

VERSION 1.7  
JANUARY 6, 2021

# MQTT CONNECTOR PROFESSIONAL

SW MANUAL  
Developer guide



4EACH S.R.O.  
[WWW.4EACH.CZ](http://WWW.4EACH.CZ)

## MQTT CONNECTOR PROFESSIONAL API REFERENCE

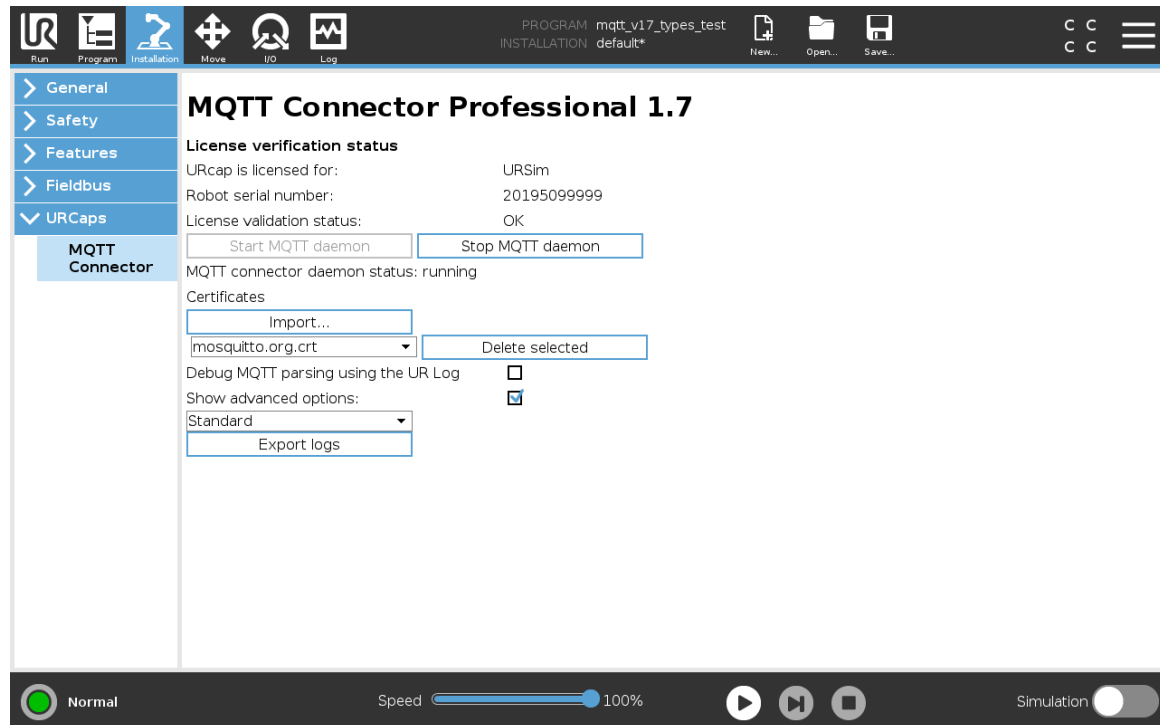
Documentation valid for URCap version 1.7

### RELEASE NOTES

This version finally brings two new useful features:

- Broker connection using CA signed server certificate and TLS v 1.2.
- New function `get_parsed_message(topic, default_value)` for direct value assignment

### INSTALLATION TAB



Installation tab incorporates buttons to start and stop MQTT daemon. Also licensing details are displayed. Physical robots require purpose built URCap with correct serial number.

Import... button allows user to import the CA signed certificate in the CRT format. All imported certificates are displayed in the list and it is also possible to delete each item by pressing the Delete selected button. The name of the certificate should be use in the `mqtt_connect_tls(cert_name)` function.

A new option “Debug MQTT parsing using the UR Log” is added for discovering issues with the MQTT payload during the use of `get_parsed_message`.

Under advanced options, error logging priority selections is present. This option is for remote support only and the end-user usually don’t need it. In default state standard option is selected.

- Standard log priority – logs only errors and critical information
- Debug log priority – logs all possible information useful for debugging and diagnostics

To change the logging priority restarting of the daemon is no longer required.

## MQTT\_INITIALIZE

initializes connection details using credentials.

It takes the following arguments:

- Hostname – the hostname or IP address of the remote broker [string]
- Port - the network port of the server host to connect to [int]
- User – username to access to remote broker [string]
- Password - password to access to remote broker [string]

Note, that some brokers will accept anonymous users, even though they provide username and password.

## MQTT\_INITIALIZE\_ANONYMOUS

initializes connection details using anonymous access (without username and password).

It takes the following arguments:

- Hostname – the hostname or IP address of the remote broker [string]
- Port - the network port of the server host to connect to [int]

## MQTT\_SET\_MAX\_QUEUE

Sets maximum outgoing messages queue length. This is important in case of connection loss, because every queued message is sent after connection is restored. This can cause temporary network overload. Parameter with value 0 sets unlimited queue length. When the queue is full, any further outgoing messages would be dropped.

- Length - maximum outgoing messages queue length [int]

## MQTT\_SET\_LAST\_WILL

Set a Will to be sent to the broker. If the client (ungracefully) disconnects without calling disconnect(), the broker will publish the message on its behalf.

- Topic - the topic that the will message should be published on [string]
- Message - the message to send as a will [string]
- Qos - the quality of service level to use for the will [int]
- Retained - if set to True, the will message will be set as the “last known good”/retained message for the topic [bool]

Note, that this function must be called after initialize function and before connect function.

## MQTT\_SET\_PUBLISH\_ON\_STOP

Set a message to be sent to the broker when UR program stops.

- Topic - the topic that the will message should be published on [string]
- Message - the message to send as a will [string]
- Qos - the quality of service level to use for the will [int]
- Retained - if set to True, the will message will be set as the “last known good”/retained message for the topic [bool]

**Timeout parameter used in previous versions of the product is deprecated. Message is always send until 10 seconds after the program is stopped.**

Note, that this function differs from last will. This function sends specified messages when the robot program changes the state from running to stopped. This can occur by user interaction; emergency stop or other factors. Messages to be send are stored locally, therefore if ungraceful disconnect occurs, no messages can be sent. Message buffer is limited to 100 entries.

## MQTT\_CONNECT

Connects the client to a broker, using default timeout period of 5 seconds.

## MQTT\_CONNECT\_TLS

Connects the client to a broker, using default timeout period of 5 seconds and CA signed server certificate. The connection is established using TLS protocol v 1.2. The certificate must be imported (please see the chapter Installation tab) and the parameter contains the filename with .crt extension without the file path (as displayed in the list of imported certificates on the installation tab).

## MQTT\_CONNECT\_TIMEOUT

Connects the client to a broker, using custom timeout period defined by parameter.

- Timeout - Ensures maximum blocking duration of function call. Time is defined in milliseconds (internally truncated to 100ms steps). [int]

## MQTT\_SUBSCRIBE

Subscribe the client to one topic.

- Topic - subscription topic to subscribe to [string]

## MQTT\_UNSUBSCRIBE

Unsubscribe the client from one topic.

- Topic - subscription topic to unsubscribe from [string]

## MQTT\_UNSUBSCRIBE\_ALL

Unsubscribe the client from all topics.

## MQTT\_GET\_MESSAGE

Function used to lookup incoming messages buffer. Returns latest received value from given topic.

- Topic – Specifies topic to search for [string]

Note, that this function has different communication interface than other methods. It doesn't return OK status code, but actual message payload. If error is present, error codes are returned.

## MQTT\_GET\_PARSED\_MESSAGE

Function used to lookup incoming messages buffer and convert to the URScript native type. Returns latest received typed value from given topic or the default value in case of any issue.

- Topic – Specifies topic to search for [string]
- DefaultValue – variable or constant for default value

Note, that this function has different communication interface than other methods. It does not return the OK status code but current typed message payload. This function recognizes the type of the default value parameter and tries to parse the incoming MQTT payload.

If an error is present or the payload cannot be parsed, the default value is returned. If the log-level is set to "Debug" the specific error code is written to the Universal robots Log.

Here is the list of MQTT payload examples for each supported URScript data type:

| URScript type    | MQTT payload example   |
|------------------|--|
| Boolean          | True   |
| Integer          | 78   |
| Float            | 3.14159265358  |
| Array of Boolean | [True, False, True, False]                                       |
| Array of Integer | [1921439, 1921440, 1921441, 1921442]                             |
| Array of Float   | [1.11, 2.22, 3.33]   |
| Pose             | p[-0.189632, -0.609684, 0.260971, -0.001221, 3.116277, 0.038892] |

### MQTT PAYLOAD NOTES:

- Do not use comma (,) character as a decimal separator (ie. ~~3,1415~~)
- Try to avoid the GET\_PARSED\_MESSAGE for STRING variables, use GET\_MESSAGE instead (it works, but all recognizable datatypes will be considered as error and payload cannot contain keywords like "NG-CODE", "nan", or "ERROR")
- MQTT payload should never contain following characters (character sequences): `|' , '<~' , '~>'`
- If the size of the array in the payload doesn't fit with the size of default value, the function returns NG-CODE-075 (type inequality).

## MQTT\_PUBLISH

This function causes a message to be sent to the broker and subsequently from the broker to any clients subscribing to matching topics, using default timeout period of 3 seconds. It takes the following arguments:

- Topic - the topic that the will message should be published on [string]
- Message - the message to send as a will [string]
- Qos - the quality of service level to use for the will [int]
- Retained - if set to True, the will message will be set as the “last known good”/retained message for the topic [bool]

## MQTT\_PUBLISH\_TIMEOUT

This function causes a message to be sent to the broker and subsequently from the broker to any clients subscribing to matching topics, using custom timeout period defined by parameter. It takes the following arguments:

- Topic - the topic that the will message should be published on [string]
- Message - the message to send as a will [string]
- Qos - the quality of service level to use for the will [int]
- Retained - if set to True, the will message will be set as the “last known good”/retained message for the topic [bool]
- Timeout - Ensures maximum blocking duration of function call. Time is defined in milliseconds, and changes in 100ms increments. [int]

## MQTT\_DISCONNECT

Disconnects from the broker cleanly. Using disconnect() will not result in a will message being sent by the broker. Disconnect will not wait for all queued message to be sent.

## DEBUGGING PRACTICE AND ERROR CODES

To ease programming, error troubleshooting and general ease of use, MQTT Connector uses error codes as return values for most of its functions. This can be helpful in situations where client-broker connectivity is not stable, or during initial setup.

The new function GET\_PARSED\_MESSAGE always returns the default value in case of any issue. The error code of this function can be found in standard log of UR, if the log-level is set to “Debug” on the installation tab (advanced options). Changing of the log-level must be done before the UR program is started.

General recommended usage

- Compare return values against constants or predefined variables using IF statement. Return values should be compared against error codes table.
- Store return values in variables. Even though value is never compared against error codes, it can be helpful to view variable values in Polyscope graphical interface.

## ERROR CODE TABLE

| URScript function       | Description                                      | Status | Error code  |
|-------------------------|--|--------|-------------|
| *** ALL FUNCTIONS ***   | NOT AUTHORIZED                                   | NG     | NG-CODE-001 |
| authorize               | SUCCESS  | OK     | OK-CODE-002 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-003 |
| *** ALL FUNCTIONS ***   | <i>deprecated</i>                                | NG     | NG-CODE-004 |
|                         | DAEMON NOT READY – INTERNAL ERROR***             | NG     | NG-CODE-005 |
|                         | CLIENT NOT INITIALIZED – call initialize         | NG     | NG-CODE-006 |
| initialize              | SUCCESS  | OK     | OK-CODE-010 |
|                         | FAILED   | NG     | NG-CODE-011 |
| connect                 | MQTT CONNECT OK                                  | OK     | OK-CODE-020 |
|                         | NO ROUTE TO HOST                                 | NG     | NG-CODE-021 |
|                         | FAILED - timeout                                 | NG     | NG-CODE-022 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-023 |
| connect_tls             | Given .crt file not found                        | NG     | NG-CODE-024 |
|                         | + all codes of connect function                  |        |             |
| disconnect              | MQTT DISCONNECT OK                               | OK     | OK-CODE-030 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-031 |
| subscribe               | SUBSCRIBE OK                                     | OK     | OK-CODE-040 |
|                         | SUBSCRIBE FAILED - already exists                | NG     | NG-CODE-041 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-042 |
| unsubscribe             | UNSUBSCRIBE OK                                   | OK     | OK-CODE-050 |
|                         | UNSUBSCRIBE FAILED - no such subscription        | NG     | NG-CODE-051 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-052 |
| unsubscribe_all         | UNSUBSCRIBE ALL OK                               | OK     | OK-CODE-060 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-061 |
| get_message             | <b>MESSAGE PAYLOAD STRING</b>                    | OK     |             |
|                         | NOT YET RECEIVED ANY MESSAGE                     | NG     | NG-CODE-071 |
|                         | NO SUCH SUBSCRIPTION                             | NG     | NG-CODE-072 |
|                         | FALSE PARAMS                                     | NG     | NG-CODE-073 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-074 |
| get_parsed_message**    | <b>PARSED PAYLOAD VALUE OR DEFAULT</b>           | OK     |             |
|                         | DEFAULT VALUE TYPE AND PAYLOAD TYPE IS NOT EQUAL | NG     | NG-CODE-075 |
|                         | all other NG codes of get_message function       |        |             |
| publish                 | SUCCESS  | OK     | OK-CODE-080 |
|                         | FAILED   | NG     | NG-CODE-081 |
|                         | MESSAGE SEND NOK - broker disconnected           | NG     | NG-CODE-082 |
|                         | FALSE PARAMS                                     | NG     | NG-CODE-083 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-084 |
| set_max_queued_messages | SUCCESS  | OK     | OK-CODE-090 |
|                         | FAILED   | NG     | NG-CODE-091 |
| set_last_will           | SUCCESS  | OK     | OK-CODE-100 |
|                         | FAILED   | NG     | NG-CODE-101 |
| set_publish_on_stop     | SUCCESS  | OK     | OK-CODE-110 |
|                         | FALSE PARAMS                                     | NG     | NG-CODE-111 |
|                         | BUFFER IS FULL                                   | NG     | NG-CODE-112 |
|                         | UNKNOWN ERROR                                    | NG     | NG-CODE-113 |

\*\* These error codes are visible in the UR Log, if the feature "Debug MQTT parsing using the UR Log" is enabled in the Installation tab (Advanced options)

\*\*\* This error is thrown if the daemon is not started. In very few cases this error can be invoked during the runtime (this issue is known as NO-CONTROLLER robot error). If the get\_message/get\_parsed\_message is called again, the function will return the correct result without necessity of stopping the robot program.