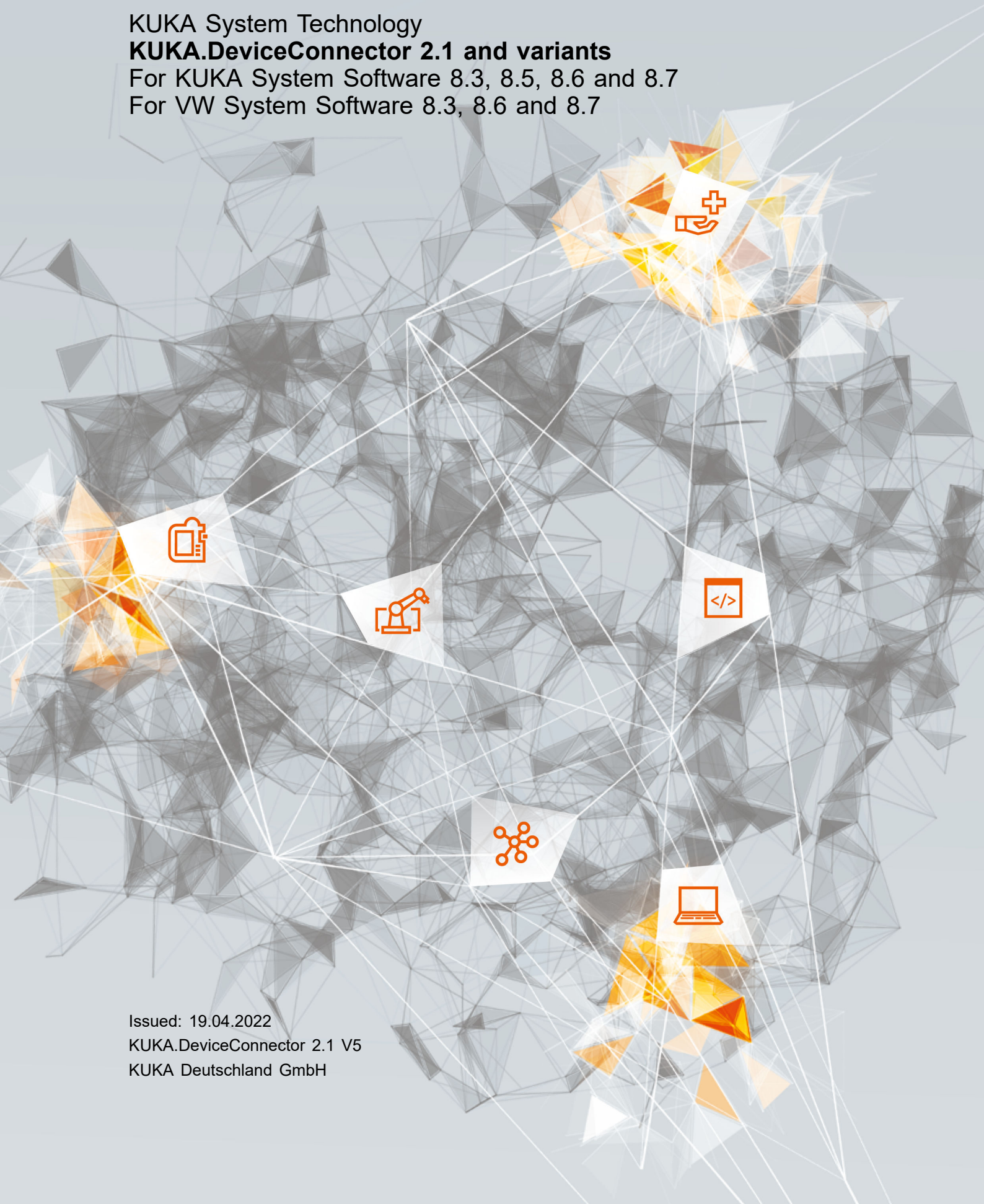
KUKA System Technology

KUKA.DeviceConnector 2.1 and variants

For KUKA System Software 8.3, 8.5, 8.6 and 8.7

For VW System Software 8.3, 8.6 and 8.7



Issued: 19.04.2022

KUKA.DeviceConnector 2.1 V5

KUKA Deutschland GmbH

KUKA Deutschland GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

KIM-PS5-DOC

Translation of the original documentation

Publication:	Pub KUKA.DeviceConnector 2.1 (PDF) en PB13101
Book structure:	KUKA.DeviceConnector 2.1 V5.1 BS11870
Version:	KUKA.DeviceConnector 2.1 V5

Contents

1 Introduction

1.1 Target group

This documentation is aimed at users with the following knowledge and skills:

- Advanced knowledge of the robot controller system
- Advanced knowledge of network connections

For optimal use of KUKA products, we recommend the training courses offered by KUKA College. Information about the training program can be found at www.kuka.com or can be obtained directly from our subsidiaries.

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

- Documentation for the robot arm
- Documentation for the robot controller
- Documentation for the smartPAD-2 or smartPAD pro (if used)
- Operating and programming instructions for the System Software
- Instructions for options and accessories
- Spare parts overview in KUKA Xpert

Each set of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

These warnings are provided for safety purposes and **must** be observed.

DANGER

These warnings mean that it is certain or highly probable that death or severe injuries **will** occur, if no precautions are taken.

WARNING

These warnings mean that death or severe injuries **may** occur, if no precautions are taken.

CAUTION

These warnings mean that minor injuries **may** occur, if no precautions are taken.

NOTICE

These warnings mean that damage to property **may** occur, if no precautions are taken.

These warnings contain references to safety-relevant information or general safety measures.
These warnings do not refer to individual hazards or individual precautionary measures.

This warning draws attention to procedures which serve to prevent or remedy emergencies or malfunctions:

SAFETY INSTRUCTION
The following procedure must be followed exactly!

Procedures marked with this warning **must** be followed exactly.

Notices

These notices serve to make your work easier or contain references to further information.

Tip to make your work easier or reference to further information.

1.4 Terms used

Term	Description
IP	Internet Protocol The Internet Protocol is used to define subnetworks by means of physical MAC addresses.
KLI	KUKA Line Interface Ethernet interface of the robot controller for external communication.
KONI	KUKA.OptionalNetworkInterface Optional Ethernet interface of the robot controller for external communication.
LDS-ME	Local Discovery Server with Multicast Extensions Local discovery servers enable OPC UA clients to find OPC UA servers in the network and connect to them.
MQTT	Message Queuing Telemetry Transport Open messaging protocol for transferring telemetry data between devices, which can be used even with high delays and limited networks
OPC UA	Open Platform Communications - Unified Architecture Open standard for data exchange between devices (vendor- and platform-independent)
PKI	Public key infrastructure System that can issue, distribute and verify digital certificates for authentication
SPOC	Single Point of Control
TCP	Transmission Control Protocol Protocol of the data exchange between devices of a network. TCP constitutes a virtual channel between 2 sockets in a network connection. Data can be transmitted on this channel in both directions.

1.5 Trademarks

mosquitto is a trademark of the Eclipse Foundation.

PROFINET® is a trademark of the PROFIBUS users' organization.

UaExpert® is a trademark of Unified Automation GmbH.

Windows is a trademark of Microsoft Corporation.

1.6 Licenses

The KUKA license conditions and the license conditions of the open-source software used can be found in the following folders:

- Under .\LICENSE on the data storage medium with the installation files of the KUKA software
- On the KUKA smartHMI, in the main menu under **Help** > **Info**, tab **Licenses**
- Under D:\KUKA_OPT*Option package name*\LICENSE on the robot controller
- In the license folder under the name of the option package in the **Options** catalog in WorkVisual

Further information about open-source licenses can be requested from the following address: opensource@kuka.com

2 Product description

2.1 Product description

Description

The KUKA.DeviceConnector 2.1 option package enables access to robot data via OPC UA and MQTT.

- OPC UA

  Data can be transferred bidirectionally within a network.
- MQTT

  Data can be transferred unidirectionally across network boundaries.

KUKA.DeviceConnector 2.1 enables customer-specific server peripherals to be integrated.

Functions

OPC UA:

- Robot controller as OPC UA server
- Connection of multiple OPC UA clients
- Reading of robot data
- Reading and writing of process variables

MQTT:

- Robot controller as MQTT client
- Connection of multiple MQTT brokers
- Reading of robot data
- Reading of process variables

2.2 Variants of the option package

The option package is available in a number of variants: These variants differ in the amount of data they provide:

- KUKA.DeviceConnector 2.1

  Basic package for access to all data defined in OPC UA standard 40010-1 for robotics and/or OPC UA standard 40001-1 for machinery and systems
- KUKA.DeviceConnector FileTransfer 2.1

  Package for access to the file system of the robot controller, including LOG and XML files
- KUKA.DeviceConnector Messages 2.1

  Package for access to all messages and Windows event logs of the robot controller
- KUKA.DeviceConnector SystemVars 2.1

  Package for access to all system variables and global KRL variables including customer-specific variables (CustomerData)
- KUKA.DeviceConnector CustomerData 2.1

  Package for access to all customer-specific variables

- KUKA.DeviceConnector SoftPLC 2.1

  Package for access to all Soft PLC data

  Precondition: KUKA.PLC ProConOS or KUKA.PLC ProConOS ltd is installed.
- KUKA.DeviceConnector Advanced 2.1

  Complete package with all available data and functions, including:

  – Soft PLC data

    Precondition: KUKA.PLC ProConOS or KUKA.PLC ProConOS ltd is installed.

  – PROFINET data

    Precondition: KUKA.PROFINET® -/S or KUKA.PROFINET® M/S is installed.

Depending on the application scenario, required data points and file information, the option packages can be installed in combination.

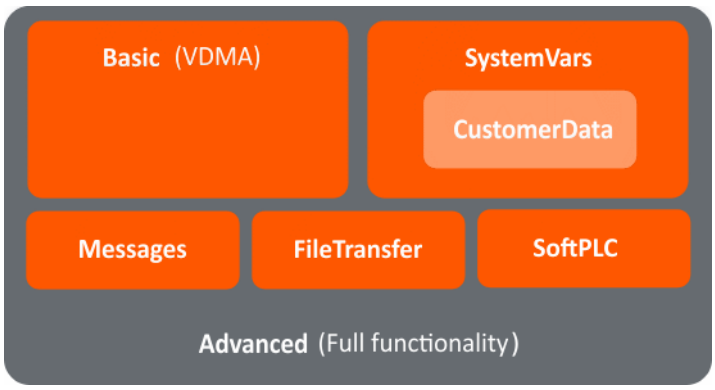


Fig. 2-1: Overview of variants

Exceptions:

- Advanced option cannot be combined (contains all available data)
- CustomerData and SystemVars options cannot be combined (CustomerData included in SystemVars)

2.3 Data processing with OPC UA client or MQTT broker

Description

In order to be able to process the data published with KUKA.DeviceConnector, an OPC UA client or MQTT broker is required.

- An OPC UA client can connect to KUKA.DeviceConnector via the KLI or KONI of the robot controller.
  – Port 4840 is reserved for OPC UA. The connection can only be established via this port.
- KUKA.DeviceConnector can connect to one or more MQTT brokers.
  – For connection configuration, see: (>>> *6.4 "MQTT configuration" Page 43*)

Demo clients

We recommend using the following demo clients for test purposes.

- For OPC UA: **UaExpert v1.4.x and higher**

  The UaExpert demo client can be downloaded free of charge from the website of Unified Automation GmbH:

  www.unified-automation.com
- For MQTT: **mosquitto 1.6.2 or higher**

2.4 Intended use and misuse

Use

The KUKA.DeviceConnector 2.1 option package enables access to robot data via OPC UA and MQTT and is used to read and write variables. There are absolutely no safety-relevant functions. Safety-relevant data cannot be influenced/written to via the interface.

KUKA.DeviceConnector 2.1 may be incorrectly configured, with the effect that some variables can no longer be output. Operation in accordance with the intended use thus also involves compliance with the configuration instructions in the documentation of the option package.

Misuse

Any use or application deviating from the intended use is deemed to be misuse and is not allowed. It will result in the loss of warranty and liability claims. KUKA is not liable for any damage resulting from such misuse.

3 Safety

This documentation contains safety instructions which refer specifically to the option package described here.

The fundamental safety information for the industrial robot can be found in the “Safety” chapter of the Operating and Programming Instructions for System Integrators or the Operating and Programming Instructions for End Users.

Comply with safety-relevant information
The safe use of this product requires knowledge of and compliance with fundamental safety measures. Death, severe injuries or damage to property may otherwise result.

- The “Safety” chapter in the operating and programming instructions of the KUKA System Software (KSS) must be observed.
- The principle of “single point of control” (SPOC) must be completely adhered to.

WARNING

Danger to life and limb due to non-secure signal changes
Signal states can be changed by accessing an OPC UA client or by transferring a project from WorkVisual. Death, severe injuries and damage to property may result from non-secure signal changes.

- Only execute dangerous signals, e.g. the signal for opening/closing a gun, if at least one of the following conditions is met:
  - Safety gate is closed.
  - Automatic mode
  - Peripheral contactor (US2) is used.
  - The variables **bSPOC_UserSafetyActive** and **bSPOC_MotionEnabled** were taken into account when mapping the signals.

Keeping the number of OPC UA connections to a minimum
KUKA.DeviceConnector allows up to 60 individual OPC UA connections through which a manipulation of the robot controller is possible. This may result in unauthorized access to the robot controller.

- Keep the number of OPC UA clients and authorized OPC UA users to a minimum.

3.1 Safety measures for single point of control

Overview

If certain components are used on the industrial robot, safety measures must be taken to ensure complete implementation of the principle of “single point of control” (SPOC).

The relevant components are:

- Submit interpreter
- PLC
- OPC server
- Remote control tools
- Tools for configuration of bus systems with online functionality
- KUKA.RobotSensorInterface

- KUKA.DeviceConnector
  (not KUKA.DeviceConnector pre-installed)

Since only the system integrator knows the safe states of actuators in the periphery of the robot controller, it is his task to set these actuators to a safe state, e.g. in the event of an EMERGENCY STOP.

Further safety measures for Single Point of Control
Depending on the specific application, further safety measures may be required to ensure complete implementation of the principle of “single point of control”. Failure to take this precaution into consideration may result in death, injuries or damage to property.

- Check whether further safety measures are required; if so, implement them.

T1, T2

In modes T1 and T2, the components referred to above may only access the industrial robot if the following signals have the following states:

Signal	State required for SPOC
$USER_SAF	TRUE
$SPOC_MOTION_ENABLE	TRUE

Submit interpreter, PLC

If motions, (e.g. drives or grippers) are controlled with the submit interpreter or the PLC via the I/O system, and if they are not safeguarded by other means, then this control will take effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

If variables that affect the robot motion (e.g. override) are modified with the submit interpreter or the PLC, this takes effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

- In T1 and T2, the system variable $OV_PRO must not be written to by the submit interpreter or the PLC.

OPC server, KUKA.DeviceConnector, remote control tools

These components can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measure:

If these components are used, outputs that could cause a hazard must be determined in a risk assessment. These outputs must be designed in such a way that they cannot be set without being enabled. This can be done using an external enabling device, for example.

Tools for configuration of bus systems

If these components have an online functionality, they can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

- WorkVisual from KUKA
- Tools from other manufacturers

Safety measure:

In the test modes, programs, outputs or other parameters of the robot controller must not be modified using these components.

4 Installation

The option package can be installed on the robot controller via the smartHMI.

4.1 System requirements

Hardware

- KR C4 or KR C5 robot controller
- or VKR C4 robot controller

Software

KR C4 robot controller:

- KUKA System Software 8.3, 8.5 or 8.6
- KUKA.DeviceConnector pre-installed 2.1

VKR C4 robot controller:

- VW System Software 8.3 or 8.6
- KUKA.DeviceConnector pre-installed 2.1

KR C5 robot controller:

- KUKA System Software 8.7
- Or VW System Software 8.7
- KUKA.DeviceConnector pre-installed 2.1

Compatibility

KUKA.DeviceConnector 2.1 must not be installed on a robot controller together with the following option package:

- KUKA.OPC Server
- KUKA.OPC UA

4.2 Installation via smartHMI

4.2.1 Installing or updating the option package

It is advisable to archive all relevant data before updating a software package.

Precondition

- User rights:
  - KSS: function group **General configuration**
    But at least the user group “Expert”
  - VSS: user group “User”
- T1 or T2 mode
- No program is selected.
- USB stick with the option package (KOP file)

NOTICE

Data loss due to USB sticks from third-party manufacturers
Data may be lost if USB sticks from manufacturers other than KUKA are used for activities on the robot controller.

- For activities on the robot controller requiring a USB stick, use a KUKA stick.
  The KUKA sticks are validated for use with the robot controller.

Procedure

1. Connect the USB stick to the robot controller or smartPAD.
2. In the main menu, select **Start-up** > **Additional software**.
3. Press **New software**: an entry for the option package (name and version) must be displayed in the **Name** column and drive **E:** or **K:** in the **Path** column.
   If not, press **Refresh**.
4. If the entry for the option package is now displayed, continue with step 5.
   Otherwise, the path from which the software is to be installed must be configured first:
   a. Press the **Configure** button.
   b. Select a line in the **Installation paths for options** area.
      Note: If the line already contains a path, this path will be overwritten.
   c. Press **Path selection**. The available drives are displayed.
   d. If the stick is connected to the robot controller: Select **E:**.
      If the stick is connected to the smartPAD: **K:** instead of **E:**
   e. Press **Save**. The **Installation paths for options** area is displayed again. It now contains the new path.
   f. Mark the line with the new path and press **Save** again.
5. Activate the check box next to the option package and press **Install**. Confirm the installation query with **OK**.
6. The request for confirmation *Do you want to activate the project […]?* is displayed. The active project is overwritten during activation. If no relevant project will be overwritten: answer the query with **Yes**.
7. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is installed and the robot controller carries out a reboot.
8. Remove the stick.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.2.2 Uninstalling the option package

It is advisable to archive all relevant data before uninstalling a software package.

Precondition

- User rights:
  - KSS: function group **General configuration**
    But at least the user group “Expert”

  - VSS: user group “User”
- T1 or T2 mode
- No program is selected.

Procedure

1. In the main menu, select **Start-up** > **Additional software**.
2. Activate the check box next to the option package and press **Uninstall**. Answer the request for confirmation with **Yes**.
3. Answer the request for confirmation *Do you want to activate the project […]?* with **Yes**.
4. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is uninstalled and the robot controller carries out a reboot.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

5 Operation

The following descriptions and procedures are based on the UaExpert demo client.

5.1 Connecting to OPC UA server via URL

Description

The robot controller serves as the OPC UA server. OPC UA clients can connect to the OPC UA server by entering the corresponding URL when adding the server. Once the connection has been established, the data are retrieved from the robot controller.

Precondition

- IP address or name of the robot controller is known.

Procedure

- When adding the server, enter the URL with the IP address or controller name as follows:
  - `opc.tcp://`*IP address*`:4840/`
  - `opc.tcp://`*Controller name*`:4840/`

5.2 Connecting to OPC UA server via LDS

Description

The robot controller serves as the OPC UA server. A local Discovery server runs on it at the same time: LDS-ME; port 50000; no port forwarding.

All LDS-ME-capable OPC UA clients in the network can automatically find and connect to the OPC UA server. Once the connection has been established, the data are retrieved from the robot controller.

Precondition

- OPC UA client has access to an LDS-ME server.

Procedure

1. Start the server search via LDS-ME on the client.
2. Select the robot controller in order to connect to the desired server.

5.3 OPC UA user groups

Description

Access to the data is controlled via user groups:

- Anonymous
  May browse all OPC UA variables.
- OpcUaObserver
  May browse and read all OPC UA variables.
- OpcUaOperator
  May browse, read and write all OPC UA variables.

The following user groups are responsible for administrative tasks:

- OpcUaConfigAdmin
  - Configure the MQTT Publisher with a suitable OPC UA client.
- OpcUaSecurityAdmin
  - Integrate the OPC UA server into the global public key infrastructure (PKI):
    - Save a new certificate for the OPC UA server.
    - Save a list of trusted client certificates.

(>>> *6.3 "Adding and updating certificates" Page 40*)

All user groups except Anonymous are protected with an initial password. The initial password for all groups is “kuka”. The password can be changed for each group.

(>>> *6.1 "Changing the initial password for OPC UA users" Page 39*)

Authentication via password

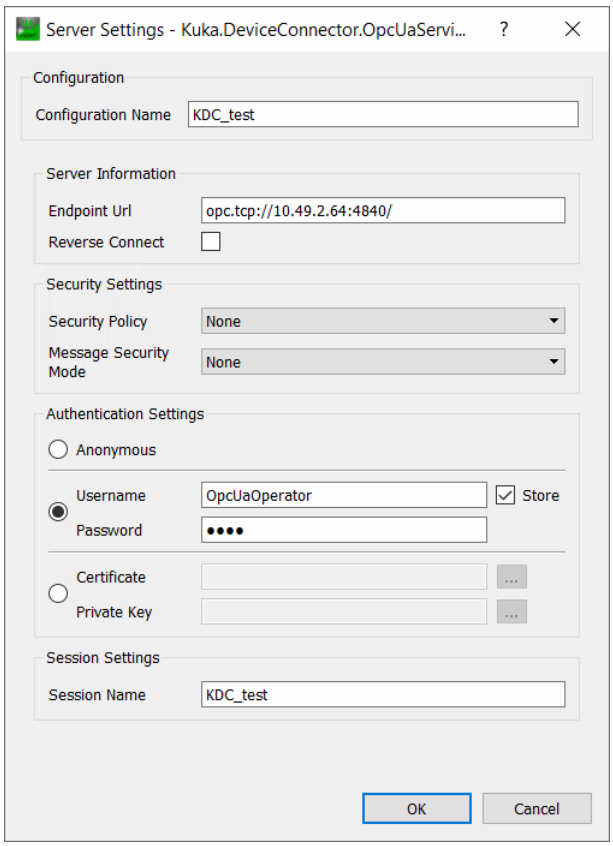


Fig. 5-1: Authentication via password

As an alternative to authentication via password, a self-signed certificate can be used.

(>>> *6.2 "Creating a self-signed certificate for authentication" Page 40*)

5.4 Basic data according to Robotics Companion Specification

These data are available with the following option packages:

- KUKA.DeviceConnector
- KUKA.DeviceConnector Advanced

In accordance with the Robotics Companion Specification, the following basic data are available under the **MotionDeviceSystem** node:

- Controllers > KRC
  Information about the robot controller
  (>>> *5.4.1 "Information about the robot controller" Page 21*)
- MotionDevices > Robot
  Information about the kinematic system
  (>>> *5.4.2 "Information about the kinematic system" Page 22*)
- SafetyStates > SafetyState
  Information about safety states of the robot controller
  (>>> *5.4.3 "Information on safety states" Page 24*)

Further information about the data can be found in the following documents from the OPC Foundation:

- OPC 40010-1 - Robotics Part 1: Vertical Integration
- OPC 40001-1 - UA for Machinery Part 1

5.4.1 Information about the robot controller

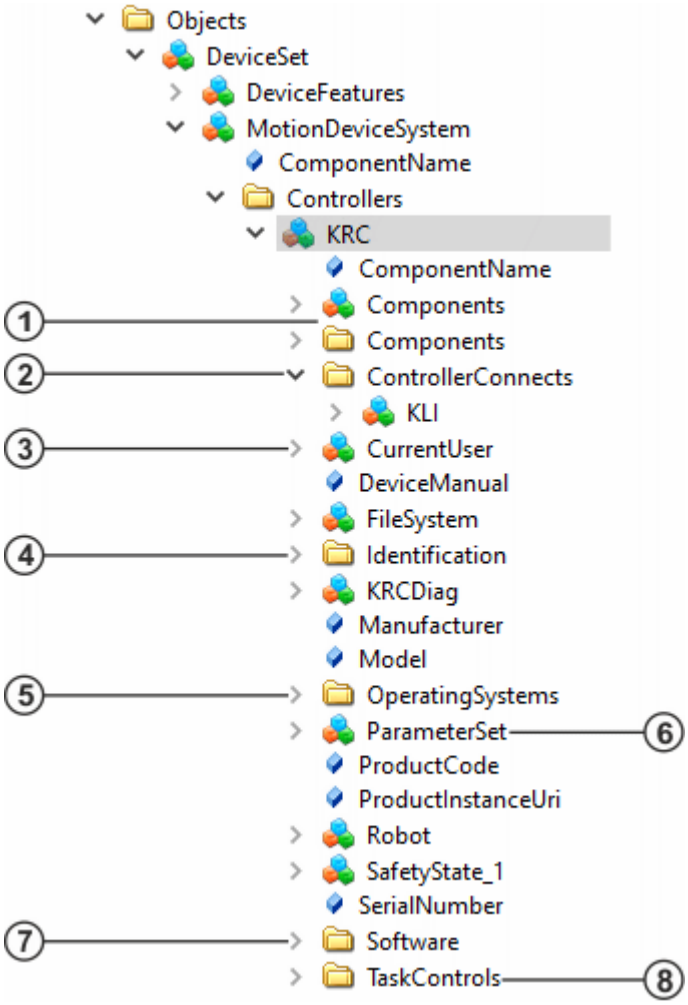


Fig. 5-2: Information about the robot controller

Item	Description
1	Electronic components of the robot controller with information on model name, serial number, firmware version, etc.
2	Network configuration
3	User currently logged on
4	Electronic identification plate
5	Operating systems of the robot controller incl. Windows registry
6	Operating information, e.g.: • Robot controller runtime • Activated project • IP address of the robot controller
7	System Software and installed option packages with version, release date, installation date, etc.
8	Information about robot interpreter and submit interpreter, e.g.: • Program name • Program status • Call stack status

5.4.2 Information about the kinematic system

A wide range of information is available for each kinematic system that is part of the robot system. The kinematic system can be a robot, linear unit or positioner, for example.

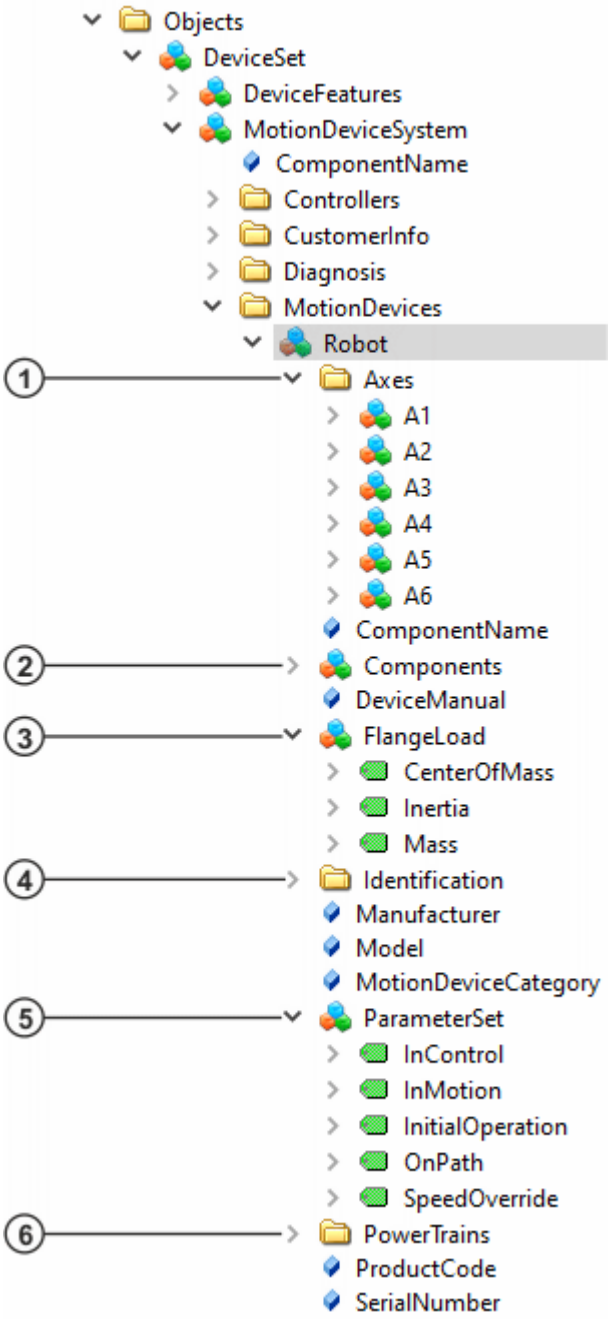


Fig. 5-3: Information about the kinematic system

Item	Description
1	Axes of the kinematic system
2	Electronic components of the kinematic system
3	Payload data on the flange
4	Electronic identification plate
5	Information about the robot motion, e.g.: • Drives under servo-control • Motion status • Program override

Item	Description
6	Information about the drives of the axes of a kinematic system, e.g.: • Brakes released/applied • Maximum torque deviation • Motor temperature

Further information is available for each axis of a kinematic system, e.g.:

- Supplementary load on axis
- Current axis position
- Current axis velocity

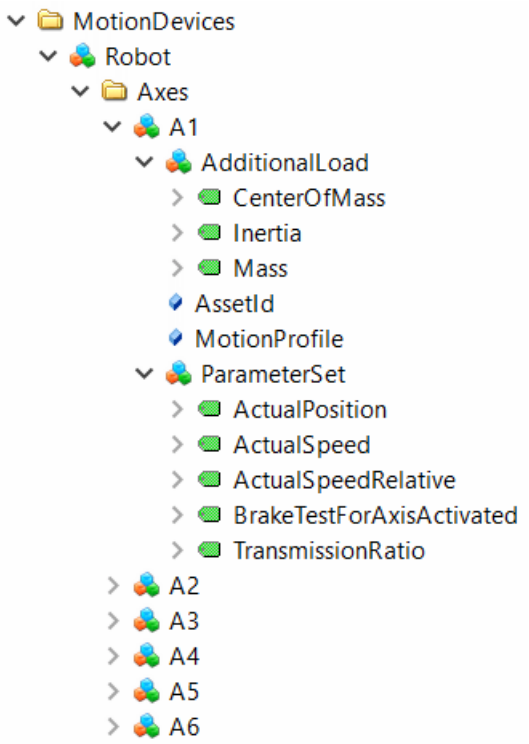


Fig. 5-4: Axis information

5.4.3 Information on safety states

The SafetyStates object provides information about safety states of the robot controller, e.g.:

- EMERGENCY STOP
- Enabling device
- Operating mode
- Safety stop

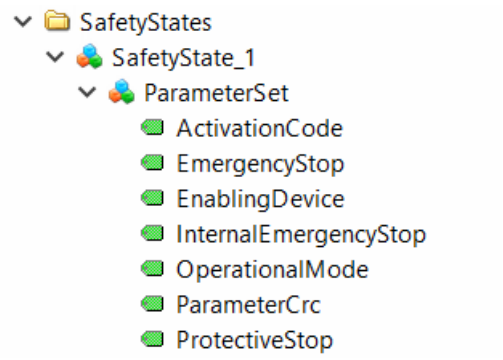


Fig. 5-5: Safety states

5.5 File transfer

This functionality is available with the following option packages:
- KUKA.DeviceConnector FileTransfer
- KUKA.DeviceConnector Advanced

Description

The OPC UA client can access the file system of the robot controller as well as KRCDiags and traces. The content of XML files is displayed in the OPC UA tree and can be read. (>>> *"Access to XML files" Page 27*)

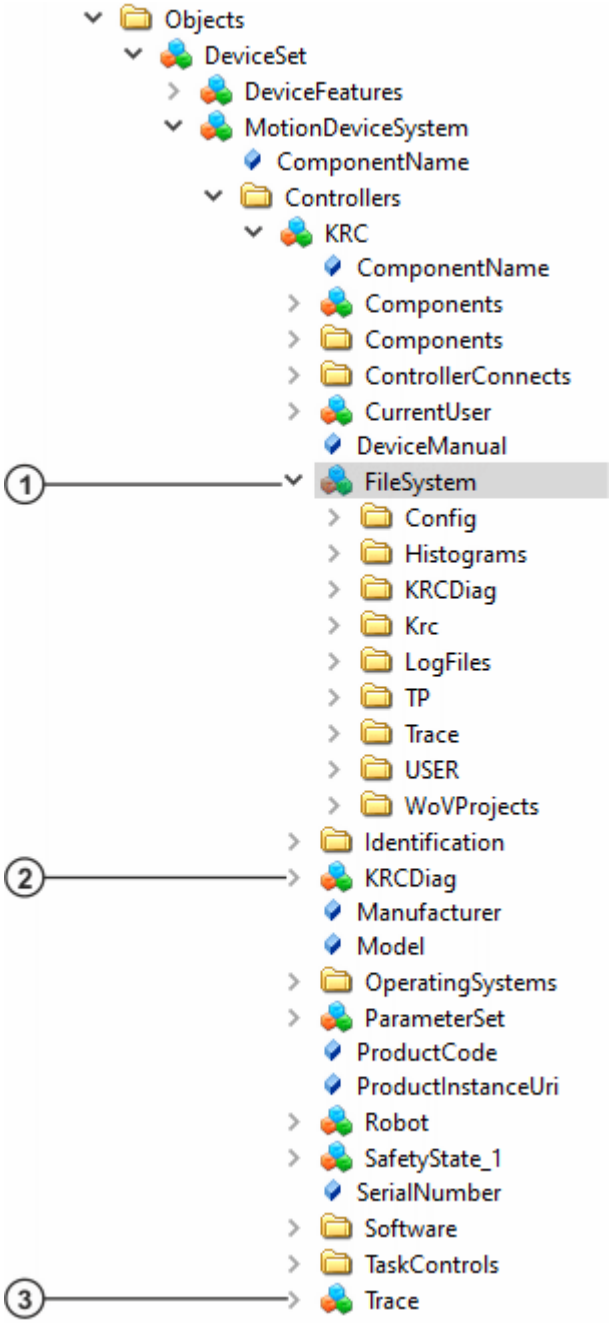


Fig. 5-6: File transfer

Item	Description
1	File system of the robot controller with all configuration files, program files, LOG files, etc. • ...\KRC\Roboter\Config\User\ with the user-specific configuration files • ...\KRCDiag with the diagnostic packages • ...\KRC\Roboter\KRC\ with the robot programs, machine data, etc. • …\KRC\Roboter\LogFiles\ with the LOG files • …\KRC\TP\ with the files of the installed option packages • …\Ikarus with the files of the Ikarus virus scanner Precondition: KUKA.Ikarus T3 is installed. • ...\KRC\Roboter\Trace\ with the trace definitions and results • …\KRC\Roboter\USER\ with the user-specific files • …\WoVProjects with the WorkVisual projects
2	KRCDiag for creating a diagnostic package
3	Trace function for recording and evaluating data (in accordance with trace configuration)

Access to XML files

Each file under **FileSystem** with the extension *.xml or *.config that is a valid XML file has a content node. This content node represents the XML structure in the OPC UA tree. Attributes, comments and text elements become variables. The variables can be read.

- The XML elements are numbered in the order in which they appear in the file, e.g. NATRule[1] … NATRule[14].
- All text elements have the browse name **#text**.
- All comments have the browse name **#comment**.

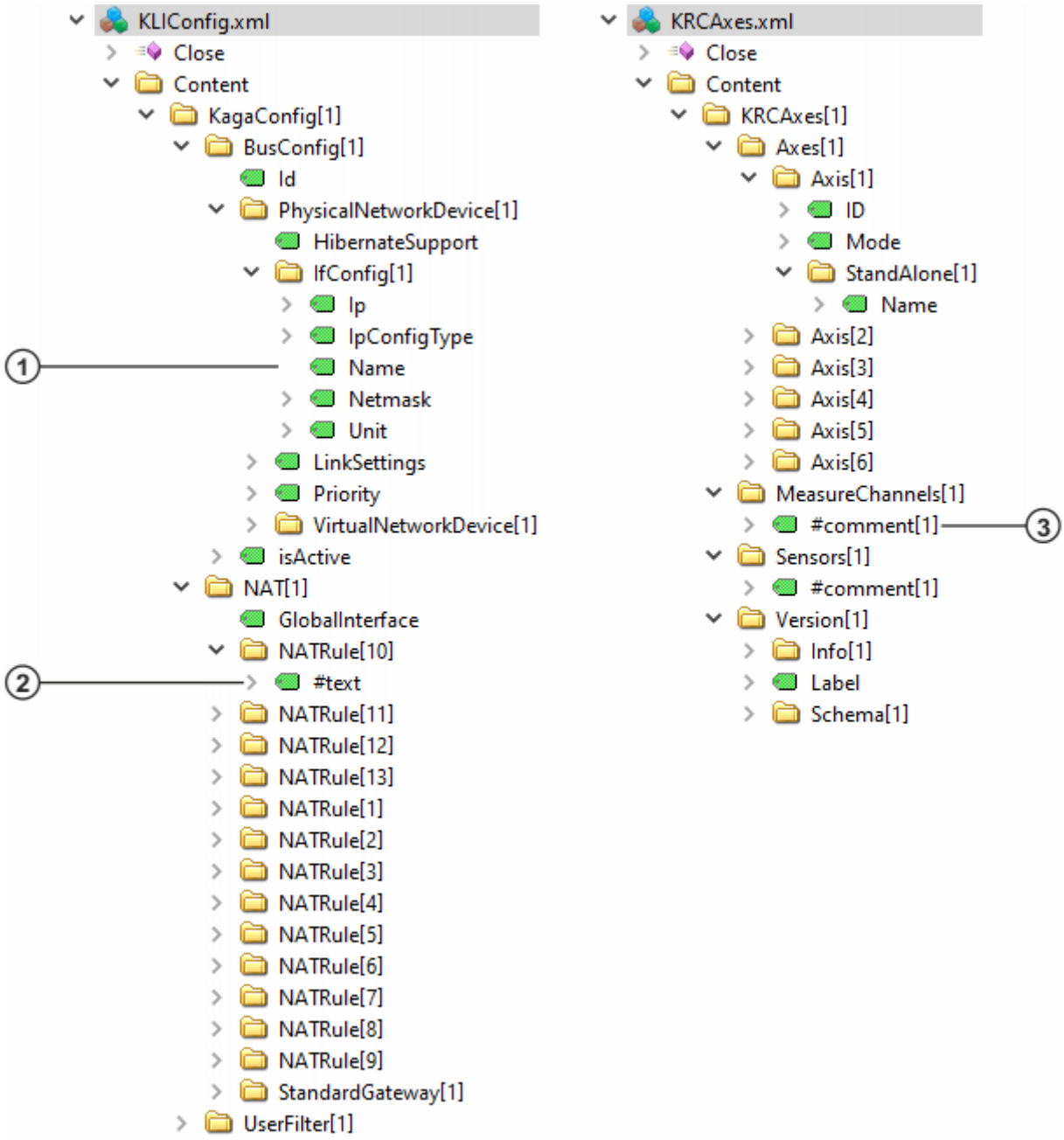


Fig. 5-7: Content of XML files

1	Attribute	2	Text element
3	Comment		

File operations

The OPC UA client provides various methods for file access:

- **Open**

  Opens a file for further file operations.

  The access mode must be specified when opening a file:

  - **1**: File is opened with read access.
  - **2**: File is opened with write access.
  - **3**: File is opened with read and write access.

  The return value is a handle number that is required for all further file operations.

- **SetPosition**

  Defines the file position from which a subsequent file operation is to be carried out.
- **GetPosition**

  Specifies the current file position. If no other file position is defined, a subsequent file operation will be performed from this position.
- **Read**

  Reads the part of a file starting at the current file position. The file position jumps forwards by the number of bytes read.
- **Write**

  Writes the part of a file starting at the current file position. The file position jumps forwards by the number of bytes written.
- **Close**

  Closes an open file. On closing, the handle number becomes invalid.

Precondition

- Read: User group OpcUaObserver
- Write: User group OpcUaOperator

Procedure

1. Open the file to read or write.
   a. Right-click on **Open** and call the method with **Call...**.
   b. Transfer the desired access mode to the method and click on **Call**. The method provides the handle number for further file operations.
   c. Close the method.
2. If necessary, define the file position from which a subsequent file operation is to be carried out with **SetPosition**.
3. Read and/or write the file with **Read** and/or **Write**.
4. Repeat steps 2 and 3 as often as required.
5. Close the file.

5.6 Messages and events

These data are available with the following option packages:
- KUKA.DeviceConnector Messages
- KUKA.DeviceConnector Advanced

Description

All messages and the entire Windows event log of the robot controller can be displayed in the **Event View** of the OPC UA client.

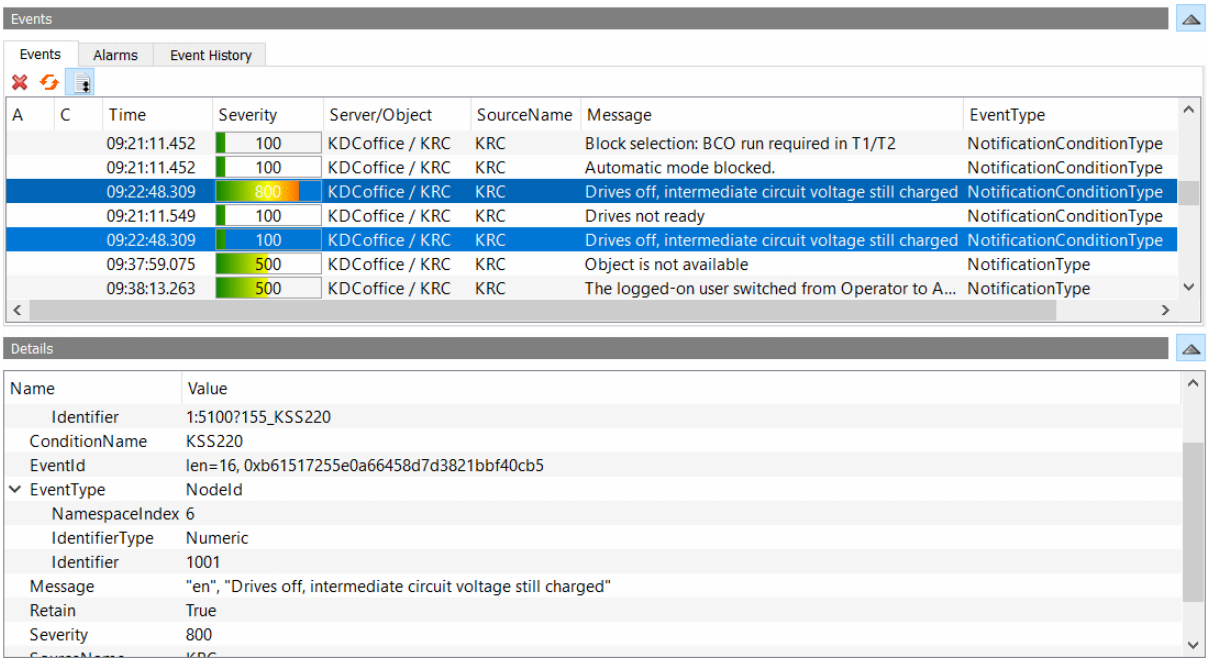


Fig. 5-8: Messages and events

The messages and events in the **Event View** are updated continuously. The message text is always English. Additionally, specific details are displayed for each message/event. For the error messages of the robot controller, for example, the error number or the Retain value:

- Retain = True when the event occurs
- Retain = False if the event is canceled, e.g. after error acknowledgement

Procedure

- To display the messages and events in the client, drag the KRC node to the **Configuration** area of an **Event View**.

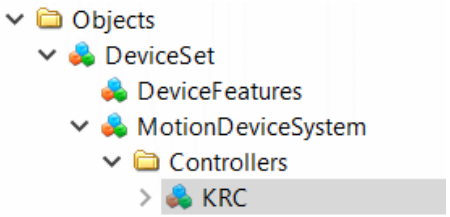


Fig. 5-9: KRC node

- To open an **Event View**, select **Document** > **Add...** in the Client menu and add the **document type Event View** with **Add**.

5.7 Process data

These data are available with the following option packages:

- KUKA.DeviceConnector SystemVars

  The SystemVars option also contains the customer-specific variables (CustomerData).
- KUKA.DeviceConnector Advanced

Description

The OPC UA client can access all system variables and global KRL variables. The variables can be write-protected and only readable or both readable and writable.

All relevant directories are available under the **ProcessData** node.

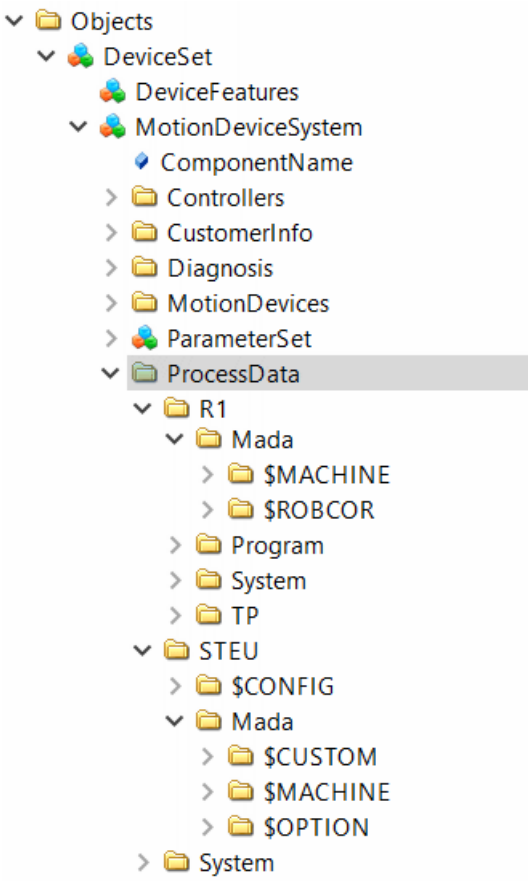


Fig. 5-10: Process data

Precondition

- Read: User group OpcUaObserver
- Write: User group OpcUaOperator

Procedure

Modifying a variable value:

- Select the variable node in the data tree and enter the new value directly under **Value** in the **Attributes** window.
- Or: Drag the variable node into a **Data Access View** and enter the new value under **Value**.

5.8 Customer-specific variables (CustomerData)

These data are available with the following option packages:

- KUKA.DeviceConnector CustomerData
- KUKA.DeviceConnector SystemVars
- KUKA.DeviceConnector Advanced

The OPC UA client can access all customer-specific variables. In the SystemVars and Advanced options, these variables are automatically enabled.

If the CustomerData option is used, the files in which the customer-specific variables are defined must be stored in the following directory on the robot controller:

- …\KRC\Roboter\KRC\R1\CUSTOMER\

The user must create the CUSTOMER folder on the robot controller.

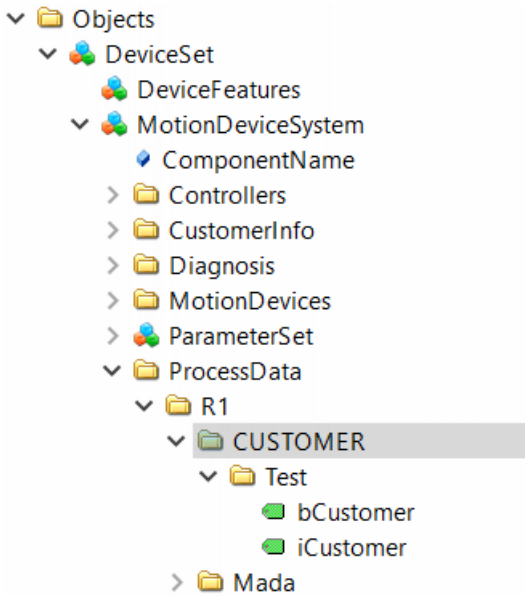


Fig. 5-11: CustomerData

5.9 Customer-specific information (CustomerInfo)

This functionality is available with the following option packages:
- KUKA.DeviceConnector Advanced

Description

The OPC UA client can save any amount of static information on the server. Using the OPC UA client, the user can create the variable nodes with the desired information and delete them again as required.

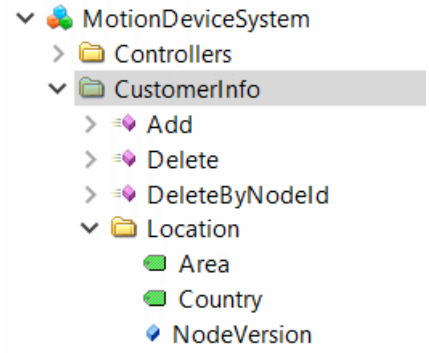


Fig. 5-12: Customer-specific information

The OPC UA client uses the following methods for this:

- **Add**
  Method for adding a new node including possible hierarchy levels

- **Delete**
  Method for deleting a node or hierarchy level using the browse name
- **DeleteByNodeId** (for advanced client users)
  Method for deleting a node or hierarchy level using the OPC UA node ID

Variable values of existing nodes can be modified subsequently.

Precondition

- User group OpcUaOperator

Procedure

Adding a node:

1. Right-click on **Add** and call the method with **Call...**.
2. Transfer the following input parameters to the method:
   - **Path**: Path to variable node
     Using the path, any number of levels (separated by the “/” symbol) can be added under the CustomerInfo node.
     If no path is specified, the variable node is created directly under the CustomerInfo node.
   - **Name**: Browse name of the variable
   - **Value**: Value of the variable
3. Select **Call**.
4. Close the method with **Close**.

Modifying a variable value:

- Select the variable node in the data tree and enter the new value directly under **Value** in the **Attributes** window.
- Or: Drag the variable node into a **Data Access View** and enter the new value under **Value**.

Deleting nodes:

1. Right-click on **Delete** and call the method with **Call...**.
2. Transfer the input parameters **Path** and **Name** to the method:
   - In order to delete a specific variable node, enter the browse name of the variable and the corresponding path to the variable node (separated by the “/” symbol).
   - If the node to be deleted is located directly below the CustomerInfo node, it is not necessary to enter a path.
   - If only a path is entered, and no name, all variables and levels located below the level specified in **Path** are deleted.
3. Select **Call**.
4. Close the method with **Close**.

Deleting a node using the node ID:

1. Right-click on **DeleteByNodeId** and call the method with **Call...**.
2. Transfer the node ID of the node that is to be deleted to the method as the input parameter:
   - "NameSpaceIndex"
   - "IdentifierType"
   - "Identifier"
3. Select **Call**.
4. Close the method with **Close**.

5.10 Soft PLC data

These data are available with the following option packages:
- KUKA.DeviceConnector SoftPLC
- KUKA.DeviceConnector Advanced
- Precondition: KUKA.PLC ProConOS or KUKA.PLC ProConOS ltd is installed.

Description

All Soft PLC data can be found under the PLC1 node. The corresponding global variables can be read and written.

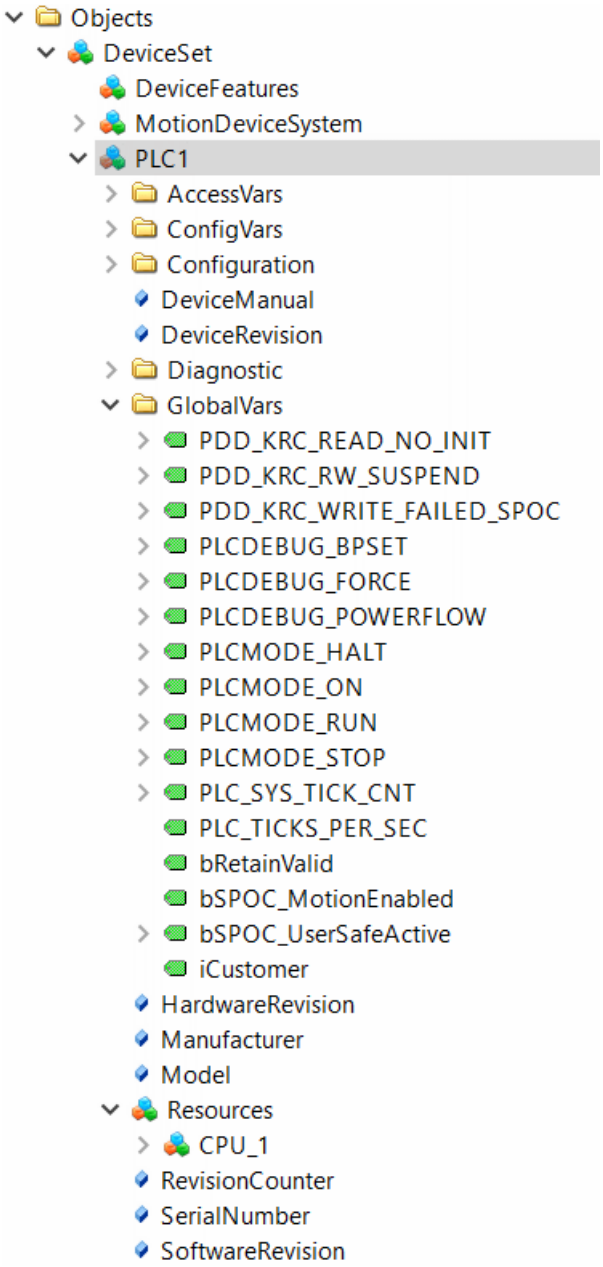


Fig. 5-13: Soft PLC data

Further information about the data can be found in the following document from the PLCopen and OPC Foundation:
- PLCopen OPC UA Information Model for IEC 61131-3

Precondition

- Read: User group OpcUaObserver
- Write: User group OpcUaOperator

Procedure

Modifying a variable value:

- Select the variable node in the data tree and enter the new value directly under **Value** in the **Attributes** window.
- Or: Drag the variable node into a **Data Access View** and enter the new value under **Value**.

5.11 PROFINET data

These data are available with the following option packages:
- KUKA.DeviceConnector Advanced
- Precondition: KUKA.PROFINET® -/S or KUKA.PROFINET® M/S is installed.

The basic information about the PROFINET periphery can be found under the PROFINET® node, e.g.:

- Device name
- IP address
- I&M data
- Energy consumption

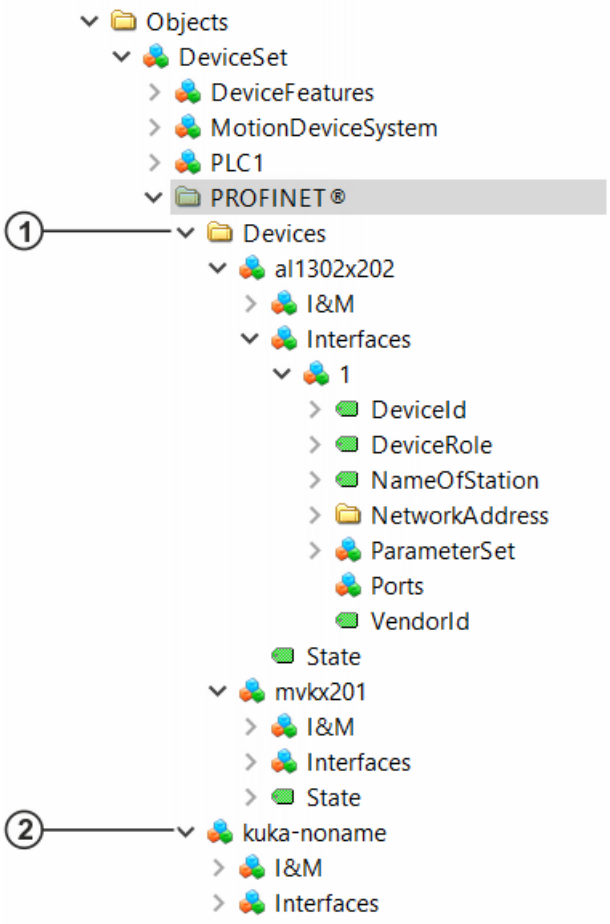


Fig. 5-14: PROFINET data

Item	Description
1	The connected PROFINET devices and their data are displayed under **Devices**.
2	This node is only available if the robot controller itself is configured as a PROFINET device.

Further information about the data can be found in the following documents from the OPC Foundation:

- OPC 30140: OPC UA for PROFINET Companion Specification
- OPC 30141: OPC UA for PROFIenergy

5.12 Diagnostic data

These data are available with the following option packages:

- KUKA.DeviceConnector Advanced

The diagnostic data, i.e. all data of the diagnostic monitor and log events, can be found under the **Diagnosis** node. The diagnostic data can be read.

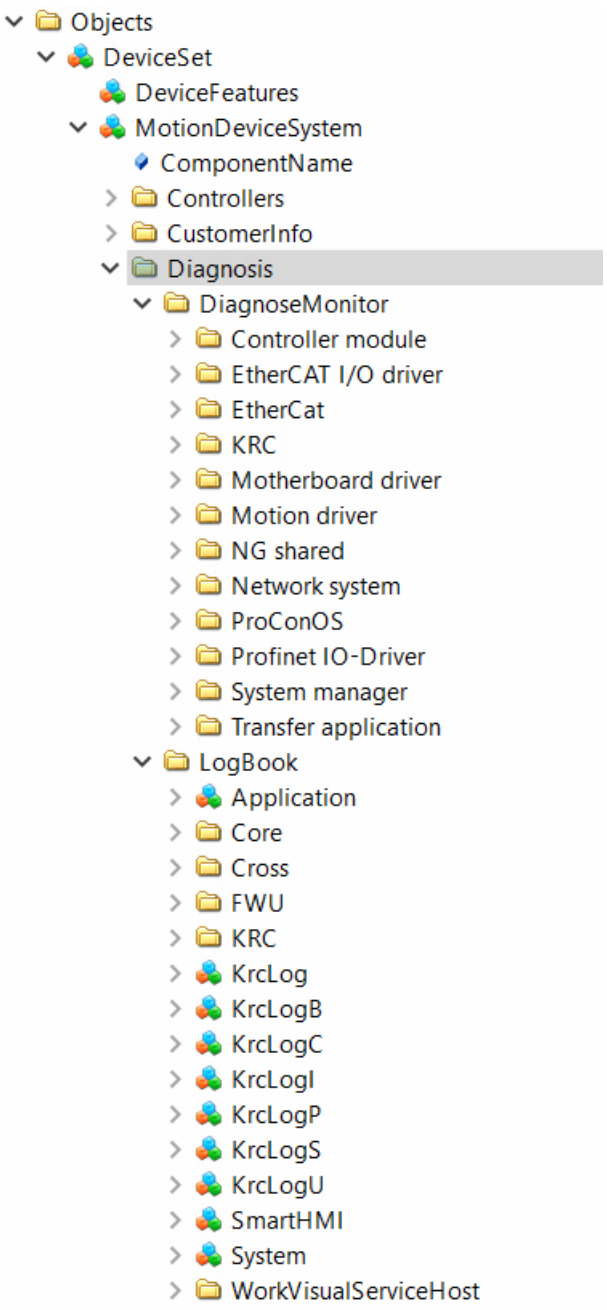


Fig. 5-15: Diagnostic data

5.13 Node IDs of KUKA-specific namespaces

Separate namespaces are defined for KUKA.DeviceConnector. These are:

- http://kuka.com/UA/Management/
- http://kuka.com/UA/Robotics/

The OPC UA nodes from this namespace have fixed node IDs that are linked to the browse path in the OPC UA tree. Fixed node IDs from the namespace http://kuka.com/UA/Robotics/ include:

- 5100: Root/Objects/DeviceSet/MotionDeviceSystem/Controllers/KRC
- 5210: Root/Objects/DeviceSet/MotionDeviceSystem/Controllers/KRC/FileSystem/Config/User

The node set definitions for the KUKA-specific namespaces can be found on the data storage medium with the installation files of the DeviceConnector:

- Directory .\DOC\nodesets\

6 Configuration

6.1 Changing the initial password for OPC UA users

Description

When starting up the system, it is recommended to change the initial password for the following OPC UA users:

- OpcUaObserver
- OpcUaOperator
- OpcUaConfigAdmin
- OpcUaSecurityAdmin

The initial password is “kuka” in each case.

Changing the initial password
If the initial password is not changed, this allows unauthorized persons to log on to the OPC UA client.

- Change the initial password during start-up of the system.
- Communicate the new password to authorized persons only.

If the changed password is lost, access to the Windows system is no longer possible – not even for KUKA.

Precondition

- Administrator access rights
- Windows interface
- Mouse and keyboard

Procedure

1. Press the Windows key and the R key simultaneously. The **Run…** window opens.
2. Enter the `cmd` command in the **Open** box and confirm with the Enter key. The command window opens.
3. Enter the following command (example with OpcUaOperator):
   ```
   c:\krc\util\krcuserpw\changepwd.exe /u="OpcUaOperator" /op="OLD_PW" /p="NEW_PW" /cp
   ```
   Here, enter the current password instead of `OLD_PW` and the desired new password instead of `NEW_PW`.
4. Confirm by pressing the Enter key.

The initial password is changed. There is no confirmation message. The change is effective immediately.

Parameter

Parameter	Description
/u= "…"	User name
/op= "…"	Current password Upper and lower case are taken into consideration.
/p= "…"	New password Upper and lower case are taken into consideration.
/cp	“Change password” command

A password must not contain quotation marks followed by another special character or space. The following examples are thus NOT possible:

- My"&password
- My" password

Log file

The change is logged in:

- C:\KRC\ROBOTER\LOG_ChangePwd.log

The errors are also indicated here in plain text. Password changes via WorkVisual are logged here, too.

6.2 Creating a self-signed certificate for authentication

Description

As an alternative to authentication with user name and password, the user can create a self-signed certificate. This enables the use of OPC UA clients that do not support logon with password.

Precondition

- An external program for creating certificates, e.g. XCA, is available.

Procedure

1. Open the program for creating certificates and start creation.
2. Enter data of the OPC UA user.
3. When defining the certificate as **OPC UA user authentication**, at least one of the two permissible role attributes must be specified:
   - Role: Opc:operator
   - Role: Opc:observer
4. Create certification key.

6.3 Adding and updating certificates

Description

By default, the OPC UA server automatically accepts each certificate of a client that establishes a connection to the server.

- It is possible to configure that only clients whose certificates are saved in a list of trusted certificates can be connected.
- KUKA.DeviceConnector also provides an update method which can be used to create and load a new certificate for the OPC UA server.

Precondition

- User group OpcUaSecurityAdmin

Procedure

1. Establish connection to the OPC UA server via an encrypted connection.

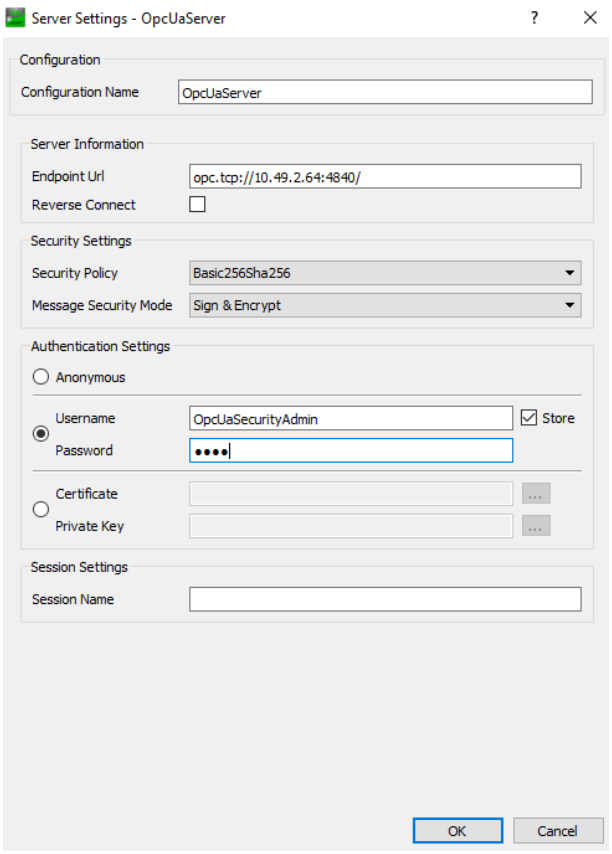


Fig. 6-1: Establishing an encrypted connection

2. Log in with user name OpcUaSecurityAdmin and the password. The **ServerConfiguration** node is displayed under the server object in the OPC UA tree.

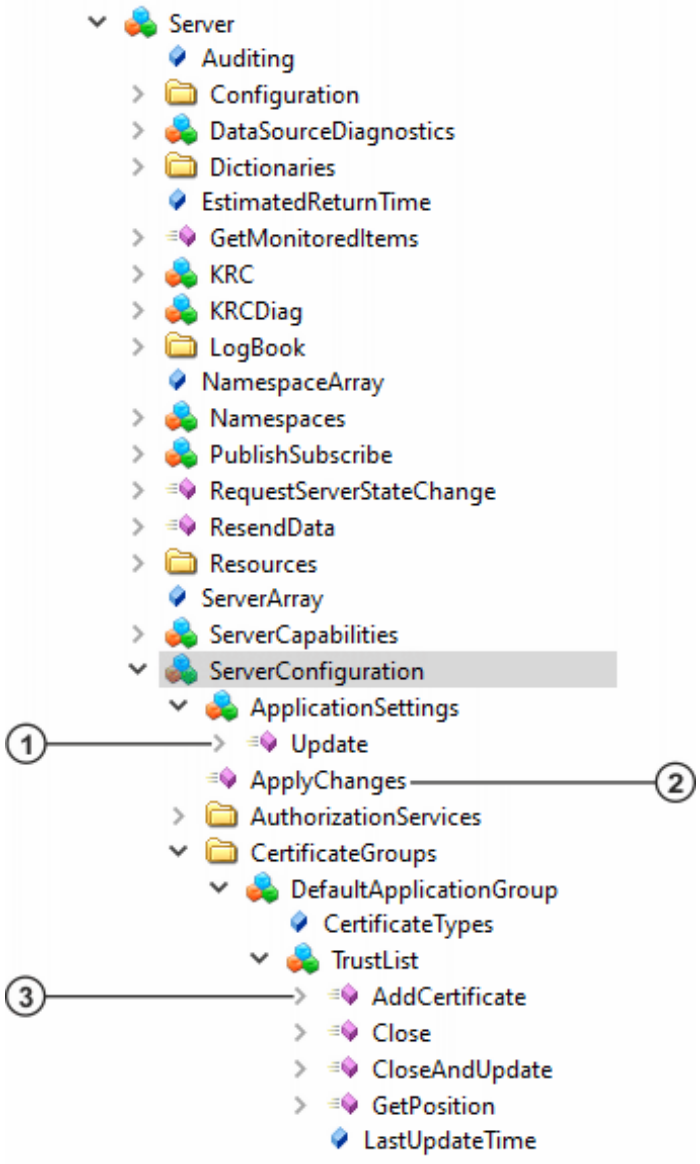


Fig. 6-2: Adding and updating certificates

1 Update certificate of OPC UA server
2 Apply changes in server configuration
3 Add certificate to list of trusted certificates

3. Certificates can be added, removed and updated under the **Server-Configuration** node:
   - Call **AddCertificate** under CertificateGroups/DefaultApplicationGroup/TrustList to add a client certificate to the list of trusted certificates.
   - Call **RemoveCertificate** under CertificateGroups/DefaultApplicationGroup/TrustList to remove a client certificate from the list of trusted certificates.
   - Call **Update** under ApplicationSettings to load a new certificate for the OPC UA server.

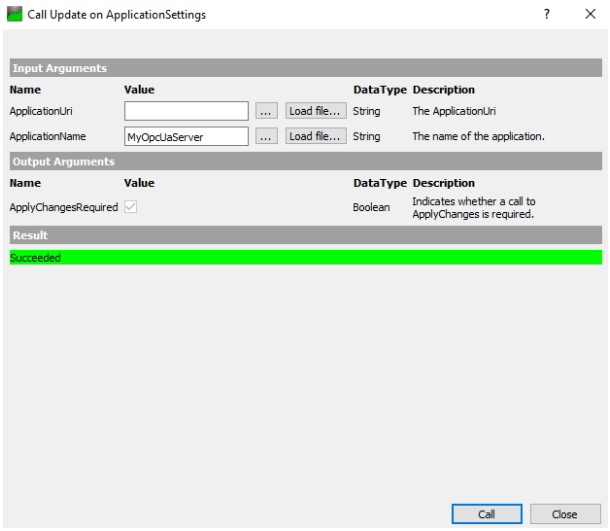


Fig. 6-3: Update ApplicationSettings

This method can be used to set any application name and application URI. If only one of the parameters is set, the other parameter value is retained.

4. To save the changes, call **ApplyChanges**. The OPC UA server is restarted.

6.4 MQTT configuration

Description

The MQTT protocol is an open messaging protocol that enables communication with a cloud. Telemetry data are exchanged between the devices and the cloud in the form of messages.

- Connections across network boundaries are possible.
- Data transmission is unidirectional.
- Ports 1883 and 8883 are reserved for MQTT.

Configuration file

Directory	C:\KRC\ROBOTER\Config\User\Common\opcua
File	*.pscj

The MQTT Publisher can be configured in JSON format. The configuration must be saved in a file with the extension *.pscj in the specified directory.

Every file with the extension *.pscj is read. Changes, new files or deleted files are automatically updated in the object model with a delay of 5 seconds.

Each configuration file contains a JSON object with the following fields:

- “ConnectionParameter”
  (>>> *6.4.1 "ConnectionParameter" Page 44*)
- List of “DataSets”
  (>>> *6.4.2 "DataSets" Page 47*)
- Optional list of “Namespaces”
  (>>> *6.4.3 "Namespaces" Page 59*)

```
{
 "Namespaces": [
```

```
    "http://opcfoundation.org/UA/DI/",
    "http://opcfoundation.org/UA/Machinery/",
    "http://opcfoundation.org/UA/Robotics/",
    "http://kuka.com/UA/Robotics/",
    …
  ],
  "ConnectionParameter": {
    "Address": "mqtts://192.168.46.158:8883",
    …
  },
  "DataSets": [
    {…},
    {…}
  ]
}
```

Placeholder

Various types of placeholders can be used in the configuration:

- Placeholders that allow the selection of specific variables from the AddressSpace without knowing the exact structure of the model. For example, the positions of all axes of all MotionDevices can be defined, irrespective of how many axes the system contains.
  - MotionDevices/+/Axes/+/ParameterSet/ActualPosition

  (>>> *6.4.2.1 "DataPoints" Page 50*)
- Placeholders that are replaced by the current value during initialization. The following placeholders are currently defined:
  - %HOSTNAME%
  - %ROBOTSERIALNO%

  The following properties in the configuration can use these placeholders:
  - “PublisherId”
  - “Topic”
  - “MqttClientId”

Example

(>>> *6.4.4 "Configuration example" Page 60*)

6.4.1 ConnectionParameter

Description

The ConnectionParameter is a JSON object that can be used for complete definition of a specific MQTT connection. The JSON object has the following properties:

Key	Use	Description
“Address”	Required	Address of the MQTT broker (URL or KRL variable) The URL is specified as follows and stored permanently in the configuration: • “mqtts://<*domain name*>[:<*port*>]” Default port: 8883 • “mqtt://<*domain name*>[:<*port*>]” Default port: 1883 Instead of specifying the URL, it is possible to refer to a KRL variable containing the address, e.g.: • “KRL:<*path*>/<*variable name*>[]” In this way, the same configuration can always be used, even if different MQTT brokers are implemented.
“ConnectionName”	Required	Name of the connection The name “KUKAConnect” is reserved and cannot be used.
“PublisherId”	Required	Publisher ID It is advisable to use a placeholder, e.g. “%HOSTNAME%”, so that the Publisher ID automatically becomes unique for multiple robots.
“UserName”	Optional	User name for authentication of the MQTT broker
“Password”	Optional	Password for authentication of the MQTT broker
“Enabled”	Optional	Specifies whether connection is enabled or not • **true** = connection is enabled • **false** = connection is disabled A disabled connection will be ignored. Default: **true** “EnabledKrlVar” can be used instead of “Enabled”. “EnabledKrlVar” makes it possible to control the connection to the MQTT broker via a KRL variable.
“EnabledKrlVar”	Optional	Name of the KRL variable used to enable and disable the connection If “EnabledKrlVar” is defined, “Enabled” will be ignored.
“ProtocolVersion”	Optional	Version of the MQTT protocol Possible values: • “V310”, “V311”, “V500” Default: “V311”
“TlsProtocolVersion”	Optional	Version of the TLS protocol Possible values: • “Tls”, “Tls11”, “Tls12”, “Tls13” Default: “Tls12”

Key	Use	Description
“TrustedCertificates”	Optional	List of trusted SSL broker certificates from MQTT brokers The certificate is used via the “fingerprint” in hex format without spaces. Multiple fingerprints can be specified separated by commas, e.g.: • “ce4a6792d4a4f299a5833960a225f7cb61887fc5, 18e9bb0237e894c2f1853edf3ff8de107905a921, fde9bafef0a33edb20ee3d05d73a26a90af751bd”
“PreferredLanguage”	Optional	Preferred language in which language-specific contents are transferred, e.g. message texts Possible values: • “auto” for the language set on the smartHMI This value is only updated if the MQTT configuration is updated. • ISO language code for preferred language, e.g. “zh-CN”, “en-US”, “de-DE”, etc. Default: “en-US”
“MessageMapping”	Optional	Message mapping for the MQTT broker connection Possible values: • “Json” JSON format: “http://opcfoundation.org/UAProfile/Transport/pubsub-mqtt-json” • “Jzip” Compressed JSON format: “http://opcfoundation.org/UAProfile/Transport/pubsub-mqtt-jzip” Default: “Json”
“MqttClientId”	Optional	MQTT client ID for the MQTT broker connection

Examples

Connection via URL:

```
"ConnectionParameter": {
 "Address": "mqtts://192.168.46.158:8883",
 "PublisherId": "%HOSTNAME%"
 "ConnectionName": "MQTTBroker"
 "Enabled": true
 "ProtocolVersion": "V311",
 "TlsProtocolVersion": "Tls12",
 "PreferredLanguage": "en-US"
}
```

Connection via KRL variables:

```
"ConnectionParameter": {
 "Address": "KRL:/R1/SYSTEM/MQTTCONFIG/KUKACONNECT_BROKERIP[]",
 "PublisherId": "%HOSTNAME%"
 "ConnectionName": "MyMQTT"
 "EnabledKrlVar": "KRL:/R1/SYSTEM/MQTTCONFIG/KUKACONNECT_ENABLED"
}
```

6.4.2 DataSets

Description

The data sets to be transferred are defined in the DataSets array. A data set can be created from a list of variables, events or a file content.

A data set is coded in a DataSetMessage for transfer. One or more DataSetMessages are grouped together for transfer in a NetworkMessage.

A data set is a JSON object with the following properties:

Key	Use	Description
“Name”	Required	A unique name of the PublishedDataSet object in the AddressSpace. The name must be unique for all DataSets in the OPC UA model.
“Description”	Optional	Description for the DataSet The description is included in the OPC UA model.
“Topic”	Required	A unique name for the MQTT topic Each topic of a KRC-based system should begin with “KUKA/KRC/”. This is followed by a unique identifier for the system. This can be the host name or the serial number (robot/cabinet). Example: • “KUKA/KRC/%HOSTNAME%/operational-data” This parameter defines the queue name in the associated WriterGroup.
“Type”	Required	Type of DataSet Possible values: • “DataItems” for a list of variables (>>> *"Example" Page 49*) • “Events” for events (>>> *6.4.2.2 "Events DataSet" Page 53*) • “HistoricalEvents” for historical events (>>> *6.4.2.2 "Events DataSet" Page 53*) • “Files” for file content (>>> *6.4.2.3 "Files DataSet" Page 55*) • “TriggeredFiles” for KRCDiags and traces (>>> *6.4.2.4 "TriggeredFiles DataSet" Page 57*)
“TriggerName”	Optional	“PublishedName” of a DataPoint in the same DataSet (>>> *6.4.2.1 "DataPoints" Page 50*) This DataPoint is used as a trigger. If the value of the triggering DataPoint changes, all other variables are read from this DataSet.

Key	Use	Description
“PublishingInterval”	Required	Publication interval in milliseconds for NetworkMessages and embedded DataSetMessages The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” This parameter defines the “PublishingInterval” in the associated WriterGroup.
“KeyFrameCount”	Optional	Multiplier of the publication interval Defines the number of elapsed publication intervals after which a KeyFrame message is sent. • **0** = no KeyFrame message is sent Default: **0**
“DefaultSamplingInterval”	Optional	Only relevant for DataSets of type “DataItems” Default value for the sampling interval of all DataPoints in a DataSet in milliseconds The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S”
“FieldContentMask”	Optional	Format of an individual field in a DataSetMessage The value is defined as a combination of the individual flag names. The individual values can be separated by means of ‘,’ or ‘|’; spaces are ignored. Alternatively, the corresponding numeric value can also be used. Examples: • “StatusCode|SourceTimestamp” • “Rawdata” • “3” or “32” This parameter defines the DataSetFieldContentMask in the associated Writer.
“MessageContentMask”	Optional	Format of the DataSetMessage header The value is defined as a combination of the individual flag names. The individual values can be separated by means of ‘,’ or ‘|’. Spaces are ignored. Alternatively, the corresponding numeric value can also be used. Example: • “DataSetWriterId|SequenceNumber” = “5” This parameter defines the MessageSettings.DataSetMessageContentMask in the associated Writer.

Key	Use	Description
“NetworkMessage-ContentMask”	Optional	Format of the NetworkMessage The value is defined as a combination of the individual flag names. The individual values can be separated by means of ‘,’ or ‘|’. Spaces are ignored. Alternatively, the corresponding numeric value can be used. Example: • “NetworkMessageHeader|DataSetMessageHeader|PublisherId” = “11” This parameter defines the MessageSettings.NetworkMessageContentMask in the associated WriterGroup.
“DataPoints”	Optional	Only relevant for DataSets of type “DataItems” Array of DataPoints Each DataPoint describes a variable that is to be monitored. (>>> *6.4.2.1 "DataPoints" Page 50*)
“Enabled”	Optional	Specifies whether the DataSet is activated. • **true** = DataSet is activated • **false** = DataSet is deactivated A deactivated DataSet will be ignored. Default: **true**
“RequestedDelivery-Guarantee”	Optional	Quality of Service (QoS) Possible values: • “AtMostOnce” Message is sent not at all, once, or more than once (QoS 0). • “AtLeastOnce” Message is sent at least once or more often (QoS 1). • “ExactlyOnce” Message is sent exactly once (QoS 2) Default: “AtMostOnce”

Example

DataSet of type “DataItems”:

```
"DataSets": [
 {
  "Name": "operational-data",
  "Enabled": true,
  "Topic": "KUKA/KRC/%HOSTNAME%/operational-data",
  "Type": "DataItems",
  "PublishingInterval": 5000,
  "DefaultSamplingInterval": 1000,
  "KeyFrameCount": 12,
  "FieldContentMask": "StatusCode|SourceTimestamp",
  "MessageContentMask": "DataSetWriterId|SequenceNumber",
```

```
    "NetworkMessageContentMask": "NetworkMessageHeader|DataSetMessageHeader",
    "RequestedDeliveryGuarantee": "ExactlyOnce"
    "DataPoints": [
      {…},
      {…}
    ]
  },
    {…}
]
```

6.4.2.1 DataPoints

Description

The following keys can be defined in the DataPoints array:

Key	Use	Description
“PublishedName”	Required	Name of the field in the DataSetMessage The name must be unique within the DataSet. If placeholders are used in the “Source” definition, the DataSet may contain multiple variables. In this case, placeholders must also be used in the definition of “PublishedName” to enable unique names: • ‘#’ is replaced by a sequential number. May be used once as a placeholder. • ‘{n}’ is replaced by the nth placeholder from the “Source” definition. To be used once for each ‘+’ placeholder from the “Source” definition. The placeholders are counted starting from “1”, e.g.: • “ActualPosition #” -> ActualPosition 1, ActualPosition 2 • “ActPos {1} {2}” -> ActPos ROBOT A1, ActPos ROBOT A2 • “ActPos_{2}” -> ActPos_A1, ActPos_A2

Key	Use	Description
“Source”	Required	Browse path to the variable that is inserted into the DataSet MotionDeviceSystem is defined as the root object of the browse path. Only variables of objects below this root object can be addressed. The root object is therefore not specified in order to define the browse path, e.g.: • *“MotionDevices/Robot/Axes/A1/ParameterSet/ActualPosition”* Within this path, individual directories can be replaced by ‘+’ as a placeholder. During evaluation, the placeholder is replaced by all names for which the path can be continued to the end. The MotionDevices object contains a folder **Axes** that contains information about all axes of a kinematic system. In order to adopt the current position of all axes in the DataSet, the following BrowsePath is sufficient: • “MotionDevices/Robot/Axes/+/ParameterSet/ActualPosition” A variable is added for each axis, regardless of how many axes the robot has. In order to include the current position of all axes of all kinematic systems in the DataSet, a placeholder for the kinematic name can be used: • “MotionDevices/+/Axes/+/ParameterSet/ActualPosition” If placeholders are used in the “Source” definition, placeholders must also be used in the definition of “PublishedName” to enable unique names. The namespace index for each segment can be specified in the path definition, e.g.: • 3:MotionDevices/+/3:Axes/+/1:ParameterSet/3:ActualPosition (>>> *6.4.3 "Namespaces" Page 59*) If no namespace index is used, the segment is compared with both the browse name and the display name of all nodes at the corresponding level. Upper and lower case are not taken into consideration. The first match to be found is used. The order of the compared nodes is not defined.
“SamplingInterval”	Optional	Sampling interval in milliseconds The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” Default: **0**

Examples

Path definition without namespace index:

```
"DataPoints": [
  {
   "PublishedName": "ActualPosition #",
   "Source": "MotionDevices/+/Axes/+/ParameterSet/ActualPosition",
   "SamplingInterval": 100
  },
  {…},
  {…}
]
```

Path definition with namespace index:
(>>> *6.4.3 "Namespaces" Page 59*)

6.4.2.2 Events DataSet

Description

For DataSets of type “Events” or “HistoricalEvents”, only the following keys are relevant:

Key	Use	Description
“EventSourceName”	Required	Browse path to the source of the event MotionDeviceSystem is defined as the root object of the browse path. Only events of objects below this root object can be addressed. The root object is therefore not specified in order to define the browse path, e.g.: • “Diagnosis/LogBook/KrcLogP” The namespace index for each segment can be specified in the path definition. (>>> *6.4.3 "Namespaces" Page 59*)
“DaysToPast”	Optional	Only required for “HistoricalEvents” Makes past events available for the number of days defined here
“PublishingInterval”	Required	Publication interval in milliseconds for NetworkMessages and embedded DataSetMessages The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” This parameter defines the PublishingInterval in the associated WriterGroup.

Key	Use	Description
“MessageContent-Mask”	Optional	Format of the DataSetMessage header The value is defined as a combination of the individual flag names. The individual values can be separated by means of ‘,’ or ‘|’. Spaces are ignored. Alternatively, the corresponding numeric value can also be used. Example: • “DataSetWriterId|SequenceNumber” = “5” This parameter defines the MessageSettings.DataSetMessageContentMask in the associated Writer.
“NetworkMessage-ContentMask”	Optional	Format of the NetworkMessage The value is defined as a combination of the individual flag names. The individual values can be separated by means of ‘,’ or ‘|’. Spaces are ignored. Alternatively, the corresponding numeric value can be used. Example: • “NetworkMessageHeader|DataSetMessageHeader|PublisherId” = “11” This parameter defines the MessageSettings.NetworkMessageContentMask in the associated WriterGroup.

Examples

DataSet of type “Events”:

```
"DataSets": [
 {
  "Name": "system-messages",
  "Enabled": true,
  "Topic": "KUKA/KRC/%HOSTNAME%/system-messages",
  "Type": "Events",
  "EventSourceName": "Diagnosis/LogBook/KrcLogP",
  "PublishingInterval": 1000,
  "MessageContentMask": "DataSetWriterId|SequenceNumber",
  "NetworkMessageContentMask": "NetworkMessageHeader|DataSetMessageHeader",
 }
]
```

DataSet of type “HistoricalEvents”:

```
"DataSets": [
 {
  "Name": "EventLogsKrcLogC",
  "Enabled": true,
  "Topic": "KUKA/KRC/%ROBOTSERIALNO%/event-logs/KrcLogC",
  "Type": "HistoricalEvents",
  "EventSourceName": "Diagnosis/LogBook/KrcLogC",
  "PublishingInterval": 1000,
  "DaysToPast": 1,
  "MessageContentMask": "DataSetWriterId|SequenceNumber",
  "NetworkMessageContentMask": "NetworkMessageHeader|DataSetMessageHeader",
 }
```

```
]
```

The FieldContentMask is always set internally to RawValue. The SamplingInterval, the KeyFrameCount and the DataPoints have no meaningful significance for events.

6.4.2.3 Files DataSet

Description

The following keys can be defined for DataSets of type “Files”:

Key	Use	Description
“FileName”	Required	Name of the file to be transferred The name may contain the placeholders ‘*’ and ‘?’. Examples: • “System*.log” • “StartKRC.log”
“Directory”	Required	Browse path to the folder in which the file is located MotionDeviceSystem/Controllers/KRC/FileSystem is defined as the root object of the browse path. Only nodes below this root object can be addressed. The root object is therefore not specified in order to define the browse path. Examples: • “KRCDiag” • “LogFiles/System” The namespace index for each segment can be specified in the path definition. (>>> *6.4.3 "Namespaces" Page 59*)
“HashAlgorithm”	Optional	Method used to generate a checksum via the file Possible values: • “MD5”, “SHA256”, “SHA384”, “SHA512” Default: “MD5”
“CompressStrategy”	Optional	Indicates whether the file contents are compressed for transfer Possible values: • “Auto” ZIP files are not compressed. • “Never” • “Always” Default: “Auto”

Key	Use	Description
“ChunkSize”	Optional	Maximum number of bytes that can be transferred together • **1024 … 16384** in steps of 1024 If the specified ChunkSize value is exceeded, the file is distributed over several MQTT blocks. Default: **16384**
“TransmissionInterval”	Optional	Minimum time interval between 2 file transfers in milliseconds The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” The interval is retained after a reboot of the system. Default: **0**
“TransmitOnlyChanges”	Optional	Specifies whether the file is transferred after a file manipulation • **true** = transfer after file manipulation The transfer is not carried out until the defined transmission interval has elapsed. • **false** = no transfer after file manipulation Default: **true**
“TransmitAtLeastOnce”	Optional	Specifies whether the file is transferred at least once, although it does not change and “TransmitOnlyChanges” has the value **true** • **true** = at least once • **false** = not at all Default: **false**
“TransmitOnlyNewestFile”	Optional	If “FileName” specifies multiple files, e.g. System*.log, only the newest file in this collection is transferred. • **true** = newest file only • **false** = all files Default: **false**

Example

```
"DataSets": [
  {
   "Name": "krcdiag",
   "Enabled": true,
   "Topic": "KUKA/KRC/%HOSTNAME%/file/KRCDiag",
   "Type": "Files",
   "MessageContentMask": "DataSetWriterId|SequenceNumber",
   "NetworkMessageContentMask": "NetworkMessageHeader|DataSetMessageHeader|
SingleDataSetMessage",
   "Directory": "KRCDiag"
   "FileName":"KRCDiag"_*.zip",
   "HashAlgorithm": "SHA256",
```

```
    "CompressStrategy": "Auto",
    "ChunkSize": 16384,
    "TransmissionInterval": "0:1:0",
    "TransmitAtLeastOnce": true,
    "TransmitOnlyChanges": true
  }
]
```

The FieldContentMask is always set internally to RawValue. The SamplingInterval, the KeyFrameCount and the DataPoints have no meaningful significance for events.

6.4.2.4 TriggeredFiles DataSet

Description

The following keys can be defined for DataSets of type “TriggeredFiles”:

Key	Use	Description
“StateMachinePath”	Required	Browse path to a “StateMachineType” under MotionDeviceSystem MotionDeviceSystem is defined as the root object of the browse path. Only nodes below this root object can be addressed. The root object is therefore not specified in order to define the browse path. Possible values: • “Controllers/KRC/KRCDiag” • “Controllers/KRC/Trace” The node must be available during initialization of the OPC UA server. The StateMachine must provide a “ResultTransitionEventType”. The namespace index for each segment can be specified in the path definition. (>>> *6.4.3 "Namespaces" Page 59*)
“TriggerMethodName”	Required	Name of the trigger method of the StateMachine that triggers file creation, e.g. “Start”
“InputArguments”	Optional	Array of input parameters for the trigger method
“HashAlgorithm”	Optional	Method used to generate a checksum via the file Possible values: • “MD5”, “SHA256”, “SHA384”, “SHA512” Default: “MD5”
“CompressStrategy”	Optional	Indicates whether the file contents are compressed for transfer Possible values: • “Auto” ZIP files are not compressed. • “Never” • “Always” Default: “Auto”

Key	Use	Description
“ChunkSize”	Optional	Maximum number of bytes that can be transferred together • **1024 … 16384** in steps of 1024 If the specified ChunkSize value is exceeded, the file is distributed over several MQTT blocks. Default: **16384**
“TriggerInterval”	Optional	Minimum time interval between 2 file transfers in milliseconds The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” The interval is retained after a reboot of the system. Default: **0**
“RetryCount”	Optional	Number of retry attempts if a call of the trigger method has failed A call can fail because the state machine is in an invalid state. Default: **0**
“RetryDelay”	Optional	Time in milliseconds between 2 retry attempts The value can also be specified as a string. The following formats are possible: • “hh:mm:ss.fff” • “P<n>DT<n>H<n>M<n.n>S” Default: 1 minute
“DeleteAfterTransfer”	Optional	Specifies whether the created file is deleted after transfer • **true** = file is deleted • **false** = file is not deleted Default: **false**

Example

```
"DataSets": [
  {
   "Name": "TraceSystemTest",
   "Enabled": false,
   "Topic": "KUKA/KRC/%HOSTNAME%/file/Trace",
   "Type": "TriggeredFiles",
   "MessageContentMask": "DataSetWriterId|SequenceNumber|Timestamp",
   "NetworkMessageContentMask": "NetworkMessageHeader|DataSetMessageHeader|
SingleDataSetMessage",
   "HashAlgorithm": "MD5",
   "CompressStrategy": "Auto",
   "ChunkSize": 16384,
   "TriggerInterval": "8:0:0",
   "StateMachinePath": "Controllers/KRC/Trace",
   "TriggerMethodName": "Start",
```

```
   "InputArguments": [
    "SystemTestTrace",
    "OpcUaTraceConfig_85.xml",
    "SystemTestTraceFolder"
    ],
   "RetryCount": 3,
   "RetryDelay": "0:0:30"
  }
]
```

The FieldContentMask is always set internally to RawValue. The SamplingInterval, the KeyFrameCount and the DataPoints have no meaningful significance for events.

6.4.3 Namespaces

Description

The data in the OPC UA tree may originate from different namespaces. These namespaces can be specified in the “Namespaces” array so that the data can be uniquely referenced using the associated namespace index.

- The namespaces can be specified in any order.
- The namespace index is derived from this order. The first namespace in the list contains the index 1, the second contains the index 2, etc.

The referencing can be used, for example, for unique definition of the individual segments of a browse path. The namespace index is specified for each segment, for example as follows:

- 3:MotionDevices/+/3:Axes/+/1:ParameterSet/3:ActualPosition

If no namespace index is used, the segment is compared with both the browse name and the display name of all nodes at the corresponding level. Upper and lower case are not taken into consideration. The first match to be found is used. The order of the compared nodes is not defined.

Example

Path definition with namespace index:

```
{
 "Namespaces": [
  "http://opcfoundation.org/UA/DI/",
  "http://opcfoundation.org/UA/Machinery/",
  "http://opcfoundation.org/UA/Robotics/",
  "http://kuka.com/UA/Robotics/",
  …
 ],
 "ConnectionParameter": {
  "Address": "mqtts://192.168.46.158:8883",
  …
 },
 "DataSets": [
  {
   "Name": "operational-data",
   "Enabled": true,
   "Topic": "KUKA/KRC/%HOSTNAME%/operational-data",
   …
   "DataPoints": [
   {
```

```
      "PublishedName": "ActualPosition_{1}_{2}",
      "Source": 3:MotionDevices/+/3:Axes/+/1:ParameterSet/3:ActualPosition
     },
    {…}
  ]
}
```

6.4.4 Configuration example

```
{
 "Namespaces": [
  "http://opcfoundation.org/UA/DI/",
  "http://opcfoundation.org/UA/Machinery/",
  "http://opcfoundation.org/UA/Robotics/",
  "http://kuka.com/UA/Robotics/",
  "http://kuka.com/UA/opcuaserver/robotics",
  "http://kuka.com/UA/Management/"
],
 "ConnectionParameter": {
  "Address": "mqtts://192.168.46.158:8883",
  "PublisherId": "%HOSTNAME%"
  "ConnectionName": "MQTTBroker"
  "Enabled": true
  "ProtocolVersion": "V311",
  "TlsProtocolVersion": "Tls12",
  "PreferredLanguage": "zh-CN"
 },
 "DataSets": [
  {
   "Name": "device-metadata",
   "Description": "Examples of source definitions using namespace indices",
   "Topic": "KUKA/KRC/%HOSTNAME%/device-metadata",
   "NetworkMessageContentMask": "11",
   "Type": "DataItems",
   "PublishingInterval": 5000,
   "DefaultSamplingInterval": 1000,
   "FieldContentMask": "3",
   "KeyFrameCount": 12,
   "MessageContentMask": "12",
   "DataPoints": [
   {
    "PublishedName": "RobotName",
    "Source": "4:ProcessData/5:System/5:$ROBNAME[]"
   },
   {
    "PublishedName": "ControllerUpTime",
    "Source": "3:Controllers/4:KRC/1:ParameterSet/4:ControllerUpTime"
   },
   ]
  },
  {
   "Name": "RobotProgram",
   "Description": "Examples of variables from the task control with complex
structures",
   "Topic": "KUKA/KRC/%HOSTNAME%/control-data",
   "NetworkMessageContentMask": "11",
   "Type": "DataItems",
   "TriggerName": "ProgramState
   "PublishingInterval": 5000,
   "DefaultSamplingInterval": 1000,
```

```
    "KeyFrameCount": 12,
    "DataPoints": [
    {
     "PublishedName": "ProgramState",
     "Source": "Controllers/KRC/TaskControls/Robot/ParameterSet/ProgramState",
     "SamplingInterval": 1500
    },
    {
     "PublishedName": "ExecutionMode",
     "Source": "Controllers/KRC/TaskControls/Robot/ParameterSet/ExecutionMode"
    },
    {
     "PublishedName": "Name",
     "Source": "Controllers/KRC/TaskControls/Robot/ParameterSet/TaskProgramName"
    },
    {
     "PublishedName": "InstructionCallerStack",
     "Source": "Controllers/KRC/TaskControls/Robot/ParameterSet/
InstructionCallerStack",
     "SamplingInterval": 500
    }
   ]
  },
   {
    "Name": "system-messages",
    "Description": "Examples of events from the controller",
    "Topic": "KUKA/KRC/%HOSTNAME%/system-messages",
    "Type": "Events",
    "PublishingInterval": 1000,
   },
   {
    "Name": "DeviceConnectorLogFiles",
    "Description": "Example of a file transfer",
    "Type": "Files",
    "Directory": "LogFiles/OPC_UA",
    "FileName": "KukaOpcUaService.log",
    "HashAlgorithm": "MD5",
    "CompressStrategy": "Auto",
    "ChunkSize": 16384,
    "TransmissionInterval": 60000,
    "TransmitAtLeastOnce": true,
    "TransmitOnlyChanges": true
   }
  ]
}
```

7 Troubleshooting

7.1 No connection to the OPC UA server

Error

The OPC UA client can no longer establish the connection to the OPC UA server on the robot controller.

Cause

Port 4840 of the KLI or KONI for OPC UA communication has been deactivated, e.g. by activating an earlier version of a WorkVisual project on the robot controller after installation of KUKA.DeviceConnector.

Solution

In order to be able to use the currently active project for OPC UA communication, the port must be enabled. The project must then be activated on the robot controller again via WorkVisual.

Precondition

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

Procedure

1. In the KLI or KONI network configuration, check whether port 4840 is present in the NAT list; if not, enable the port:
   a. In the main menu, select **Start-up** > **Network configuration**. The **Network configuration** window opens.
   b. Press **Advanced...**. The window for advanced network configuration opens.
   c. Select the **NAT** tab. A list of all the shared ports of the Windows interface is displayed in the **Available ports:** area.
   d. If the port is to be enabled:
      i. Press **Add port**. A new port with the number “0” is added to the list.
      ii. Enter 4840 in the **Port number:** box.
      iii. Select **tcp** in the **Permitted protocols:** box.
      iv. Press **Save**.

      A maximum total of 40 ports can be enabled.
   e. Close the **Network configuration** window using the Close icon.
   f. Only if modifications have been made:

      Reboot the robot controller so that the changes take effect. To do so, if PROFINET is used, select **Shutdown** in the main menu and select the option **Reload files**.
2. Load the active project from the robot controller in WorkVisual.
3. Transfer the project back from WorkVisual to the robot controller and activate it.

7.2 MQTT connection error

7.2.1 Incorrect target address

Error

No connection can be established to the MQTT broker.

Cause

A typical cause of an error in establishing the connection is an incorrect address.

Solution

The address under `ConnectionParameter` must match the target address of the MQTT broker.

The address entered must match the target address. This means, for example, that the encryption parameters must be correct, e.g. mqtt://<IP address> or mqtts://<IP address>.

Precondition

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

Procedure

1. Check in the code whether the address under `ConnectionParameter` matches the target address of the MQTT broker.

```
{
 "ConnectionParameter":
 {
  "Address": "mqtts://192.168.46.158:8883",
  PubisherId": "%HOSTNAME%"
  }
```

2. Correct the target address.
3. Load the active project from the robot controller in WorkVisual.
4. Transfer the project back from WorkVisual to the robot controller and activate it.

7.2.2 Invalid certificate

Error

No connection can be established to the MQTT broker.

Cause

One possible cause is the use of an invalid SSL certificate (e.g. because it has expired) by the MQTT broker. The problem also occurs if the communication protocol is not correct, e.g. *mqtt://<IP address>* instead of *mqtts://<IP address>*.

Solution

A valid SSL certificate must be issued for the MQTT broker.

The address entered must match the target address. This means, for example, that the encryption parameters must be correct, e.g. mqtt://<IP address> or mqtts://<IP address>.

Precondition

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

Information on issuing a valid SSL certificate can be found in the documentation of the MQTT broker.

8 Messages

8.1 Information about the messages

The “Messages” chapter contains selected messages. It does not cover all the messages displayed in the message window.

8.2 System messages from module: DeviceConnector

8.2.1 DeviceConnector 00001

Message code	DeviceConnector 00001
Message text	Unable to establish connection {0} to {1}.
Message type	Status message 
Possible cause(s)	**Cause:** URL address of the MQTT broker is incorrect (>>> Page 67) **Solution:** Adapting the address of the MQTT broker via the variable display (>>> Page 67) **Cause:** URL address of the MQTT broker is incorrect (>>> Page 68) **Solution:** Adapting the address of the MQTT broker via the configuration file (>>> Page 68)

Cause: URL address of the MQTT broker is incorrect

Description

If the URL address of the MQTT broker is specified incorrectly, it is not possible to establish a connection between the robot controller and the cloud.

Solution: Adapting the address of the MQTT broker via the variable display

Description

The connection to the KUKA Cloud can be established via an MQTT broker. The following variables must be configured for this purpose (via variable display on KUKA smartHMI):

Variable	Description
KUKACONNECT_BROKERIP	Enter IP address of the MQTT broker.
KUKACONNECT_ENABLED	Set to TRUE to activate MQTT.

If the connection to the MQTT broker cannot be established, it is necessary to check whether the variables are correctly configured.

Precondition

The following user rights are required for modifying a variable:

- KSS 8.5 or higher: Function group **General configuration**
- KSS 8.3: User group “Expert”
- VSS: User group “User”

Procedure

1. In the main menu, select **Display** > **Variable** > **Single**.
   The **Variable display – Single** window opens.
2. Enter the name of the variable in the **Name** box and confirm with the Enter key.
3. The current value of the variable is displayed in the **Current value** box. If nothing is displayed, no value has yet been assigned to the variable.
4. To modify the variable: Enter the desired value in the **New value** box.
5. Press the **Set value** button. The new value is displayed in the **Current value** box.

Cause: URL address of the MQTT broker is incorrect

Description

If the URL address of the MQTT broker is specified incorrectly, it is not possible to establish a connection between the robot controller and the cloud.

Solution: Adapting the address of the MQTT broker via the configuration file

Description

The connection of the robot controller to the MQTT broker is configured via a configuration file. If the connection to the MQTT broker cannot be established, it is necessary to check whether the following variable parameters are correctly configured:

Key	Use	Description
“Address”	Required	Address of the MQTT broker (URL or KRL variable)
“ConnectionName”	Required	Name of the connection The name “KUKAConnect” is reserved and cannot be used.
“PublisherId”	Required	Publisher ID It is advisable to use a placeholder, e.g. “%HOSTNAME%”, so that the Publisher ID automatically becomes unique for multiple robots.

Configuration file

Directory	C:\KRC\ROBOTER\Config\User\Common\opcua
File	*.pscj

Procedure

1. Open the configuration file.
2. Configure the connection parameters correctly.

8.2.2 DeviceConnector 00003

Message code	DeviceConnector 00003
Message text	Unable to find KRL variable “{0}” ({1})
Message type	Status message 
Possible cause(s)	**Cause:** Configuration file .pscj contains errors (>>> Page 69) **Solution:** Adapt configuration file .pscj (>>> Page 69)

Cause: Configuration file .pscj contains errors

The KRL variable cannot be found if the configuration file does not meet the following three conditions:

- The configuration file must be of the file type *.pscj.
- The file path of the KRL variable must be correctly specified in the configuration file.
- The URL for the file path of the KRL variables must be of the data type String.

Checking instructions

1. Open the file location of the configuration file **C:/KRC/Robot/User/Common/OpcUa/**.
2. Check whether the file extension ***.pscj** is present.
3. Open the configuration file ***.pscj**.
4. Check whether the file path **ConnectionParameter > Address** exists.

Solution: Adapt configuration file .pscj

Procedure

1. Call the storage location of the configuration file: C:/KRC/Roboter/User/Common/OpcUa/
2. Ensure that the file ends in *.pscj.
3. Open the configuration file.
4. Check the file path under **ConnectionParameter > Address** and adapt it if necessary.

8.2.3 DeviceConnector 00006

Message code	DeviceConnector 00006
Message text	Failed to read file "{0}".

Message type	Notification message
Possible cause(s)	**Cause:** Configuration file .pscj contains errors (>>> Page 70) **Solution:** Adapt configuration file .pscj (>>> Page 70)

Cause: Configuration file .pscj contains errors

The KRL variable cannot be found if the configuration file does not meet the following three conditions:

- The configuration file must be of the file type *.pscj.
- The file path of the KRL variable must be correctly specified in the configuration file.
- The URL for the file path of the KRL variables must be of the data type String.

Checking instructions

1. Open the file location of the configuration file **C:/KRC/Robot/User/Common/OpcUa/**.
2. Check whether the file extension ***.pscj** is present.
3. Open the configuration file ***.pscj**.
4. Check whether the file path **ConnectionParameter > Address** exists.

Solution: Adapt configuration file .pscj

Procedure

1. Call the storage location of the configuration file: C:/KRC/Roboter/User/Common/OpcUa/
2. Ensure that the file ends in *.pscj.
3. Open the configuration file.
4. Check the file path under **ConnectionParameter > Address** and adapt it if necessary.

8.2.4 DeviceConnector 00007

Message code	DeviceConnector 00007
Message text	Unexpected error in “{0}”
Message type	Notification message
Possible cause(s)	**Cause:** Unexpected error (>>> Page 71) **Solution:** Contact KUKA Service (>>> Page 71)

Cause: Unexpected error

If the OPC UA or MQTT communication breaks off during the restart of the controller, an unexpected error message is displayed.

Solution: Contact KUKA Service

8.2.5 DeviceConnector 00008

Message code	DeviceConnector 00008
Message text	The license file {0} is no longer valid.
Message type	Notification message 
Possible cause(s)	**Cause:** License file invalid (>>> Page 71) **Solution:** Installing new license file (>>> Page 71)

Cause: License file invalid

Description

If a license file is not valid, the required data cannot be called up by the device used. This occurs when an incorrect license file is used, when the duration of validity for a license file has expired or when the license file and the serial number of the device do not match.

Solution: Installing new license file

Description

License files offer the possibility of adding functions and additional data points to the scope of KUKA.DeviceConnector.

License files are uniquely assigned to devices via the device serial number.

To allow for the license file to be used, it must be stored in the correct directory after it is acquired and the system must be reconfigured.

Precondition

- User rights: Function group **General configuration**
- T1 or T2 mode
- No program is selected.

Procedure

1. Apply for the license file at DigitalServices.Robotics.De@kuka.com.
2. Copy the license file to C:/KRC/ROBOTER/config/user/common/opcua/accessconfig/.
3. Reconfigure the I/O driver

   Further information can be found in the corresponding chapter in the system software documentation.

8.2.6 DeviceConnector 00009

Message code	DeviceConnector 00009
Message text	The license file {0} is invalid.
Message type	Notification message 
Possible cause(s)	**Cause:** License file invalid (>>> Page 72) **Solution:** Installing new license file (>>> Page 72)

Cause: License file invalid

Description

If a license file is not valid, the required data cannot be called up by the device used. This occurs when an incorrect license file is used, when the duration of validity for a license file has expired or when the license file and the serial number of the device do not match.

Solution: Installing new license file

Description

License files offer the possibility of adding functions and additional data points to the scope of KUKA.DeviceConnector.

License files are uniquely assigned to devices via the device serial number.

To allow for the license file to be used, it must be stored in the correct directory after it is acquired and the system must be reconfigured.

Precondition

- User rights: Function group **General configuration**
- T1 or T2 mode
- No program is selected.

Procedure

1. Apply for the license file at DigitalServices.Robotics.De@kuka.com.
2. Copy the license file to C:/KRC/ROBOTER/config/user/common/opcua/accessconfig/.
3. Reconfigure the I/O driver

   Further information can be found in the corresponding chapter in the system software documentation.

8.2.7 DeviceConnector 00010

Message code	DeviceConnector 00010
Message text	The license file {0} is invalid. Incorrect serial number.

Message type	Notification message 
Possible cause(s)	**Cause:** License file invalid (>>> Page 73) **Solution:** Installing new license file (>>> Page 73)

Cause: License file invalid

Description

If a license file is not valid, the required data cannot be called up by the device used. This occurs when an incorrect license file is used, when the duration of validity for a license file has expired or when the license file and the serial number of the device do not match.

Solution: Installing new license file

Description

License files offer the possibility of adding functions and additional data points to the scope of KUKA.DeviceConnector.

License files are uniquely assigned to devices via the device serial number.

To allow for the license file to be used, it must be stored in the correct directory after it is acquired and the system must be reconfigured.

Precondition

- User rights: Function group **General configuration**
- T1 or T2 mode
- No program is selected.

Procedure

1. Apply for the license file at DigitalServices.Robotics.De@kuka.com.
2. Copy the license file to C:/KRC/ROBOTER/config/user/common/opcua/accessconfig/.
3. Reconfigure the I/O driver

   Further information can be found in the corresponding chapter in the system software documentation.

9 KUKA Service

9.1 Requesting support

Introduction

This documentation provides information on operation and operator control, and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Information

The following information is required for processing a support request:

- Description of the problem, including information about the duration and frequency of the fault
- The greatest possible amount of information about the hardware and software components of the overall system

  The following list gives an indication of the information which is relevant in many cases:

  – Model and serial number of the kinematic system, e.g. the manipulator
  – Model and serial number of the controller
  – Model and serial number of the energy supply system
  – Designation and version of the system software
  – Designations and versions of other software components or modifications
  – System Software diagnosis package

    Additionally for KUKA Sunrise: Existing projects including applications

    For versions of KUKA System Software older than V8: Archive of the software (Diagnosis package is not yet available here.)
  – Application used
  – External axes used

9.2 KUKA Customer Support

The contact details of the local subsidiaries can be found at:
www.kuka.com/customer-service-contacts

Index

V

W