name>	Each instance corresponds to an ABB robot and <name> equals the Alias name given to the robot in the OPC UA server configuration.

MotionDeviceSystem

Feature	Browse Name	Description
MotionDevices	MotionDevices	A container for instances of MotionDeviceType
Controllers	Controllers	A container for instances of ControllerType
SafetyStates	SafetyStates	A container for instances of SafetyStateType

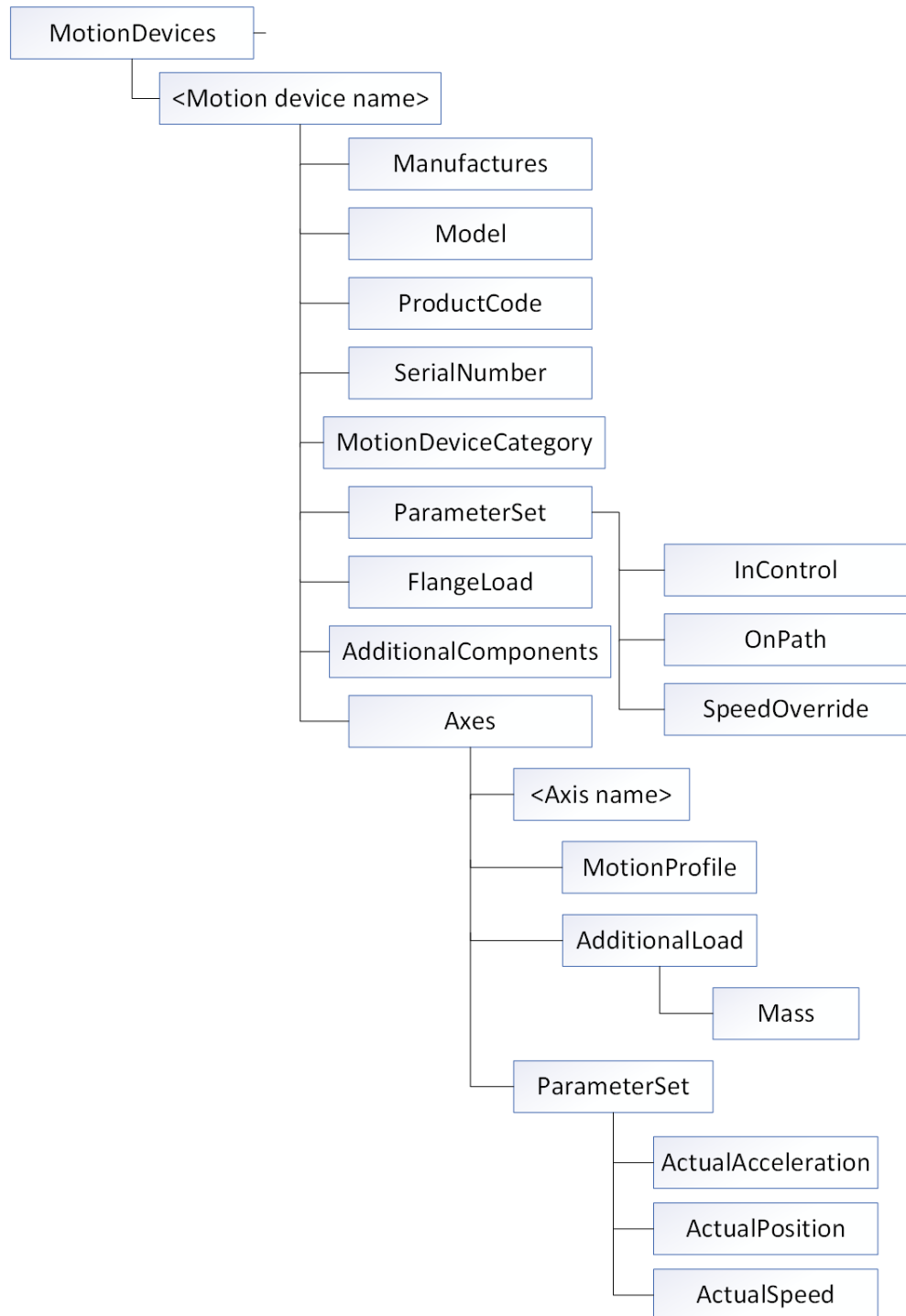
Continues on next page

6 Appendix

6.1 Appendix A - Robotics companion specification

Continued

MotionDevices



xx2000002343

Feature	Browse Name	Description
MotionDevice	<name>	Each MotionDevice instance corresponds to a Mechanical Unit in the ABB robot. <name> is equal to the name of the Mechanical Unit.

Continues on next page

MotionDevice

Feature	Browse Name	Description
MotionDeviceCategory	MotionDeviceCategory	A categorization of the type of motion device based on ISO 8373, e.g. ARTICULATED_ROBOT
Manufacturer	Manufacturer	Name of manufacturer, i.e. "ABB"
Model	Model	Maps to the Model property of the Mechanical Unit, e.g. IRB5500_HWT
ProductCode	ProductCode	The article number for the Mechanical Unit, if available. Empty string otherwise.
SerialNumber	SerialNumber	The serial number if the Mechanical Unit, if available. Empty string otherwise.
FlangeLoad	FlangeLoad/Mass	The current Payload Mass of the Mechanical Unit.
ParameterSet	ParameterSet/OnPath	Not supported – always (null)
	ParameterSet/InControl	"true" if Motors ON, "false" otherwise
	ParameterSet/SpeedOverride	The Speed Ratio of the system 0 – 100%
Axes	Axes	A container for instances of AxisType
PowerTrains	PowerTrains	A container for instances of PowerTrainType
AdditionalComponents	AdditionalComponents	Empty folder, not in use.

Axes

Feature	Browse Name	Description
Axis	<name>	Each instance corresponds to an axis of the Mechanical Unit. <name> is equal to the name of the axis, e.g. Rax_1 or Eax_6

Axis

Feature	Browse Name	Description
MotionProfile	MotionProfile	Property describing the type of motion for this axis, e.g. "ROTARY".
AdditionalLoad	AdditionalLoad/Mass	Not supported – always 0.0
ParameterSet	ParameterSet/ActualPosition	Current position of axis
	ParameterSet/ActualSpeed	Not supported – always (null)
	ParameterSet/ActualAcceleration	Not supported – always (null)

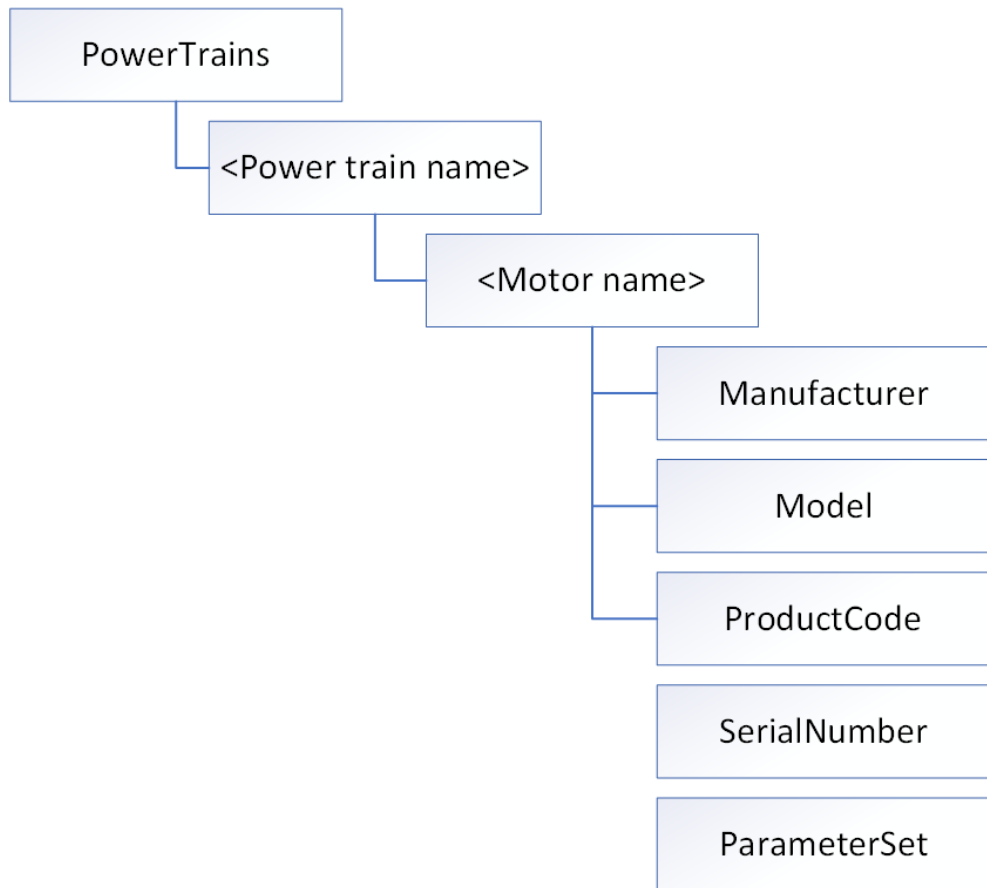
Continues on next page

6 Appendix

6.1 Appendix A - Robotics companion specification

Continued

PowerTrains



xx2000002344

Feature	Browse Name	Description
PowerTrain	<name>	Each instance corresponds to a PowerTrain of the Mechanical Unit. <name> is equal to the joint name of the robot or the external axis of the mechanical unit that the power train drives. E.g. rob_1_1

PowerTrain

Feature	Browse Name	Description
Motor	<name>	Each instance of the MotorType corresponds to a Motor of the Axis. Normally there is one Motor per Axis. <name> is equal to the name of the Power-Train instance it belongs to.

Motor

Feature	Browse Name	Description
Manufacturer	Manufacturer	Name of manufacturer, i.e. "ABB"
Model	Model	Not supported – always (null)

Continues on next page

Feature	Browse Name	Description
ProductCode	ProductCode	Article number of Motor, retrieved from the "Use Motor Type" field of the Motor configuration.
SerialNumber		Not supported – always (null)
ParameterSet	ParameterSet/BrakeReleased	Not supported – always (null)
	ParameterSet/MotorTemperature	Not supported – always (null) This is a mandatory variable, but as ABB robots have only PTCs and not analog temperature sensors in the motors, there is no temperature to read.
	ParameterSet/EffectiveLoadRate	Not supported – always (null)

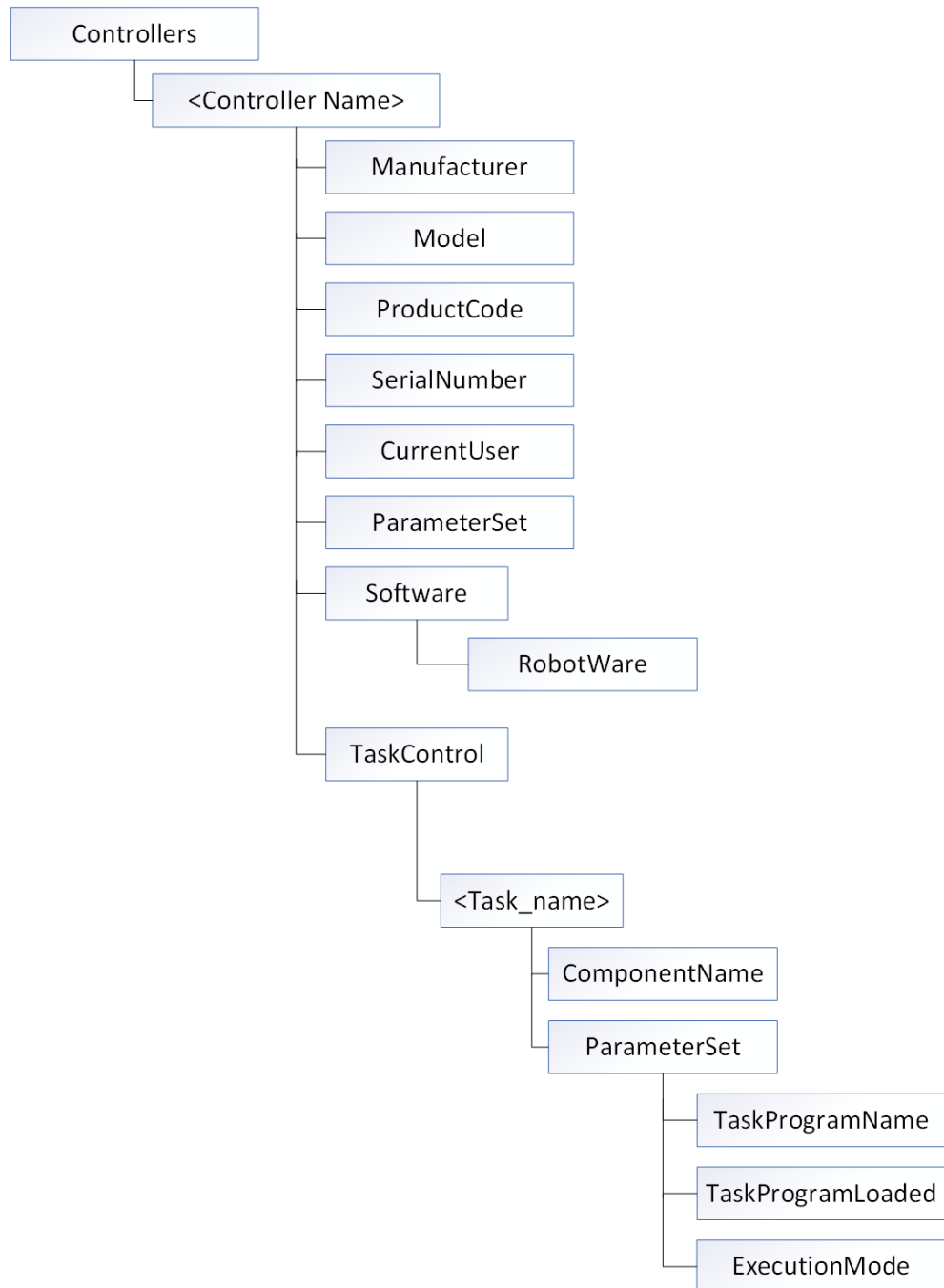
Continues on next page

6 Appendix

6.1 Appendix A - Robotics companion specification

Continued

Controllers



xx2000002345

Feature	Browse Name	Description
Controller	<name>	There is always only one instance of ControllerType for ABB systems. <name> is equal to the Controller Name from the Controller Properties

Continues on next page

Controller

Feature	Browse Name	Description
Manufacturer	Manufacturer	Name of manufacturer, i.e. "ABB"
Model	Model	Name of controller model, e.g. "IRC5"
ProductCode	ProductCode	Article number of controller. Not available digitally, so value is set to empty string.
SerialNumber	SerialNumber	Serial number of controller. Not available digitally, so value is set to empty string.
CurrentUser	CurrentUser/Level	String containing list of grants assigned to the current user.
	CurrentUser/Name	Name of the current user. For example, "Default User"
ParameterSet	ParameterSet/TotalPowerOnTime	Provides the elapsed production time since the last SIS reset represented as an OPC UA DurationString. For details, see https://reference.opcfoundation.org/v104/ISA-95/v100/docs/6.2.6/ A Virtual Controller will always show PT0H (that is, a zero duration)
	ParameterSet/StartUpTime	Not supported – always (null)
	ParameterSet/UpsState	Not supported – always (null)
	ParameterSet/TotalEnergyConsumption	Not supported – always (null)
	ParameterSet/CabinetFanSpeed	Not supported – always (null)
	ParameterSet/CPUFanSpeed	Not supported – always (null)
	ParameterSet/InputVoltage	Not supported – always (null)
	ParameterSet/Temperature	Not supported – always (null)
Components	Components	Empty folder, not in use.
Software	Software	A container for instances of SoftwareType
TaskControls	TaskControls	A container for instances of TaskControlType

Software

Feature	Browse Name	Description
Software	<name>	A list of software on the robot controller. For ABB controllers this list contains only one instance named RobotWare

Software: RobotWare

Feature	Browse Name	Description
Manufacturer	Manufacturer	Name of manufacturer, i.e. "ABB"
Model	Model	Name of software, typically "RobotWare"

Continues on next page

6 Appendix

6.1 Appendix A - Robotics companion specification

Continued

Feature	Browse Name	Description
SoftwareRevision	SoftwareRevision	Version number of software, e.g. "6.11.0.1"

TaskControls

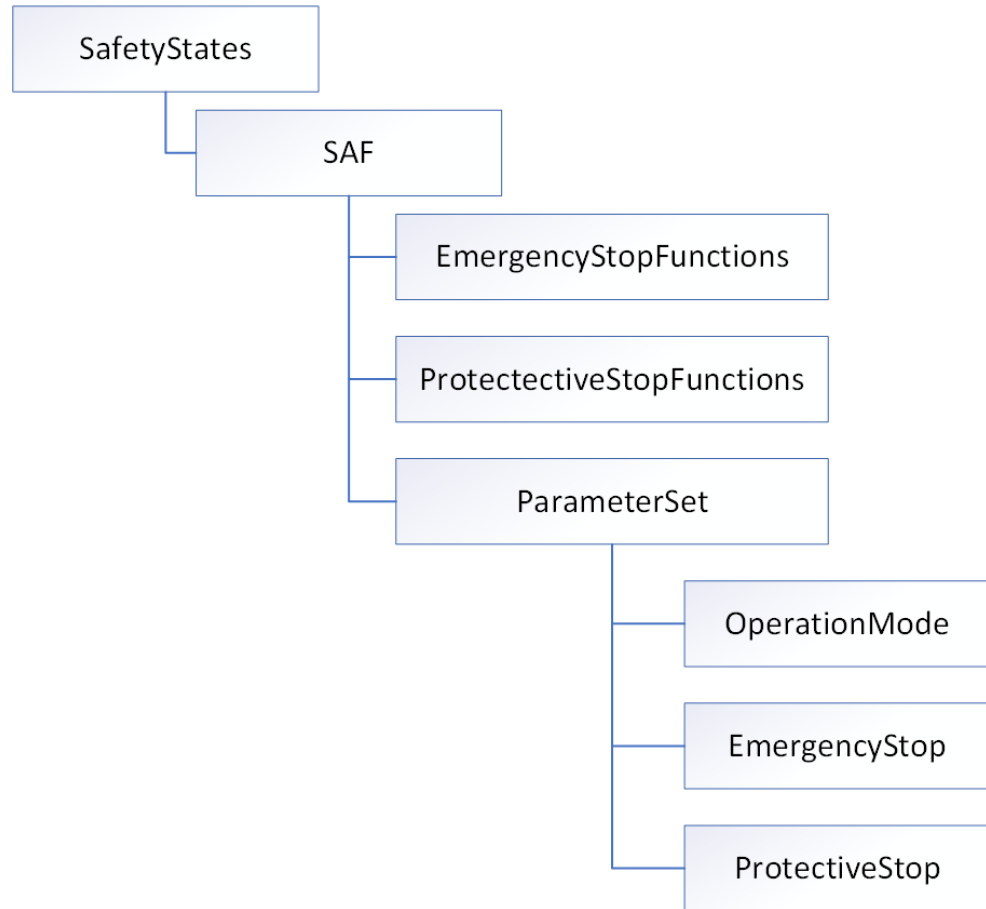
Feature	Browse Name	Description
TaskControl	<name>	Each TaskControlType instance listed corresponds to a RAPID task. <name> is equal to the name of the corresponding RAPID task.

TaskControl

Feature	Browse Name	Description
ComponentName	ComponentName	Contains the name of the corresponding RAPID task, i.e. the same as the Browse Name for the TaskControl itself.
ParameterSet	Parameter-Set/TaskProgram-Name	
	Parameter-Set/TaskProgram-Loaded	True if the RAPID task has a defined execution context, i.e. that some RAPID code is loaded and the program pointer is defined.
	ParameterSet/ExecutionMode	Not supported – always (null)

Continues on next page

SafetyStates



xx2000002346

Feature	Browse Name	Description
SafetyState	<name>	For ABB controllers this list contains only one instance named "SAF"

SafetyState: SAF

Feature	Browse Name	Description
ParameterSet	ParameterSet/OperationalMode	Corresponds to the Operating Mode of the robot controller.
	ParameterSet/EmergencyStop	True if emergency stop is activated, false otherwise.
	ParameterSet/ProtectiveStop	True if one or more protective stops are activated, false otherwise.
EmergencyStopFunctions	EmergencyStopFunctions	Empty folder, not in use.
ProtectiveStopFunctions	ProtectiveStopFunctions	Empty folder, not in use.

6 Appendix

6.2 Appendix B - Performance tests summary

6.2 Appendix B - Performance tests summary

Overview

IRC5 OPC UA Server running on PC with following specifications:

- OS: Windows 10 Pro
- System Type: 64-Bit OS, x64-based processor
- Processor: Intel i5-6500T CPU @ 2.5GHz
- RAM: 8 GB

Sl. No.	Test case description	Test Result
1	IRC5 OPC UA Server with one robot controller having 11 subscription changes per second (7 Rapid variables + 3 output signals + 1 event log)	Pass
2	IRC5 OPC UA Server with one robot controller having 31 subscription changes per second (27 Rapid variables + 3 output signals + 1 event log)	Pass
3	IRC5 OPC UA Server with one robot controller having 51 subscription changes per second (47 Rapid variables + 3 output signals + 1 event log)	Pass
4	IRC5 OPC UA Server with one robot controller having 71 subscription changes per second (67 Rapid variables + 3 output signals + 1 event log)	Pass
5	IRC5 OPC UA Server with one robot controller having 101 subscription changes per second (97 Rapid variables + 3 output signals + 1 event log)	Pass
6	IRC5 OPC UA Server with one robot controller having 151 subscription changes per second (147 Rapid variables + 3 output signals + 1 event log)	Pass
7	IRC5 OPC UA Server with one robot controller having 201 subscription changes per second (197 Rapid variables + 3 output signals + 1 event log)	Pass
8	IRC5 OPC UA Server with one virtual controller having 1000 I/O signals and 1000 I/O signals subscription changes per second	Pass
9	IRC5 OPC UA Server with four virtual controllers each having 13 subscription changes per second (10 I/O signals + 3 Rapid variables)	Pass
10	IRC5 OPC UA Server with eight virtual controllers each having 13 subscription changes per second (10 I/O signals + 3 Rapid variables)	Pass
11	IRC5 OPC UA Server with twelve virtual controllers each having 13 subscription changes per second (10 I/O signals + 3 Rapid variables)	Pass

6.3 Appendix C - IoT Gateway configuration file

IoTGateway.config

```
<IoTGatewayConfig >
  <Username>Default User</Username>
  <Password>robotics</Password>
  <Language>en</Language>
  <SubscriberPollTime>1000</SubscriberPollTime>
  <Servername>ABB.IRC5.OPCUA.Server</Servername>
  <Portnumber>61510</Portnumber>
  <AliasList>
    <AliasItem>
      <LocalRemoteController>Local</LocalRemoteController>
      <Alias>vera_system</Alias>
      <SystemID>c518004e-64ed-431f-a288-cf9ce8107f43</SystemID>
      <CtrlID />
      <SystemName>vera_7.2</SystemName>
      <CtrlName />
      <Address />
    </AliasItem>
    <AliasItem>
      <LocalRemoteController>Local</LocalRemoteController>
      <Alias>vera_not_supported</Alias>
      <SystemID>0b71724a-2e4d-4485-a943-c4ff58a2a83a</SystemID>
      <CtrlID />
      <SystemName>vera_7.2_1</SystemName>
      <CtrlName />
      <Address>C:\Users\INSIMAR\Documents\RobotStudio\Virtual
        Controllers\vera_7.2_1</Address>
    </AliasItem>
  </AliasList>
</IoTGatewayConfig>
```

Alias definition

Each alias definition within this file consists of the following lines of XML syntax:

```
<AliasItem>
  <LocalRemoteController>Local</LocalRemoteController>
  <Alias>vera_system</Alias>
  <SystemID>c518004e-64ed-431f-a288-cf9ce8107f43</SystemID>
  <CtrlID />
  <SystemName>vera_7.2</SystemName>
  <CtrlName />
  <Address />
</AliasItem>
```

PCI value

Each alias definition has a PCI value, which specifies one of the following:

- **Connected** - the specified robot controller is connected and has the required RobotWare option installed.

Continues on next page

6 Appendix

6.3 Appendix C - IoT Gateway configuration file

Continued

- Disconnected - either the alias cannot be resolved to a single robot controller on the network, or there is no such robot controller connected to the network.
- NoPCI - Missing RobotWare options.

Parameters

Each alias definition consists of seven parameters, which are specified as XML element attributes; these parameters correspond to:

- robot controller is Local or Remote (the LocalRemoteController attribute of the AliasItem element).
- The alias name is the Name attribute of the Alias element; for example, `vera_system`. This is the name you want the robot to be identified as by the IoT Gateway.
- The robot's IP Address (the IP attribute of the AliasItem element); for example, `130.110.69.254`
- The robot's Controller ID (the CtrlID attribute of the AliasItem element).
- The robot's SystemID (the SystemID attribute of the AliasItem element).
- The robot's Controller Name (the CtrlName attribute of the AliasItem element);
- The robot's System Name (the SystemName attribute of the AliasItem element); for example, `vera_7.2`



Note

Not all parameters necessarily need to contain values, according the association rules described in the section [Aliases on page 21](#).

6.4 Appendix D - ABB Robotics OPC UA proprietary information model

6.4.1 Overview

This section describes the OPC UA information model for ABB robot controllers.

6 Appendix

6.4.2 OPC Unified Architecture for ABB Robotics Controller

6.4.2 OPC Unified Architecture for ABB Robotics Controller

ObjectType RobotControllersType

A container for Robot Controller objects.

Table 1: ObjectType RobotControllersType

Attribute	Value
BrowseName	RobotControllersType
IsAbstract	False

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	No-deClass	BrowseName	Data-Type	TypeDefinition	ModellingRule	Access
HasComponent	Object	S_Alias_name_		RobotControllerType	OptionalPlaceholder	

S_Alias_name_: A robot controller is identified by its alias name that must be unique.

ObjectType RobotControllerType

Top level object type for an ABB Robotics Controller.

Table 2: ObjectType RobotControllerType

Attribute	Value
BrowseName	RobotControllersType
IsAbstract	False

Subtype of BaseObjectType of <http://opcfoundation.org/UA/>

Reference	No-deClass	Browse-Name	Data Type	TypeDefinition	ModellingRule	Access
HasProperty	Property	BootVersion	String	PropertyType	Mandatory	Read
HasProperty	Property	ControllerAddress	String	PropertyType	Mandatory	Read
HasProperty	Property	ControllerID	String	PropertyType	Mandatory	Read
HasProperty	Property	Controller-Name	String	PropertyType	Mandatory	Read
HasProperty	Property	SystemID	Guid	PropertyType	Mandatory	Read
HasProperty	Property	SystemName	String	PropertyType	Mandatory	Read
HasComponent	Variable	Collision-DetectState	CollisionDetect-StateEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	ControllerExecutionState	ControllerExecution-StateEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	Controller-State	ControllerExecution-StateEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	InterfaceState	InterfaceStateEnum	BaseDataVariableType	Mandatory	Read

Continues on next page

Reference	No-deClass	Browse-Name	Data Type	TypeDefinition	ModellingRule	Access
HasComponent	Variable	Operating-Mode	OperatingModeEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	SpeedRatio	Int32	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	SystemClock	DateTime	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	RapidProgramUsed-Memory	UInt32	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	RapidProgramFree-Memory	UInt32	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	MasterRAPID	MastershipEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	MasterCFG	MastershipEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Object	IO_System		IOSystemType	Mandatory	
HasComponent	Object	RAPID		RAPIDType	Mandatory	

BootVersion: A read-only string that contains the value of the robot controller's RobotWare operating system version.

ControllerAddress: A read-only string that specifies either a) the IP network address of the Real Controller (RC), or b) the localhost loopvback address (127.0.0.1) for a Virtual Controller (VC) running on the PC.

ControllerID: By default, set to the serial number of the controller and is thereby a unique identifier of the controller. This is a read-only value.

ControllerName: An identification of the controller that is independent of the system or the software running on the controller. This name comes from the robot controller and may be the same as the AliasName, however while the AliasName must be unique, there is no such requirement on the ControllerName. This is a read-only value.

SystemID: A read-only GUID that contains the identifier that globally and uniquely identifies a robot controller/system combination.

SystemName: A read-only string that contains the name of the RobotWare system currently loaded. This is the name assigned by the user when creating a system in e.g. Installation Manager.

CollisionDetectState: A read-only value that contains the state of the collision detection mechanism in the robot controller. See the definition of CollisionDetectStateEnum for details.

ControllerExecutionState: A read-only value that contains the execution state (Running or Stopped) of the robot controller.

ControllerState: A read-only value that contains the state of the robot controller. See the definition of ControllerStateEnum for details.

Continues on next page

6 Appendix

6.4.2 OPC Unified Architecture for ABB Robotics Controller

Continued

InterfaceState: A read-only value indicating the state of the communication interface to the robot controller. This state is maintained by the OPC UA server. See the definition of InterfaceStateEnum for details.

OperatingMode: A read-only value that contains the robot controller operational mode. See the definition of OperatingModeEnum for details.

SpeedRatio: A read-only value that defines the speed ratio of the robot controller in percent, range 0 - 100.

SystemClock: A read-only value that contains the robot controller's system clock value. It is only valid when the interface to the controller is operational.

RapidProgramUsedMemory: A read-only value that defines the amount of memory in bytes being used by the robot controller's RAPID program.

RapidProgramFreeMemory: A read-only value that defines the amount of memory in bytes available to the robot controller's RAPID program.

MasterRAPID: A read-only value that shows if the mastership of RAPID is held by another client. See definition of MastershipEnum for details.

MasterCFG: A read-only value that shows if the mastership of CFG is held by another client. See definition of MastershipEnum for details.

IO_System: Represents the I/O system of the controller.

RAPID: Container for all RAPID tasks in the controller.

ObjectType IOSystemType

Object type describing the the I/O system of the robot controller.

Table 3: ObjectType IOSystemType

Attribute	Value
BrowseName	IOSystemType
IsAbstract	False

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	NodeClass	BrowseName	Data-Type	TypeDefinition	ModellingRule	Access
HasComponent	Object	IO_Signals		IOSignalsType	Mandatory	

IO_Signals: Container for all I/O signals in the controller.

ObjectType IOSignalsType

A container for I/O signals.

Table 4: ObjectType IOSignalsType

Attribute	Value
BrowseName	IOSignalsType
IsAbstract	False

Continues on next page

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	No-deClass	Browse-Name	Data Type	TypeDefini-tion	Modellin-gRule	Access
HasCom-ponent	Variable	S_Sig-nal_name_	BaseData-Type	DataItem-Type	OptionalPlace-holder	Read-Write

S_Signal_name_: Represents an IO signal.



Note

Depending on the configuration of signal, it may be possible to write to a input or output signal. The 'Type of signal', 'Access level', and 'Safe level' parameters all influence whether it is possible to write to a 'signal' or not. OPC UA Client see all the signals from Robot controller's point of view, that is, O/P signal is an output from the Robot controller to some external equipment and vice versa for I/P signals.

For more information on system parameters, please refer to *Technical reference manual - System parameters, section I/O*.

ObjectType RAPIDType

Object type describing the RAPID sub-system of the robot controller.

Table 5: ObjectType RAPIDType

Attribute	Value
BrowseName	RAPIDType
IsAbstract	False

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	No-deClass	BrowseName	Data-Type	TypeDefini-tion	ModellingRule	Ac-cess
HasCompon-ent	Object	S_Task_name_		RAPIDTask-Type	MandatoryPlace-holder	

S_Task_name_: Represents a RAPID task in the controller.

ObjectType RAPIDTaskType

Represents a RAPID task in the controller. It acts as a container for any modules loaded in the task.

Table 6: ObjectType RAPIDTaskType

Attribute	Value
BrowseName	RAPIDTaskType
IsAbstract	False

Continues on next page

6 Appendix

6.4.2 OPC Unified Architecture for ABB Robotics Controller

Continued

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	No-deClass	Browse-Name	Data Type	TypeDefinition	ModellingRule	Access
HasComponent	Variable	TaskExecutionState	TaskExecutionStateEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Variable	TaskState	TaskExecutionStateEnum	BaseDataVariableType	Mandatory	Read
HasComponent	Object	S_Module_name_		RAPIDModuleType	Optional-Placeholder	
HasComponent	Variable	ProgramPointer	ProgramPosition	BaseDataVariableType	Mandatory	Read

TaskExecutionState: A read-only value that contains the execution state of the RAPID task. See the definition of **TaskExecutionStateEnum** for details.

TaskState: A read-only value that contains the state of the RAPID task. See definition of **TaskStateEnum** for details.

S_Module_name_: Represents a RAPID module in a RAPID task.

ProgramPointer: ProgramPointer is introduced under RAPID address space to get correct module name, routine, and line number (exposed by the OPC UA server for each RAPID task).

ObjectType RAPIDModuleType

An object representing a RAPID module. It acts as a container for all persistent variables in the module.

Table 7: ObjectType RAPIDModuleType

Attribute	Value
BrowseName	RAPIDModuleType
IsAbstract	False

Subtype of FolderType of <http://opcfoundation.org/UA/>

Reference	No-deClass	BrowseName	Data Type	TypeDefinition	ModellingRule	Access
HasComponent	Variable	S_PERS_name_	BaseDataVariableType	DataItem- Type	Optional-Placeholder	Read-Write

S_PERS_name_: Represents a persistent (PERS) variable in a RAPID module. Clients can both read and write persistent variables. A successful write to a persistent variable requires that no other client has mastership of RAPID. See description of **MasterRAPID** variable.

CollisionDetectStateEnum Values

Defines possible states of the collision detection mechanism in the robot controller.

Table 8: CollisionDetectStateEnum Values

Continues on next page

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Unknown	0	Unknown.
Initiated	1	Collision detection has been initiated.
Started	2	Collision detection has been started.
Confirmed	3	Collision detected/confirmed.
Acknowledged	4	Collision detected and acknowledged.

Continues on next page

6 Appendix

6.4.2 OPC Unified Architecture for ABB Robotics Controller

Continued

ControllerExecutionStateEnum Values

Defines possible execution states of the robot controller.

Table 9: ControllerExecutionStateEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Unknown	0	Status is unknown.
Running	1	At least one normal RAPID task is executing or performing regain.
Stopped	2	No normal RAPID task is executing or performing regain.

ControllerStateEnum Values

Defines possible states of the robot controller.

Table 10: ControllerStateEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Init	0	Initialize state.
MotorsOff	1	Motors off state.
MotorsOn	2	Motors on state.
GuardStop	3	Guard stop state.
EmergencyStop	4	Emergency stop state.
EmergencyStopReset	5	Emergency stop reset state.
SystemFailure	6	System failure state.
Unknown	99	Unknown state.

InterfaceStateEnum Values

Defines possible states of the interface to the robot controller.

Table 11: InterfaceStateEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Disconnected	0	The interface to the robot controller is disconnected and non-functional.
Connected	1	The interface to the robot controller is connected and operational.
NoPCInterfaceOption	2	The robot controller does not have the PC Interface RobotWare option that creates the interface to the controller.
UnresolvableAlias	3	The system cannot resolve the indicated alias to a single robot controller on the network.

Continues on next page

OperatingModeEnum Values

Defines possible operational modes of the robot controller.

Table 12: OperatingModeEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Auto	0	Automatic mode (production).
Init	1	Initialize mode.
ManualReduced-Speed	2	Manual reduced speed mode.
ManualFullSpeed	3	Manual full speed mode.
AutoChange	4	A change to automatic mode has been requested.
ManualFullSpeed-Change	5	A change to manual full speed has been requested.
NotApplicable	6	Controller operating mode is not applicable in current controller state.

TaskExecutionStateEnum Values

Defines possible task execution states.

Table 13: TaskExecutionStateEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Ready	0	The task has no PCP or execution context.
Stopped	1	Task is not executing or not performing regain. PCP and execution context are defined in task.
Running	2	Task is executing or performing regain.
UnInitiated	3	The program server is not initialized. State only assumed during startup.
Unknown	4	Status is unknown.

TaskStateEnum Values

Defines possible task states.

Table 14: TaskStateEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
Empty	0	No modules are loaded in the task.
Loaded	1	Modules are loaded, but not linked.
Linked	2	Modules are loaded and linked.
Initiated	3	The program server is not initialized. State only assumed during startup.

Continues on next page

6 Appendix

6.4.2 OPC Unified Architecture for ABB Robotics Controller

Continued

MastershipEnum Values

Defines possible mastership values.

Table 15: MastershipEnum Values

Subtype of Enumeration of <http://opcfoundation.org/UA/>

Name	Value	Comment
NoMaster	0	No client has mastership
HeldRemote	1	A remote client has mastership.
HeldLocal	2	A local client has mastership (typically the TPU)
HeldInternal	3	The controller itself has mastership.

Index

A

- Add New Alias, 23
- Alias definition, 149
- Aliases, 21
 - add, 23
 - edit, 27
 - setup, 12
- Alias Name, 37
- application certificate, 45

B

- Built-in helpers, 134

C

- certificate management, 42
 - client certificate, 42
 - server application instance certificate, 44
- configuration application, 18
- Connection Criteria, 23
- Controller ID, 22
- Controller Name icons, 26

D

- data variables, 135

E

- edit alias, 27

H

- handlebars, 132
 - language features, 133

I

- IoT Gateway

- configuration file, 31
 - logs, 36
 - plug-in, 32
 - prerequisite, 12
 - product installation, 12
 - troubleshooting, 137
- IoTGateway.config, 149
- IoT Gateway plug-in
 - username, 33

M

- Main Screen Components, 18
- MQTT client configuration, 66
- MQTT Publisher, 63
 - Configuration, 65
- Multiple controllers matching the same Alias definition, 21

O

- OPC UA for ABB robotics, 152

P

- Payload configuration, 94

R

- Robot configuration, 77

S

- server control, 35
- Simple expressions, 133
- Status Icons, 20
- System ID, 22
- System Name, 22

T

- Toolbar Buttons, 19
- Trigger configuration, 79



ABB AB

Robotics & Discrete Automation

S-721 68 VÄSTERÅS, Sweden

Telephone +46 10-732 50 00

ABB AS

Robotics & Discrete Automation

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong New District

SHANGHAI 201319, China

Telephone: +86 21 6105 6666

ABB Inc.

Robotics & Discrete Automation

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

abb.com/robotics