acontis technologies GmbH

SOFTWARE

EC-Master

Feature Pack MQTT

Version 3.2

Edition: August 1, 2025

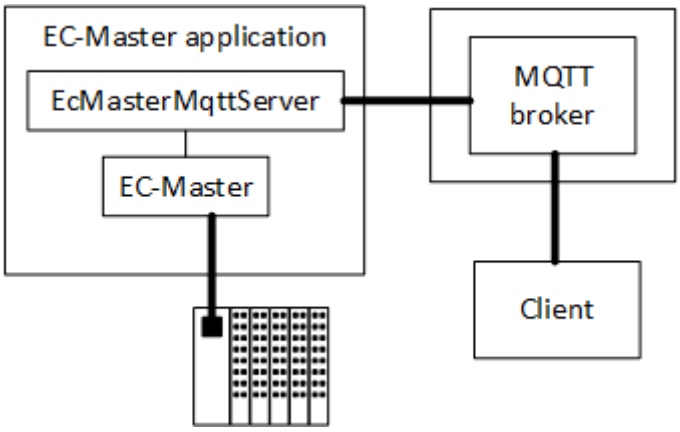
Contents

1 Introduction

The MQTT (Message Queueing Telemetry Transport) functionality allows to query configuration of and to control EC-Master stack running on a target.

1.1 Typical setup

In common case EC-Master is connected to a MQTT broker and serves requests over EcMasterMqttServer library.



2 Application programming interface, reference

2.1 emMqttSrvGetVersion

EC_T_DWORD EC_NAMESPACE::**emMqttSrvGetVersion**(EC_T_VOID)
Version of EcMaster MQTT Server Software.

Returns
Version Number as DWORD.

2.2 emMqttSrvStart

EC_T_DWORD EC_NAMESPACE::**emMqttSrvStart**(
EC_T_MQTT_SRVPARMS *pParms,
EC_T_PVOID *ppHandle
)
Initializes and start MQTT Server Instance.

Parameters

- **pParms** – [in] Server start-up parameters
- **ppHandle** – [out] Handle to opened instance, used for ctrl access

Returns
EC_E_NOERROR or error code

struct **EC_T_MQTT_SRVPARMS**

Public Members

EC_T_DWORD **dwSignature**
[in] Set to ECMASTERMQTTSERVER_SIGNATURE

EC_T_DWORD **dwSize**
[in] Set to sizeof(EC_T_MQTT_SRVPARMS)

EC_T_LOG_PARMS **LogParms**
[in] Pointer to logging parameters

EC_T_CHAR ***pszMqttServerUri**
[in] URI of MQTT broker, i.e. “tcp://172.17.10.53:1883”

EC_T_CHAR ***pszMqttClientId**
[in] MQTT client ID, if empty a generated name will be used

EC_T_BOOL **bSkipCheckCertificate**
[in] If EC_TRUE the SSL certificate will be ignored

EC_T_CHAR ***szInstanceName**
[in] Name of master instance, i.e. ‘Instance1’

EC_T_DWORD **dwWorkerThreadPrio**
[in] Working thread priority

EC_T_CPUSET **WorkerThreadCpuSet**
[in] Working thread CPU set

EC_T_DWORD **dwWorkerThreadStackSize**
[in] Working thread stack size, 0 = default

EC_T_DWORD **dwCommThreadPrio**
[in] Communication thread priority

EC_T_CPUSET **CommThreadCpuSet**
[in] Communication thread CPU set

EC_T_DWORD **dwCommThreadStackSize**
[in] Communication thread stack size, 0 = default

EC_T_DWORD **dwMaxMsgCnt**
[in] Max. amount of MQTT messages in buffer, 0 = default

EC_T_DWORD **dwMaxMsgSize**
[in] Max. size of messages buffer, 0 = default

EC_T_CNF_TYPE **eCnfType**
[in] Type of master configuration

EC_T_BYTE ***pbyCnfData**
[in] Master configuration data

EC_T_DWORD **dwCnfDataLen**
[in] Size of master configuration

enum **EC_T_CNF_TYPE**
Values:

enumerator **eCnfType_Unknown**

enumerator **eCnfType_Filename**
pbyCnfData: ENI filename to read

enumerator **eCnfType_Data**
pbyCnfData: ENI data

enumerator **eCnfType_Datadiag**
pbyCnfData: ENI data for diagnosis

enumerator **eCnfType_GenPreopENI**
Generate ENI based on bus-scan result to get into PREOP state

enumerator **eCnfType_GenPreopENIWithCRC**
Same as eCnfType_GenPreopENI with CRC protection

enumerator **eCnfType_GenOpENI**
Generate ENI based on bus-scan result to get into OP state. The default PDO mapping read from the

slaves is activated. See ETG2010 “SII Specification”, Table 14 “Structure Category TXPDO and RXPDO for each PDO”

enumerator **eCnfType_None**
Reset configuration

enumerator **eCnfType_ConfigData**
pbyCnfData: Binary structured configuration

enumerator **eCnfType_GenOpENINoStrings**
Generate ENI based on bus-scan result to get into OP state , does not read strings from EEPROM

enumerator **eCnfType_FileByApp**
File access provided by user application, See EC_T_CNF_FILEBYAPP_DESC

enumerator **eCnfType_GenEBI**
Generate EBI based on bus-scan result

2.3 emMqttSrvStop

EC_T_DWORD EC_NAMESPACE::**emMqttSrvStop**(
EC_T_PVOID pvHandle,
EC_T_DWORD dwTimeout
)
Stop and de-initialize MQTT Server Instance.

Parameters

- **pvHandle** – [in] Handle to previously started Server
- **dwTimeout** – [in] Timeout [ms] used to shut down all spawned threads, it’s multiplied internally by the amount of threads spawned.

Returns
EC_E_NOERROR or error code