mapp Services

1 mapp Tweet: Messaging system

mapp Tweet is a messaging system that can transmit and receive data in the form of text messages.

1.1 Concept

<table>
<thead>
<tr>
<th>User stories</th>
<th>Description</th>
</tr>
</thead>
</table>
| Machine operator | • The machine can inform the operator of its current status – “Packaging material almost empty” for example – if the operator is responsible for multiple machines.  
  • It can be used to forward critical alarms directly to the appropriate recipients via text message (SMS).  
  • The machine should be able to send its messages to the service technician if necessary. This might be used to announce an upcoming service interval.  
  • It should be possible to query the current status of the machine or a process variable. This might be the current PackML state or the temperature variable of a heating zone. |
| Service technician | • As a supervisor, I would like to be informed in the event that the machine stops unexpectedly.  
  • As a supervisor, I would like to be notified if neither the machine operator nor the service technician respond to an alarm on the machine.  
  • It should be possible to forward critical alarms directly to the appropriate recipients via text message. |

1.1.1 SiteManager configuration

In order to be able to use MpTweet, SiteManager 0RMSM1135 can be used. In this case, the remote maintenance module only serves as a modem. For technical information about the module, see B&R SiteManager 0RMSM1135. The SiteManager is connected to the controller via Ethernet. SiteManager can send an SMS text message to a defined telephone number.
The SiteManager must be configured using the "Appliance launcher tool" in order to send SMS text messages. Changes must then be made using a web browser.

**Configuration via "Appliance launcher tool"**

**Step 1**

To configure the SiteManager, you will need the "Appliance launcher tool". This tool can be downloaded [here](#).

**Step 2:**
The next step is to switch on the SiteManager. To connect using the SiteManager, connect port DEV1 to the network. It is important to ensure that the hardware is on the same subnet as the PC. Alternatively, the SiteManager can be connected directly using an Ethernet cable.

**Step 3:**

Next, open the "Appliance launcher tool". All connected devices are listed using the search function. Once the SiteManager has been found, you can continue by clicking the "Next" button. WLAN must be switched off when searching!

![Appliance Launcher](image)

**Step 4:**

Here you need to enter the IP address and subnet mask:
Step 5:
The next step is to enter the username and password. For the username, use *admin*. For the password, enter the MAC address found on the product label. Then click "Next".
Step 6:
In the next window, enter the SIM PIN code and APN (access point name). The APN depends on the respective mobile network provider and can be found online.
You would now have the option of configuring a GateManager. This is not necessary in order to use MpTweet, however.

**Step 7:**

Finally, save everything and reboot the SiteManager.
Configuration via web browser

Now we need to make settings in the "Setup assistant". This assistant can be launched from a web browser.

**Step 1**

The first step is to open a web browser, e.g. Google Chrome. Enter `https://` followed by the IP address of the SiteManagers. You must be in the same network for this!

Step 2

It is possible that a message is displayed when loading the page that states the connection is not secure. To display the page, click on "Advanced" and then "Continue to 172.16.1.209".

**Step 3**

Authentication is required in the next step. Enter `admin` for the username and the MAC address on the SiteManager for the password. The MAC address is specified without periods or spaces. For example, if the MAC address is "00.CO.A1.01.4E.60", then it is specified as "00COA1014E60".

**Step 4**

The "Setup assistant" appears after successful authentication. The administrator password must now be changed.
Enter a new password in the next dialog box. The new password must be between 8 and 47 characters. It must contain at least one numeric digit. Special characters are permitted.
After the password is changed successfully, the authentication window is displayed again. Use the new password here.

Click on the *Continue setup* button to return to the main menu.

**Step 5**

Now specify the controller connected to the SiteManager. The controller is defined using "Device agents".
Specify the parameters for the connected CPU in the next window. This includes defining the device name, device type and IP address of the CPU. If the SiteManager and CPU are connected over Ethernet, specify "Ethernet" here. In our example, the name of the controller has been added as a comment. Now save everything.

Once the "Device agent" has been defined, click on the **Continue setup** button to return to the main menu.

**Step 6**

Since the SiteManager is being used as a modem to send SMS text messages, Uplink2 must be configured.
Specify the mode (SMS only), APN and SIM PIN code. These were also used earlier in the "Appliance launcher tool".
Now save everything and click on **Continue setup** to return to the main menu.

**Step 7**

The notification mode is now configured using the GateManager. This is done by enabling the "Alert mode" and selecting "None" for the identification. The port can be changed with parameter "Agent alerts TCP/UDP port". This is 26864 by default. The port number must be specified later in the MpTweet configuration.
Step 8

The SiteManager is restarted in the last step. This completes the configuration of the SiteManager; it can now be used.
1.1.2 Configuration in Automation Studio

MpTweet is supported by the following hardware:

- B&R SiteManager 0RMSM1135
Configuration in Automation Studio:

Once the modem to be used has been configured successfully, you can make the final settings in Automation Studio.

If using the SiteManager, select "B&R Site Manager 1135" under Modem in the MpTweet configuration. It is also necessary to define whether to use the TCP or UDP protocol. In order to establish a connection to the modem, the IP address and port number of the SiteManager must be specified. The IP address of the SiteManager is configured in the "Appliance launcher tool". The port number is 26864.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Interval</td>
<td>60</td>
<td>sec</td>
<td>Interval defining how often MpTweet receives information from modem</td>
</tr>
<tr>
<td>Modern</td>
<td>B&amp;R Site Manager 1135</td>
<td></td>
<td>Modern settings</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP</td>
<td></td>
<td>Device address from the system configuration</td>
</tr>
<tr>
<td>Connection setup</td>
<td>Site Manager IP address</td>
<td></td>
<td>Configuration of the serial connection, through e.g. X20IF1020</td>
</tr>
<tr>
<td>Port number</td>
<td></td>
<td></td>
<td>IP of the site manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Port number used for communication</td>
</tr>
</tbody>
</table>

1.1.3 Using the AT command set

In order to send and receive SMS text messages, MpTweet is able to communicate with different modem types over a serial interface using the AT command set.

MpTweet has been tested with the AT command set on the following modems:

- CINTERION MC52i Terminal, Release 01.201
- INSYS GSM small 2.0, Revision 01.100
- Westermo GDW-11, Art.No. 3615-0001, Release Q24PL001

With the modem, the PIN for the SIM card must be stored and the serial interface configured (baud rate, stop bit, etc.).

1.1.4 Connection to alarm management

Notification can be sent to a user in the event of an alarm. Only a few settings are necessary for this:

Alarm configuration

MpAlarmX must be used as the alarm system. The MpAlarmXCore configuration is added. In our example, we have created alarm TempHigh. In order for a message to be sent when the alarm is triggered, "SendMessage" must be selected under section Mapping.
Message configuration

If alarm "TempHigh" is triggered, a message is sent automatically to the first person to be notified. For this reason, we have to specify a user under Subscriptions and hierarchy. Additional settings can also be made in this section; more information about these can be found in the MpTweetCore configuration.
1.1.5 Auditing the messaging system

Events for these components can be recorded using MpAudit. For more information about how to create an entry, see section Creating entries.

The identification number can be incorporated in the respective entry using the token %act.

The following events can occur for these mapp components:

**MpTweet event**

This event type allows the time when a message was sent or received to be recorded.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>MpTweet components</td>
</tr>
</tbody>
</table>

**Events**

- **msg** Message text
- **rxn** Name of the user who received a message
- **rxn** Telephone number of the user who received a message
- **txn** Name of the user who sent a message
- **txc** Telephone number of the user who sent a message
- **act** Action that was executed in a mapp component. The ID for the action is returned (e.g. 1 for "Message sent").

For more information, see the following table:

<table>
<thead>
<tr>
<th>Event</th>
<th>Identification number (returned via %act)</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent message</td>
<td>1</td>
<td><img src="table1.png" alt="Identification numbers and additional information for sent messages." /></td>
</tr>
<tr>
<td>Received message</td>
<td>2</td>
<td><img src="table1.png" alt="Identification numbers and additional information for received messages." /></td>
</tr>
</tbody>
</table>

**Examples**

- "Message [%msg] sent to [%rxn]" evaluated: "Message Hello sent to Sam".

Table 1: MpTweet event

1.2 Guides

1.2.1 Getting started

1.2.1.1 Creating a messaging system

This section uses MpTweetCore to create a messaging system. A machine command should be started via SMS text message using MpTweetCommandRequestCheck and MpTweetCommandRequestDone.

1.2.1.1.1 Creating a project

You must first create a new project in Automation Studio. For more information, see project with Automation Runtime Simulation.

1.2.1.1.2 Add mapp component

**Adding the MpTweetCore configuration**

The MpTweetCore configuration is added to create a messaging system.
Editing the configuration for the messaging system

It contains the mapp Link, which uniquely identifies the mapp component within the mapp environment. The mapp Link establishes the connection between programming and configuration. The modem to be used is specified under "Connection setup". B&R SiteManager is used. To establish a connection to the modem, the modem's IP address and port number 26864 must be specified. How to configure B&R SiteManager is explained in section SiteManager configuration.

To be able to start a machine command via SMS text message, a request must be created under "Requests". A request is created with ID "StartHeater". The message that must be sent to start the command is "Start heater". Once the heater has reached the required temperature, response "Temperature reached" should be transmitted to the user.
1.2.1.3 Adding a program
The next step is to add a Ladder Diagram program. For more information, see here.

1.2.1.4 Adding function blocks and functions
In the next step, the function blocks and functions are added and configured.

Adding MpTweetCore to the project

Searching for MpTweet in the Ladder Diagram Catalog
MpTweet can be looked for in the Ladder Diagram Catalog. All MpTweet components are listed.

Selecting the function block
Function block MpTweetCore is added to send and receive SMS text messages.
Configuring MpTweetCore

Connecting MpLink
Using input "MpLink", we establish a connection to the configuration we created earlier named "gTweet".
Enabling the function block
The function block is enabled using input "Enable".

Adding parameters
The required MpTweetCore parameters are then added. The parameters must be of data type "MpTweetCoreParameter".
Adding MpTweetCommandRequestCheck to the program

To be able to check incoming requests, function MpTweetCommandRequestCheck is added.

Configuring MpTweetCommandRequestCheck

Connecting MpLink

The same mapp Link that was connected to MpTweetCore is used as the mapp Link.
Defining "RequestID"

"RequestID" specifies the request ID "StartHeater" to be checked.

Switching on the heating unit

The output of the function becomes TRUE as soon as the request is received. If this is the case, the heating unit should be switched on using command "Heater".

Adding MpTweetCommandRequestDone to the program

If the desired temperature is reached, a response should be sent to Sam. This is done using function MpTweetCommandRequestDone.
Configuring MpTweetCommandRequestDone

Adding "EN/ENO"

Option "EN/ENO" is enabled by right-clicking on the function.
The function should be executed as soon as the temperature (TempReached) has been reached. The variable is of data type BOOL.

Connecting MpLink

The same mapp Link that was connected to MpTweetCore is used as the mapp Link.

Determining "RequestID"

The same "RequestID" is used that was also used on MpTweetCommandRequestCheck.

Determining "ReturnType"

ReturnType specifies whether the transmission process was successful. "mpTWEET_CMD_REQUEST_SUCCESS" is specified.
MpTweetCommandRequestDone status

Variable "Status" of data type DINT is created to see the current status of the function.

1.2.1.5 Generating the file structure

Now generate a file structure for the memory card for ARsim. For more information, see here.

1.2.1.6 Testing the messaging system

After the changes are downloaded, the program can be tested.

If a user now transmits a message to the machine, the request is detected and the heating unit is switched on.

As soon as the desired temperature has been reached, a response is sent back to the user.
1.3 Configuration

1.3.1 MpTweetCore configuration

Connection setup

Section "Connection setup" defines the way that messages should be sent. This can be via a modem (for text messages) or a server (for emails). For information on how to configure the server/modem, see section Configuration in Automation Studio. 26864 must be defined as the port number.

The "Receive Interval" is entered in seconds and determines how frequently MpTweet queries the modem for information such as whether there are requests pending.

Recipient

All possible recipients are listed under "Receiver List". The name, telephone number and language are specified for each recipient. The name uniquely identifies the recipient within the application. The telephone number can be used to reach the recipient.

The telephone number can be entered in various formats. The following formats are supported:

- Country calling code: The country calling code can be indicated by the character "+" or 00: +43, 0043.
- Spaces: The telephone number can be specified with or without spaces: +43676821257963, +43 676 821 257 963.
- Other characters: The telephone number can also include other the characters "-" and ": +43 676 58058-0, +43 676/580580.

"User Language" defines the language in which the message should be sent.
For more information on how to send a message in different languages, see "Messages".

**Messages**

Messages that the machine should send should are listed under "Message List". For each message, a name and a message text must be entered. Parameter "Name" uniquely identifies the message in the application. It is specified under parameter "Message" of function MpTweetSendMessage, for example. "Text" contains the message to be sent. The message text can be specified in two different ways:

- **Message text directly in the configuration**: The message text is entered in the configuration under "Text". If you do this, only one language is possible.

- **Message text localized using the text system**: The Automation Studio text system is used to localize message texts. In this case, the text is not entered directly in the configuration, but referenced via the text system. A file containing the message texts must first be added in the Logical View. This file contains versions of the message texts in each project language. The connection is established via the namespace of the text and the text ID. The namespace can be defined in the properties for the file containing the texts. The text ID is then defined for each text in the file to be localized. The namespace and text ID are specified inside curly brackets. The connection to the text system is established with the $ character: {Namespace/TextID}.

A file containing the message texts must first be added in the Logical View. This file contains versions of the message texts in each project language. The connection is established via the namespace of the text and the text ID. The namespace can be defined in the properties for the file containing the texts. The text ID is then defined for each text in the file to be localized. The namespace and text ID are specified inside curly brackets. The connection to the text system is established with the $ character: {Namespace/TextID}.

Figure 1: Use of text system  
Figure 2: Project languages

Figure 3: Namespacelocalizable text file

Figure 4: Text ID and localized texts
Messaging hierarchy

It is possible to define a messaging hierarchy. This is done under "Subscriptions and hierarchy". The order specified here is used to establish a hierarchy for sending alarm notifications. The name of an existing recipient and the type of subscription are specified. The type of subscription defines whether the recipient simply receives the messages ("Receive only") or if they should be required to confirm receipt ("Confirmation of receipt").

If the specified recipient is expected to provide confirmation, then the expected response must be specified under "Expected Response". "Response Timeout" specifies the how long to wait for a response/confirmation from the recipient. Under "Attempts" it is possible to define how many times the alarm message should be resent to the same recipient after the "Response Timeout" has expired. If the recipient fails to respond after the defined number of attempts, then the alarm message is forwarded to the next recipient in the list.

Requests

Requests to MpTweet are defined using "Requests". Requests can include things like querying the current status of the machine or transmitting a command. An example of how a request can be implemented in a specific application is illustrated in the use case Sending a machine command.

The following parameters are specified:

- **Request ID**: The "RequestID" uniquely identifies the request within the application.
- **Type**: Specifies whether the request is a status query or a command.
- **Request Message**: Message text that must be sent in order to receive the corresponding information. If a request is sent to MpTweet, then the message text must exactly match the text specified here. This includes case sensitivity (lowercase and uppercase letters)!
- **Return message**: Response message that is returned to the sender. It may contain the current status of the machine or notification that a command was executed successfully.

1.4 Use cases

This section outlines several different use cases for MpTweet components.

1.4.1 Use case 1: Transmitting a machine command

**Requirement**

A machine must heat up to a certain temperature before it can begin production. The process of heating up takes half an hour. When the operator arrives at the beginning of a shift, he must manually switch on the heat and then wait half an hour before the machine is productive. Now it is possible to send a text message to the machine that triggers the heating process remotely. The operator can send the message half an hour before the beginning of his shift, and the machine will be ready for production right when the shift begins.

**Solution**

**Component list**

- **MpTweetCore** (own MpLink): This component establishes a connection to a specified modem that makes it possible to transmit and receive messages.
- **MpTweetCommandRequestCheck** (MpLink from MpTweetCore): Checks whether a transmitter's request is pending
- **MpTweetCommandRequestDone** (MpLink from MpTweetCore): Indicates whether a request has been executed successfully
The MpTweetCore configuration is added. In order for the machine to switch on preheating on command, a request called "Preheat" is created under Requests. Since the request is for a command to be executed, the Type is set to "Command Request". If the machine operator sends a message, which is specified under Request Message, the machine begins preheating. Once the machine reaches the desired temperature, the message specified under Return Message is returned to the sender.

For information about configuring the modem, see section Configuration in Automation Studio.

Using the mapp components

The MpTweetCore component is added in order to establish a connection with the modem specified in the configuration. The MpTweetCommandRequestCheck function checks whether any requests are pending. MpTweetCommandRequestDone indicates whether a request has been executed successfully and can inform the sender of this. In order for these two functions know which request to check, the "Preheat" ID defined in the configuration is specified on input "RequestID".

After the components are added, they are connected to each other as described under "Connection diagram" and then configured.

Switching on the heating must be implemented in the application, and might look something like this:

```plaintext
PROGRAM  _CYCLIC
    IF (MpTweetCommandRequestCheck(gTweet, 'Preheat')) THEN
        HeatingZone.Enable := TRUE;
        HeatingZone.SetTemperature := HMI.SetTemperature;
    END_IF
    IF (MpTweetCommandRequestCheck(gTweet, 'Preheat')) AND (HeatingZone.TemperatureReached) THEN
        MpTweetCommandRequestDone(gTweet, 'Preheat', mpTWEET_CMD_REQUEST_SUCCESS);
    END_IF
END_PROGRAM
```

1.4.2 Use case 2: Transmitting an alarm notification

Requirement

A machine fills a soft drink into bottles. The machine operator needs to ensure that the tank containing the soft drink does not drop below a lower limit value. Since the machine operator is responsible for multiple machines at the same time, he needs to be notified via text message as soon as the fill level drops below the limit. This allows him to refill the tank before it becomes necessary to stop the filling process.
Solution

Component list

- **MpAlarmXCore** (own MpLink): Collects all alarms from mapp components
- **MpAlarmXSet** (MpLink from MpAlarmXCore): Sets a user alarm
- **MpAlarmXReset** (MpAlarmXReset from MpAlarmXCore): Resets a user alarm
- **MpTweetCore** (own MpLink): This component establishes a connection to a specified modem that makes it possible to transmit and receive messages.

Connection diagram

![Connection Diagram](image)

Alarm configuration

The MpAlarmXCore configuration is added.

User alarm "LowProduct" is created. In order for a message to be sent when an alarm occurs, action "Send message" must be selected in the alarm configuration under *Mapping*. This will send the message with the text specified in the alarm configuration under *Message*.

![Alarm Configuration](image)

MpTweet configuration

The MpTweet configuration is added.

In order for a machine operator to receive the alarm message, they must be specified in the *Receiver List*. It is important that the recipient is also specified under *Subscriptions and hierarchy*, since alarm messages are sent to the recipients in this list.

![MpTweet Configuration](image)
Using the mapp components

The **MpTweetCore** component is added in order to establish a connection with the modem specified in the configuration. For information on how to configure the modem, see section SiteManager configuration.

The components **MpAlarmXCore**, **MpAlarmXSet** and **MpAlarmXReset** are added for alarm management.

After the components are added, they are connected to each other as described under "Connection diagram" and then configured.

For information on how to enable and disable alarms, see the use case User alarm management.

If the "ProductLow" alarm is active, the machine operator receives a text message containing the alarm text specified in the alarm configuration.

### 1.5 Libraries

#### 1.5.1 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpTweetCore</td>
<td>This function block is used to establish a connection with a modem and to send and receive messages.</td>
</tr>
<tr>
<td>MpTweetCoreUI</td>
<td>This function block allows you to create or edit new recipients via the HMI application. The notification order for alarms can also be edited here.</td>
</tr>
<tr>
<td>MpTweetSendMessage</td>
<td>This function is used to send messages.</td>
</tr>
<tr>
<td>MpTweetCommandRequestCheck</td>
<td>This function checks whether a command sent via text message has been received.</td>
</tr>
<tr>
<td>MpTweetCommandRequestDone</td>
<td>This function checks whether a command received via text message has been processed.</td>
</tr>
</tbody>
</table>

#### 1.5.1.1 MpTweetCommandRequestCheck

This function checks whether a command has been received via text message.
Function

MpComIdentType MpTweetCommandRequestCheck
MpLink STRING[255] RequestID
BOOL Return value

Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>RequestID</td>
<td>STRING[255]</td>
<td>Type of request to be checked for. This type is previously defined via &quot;RequestID&quot; in the configuration.</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>BOOL</td>
<td>Specifies whether a request of this type is pending.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.1 Description

This function uses the MpLink of an MpTweetCore configuration. The user can predefine various requests in the configuration. Examples of possible requests:

- What is the machine's current PackML state
- Is the machine running or stopped
- What is the current production data
- What alarms are currently active
- Request to initialize the machine
- Request to reduce the production speed

This function checks whether a certain request has been received. The type of request is specified using parameter "RequestID". The return value informs the user whether a request of this type is currently pending.

1.5.1.2 MpTweetCommandRequestDone

This function checks whether a command received via text message has been processed.

Function

MpComIdentType MpTweetCommandRequestDone
MpLink STRING[255] RequestID
DINT Return value

Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>RequestID</td>
<td>STRING[255]</td>
<td>RequestID of the request to be confirmed. Defined in the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>ReturnType</td>
<td>MpTweetCommandRequestEnum</td>
<td>Specifies whether the transmission was successful</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>DINT</td>
<td>Returns status information.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).
For mapp function blocks, asynchronous handling does not have to be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.2.1 Description

This function uses the MpLink of an MpTweetCore configuration. This function informs the sender of a request about the status of the request. The various possible requests are defined in the configuration. The MpTweetCommandRequestCheck function can be used at runtime to check whether a certain request has been received. The reaction to a request must be defined. For each request, whether it queries a status or issues a command, there is corresponding feedback for the sender. This, too, must be defined in the configuration. This function enables the sending of a response for a specific request ("RequestId"). The process could look like this:

1. At the beginning of the shift, the operator sends the request "WarmUp" to the machine.
2. The machine checks regularly for pending requests (using the MpTweetCommandRequestCheck function).
3. The machine detects that the operator has sent the "WarmUp" request.
4. In response, the machine's various heating zones are heated up to their setpoint values.
5. As soon as the temperature setpoint has been reached, a feedback message is sent to the operator (using the MpTweetCommandRequestDone function).
6. The operator receives a message with the information "WarmUpDone".

1.5.1.3 MpTweetCore

This function block is used to establish a connection with a modem and to send and receive messages.

**Function block**

<table>
<thead>
<tr>
<th>Optional parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MpTweetCore</strong></td>
</tr>
<tr>
<td>&amp;MpComIdentType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MpLink</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MP &amp;MpTweetCoreParType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MpTweetCoreInfoType</td>
</tr>
<tr>
<td>MpComIdentType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MP &amp;MpTweetCoreParType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MpTweetCoreInfoType</td>
</tr>
</tbody>
</table>
### Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>IN</td>
<td>Parameters</td>
<td>Pointer to MpTweetCoreParaType</td>
<td>Function block parameters.</td>
</tr>
<tr>
<td>IN</td>
<td>Update</td>
<td>BOOL</td>
<td>Updates the parameters.</td>
</tr>
<tr>
<td>IN</td>
<td>Send</td>
<td>BOOL</td>
<td>Command to transmit a message.</td>
</tr>
<tr>
<td>IN</td>
<td>Ping</td>
<td>BOOL</td>
<td>Checks the connection to the modem.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>UpdateDone</td>
<td>BOOL</td>
<td>Parameter update completed.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>PingDone</td>
<td>BOOL</td>
<td>Indicates whether the ping command was successful.</td>
</tr>
<tr>
<td>OUT</td>
<td>MessageSent</td>
<td>BOOL</td>
<td>Message was transmitted successfully</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpTweetCoreInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

### mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

#### 1.5.1.3.1 Description

The MpLink of an MPTweetCore configuration is used for this function block.

This component can be used to establish a connection with a configured modem. This connection makes it possible to send and receive messages. When MpTweetCore is enabled using input "Enable", then the connection to the server/modem is established automatically. While a connection is being established, this is indicated on output "StatusID". Output "Active" of the component indicates when the connection has been established successfully. If the connection is lost later on, this is indicated by an error on output "StatusID".

#### Checking the connection via ping

The modem/server that MpTweetCore should connect with and the corresponding settings are entered in the configuration of MpTweetCore. The "Ping" command can be used to check the connection with the configured modem. Output "PingDone" indicates whether the connection with the server/modem has been established successfully.

#### Sending and receiving messages

MpTweetCore checks regularly for pending requests. The receipt interval (default = 60 seconds) can be changed in the configuration. In addition to receiving messages, this function block can also send messages. To do this, define the message and recipient in the parameter structure. You can either enter the message directly or specify a MessageID. This must first be created in the configuration. In this case, the configuration also contains the text that corresponds with the MessageID, which can be localized using the text system. The recipient can either be specified directly by entering a telephone number or defined previously in the configuration. If the telephone number is specified directly, then it can be defined in various formats. For supported formats, see MpTweetCore configuration under "Recipient".

#### Additional information

Additional information can be viewed using the Info structure.

- **Sent and received messages**: Parameter "SentMessages" shows how many messages have been sent since MpTweetCore was enabled. "ReceivedMessages" shows how many messages have been received via MpTweetCore during the same time period.
- **Pending request**: Parameter "PendingRequests" shows how many requests are still pending.
- **Current messages**: Parameter "Messages" shows the current messages. The last 10 pending requests are shown. If there are more than 10 pending requests, then the oldest is discarded when a new request is received. The last 10 received messages are managed in the "Info" structure according to the same principle.
1.5.1.4 MpTweetCoreUI

This function block allows you to create or edit new recipients via the HMI application. The notification order for alarms can also be edited here.

Function block

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType</td>
<td>MpLink</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration).</td>
</tr>
<tr>
<td>BOOL Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>BOOL ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>&amp;MpTweetCoreUIConnectType</td>
<td>UIConnect</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</td>
</tr>
<tr>
<td>BOOL Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>BOOL Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>DINT StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>MpTweetInfoType</td>
<td></td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.4.1 Description

The MpLink for this function block is the same that is used for function block MpTweetCore. The component represents a connection between a messaging system and an HMI application. Data is exchanged between MpTweetCoreUI and the HMI application using structure MpTweetCoreUIConnectType. An MpTweetCore component must be active to use MpTweetCoreUI.

MpTweetCoreUIConnectType

Structure "UIConnect" is divided into the following areas:

- **Receiver**: Structure MpTweetUIReceiverType provides a table listing existing recipients and their telephone numbers. It is also possible to add, edit or delete recipients.

- **SubscriptionsAndHierarchy**:MpTweetUISubscriptionType provides a table listing recipients of alarm notifications. The notification order and recipient settings can be edited here.

MpTweetCoreUISetupType

"ReceiverListSize" in the structure defines how many recipient are displayed on one page of the HMI application. "ReceiverListScrollWindow" determines how many entries from the list are initially displayed when scrolling up and down.

Parameter "Confirmation" can be used to enable or disable various dialog boxes.

1.5.1.5 MpTweetSendMessage

This function is used to send messages.
Function

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpLink</td>
<td>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration)</td>
</tr>
<tr>
<td>Message</td>
<td>STRING[255]</td>
<td>Here you can either enter the message to be sent directly, or specify a previously configured &quot;MessageID&quot; that references a text in the configuration.</td>
</tr>
<tr>
<td>Receiver</td>
<td>STRING[100]</td>
<td>Here you can either enter a telephone number or reference a previously configured recipient.</td>
</tr>
</tbody>
</table>

Return value: DINT

Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpTweetCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Message</td>
<td>STRING[255]</td>
<td>Here you can either enter the message to be sent directly, or specify a previously configured &quot;MessageID&quot; that references a text in the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>Receiver</td>
<td>STRING[100]</td>
<td>Here you can either enter a telephone number or reference a previously configured recipient.</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>DINT</td>
<td>Returns status information.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.5.1 Description

This function uses the MpLink of an MpTweetCore configuration.

There are two ways to use parameter "Message". The first is to enter the message text directly. This is then sent directly to the recipient. The second way is to reference a MessageID, which must first be defined in the configuration. The configuration also contains the text that corresponds with the MessageID, which can be localized using the text system.

Parameter "Receiver" defines the recipient of the message. The recipient can either be specified directly by entering a telephone number or defined previously in the configuration.

If the telephone number is specified directly, then it can be defined in various formats. For supported formats, see MpTweetCore configuration under "Recipient".

1.5.2 Data types and enumerators

1.5.2.1 Data types

1.5.2.1.1 MpTweetCoreInfoType

This data type provides additional information for the MpTweetCore component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentMessages</td>
<td>UDINT</td>
<td>Number of sent messages</td>
</tr>
<tr>
<td>ReceivedMessages</td>
<td>UDINT</td>
<td>Number of received messages</td>
</tr>
<tr>
<td>PendingRequests</td>
<td>USINT</td>
<td>Number of pending requests</td>
</tr>
<tr>
<td>Messages</td>
<td>MpTweetInfoMessageType</td>
<td>Information about the messages</td>
</tr>
<tr>
<td>Diag</td>
<td>MpTweetDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

1.5.2.1.2 MpTweetCoreParType

Parameter structure for MpTweetCore.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>MpTweetSendMessageType</td>
<td>Specifies the message</td>
</tr>
<tr>
<td>PingTimeout</td>
<td>TIME</td>
<td>Timeout for a “ping” attempt</td>
</tr>
</tbody>
</table>

1.5.2.1.3 MpTweetCoreUIConnectType

This data type can be used to create a connection between MpTweetCore and the HMI application.
### Parameter Data type Description

<table>
<thead>
<tr>
<th>Status</th>
<th>MpTweetCoreUIStatusEnum</th>
<th>Current operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver</td>
<td>MpTweetUIReceiverType</td>
<td>List of recipients</td>
</tr>
<tr>
<td>SubscriptionsAndHierarchy</td>
<td>MpTweetUISubscriptionType</td>
<td>List of recipients for alarm notification</td>
</tr>
<tr>
<td>Message box</td>
<td>MpTweetUIMessageBoxType</td>
<td>Controls dialog boxes</td>
</tr>
<tr>
<td>DefaultLayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer of the visualization page where the recipe management system is being displayed</td>
</tr>
</tbody>
</table>

**VC4 connection:** StatusDatapoint from Layer

### 1.5.2.1.4 MpTweetCoreUISetupType

**Additional configuration options for the HMI application**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReceiverListSize</td>
<td>UINT</td>
<td>20</td>
<td>Number of recipients to be displayed on one page of the list in the HMI application</td>
</tr>
<tr>
<td>ReceiverListScrollWindow</td>
<td>USINT</td>
<td>1</td>
<td>Determines how many entries from the list are initially displayed when scrolling up and down</td>
</tr>
<tr>
<td>Confirmation</td>
<td>MpTweetUISetupConfirmType</td>
<td></td>
<td>Displays the confirmation window</td>
</tr>
</tbody>
</table>

### 1.5.2.1.5 MpTweetDiagType

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusID</td>
<td>MpTweetStatusIDType</td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>

### 1.5.2.1.6 MpTweetInfoMessageType

This data type provides more detailed information about the current messages.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PendingRequests</td>
<td>ARRAY[0..9] of MpTweetPendingRequestType</td>
<td>Information about pending requests</td>
</tr>
<tr>
<td>Received</td>
<td>ARRAY[0..9] of MpTweetReceivedMessageType</td>
<td>Information about received requests</td>
</tr>
<tr>
<td>PendingConfirmationReceipts</td>
<td>ARRAY[0..9] of MpTweetPendingConfirmationType</td>
<td>Information about pending confirmations</td>
</tr>
</tbody>
</table>

### 1.5.2.1.7 MpTweetInfoType

This data type provides additional information for the MpTweetCoreUI component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diag</td>
<td>MpTweetDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

### 1.5.2.1.8 MpTweetPendingConfirmationType

This data type is used to configure the requested confirmation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>STRING[100]</td>
<td>Specifies the sender</td>
</tr>
<tr>
<td>TimeOut</td>
<td>BOOL</td>
<td>Indicates whether the timeout for a confirmation has expired. If TRUE, then the message is sent to the next recipient.</td>
</tr>
<tr>
<td>Receiver</td>
<td>STRING[100]</td>
<td>Specifies the recipient</td>
</tr>
<tr>
<td>SentTimeStamp</td>
<td>DATE_AND_TIME</td>
<td>Timestamp for the first time the message was sent</td>
</tr>
</tbody>
</table>

### 1.5.2.1.9 MpTweetPendingRequestType

This data type shows a current request.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestID</td>
<td>STRING[320]</td>
<td>Active RequestID</td>
</tr>
<tr>
<td>RequestMessage</td>
<td>STRING[320]</td>
<td>Message that was requested externally</td>
</tr>
<tr>
<td>ReturnMessage</td>
<td>STRING[320]</td>
<td>Message sent in response to request</td>
</tr>
<tr>
<td>ReceivedTimeStamp</td>
<td>DATE_AND_TIME</td>
<td>Time that the request was received</td>
</tr>
<tr>
<td>Sender</td>
<td>STRING[100]</td>
<td>Sender of the request (ReceiverID or telephone number)</td>
</tr>
</tbody>
</table>

### 1.5.2.1.10 MpTweetReceivedMessageType

This data type contains information about a message to be received
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>STRING[100]</td>
<td>Specifies the sender (ReceiverID or telephone number)</td>
</tr>
<tr>
<td>Message</td>
<td>STRING[255]</td>
<td>Specifies the message</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>DATE_AND_TIME</td>
<td>Timestamp for when the message was received</td>
</tr>
</tbody>
</table>

### 1.5.2.1.11 MpTweetSendMessageType

This data type is used to configure the message to be sent.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Receiver      | STRING[100]    | Specifies the recipient (telephone number or recipient defined in the configu-
|               |                | ration).                                                                    |
| Message       | STRING[255]    | Specifies the message (the message itself, or a MessageID to reference a tex-
|               |                | t in the configuration)                                                     |

### 1.5.2.1.12 MpTweetStatusIDType

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>MpTweetErrorEnum</td>
<td>Error code for mapp component</td>
</tr>
</tbody>
</table>
| Severity      | MpComSeveritiesEnum | Describes the type of information supplied by the status ID (success, infor-
|               |                | mation, warning, error)                                                    |
| Code          | UINT            | Code for the status ID. This error number can be used to search for addi-
|               |                | tional information in the help system.                                    |

### 1.5.2.1.13 MpTweetUIMessageBoxType

Data type used to control the display of a dialog box.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Visibility of the dialog box (status data point for the dialog box layer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: StatusDatapoint from Layer</td>
</tr>
<tr>
<td>Type</td>
<td>MpTweetUIMessageEnum</td>
<td>Type of dialog box</td>
</tr>
</tbody>
</table>
| ErrorNumber   | UINT            | Current error number to be displayed (corresponds to the contents of "Di-
|               |                | ag.StatusID.Code")                                                          |
|               |                | VC4 connection: Datapoint from Numeric                                       |
| StatusID      | DINT            | Current error number to be displayed (corresponds to the contents of "Sta-
|               |                | tusID")                                                                     |
|               |                | VC4 connection: Datapoint from Numeric                                       |
| Confirm       | BOOL            | Confirms the operation                                                       |
| Cancel        | BOOL            | Cancels the operation                                                        |
1.5.2.1.14 MpTweetUIReceiptConfirmationType

This data type defines the response to an alarm notification.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpectedResponse</td>
<td>STRING[20]</td>
<td>Expected response</td>
</tr>
<tr>
<td>ResponseTimeout</td>
<td>USINT</td>
<td>Defines the response time. When this time expires, the alarm notification is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sent to the same or next defined recipient [min].</td>
</tr>
<tr>
<td>Attempts</td>
<td>USINT</td>
<td>Specifies how often an alarm notification should be transmitted to a recipient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>after the response time expires</td>
</tr>
<tr>
<td>Lock</td>
<td>BOOL</td>
<td>If mode &quot;ReceiveOnly&quot; is being used for alarm notifications for a selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user, this parameter can be used to block out parameters &quot;ExpectedResponse&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;ResponseTimeout&quot; and &quot;Attempts&quot;</td>
</tr>
</tbody>
</table>

VC4 connection: LockingDatapoint from String and Numeric/ Locking: Datapoint >= Level / Level: 1

1.5.2.1.15 MpTweetUIReceiverCreateDlgType

Data type used to control the display of a dialog box.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Visibility of the dialog box (status data point for the dialog box layer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection</strong>: StatusDatapoint from Layer</td>
</tr>
<tr>
<td>Details</td>
<td>MpTweetUIReceiverInfoType</td>
<td>Information about the selected recipient</td>
</tr>
<tr>
<td>Confirm</td>
<td>BOOL</td>
<td>Confirms the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection</strong>: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
<tr>
<td>Cancel</td>
<td>BOOL</td>
<td>Cancels the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection</strong>: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
</tbody>
</table>
1.5.2.1.16 MpTweetUIReceiverCreateType

Combines the display and data for a dialog box to create a recipient

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowDialog</td>
<td>BOOL</td>
<td>Command that opens the dialog box</td>
</tr>
<tr>
<td>Dialog box</td>
<td>MpTweetUIReceiverCreateDlgType</td>
<td>Dialog box for creating a recipient</td>
</tr>
</tbody>
</table>

1.5.2.1.17 MpTweetUIReceiverInfoType

This data type contains detailed information about the recipient.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING[100]</td>
<td>Name of the recipient</td>
</tr>
<tr>
<td>PhoneNumber</td>
<td>STRING[20]</td>
<td>Phone number of the recipient</td>
</tr>
<tr>
<td>Language</td>
<td>STRING[20]</td>
<td>Language of the recipient</td>
</tr>
</tbody>
</table>

1.5.2.1.18 MpTweetUIReceiverListType

This data type contains a list of current recipients and associated navigation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReceiverNames</td>
<td>ARRAY[0..19] of STRING[100]</td>
<td>Name of the recipient</td>
</tr>
<tr>
<td>MaxSelection</td>
<td>UINT</td>
<td>Index of the last recipient</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>UINT</td>
<td>Index of the currently selected recipient</td>
</tr>
<tr>
<td>PageUp</td>
<td>BOOL</td>
<td>Jumps to the start of the current page and then scrolls up one page at a time</td>
</tr>
<tr>
<td>PageDown</td>
<td>BOOL</td>
<td>Jumps to the end of the current page and then scrolls down one page at a time</td>
</tr>
<tr>
<td>StepUp</td>
<td>BOOL</td>
<td>Selects the previous entry in the list</td>
</tr>
<tr>
<td>StepDown</td>
<td>BOOL</td>
<td>Selects the next entry in the list</td>
</tr>
<tr>
<td>RangeStart</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed. Used for the starting value of &quot;Range&quot; for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. &quot;Range&quot; should correspond to the number of entries that are displayed on one page.</td>
</tr>
<tr>
<td>RangeEnd</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed. Used for the ending value of &quot;Range&quot; for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. &quot;Range&quot; should correspond to the number of entries that are displayed on one page.</td>
</tr>
</tbody>
</table>

1.5.2.1.19 MpTweetUIReceiverType

This data type contains a list with information about the recipient.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>MpTweetUIReceiverListType</td>
<td>List of all recipients and associated navigation</td>
</tr>
<tr>
<td>Info</td>
<td>MpTweetUIReceiverInfoType</td>
<td>Specifies information about a selected recipient</td>
</tr>
<tr>
<td>Create</td>
<td>MpTweetUIReceiverCreateType</td>
<td>Creates a new recipient</td>
</tr>
<tr>
<td>Edit</td>
<td>MpTweetUIReceiverCreateType</td>
<td>Edits an existing recipient</td>
</tr>
<tr>
<td>Delete</td>
<td>BOOL</td>
<td>Deletes the selected recipient</td>
</tr>
</tbody>
</table>

1.5.2.1.20 MpTweetUISetupConfirmType

This data type can be used to define when confirmation windows should be displayed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReceiverCreate</td>
<td>BOOL</td>
<td>Enables a confirmation window when creating a recipient</td>
</tr>
<tr>
<td>ReceiverDelete</td>
<td>BOOL</td>
<td>Enables a confirmation window when deleting a recipient</td>
</tr>
<tr>
<td>ReceiverEdit</td>
<td>BOOL</td>
<td>Enables a confirmation window when editing a recipient</td>
</tr>
</tbody>
</table>

1.5.2.1.21 MpTweetUISubscriberType

This data type contains detailed information about the recipient of alarm notifications.
1.5.2.1.22 MpTweetUISubscriptionAlarmType
This data type contains a list of alarm notification recipients.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscriber</td>
<td>ARRAY[0..19] of MpTweetUISubscriberType</td>
<td>List of all available recipients</td>
</tr>
<tr>
<td>EditCompletion</td>
<td>BOOL</td>
<td>If a parameter in the alarm notification is changed, e.g. &quot;Attempts&quot;, this change must be applied in the MpTweet configuration. &quot;EditCompletion&quot; triggers this process. This parameter is connected to the completion data point of a control. This could be a Numeric or String control, for example.</td>
</tr>
</tbody>
</table>

1.5.2.1.23 MpTweetUISubscriptionType
This data type specifies alarm notification recipients and the order in which they are notified.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>MpTweetUISubscriptionAlarmType</td>
<td>List of the notification order when alarms occur</td>
</tr>
</tbody>
</table>

1.5.2.2 Enumerators

1.5.2.2.1 MpTweetCommandRequestEnum
This enumerated data type specifies how the response to a request was sent.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpTWEET_CMD_REQUEST_SUCCESS</td>
<td>Response to request sent successfully</td>
</tr>
</tbody>
</table>

1.5.2.2.2 MpTweetCoreUILanguageEnum
This enumerated data type indicates the language that should be used.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpTWEET_CORE_UI_EN</td>
<td>English</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_DE</td>
<td>German</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_ES</td>
<td>Spanish</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_FR</td>
<td>French</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_USER1</td>
<td>User-defined language</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_USER2</td>
<td>User-defined language</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_USER3</td>
<td>User-defined language</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_USER4</td>
<td>User-defined language</td>
</tr>
</tbody>
</table>

1.5.2.2.3 MpTweetCoreUIStatusEnum
This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpTWEET_CORE_UI_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_ACTIVE</td>
<td>A process is currently active.</td>
</tr>
<tr>
<td>mpTWEET_CORE_UI_ERROR</td>
<td>The last operation triggered an error.</td>
</tr>
</tbody>
</table>

1.5.2.2.4 MpTweetUIMessageEnum
This enumerated data type indicates the task of the dialog box.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpTWEET_UI_MSG_NONE</td>
<td>No dialog box</td>
</tr>
<tr>
<td>mpTWEET_UI_MSG_ERROR</td>
<td>Dialog box: Error</td>
</tr>
<tr>
<td>mpTWEET_UI_MSG_CONFIRM_DELETE</td>
<td>Dialog box: Confirmation of user deletion</td>
</tr>
<tr>
<td>mpTWEET_UI_MSG_CONFIRM_CREATE</td>
<td>Dialog box: Confirmation of user creation</td>
</tr>
<tr>
<td>mpTWEET_UI_MSG_CONFIRM_EDIT</td>
<td>Dialog box: Confirmation of user edit</td>
</tr>
</tbody>
</table>

1.5.2.2.5 MpTweetUIStatusEnum
This enumerated data type provides information about the current activity of a component.
### 1.5.2.2.6 MpptweetUISubscriberTypeEnum

This enumerated data type specifies how a received message should be responded to.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpTWEET_RECEIVE_ONLY</td>
<td>Only receive messages</td>
</tr>
<tr>
<td>mpTWEET_CONFIRMATION_OF_RECEIPT</td>
<td>The received message should be responded to.</td>
</tr>
</tbody>
</table>

### 1.5.3 Status numbers

#### 1.5.3.1 1083355143: MpTweetCore not yet active

**Description:**
This component can only become active if MpTweetCore is active.

**Reaction:**
The function block indicates this information on output "StatusID".

**Cause/Solution:**
- Enable MpTweetCore.

**Constant:**
mpTWEET_INF_WAIT_CORE_FB

**These function blocks / functions can report this error:**
- MpTweetCoreUI

#### 1.5.3.2 1083355138: Connection being established

**Description:**
The connection to the server/modem is currently being established.

**Reaction:**
The function block indicates this information on output "StatusID".

**Cause/Solution:**
- Switch on the modem/server

**Constant:**
mpTWEET_INF_WAIT_SERVER

**These function blocks / functions can report this error:**
- MpTweetCore

#### 1.5.3.3 -1064239103: Could not create component

**Description:**
The mapp component could not be created and is not enabled. See the Logger for additional information.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
- Service for configuring the function block not available -> Problem with MpCom
- Unable to read registry -> Problem with MpCom
- Details about the cause of error in the logger

These function blocks / functions can report this error:
- MpTweetCore

Constant:
mpTWEET_ERR_ACTIVATION

1.5.3.4 -1064239102: MpLink is null pointer

Description:
Input "MpLink" is not connected, null pointer.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check input "MpLink" on the function block.

These function blocks / functions can report this error:
- MpTweetCore
- MpTweetCommandRequestCheck
- MpTweetCommandRequestDone
- MpTweetSendMessage

Constant:
mpTWEET_ERR_MPLINK_NULL

1.5.3.5 -1064239101: MpLink connection not permitted

Description:
The value on input "MpLink" is not allowed.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The component is connected directly to mpCOM_MAIN or mpCOM_STANDALONE. This is not supported.

These function blocks / functions can report this error:
- MpTweetCore
- MpTweetCommandRequestCheck
- MpTweetCommandRequestDone
- MpTweetSendMessage

Constant:
mpTWEET_ERR_MPLINK_INVALID
1.5.3.6 -1064239100: MpLink modified

Description:
The value on input "MpLink" was modified while the components were running ("Enable" = TRUE).

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• The value of input "MpLink" can only be changed while the component is inactive ("Enable" = FALSE).

These function blocks / functions can report this error:
• MpTweetCore
• MpTweetCommandRequestCheck
• MpTweetCommandRequestDone
• MpTweetSendMessage

Constant:
mpTWEET_ERR_MPLINK_CHANGED

1.5.3.7 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" on the function block input is invalid.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Writing to the values in structure MpComIdentType is not permitted.
• A corresponding configuration for this mapp component must be available (see first paragraph of the description for the functions / function blocks listed below).

These function blocks / functions can report this error:
• MpTweetCore
• MpTweetCommandRequestCheck
• MpTweetCommandRequestDone
• MpTweetSendMessage

Constant:
mpTWEET_ERR_MPLINK_CORRUPT

1.5.3.8 -1064239098: MpLink already in use

Description:
This MpLink is already in use.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:

- If "Overload" is the selected download mode, then the mapp component should be disabled in the exit subroutine with "Enable" = FALSE. It is also possible to use "Copy" or "One cycle" mode. In these cases, it is not necessary to disable the mapp component.
- Check which components are already using this MpLink.
- Create a new MpLink.

These function blocks / functions can report this error:

- MpTweetCore
- MpTweetCommandRequestCheck
- MpTweetCommandRequestDone
- MpTweetSendMessage
- MpTweetCoreUI

Constant:

mpTWEET_ERR_MPLINK_IN_USE

1.5.3.9 -1064239097: Parameter structure is a null pointer

Description:
Input "Parameter" is not connected, null pointer.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:

- Check input "Parameter" on the function block.

These function blocks / functions can report this error:

- MpTweetCore

Constant:

mpTWEET_ERR_PAR_NULL

1.5.3.10 -1064239091: Invalid configuration

Description:
Could not read configuration while creating components. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:

- The configuration is damaged.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:

- MpTweetCore

Constant:

mpTWEET_ERR_CONFIG_INVALID
1.5.3.11 -1064128512: Invalid recipient

Description:
The specified recipient is invalid.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check the telephone number.
- Check the ReceiverID (for recipients defined in the configuration)

Constant:
mpTWEET_ERR_RECEIVER_INVALID

These function blocks / functions can report this error:
- MpTweetCore
- MpTweetSendMessage

1.5.3.12 -1064128511: Invalid request

Description:
An unknown/invalid request has been received

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The request is not listed in the configuration.
- The sender should check the syntax of the message.

Constant:
mpTWEET_ERR_REQUEST_INVALID

These function blocks / functions can report this error:
- MpTweetCore

1.5.3.13 -1064128509: Could not send message

Description:
The message could not be sent.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check the connection to the server/modem.

Constant:
mpTWEET_ERR_MESSAGE_SEND_FAIL
These function blocks / functions can report this error:
  • MpTweetCore

1.5.3.14 -1064128508: Ping not successful

Description:
The server/modem did not respond within the defined timeout.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
  • Increase the timeout value and try again.
  • Check the connection to the server/modem.

Constant:
mpTWEET_ERR_PING_TIMEOUT

These function blocks / functions can report this error:
  • MpTweetCore

1.5.3.15 -1064128507: Lost communication to modem

Description:
The connection to the server/modem was lost.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
  • Check the connection to the server/modem.

Constant:
mpTWEET_ERR_COMMUNICATION_FAIL

These function blocks / functions can report this error:
  • MpTweetCore

1.5.3.16 -1064128506: Missing value on UIConnect

Description:
NULL was appended to "UIConnect".

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
  • Input "UIConnect" forgotten

Constant:
mpTWEET_ERR_MISSING_UICONNECT
These function blocks / functions can report this error:

- MpTweetCoreUI

1.5.4 Alarms

1.5.4.1 mpTWEET_ALM_MESSAGE_SEND_FAIL: Failed to send message

Description:
The attempt to send a message failed.

Reaction:
A mapp alarm is triggered if a MpAlarmX component is active.

Cause/Solution:
- Check the connection to the server ("ping").
- Check the recipient.
- Check the "MessageID".

1.5.4.2 mpTWEET_ALM_PING_TIMEOUT: Ping attempt timed out

Description:
The ping attempt timed out.

Reaction:
A mapp alarm is triggered if a MpAlarmX component is active.

Cause/Solution:
- Check the connection to the server ("ping").