AC500 V3 – MQTT & JSON
Best practice webinar
Dominik Franz, Global Technical Support
1 Overview
2 Using new JSON library
3 Using new MQTT library
4 Hands-on session
5 References & Q&A
1 Overview
2 Using new JSON library
3 Using new MQTT library
4 Hands-on session
5 References & Q&A
Overview
MQTT (Message Queuing Telemetry Transport) architecture

- Temperature probe
- AC500 PLC
- Mobile device
- Laptop

Publish to Temperature: “Probe1”: 23
Subscribe to topics
Publish & Receive data
Subscribe Temperature
Publish “Probe1”: 23
Overview

JSON (JavaScript Object Notation)

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoConToDrive</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>Ready</td>
<td>BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>Operating</td>
<td>BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>AtSetpoint</td>
<td>BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>Tripped</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>Alarm</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>ActSpeed</td>
<td>INT</td>
<td>4836</td>
</tr>
<tr>
<td>ActValue2</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td>Message</td>
<td>String</td>
<td>Operation</td>
</tr>
</tbody>
</table>

JSON String

```json
{
    "NoConToDrive": false,
    "Ready": true,
    "Operating": true,
    "AtSetpoint": true,
    "Tripped": false,
    "Alarm": false,
    "ActSpeed": 4836,
    "ActValue2": 0,
    "Message": "Operation"
}
```
Overview
AC500 offering

MQTT library
- Connecting to MQTT broker
- Using encrypted, secured communication (TLS v. 1.2)
- User definable keep alive / timeout times & last will
- Parallel connection to different brokers possible
- Publishing messages to different topics
- Subscribing to topics, including wildcards, and receiving messages
- Supporting Unicode charset (UTF16)

JSON library
- Create JSON objects and arrays
- Parse JSON objects and array
- Nested objects / arrays possible
- Parsing/ serializing objects from / into string or file
- Supporting Unicode charset (UTF16)
System Overview

Libraries

New vs old library

New V3 libraries available as of AB2.7.0
Recommended for further usage

Old library is still available & supported
V2 compatible
Will move to “classic” state depending on feedback
Overview

2 Using new JSON library

3 Using new MQTT library

4 Hands-on session

5 References & Q&A
# New functionalities

Comparison of the JSON (JavaScript Object Notation) libraries

<table>
<thead>
<tr>
<th>Old AC500 JSON</th>
<th>New JSON Utilities SL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available for V2 &amp; V3</td>
<td>Available for V3 as of AB2.7.0</td>
</tr>
<tr>
<td>Supports BOOL, DINT, LREAL &amp; STRING</td>
<td>Supports “ANY” datatype</td>
</tr>
<tr>
<td>Limited to “STRING” = ASCII characters</td>
<td>Using “WSTRING” supports all Unicode characters</td>
</tr>
<tr>
<td>Dynamic memory allocation → free required</td>
<td>Fixed memory allocation with user defined size</td>
</tr>
<tr>
<td></td>
<td>Can also read/ write JSON files</td>
</tr>
<tr>
<td></td>
<td>Supports “Null” keys</td>
</tr>
</tbody>
</table>
Using new JSON library

Program flow

1. Instantiate JSONDataFactory
2. Create JSON root object
3. Add Keys to object
4. JSONByteArrayWriter
5. Json reset

Resulting string

```json
{
  "device": "AC500",
  "running": true,
  "temp1": 23.12,
  "temp2": 100.98765,
  "temp3": 65.8475,
  "timestamp": "2024-01-15 10:00:02",
  "alarm1": "This is a first message",
  "alarm2": "This is a second message",
  "rms": 0.00004585
}
```
Using new JSON library

Program flow

1. Instantiate JSONDataFactory
2. Create JSON root object
3. Add Keys to object
4. JSONByteArrayWriter
5. Json reset

Pseudo Code

fbFactory: JSON.JSONDataFactory;
pJsonData: POINTER TO JSON.JSONData := fbFactory.Create();
fJsonBuilder(pJsonData := pJsonData, diRootObj => diRoot);
fJsonBuilder.SetKeyWithValue("Device", "AC500");
fJsonBuilder.SetKeyWithValue("running", TRUE);
fJsonBuilder.SetKeyWithValue("temp1", 23.12);
jsonWriter(ADR(JsonString), jsonData := pJsonData^);

fbJsonBuilder.Reset();
## New functionalities
Comparison of the MQTT (Message Queuing Telemetry Transport) libraries

<table>
<thead>
<tr>
<th>Old AC500 MQTT</th>
<th>New MQTT Client SL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available for V2 &amp; V3</td>
<td>Available for V3 as of AB2.7.0</td>
</tr>
<tr>
<td>Connection needs to be handled by the user</td>
<td>Connection handled in MQTTClient function block</td>
</tr>
<tr>
<td>Certificate for encryption can be a file, buffer or cert store Thumbprint</td>
<td>Certificate for encryption can be in the cert store. Peer verification as defined in TLS settings.</td>
</tr>
<tr>
<td>Limited to STRING characters</td>
<td>All WSTRING characters possible</td>
</tr>
<tr>
<td>Only IP addresses possible</td>
<td>Can resolve Hostname (DNS must be configured)</td>
</tr>
<tr>
<td></td>
<td>Supports MQTT Version 5</td>
</tr>
<tr>
<td></td>
<td>Supports MQTT via Websockets</td>
</tr>
<tr>
<td></td>
<td>Supports Multitask configuration</td>
</tr>
</tbody>
</table>
Using new JSON library

Program flow

1. Connect to broker

2. Subscribe to topic

3. Publish data

4. Receive data

Pseudo Code

```plaintext
fbMqttClient('MyAC500V3PLC', 'test.mosquitto.org');

fbMqttSubscribe(fbMqttClient, "AC500V3/get", ADR(recPayload));

fbMqttPublish(fbMqttClient, "AC500V3/messages", ADR(payload));

IF fbMqttSubscribe.xReceived THEN
    //handle recPayload
END_IF
```
1 Overview
2 Using new JSON library
3 Using new MQTT library
4 Hands-on session
5 References & Q&A
Hands-on session
Function Block Diagram - minimal program

Connect to broker

Publish a message
Connect to broker

```javascript
var mqttEnable = PFON; // Get to true during runtime to enable MQTT protocol
sHostName = STRING := 'test.mosquitto.org'; // Broker to connect to. Can either be an IP address or the hostname itself. In this case the EMS server has to be set in ID Config tool
sLastWill = STRING(255) := '{"message":"Last connection","value":0}'; // Last will which is sent to broker to share once the connection to the client is lost
uiKeepAlive = UINT := 60; // If no message / ping is send within 60 s a broker will close the connection
uiTimeout = UINT := 10000000; // 20 seconds used for connect, publish and receive
wsSendTopic = WSTRING(1024) := "AC500V3/messages"; // Topic where the data is published to
wsRecTopic = WSTRING(1024) := "AC500V3/receive"; // Topic from which data is received

// This content set to Verification Note to 2 > verify peer > check if server root certificate is installed in "Trusted Certificates" in the security screen
mqttContext = MQTT.MQS.TLSContx := {ePurpose:=MQTT.MQS.PURPOSE.CLIENT_SIDE, uiVerificationNode:=2};

// SSL3 for secure communication
// Keepalive time = 1/2 --- 20 sec ping interval, by default reestablish session timeout
if(mqttClient != MQTT.MQTTClient := {uiPort := 8883, xUseTLS := TRUE, uiKeepAlive := uiKeepAlive, uiPingInterval := TO_TIME(uiKeepAlive*1000/3), xCleanSession := FALSE, aiTimeout := aiTimeout,
// Will message or test message can be send during first connect. Here an example
pWillMessage := ADR(sLastWill), uiWillMessage := TO_UINY(LEN(sLastWill)), sWillRetain := TRUE, eWillQos := MQTT.MQTT.QOS.0x01, wsWillTopic := wsSendTopic,
WAIT(1000); mqttContext := mqttTLSCtx};

if(mqttClient != MQTT.MQTTClient :=
    xEnable := mqttEnable.FALS,
    wsUsrname := ,
    wsPassword := ,
    sClientID := 'MyAC500VPLC',
    sHostName := 'test.mosquitto.org',
    sConnectedToBroker := sConnected.FALS} // sConnected will be true if connection could be established successfully
```
Hands-on session
Publish Data

Generate & Publish data

```plaintext
fbJsonBuilder.SetKeyWithValue(wsKey := "NoConToDrive", Value := gaDrives[DriveId].xNoConToDrive, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "Ready", Value := gaDrives[DriveId].xReady, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "Operating", Value := gaDrives[DriveId].xOperating, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "AtSetpoint", Value := gaDrives[DriveId].xAtSetpoint, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "Tripped", Value := gaDrives[DriveId].xTripped, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "Alarm", Value := gaDrives[DriveId].xAAlarm, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "ActSpeed", Value := gaDrives[DriveId].xAActSpeed, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "ActValue2", Value := gaDrives[DriveId].xAActValue2, eError => eJsonErr);
fbJsonBuilder.SetKeyWithValue(wsKey := "Message", Value := gaDrives[DriveId].xMessage, eError => eJsonErr);

jsonWriter(xExecute := TRUE, pwData := ADR(refPayload), udiSize := SIZEOF(refPayload), wsLinebreak := ",", jsonData := pJsonDataPub^);
IF jsonWriter.xDone THEN
  jsonWriter(xExecute := FALSE, jsonData := pJsonDataPub^);
  fbJsonBuilder.Reset(eError => eJsonErr);

fbPublish(); // After message was build and message is not an empty JSON string
  xExecute := (eCreation = JsonStatus.DONE) AND (udiPayloadLen>2),
  pbPayload := ADR(wsPublishData),
  udiPayloadSize := udiPayloadLen ^ 2
  mqttClient := fbClient, wsTopicName := wsPublishTopic);
```
Hands-on session
Publish Data

Subscribe to a topic, receive and parse data

```java
//Subscription to topic and receiving of data with parsing into
fbSubscription(
  xEnableFALSE := xConnectedFALSE,
  pbPayload[1607FC8C82] := ADR(wsReceivedData____),
  udiMaxPayloadSize := SIZEOF(wsReceivedData____),
  mqtClient := fbClient,

IF fbSubscription.xSubscribeActiveFALSE AND fbSubscription.xReceivedFALSE THEN

  jsonReader(xExecute := TRUE, pwData := ADR(refPayload), jsonData := pJsonDataRec^);

  //Get SwitchOn from object
  fbJsonFind(xExecute := TRUE, wsKey := "SwitchOn", jsonData := pJsonDataRec^);
  IF fbJsonFind.xDone AND fbJsonFind.jsonElement.eType = JSON.JSONType.BOOL_VALUE THEN
    gaDrives[DriveId].xSwitchOn := fbJsonFind.jsonElement.value.xValue;
    fbJsonFind(xExecute := FALSE, jsonData := pJsonDataRec^);
  END_IF
END_IF
```
1 Overview

2 Using new JSON library

3 Using new MQTT library

4 Hands-on session

5 References & Q&A
References & Q&A

Application Examples
- AC500 V3 – JSON – How to use the library
- AC500 V3 – MQTT – How to use the library
- AC500 V3 as Drive MQTT Gateway

Application Notes
- AC500 MQTT & Mosquitto
- AC500 MQTT & Azure
- AC500 MQTT FAQs