mapp Services

1 mapp Audit: Event management

mapp Audit can log different events. These can come from the HMI application, MpUser or user-defined events. The user can determine the format in which the events are stored in a file.

MpAudit provides many functions that make it easier to implement 21 CFR Part 11 requirements (e.g. logging unauthorized access attempts).

On our YouTube channel, the tutorials [mapp Audit - Part 1](#) and [mapp Audit - Part 2](#) explain how event management can be implemented.

Our B&R online tutorial platform includes tutorials about [mapp Audit](#).

1.1 Concept

### User stories

**OEM**

- As an OEM, I would like my machine to meet all FDA regulations; I am supported by MpAudit here.
- As an OEM, I would like a list of all events that occur on my machine throughout the day/week (e.g. user logged in, user started machine, user changed speed from 22.0 to 25.0, user logged out).
- As an OEM, I would like for the audit trail to be stored so that it cannot be manipulated.
- As an OEM, I would like for all events to be archived.
- As an OEM, I would like to be able to view informative event texts in my own language.

**Service technician**

- As a service technician, I would like during a service call to be able to see the last events that occurred on the machine. This makes it possible for me to find possible changes to certain parameters or similar relevant data that could be tied to the cause of the service call.

**Application engineer**

- As an application engineer, I would like to be able to record various events in my audit trail. This includes OPC UA events, value changes, actions on the HMI system, actions in user management, etc.
- As an application engineer, I would like to be able to generate an event at any time.
- As an application engineer, I do not want to have to worry about value changes, OPC UA events or actions on the HMI system.

#### 1.1.1 Event types

In order for different event types to be recorded, they must be enabled in the advanced properties of the MpAudit configuration:
In most cases, the event details should be included in the event. This can be done using token \{\par\} or [%par]. Token "par" stands for different parameters that are identified by a unique identifier.

The following tables show which components must be used to record the desired event types. They also list which tokens can be used with each event type.

If one of the tokens is given after the & or % character, then the respective information is used when the entries are generated. The text comes from either the configuration, the text system or a VC4-based HMI application. For more information about the various text sources, see here.

**Only one format can be used.**

This is either the new format (e.g. \{\par\}) or the old format (e.g. [%par]).

The two formats cannot be used together. For more information, see Using the new format.

**Value change**

This event type records changes to specified parameters.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MpAuditRegPar</td>
</tr>
<tr>
<td></td>
<td>Event ID</td>
</tr>
<tr>
<td></td>
<td>ev</td>
</tr>
</tbody>
</table>

Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.

Name of the user logged when the event occurred.

Name of the mapp component to which the event is assigned via MpLink.

**Table 1: Value change**
VC4 value change

The VC4 value change allows changes in the VC4-based HMI application to be recorded.

For more information, see section VC4 event system under MpAudit and "VC4 as the text source" in section Creating entries.

The unit cannot also be shown with value changes in VC4!

A use case is also available on this subject.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>MappAuditRegPar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dpid</th>
<th>Identification of the value that has changed. For all value changes, the value on input &quot;Identifier&quot; is used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>New value (&quot;New&quot;) on a value change.</td>
</tr>
<tr>
<td>old</td>
<td>Old value (&quot;Old&quot;) on a value change.</td>
</tr>
</tbody>
</table>

Examples of the new format:
-$format can be added after %par to define the format: $.
- "Value of {&dpid} changed. Old: {&old}.2f, New: {&new}.2f" evaluated: "Value of SetTemp changed. Old: 80.81, New: 90.00".

".2f" defines how many positions should be displayed after the decimal point.

Examples of the old format:
-
- ".2f" can be added after %par to define the format:
- "Value of [%dpid] changed. Old: [%old$2.1], New: [%new$2.2]" evaluated: "Value of SetTemp changed. Old: 80.1, New: 90.00".

$2.2 and $2.1 define how many positions should be displayed after the decimal point.

<table>
<thead>
<tr>
<th>Table 1: Value change</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>MappAuditVC4Event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>op</th>
<th>Name of the user logged when the event occurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev</td>
<td>Event ID.</td>
</tr>
<tr>
<td>idx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.</td>
</tr>
<tr>
<td>map</td>
<td>Name of the mapp component to which the event is assigned via MpLink.</td>
</tr>
<tr>
<td>dpid</td>
<td>In this case, either the user ID or description of the user ID is used depending on input &quot;Localize&quot; of function block MappAuditVC4Event. The description of the user ID is solved via an additional text group. This text group is connected to the data source:</td>
</tr>
<tr>
<td>new</td>
<td>New value (&quot;New&quot;) on a value change.</td>
</tr>
<tr>
<td>old</td>
<td>Old value (&quot;Old&quot;) on a value change.</td>
</tr>
<tr>
<td>unit</td>
<td>Index of the unit group / Text group that was used for input.</td>
</tr>
<tr>
<td>uext</td>
<td>Index within the unit group that identifies the exact unit that was used.</td>
</tr>
<tr>
<td>sel</td>
<td>Index within the text group that was used via a drop-down menu.</td>
</tr>
<tr>
<td>typ</td>
<td>Type (&quot;Type&quot;) triggered by a user-defined event.</td>
</tr>
</tbody>
</table>

Examples of the new format:

Examples of the old format:

<table>
<thead>
<tr>
<th>Table 2: VC4 value change</th>
</tr>
</thead>
</table>

User-defined event

This event type supports the recording of user-defined events.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>MappAuditCustomEvent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>op</th>
<th>Name of the user logged when the event occurred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev</td>
<td>Event ID.</td>
</tr>
<tr>
<td>idx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.</td>
</tr>
<tr>
<td>map</td>
<td>Name of the mapp component to which the event is assigned via MpLink.</td>
</tr>
</tbody>
</table>

| Table 3: User-defined event |
mapp Services

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>MpAuditCustomEvent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>typ</th>
<th>Message (&quot;Type&quot;) triggered by a user-defined event</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>Message (&quot;Message&quot;) from a user-defined event</td>
</tr>
<tr>
<td>cnt</td>
<td>Comment (&quot;Comment&quot;) from a user-defined event</td>
</tr>
</tbody>
</table>

**Examples of the new format**
- "User-defined event: &[typ] enabled by &[msg]" evaluated: "User-defined event: Turbine 1 enabled by Motor 5".

**Examples of the old format**
- "User-defined event: [%typ] enabled by [%msg]" evaluated: "User-defined event: Turbine 1 enabled by Motor 5".

Table 3: User-defined event

**User management event**

This event type allows user management events such as deleting/adding a user to be recorded.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>MpUser components</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
</tr>
<tr>
<td>ev</td>
</tr>
<tr>
<td>idx</td>
</tr>
<tr>
<td>map</td>
</tr>
<tr>
<td>usr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>act</th>
<th>Action that was executed in a mapp component. The ID for the action is returned (e.g. 10 for &quot;Invalid username entered&quot;). For more information, see the following table.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Identification number (returned via %act)</td>
</tr>
<tr>
<td>Unknown action</td>
<td>0</td>
</tr>
<tr>
<td>User has logged in</td>
<td>1</td>
</tr>
<tr>
<td>User has logged out</td>
<td>2</td>
</tr>
<tr>
<td>User has been automatically logged out</td>
<td>3</td>
</tr>
<tr>
<td>User has been added</td>
<td>4</td>
</tr>
<tr>
<td>User has been deleted</td>
<td>5</td>
</tr>
<tr>
<td>User has been blocked</td>
<td>6</td>
</tr>
<tr>
<td>User has been unblocked</td>
<td>7</td>
</tr>
<tr>
<td>User has been blocked for entering an incorrect password too many times</td>
<td>8</td>
</tr>
<tr>
<td>User has changed the password</td>
<td>9</td>
</tr>
<tr>
<td>An invalid username has been entered</td>
<td>10</td>
</tr>
<tr>
<td>An invalid password has been entered</td>
<td>11</td>
</tr>
</tbody>
</table>

**Examples of the new format**
- "&[op] - Action: User &[usr] was deleted" evaluated: "Admin - Action: User Mike was deleted", meaning that user Admin deleted user Mike.

**Examples of the old format**
- "%[op] - Action: User %[usr] was deleted" evaluated: "Admin - Action: User Mike was deleted", meaning that user Admin deleted user Mike.

Table 4: User management event

**Electronic signature event**

This event type allows events involving the electronic signature to be recorded.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>MpUserSignature</td>
</tr>
<tr>
<td></td>
<td>MpUserXSignature</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmt</td>
</tr>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

Table 5: Electronic signature event
### mapp Services

#### Event ID

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>MpUserSignature</td>
</tr>
</tbody>
</table>

#### Components

- **idx**: Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.
- **err**: User management error number (can be used to display whether the username or password was incorrect, for example)
- **act**: Returns the action ID that must be specified in the configuration for a signature
- **typ**: Action that was executed in a mapp component. The ID for the action is returned (e.g. 2 for "Electronic signature canceled"). For more information, see the following table.

<table>
<thead>
<tr>
<th>Event</th>
<th>Identification number (returned via &quot;typ&quot;)</th>
<th>Event as text (returned via &quot;typt&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples of the new format

- "Signature ID {&act}: {&typ} ("{&typt}") " evaluated: "Signature ID 3: Electronic signature performed successfully (OK)", meaning that the signature with ID3 was done successfully.

Examples of the old format

- "Signature ID [\%act]: [\%typ] ([\%typt]) " evaluated: "Signature ID 3: Electronic signature performed successfully (OK)", meaning that the signature with ID3 was done successfully.

#### Table 5: Electronic signature event

### MpAlarmX event

This event type allows events involving the alarm system, such as when an alarm occurs, to be recorded. Option "Auditing" must be enabled in the MpAlarmXHistory configuration in order to record events:

![Config_1.mpalarmxhistory](image)

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>MpAlarmX components</td>
</tr>
</tbody>
</table>

#### Events

- **ev**: Event ID
- **idx**: Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.
- **name**: Alarm name
- **code**: Alarm code
- **stold**: Last state of the alarm

Table 6: MpAlarmX event
mapp Services

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
<th>MpAlarmX components</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>stnew</td>
<td>Current state of the alarm</td>
</tr>
<tr>
<td></td>
<td>sev</td>
<td>Alarm level (severity)</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>Time at which the alarm state changed. This can deviate from the time at which the event was logged.</td>
</tr>
</tbody>
</table>

**Example of the new format**


**Example of the old format**


Table 6: MpAlarmX event

**MpPackML mode change**

This event type allows MpPackML mode changes to be tracked.

Option "Auditing" must be enabled in the MpPackMLCore configuration in order to record events:

Table 7: MpPackML mode change

**MpPackML state change**

This event type allows MpPackML state changes to be tracked.

Option "Auditing" must be enabled in the MpPackMLCore configuration in order to record events:
Event ID
65

Components
MpPackML components

Events

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
<th>MpPackML components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev</td>
<td>Event ID</td>
<td></td>
</tr>
<tr>
<td>idx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>Old PackML state</td>
<td></td>
</tr>
<tr>
<td>resp</td>
<td>Source of the PackML mode or state change (e.g. &quot;MpPackMLBasicUI&quot;, &quot;MpPackMLMode&quot;, &quot;PackML tag structure&quot;)</td>
<td></td>
</tr>
<tr>
<td>new</td>
<td>New PackML state</td>
<td></td>
</tr>
</tbody>
</table>

Examples of the new format


Examples of the old format


Table 8: MpPackML state change

MpRecipe event

This event type allow events involving recipe management, such as which recipe was loaded, to be recorded.

Option "Auditing" must be enabled in the MpRecipeCsv configuration or MpRecipeXml configuration in order to record events:
mapp Services

![Image](image.png)

### MpRecipe components

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>MpRecipe components</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>op</th>
<th>Name of the user logged when the event occurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>idx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.</td>
</tr>
<tr>
<td>ev</td>
<td>Event ID</td>
</tr>
<tr>
<td>file</td>
<td>File that was used (e.g. recipe that was loaded)</td>
</tr>
<tr>
<td>dev</td>
<td>Data storage device that was used (e.g. USB flash drive from which a recipe was loaded)</td>
</tr>
<tr>
<td>act</td>
<td>Action that was executed in a mapp component. The ID for the action is returned (e.g. 1 for &quot;Completed loading of recipe&quot;). For more information, see the following table.</td>
</tr>
</tbody>
</table>

#### Event Identification number (returned via %act) Additional information

<table>
<thead>
<tr>
<th>Event</th>
<th>Identification number (returned via %act)</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>The loading of a recipe has been started.</td>
<td>0</td>
<td>• (%op)/[%op] - User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%file)/[%file] - Recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%dev)/[%dev] - Data storage device</td>
</tr>
<tr>
<td>The loading of a recipe has been completed.</td>
<td>1</td>
<td>• (%op)/[%op] - User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%file)/[%file] - Recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%dev)/[%dev] - Data storage device</td>
</tr>
<tr>
<td>Saving a recipe</td>
<td>2</td>
<td>• (%op)/[%op] - User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%file)/[%file] - Recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%dev)/[%dev] - Data storage device</td>
</tr>
<tr>
<td>Renaming a recipe</td>
<td>3</td>
<td>• (%op)/[%op] - User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%old)/[%old] - Old name of recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%new)/[%new] - New name of recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%dev)/[%dev] - Data storage device</td>
</tr>
<tr>
<td>Deleting a recipe</td>
<td>4</td>
<td>• (%op)/[%op] - User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%file)/[%file] - Recipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (%dev)/[%dev] - Data storage device</td>
</tr>
</tbody>
</table>

**Examples of the new format**

- "Recipe (%file) on data storage device (%dev) was saved" evaluated: "Recipe Cappuccino on data storage device USB1 was saved."

**Examples of the old format**

- "Recipe [%file] on data storage device [%dev] was saved" evaluated: "Recipe Cappuccino on data storage device USB1 was saved."

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>OPC UA value change</td>
</tr>
</tbody>
</table>

### OPC UA value change

This event type allows variables enabled by OPC UA to be monitored for value changes. For information about how to configure the OPC UA server, see section OPC UA events.

<table>
<thead>
<tr>
<th>op</th>
<th>Name of the user logged when the event occurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev</td>
<td>Event ID</td>
</tr>
<tr>
<td>idxx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1; the 50th event has index 50 and the 100th has index 100.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>OPC UA value change</td>
</tr>
</tbody>
</table>
**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>MpMap</td>
</tr>
</tbody>
</table>

**Data sheet V**

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>MpMapAuditStartBatch</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>2457</td>
<td>MpMapAuditClearBuffer</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>MpMapTweet</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>MpMap</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>MpMapAuditStartBatch</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>2457</td>
<td>MpMapAuditClearBuffer</td>
</tr>
</tbody>
</table>

**Event ID**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>MpMapTweet</td>
</tr>
</tbody>
</table>

**Examples of the new format**

- \"OPC UA change: Parameter \"Coffee\" changed from \{\&old\} to \{\&new\}\" evaluated: \"OPC UA change: Parameter \"Coffee\" changed from 10g to 5g\".

**Examples of the old format**

- \"OPC UA change: Parameter \"Coffee\" changed from \[%old\] to \[%new\]\" evaluated: \"OPC UA change: Parameter \"Coffee\" changed from 10g to 5g\".

**Batch event**

This event type allows the start of a new batch to be recorded.

**Clear buffer event**

This event type can be used to record when the buffer containing all currently logged events was reset.

**MpTweet event**

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

**Table 10: OPC UA value change**

**Table 11: Batch event**

**Table 12: Event**

**Option \"Auditing\" must be enabled in the MpTweetCore configuration in order to record events:**
### Table 13: MpTweet event

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
<td>MpTweet function blocks</td>
</tr>
</tbody>
</table>

#### Events

<table>
<thead>
<tr>
<th>IDx</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>idx</td>
<td>Returns the index of the currently generated event. The value is incremented on each new event. The first generated event in the event list thus has value 1, the 50th event has index 50 and the 100th has index 100.</td>
<td></td>
</tr>
<tr>
<td>msg</td>
<td>Message text</td>
<td></td>
</tr>
<tr>
<td>rxn</td>
<td>Name of the user who received a message</td>
<td></td>
</tr>
<tr>
<td>rxc</td>
<td>Telephone number of the user who received a message</td>
<td></td>
</tr>
<tr>
<td>txn</td>
<td>Name of the user who sent a message</td>
<td></td>
</tr>
<tr>
<td>txc</td>
<td>Telephone number of the user who sent a message</td>
<td></td>
</tr>
<tr>
<td>act</td>
<td>Action that was executed in a mapp component. The ID for the action is returned (e.g. 1 for &quot;Message sent&quot;). For more information, see the following table.</td>
<td></td>
</tr>
</tbody>
</table>

#### Additional information

<table>
<thead>
<tr>
<th>Event</th>
<th>Identification number (returned via %act)</th>
<th>Additional information</th>
</tr>
</thead>
</table>
| Sent message           | 1                                        | • (%msg)/[%msg] - Message text  
• (%rxn)/[%rxn] - Name of the user who received the message  
• (%rxc)/[%rxc] - Telephone number of the user who received the message  |
| Received message       | 2                                        | • (%msg)/[%msg] - Message text  
• (%txn)/[%txn] - Name of the user who sent the message  
• (%txc)/[%txc] - Telephone number of the user who sent the message  |

#### Examples of the new format

- "Message (%msg) sent to (%rxn)" evaluated: "Message Hello sent to Sam".

#### Examples of the old format

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

### 1.1.2 Creating entries

One entry is created for each event. The entry is managed by MpAudit and stored in memory. In order to store as many entries as possible, no unnecessary memory is used when a new event occurs. For example, only the raw data of the event is saved (event type, new value, old value, index, comment, etc.), not its entire text. A readable
entry is generated from the raw data only when the data is exported or when preparing for the HMI application. The text for each entry can come from two different sources for this. The source for the text is configured using `MpAuditTrailConfig`. The text source can be selected there in the configuration structure under "TextSource" (MpAuditTextSourceType).

1.1.2.1 Configuration as the text source

If the element `mpAUDIT_TEXT_SOURCE_NONE` is defined as the text source under "Type", then the text will come directly from the configuration. For each entry, the text will be added under "OutputFormat".

<table>
<thead>
<tr>
<th>Type</th>
<th>&quot;OutputFormatSource&quot;</th>
<th>&quot;OutputFormat&quot;</th>
<th>Entry</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpAUDIT_TEXT_SOURCE_NONE</code></td>
<td>&quot;&quot;</td>
<td>&quot;An entry has been created.&quot;</td>
<td>&quot;An entry has been created.&quot;</td>
<td>The text is taken directly from the configuration.</td>
</tr>
</tbody>
</table>

1.1.2.2 VC4 as the text source

If the element `mpAUDIT_TEXT_SOURCE_VC4` is defined as the text source under "Type", then the text will come from a text from a text group. The text is taken from the text group under "OutputFormatSource" (MpAuditTextSourceType, corresponds to "Format - Text group index" in the Automation Studio configuration) by its index. A "g" in front of the index differentiates between global text groups (shared resources) and local text groups (visualization resources). In the defined text group, the exact text index is specified using "OutputFormat" (MpAuditTextSourceType, corresponds to "Format - Text index" in the Automation Studio configuration).

All text groups:

Text group "EventSystem" as well as text group "LocalEvents":

<table>
<thead>
<tr>
<th>Type</th>
<th>&quot;OutputFormatSource&quot;</th>
<th>&quot;OutputFormat&quot;</th>
<th>Entry</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpAUDIT_TEXT_SOURCE_VC4</code></td>
<td>'g2'</td>
<td>'3'</td>
<td>&quot;Global Event 4&quot;</td>
<td>The &quot;g&quot; indicates a text group from the shared resources. The &quot;2&quot; after the &quot;g&quot; determines which text group. The &quot;3&quot; under &quot;OutputFormat&quot; specifies which text in the text group should be used.</td>
</tr>
<tr>
<td><code>mpAUDIT_TEXT_SOURCE_VC4</code></td>
<td>'g2'</td>
<td>'6'</td>
<td>&quot;Global Event 7&quot;</td>
<td>The &quot;g&quot; indicates a text group from the shared resources. The &quot;2&quot; after the &quot;g&quot; determines which text group. The &quot;3&quot; under &quot;OutputFormat&quot; specifies which text in the text group should be used.</td>
</tr>
<tr>
<td><code>mpAUDIT_TEXT_SOURCE_VC4</code></td>
<td>'2'</td>
<td>'2'</td>
<td>&quot;Local Event 3&quot;</td>
<td>The &quot;2&quot; without &quot;g&quot; determines which text group from the visualization resources should be used. The &quot;2&quot; under &quot;OutputFormat&quot; specifies which text in the text group should be used.</td>
</tr>
<tr>
<td><code>mpAUDIT_TEXT_SOURCE_VC4</code></td>
<td>'2'</td>
<td>'3'</td>
<td>&quot;Local Event 4&quot;</td>
<td>The &quot;2&quot; without &quot;g&quot; determines which text group from the visualization resources should be used. The &quot;2&quot; under &quot;OutputFormat&quot; specifies which text in the text group should be used.</td>
</tr>
</tbody>
</table>
Referencing another text group

Referencing another text group makes it possible to display another text depending on the contents of the token. This is beneficial if different events should display different texts in an entry. For example, the text of an entry may be different when a value has changed and when the user has entered an incorrect password. Another advantage of this approach is that tokens that are returned as numbers can be replaced with text (e.g. the actions from user management 0 - 11 can be replaced with text). The text group is specified after the token with $[TEXT=data]$. The token itself defines which index within the text group should be used. The following example illustrates referencing:

- [%[TIME=%D %R]] [%ev[TEXT=8]] - The time is output first. The event ID is output with the %ev token. Then (local) text group 8 is defined. The contents of text group 8 and index %ev within the text group will be displayed as a text after the time.

In this case, the following texts would be output depending on the event [%ev]:

- If the event ID is 2: '27/11/1989 01:32 Action out of user management happened'
- If the event ID is 16: '27/11/1989 01:32 Value changed'
- If the event ID is 17: '27/11/1989 01:32 VC4 Value changed'
- If the event ID is 18: '27/11/1989 01:32 Customer Event happened'

Multiple referencing is also possible. This could be used when an event occurs in user management, for example, to show exactly which. To do this, another text group must be referenced by event ID 2 that covers the various user management actions.

If the event with ID 2 now occurs, then text group 8 will go on to reference text group 9. The corresponding text will then be added to the entry depending on [%act].

Additional possibilities for multiple referencing can be found in the following example:

### 1.1.2.3 Text system as the text source

If the element mpAUDIT_TEXT_SOURCE_TEXTSYS is defined as the text source under "Type", then the text will come from the Automation Studio text system. Which text is used is determined by the following parameters:

- **Format text source**: This defines where in the text system the event text will be searched for. The namespace and text ID are used for identification.

Linking the text system could look something like this:

A .tmx file with the actual event texts is added to the Logical View beforehand. This file contains the event text in each project language. The connection is established via the namespace of the text and the text ID. The text ID is defined for each text in the file to be localized. The namespace and text ID are specified for parameter "Format text source":

**Namespace/EventID**
Referencing another text ID (new format)

This section describes how to reference another text ID using the new format. For information about how the new format is used, see section Using the new format.

Referencing another text ID makes it possible to display another text depending on the contents of the token. This is beneficial if different events should display different texts in an entry. For example, the text of an entry may be different when a value has changed and when the user has entered an incorrect password. Another advantage of this approach is that tokens that are returned as numbers can be replaced with text (e.g. the actions from user management 0 - 11 can be replaced with text). The whole thing is specified as follows:

- `{=$Namespace/{&key}}`

"{=$" specifies the namespace, which is followed by the token. The token defines which text ID within the specified namespace is used.

The following example illustrates referencing:

- `{&evtime[TIME=%u%R]} {=$Audit/Event/{&ev}}` - The time is output first. Then another text is referenced. The namespace in which the search should take place is defined in the curly brackets. A text ID is searched for in this namespace that corresponds to the contents of ":=ev".
In this case, the following texts would be output depending on event \(\&ev\):

- If the event ID is 2: ‘27/11/1989 01:32 User action’
- If the event ID is 16: ‘27/11/1989 01:32 Changed DP’
- If the event ID is 17: ‘27/11/1989 01:32 Changed VCDP’
- If the event ID is 18: ‘27/11/1989 01:32 Custom Event’

Multiple referencing is also possible. This could be used when an event occurs in user management (event ID = 2), for example, to show exactly which event (e.g. user deleted, new user created, user blocked, etc.). To do this, another text ID (possibly in another namespace) must be referenced by event ID 2 that covers the various user management actions.

If the event with ID 2 now occurs, then the Audit/Event namespace will continue to refer to the Audit/UserAction namespace. The corresponding text ID will then be searched for and the associated text added to the entry depending on \(\&act\).

Additional possibilities for multiple referencing can be found in the following example:
Referencing another text ID (old format)

Referencing another text ID makes it possible to display another text depending on the contents of the token. This is beneficial if different events should display different texts in an entry. For example, the text of an entry may be different when a value has changed and when the user has entered an incorrect password. Another advantage of this approach is that tokens that are returned as numbers can be replaced with text (e.g. the actions from user management 0 - 11 can be replaced with text). The namespace is specified after the token with $[TEXT=data]$. The token itself defines which text ID within the specified namespace is used. The following example illustrates referencing:

- [%[TIME=%u%R]] [%ev$[TEXT=Audit/Event]] - The time is output first. Then another text is referenced. The namespace in which the search should take place is defined in the square brackets. A text ID is searched for in this namespace that corresponds to the contents of %ev.

In this case, the following texts would be output depending on the event [%ev]:

- If the event ID is 2: '27/11/1989 01:32 User action'
- If the event ID is 16: '27/11/1989 01:32 Changed DP'
- If the event ID is 17: '27/11/1989 01:32 Changed VCDP'
- If the event ID is 18: '27/11/1989 01:32 Custom Event'
Multiple referencing is also possible. This could be used when an event occurs in user management (event ID = 2), for example, to show exactly which event (e.g. user deleted, new user created, user blocked, etc.). To do this, another text ID (possibly in another namespace) must be referenced by event ID 2 that covers the various user management actions.

If the event with ID 2 now occurs, then the Audit/Event namespace will continue to refer to the Audit/UserAction namespace. The corresponding text ID will then be searched for and the associated text added to the entry depending on [%act].

Additional possibilities for multiple referencing can be found in the following example:
Adding event data to the entry

In most cases, the event details should be included in the event. This can be done using the [%par] token. The %par token stands for different parameters that are identified by a unique identifier. For information about available event types and tokens, see section Event types.

Adding to the time to an entry

Similar to this principle, it is also possible to add the time to an entry:

Old format

• [%[TIME=\textbf{format}]]

New format (text system as the text source)

• \{&evtime[\textbf{TIME=\textbf{format}}]\}

"format" specifies how the time is displayed.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%u</td>
<td>Use UTC time (otherwise, local time will be used)</td>
</tr>
<tr>
<td>%H</td>
<td>Hour in 24-hour format. Leading zero if necessary: 00 - 23</td>
</tr>
<tr>
<td>%i</td>
<td>Hour in 12-hour format. Leading zero if necessary: 01 - 12</td>
</tr>
<tr>
<td>%k</td>
<td>Hour in 24-hour format: 0 - 23</td>
</tr>
<tr>
<td>%M</td>
<td>Minute within an hour. Leading zero if necessary: 00 - 59</td>
</tr>
<tr>
<td>%S</td>
<td>Seconds within a minute. Leading zero if necessary: 00 - 59</td>
</tr>
<tr>
<td>%L</td>
<td>Milliseconds within a second. Leading zero if necessary: 000 - 999</td>
</tr>
<tr>
<td>%z</td>
<td>Local shift to UTC time</td>
</tr>
<tr>
<td>%s</td>
<td>Seconds since 1 January 1970</td>
</tr>
<tr>
<td>%C</td>
<td>Four-digit year divided by 100. Displayed with two digits. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%y</td>
<td>Year as a four-digit number. Zero at the beginning if necessary</td>
</tr>
<tr>
<td>%m</td>
<td>Month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%d</td>
<td>Day of the month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%R</td>
<td>Time in 24-hour format: &quot;%H:%M&quot;</td>
</tr>
<tr>
<td>%T</td>
<td>Time in 24-hour format: &quot;%H:%M:%S&quot;</td>
</tr>
<tr>
<td>%D</td>
<td>Date in the following format: &quot;%m/%d/%y&quot;</td>
</tr>
<tr>
<td>%F</td>
<td>Date in ISO 8601 format: &quot;%Y-%m-%d&quot;</td>
</tr>
<tr>
<td>%c</td>
<td>Date and time in the following format: &quot;%F %T,%L&quot;</td>
</tr>
</tbody>
</table>
After the = character, how the time is displayed can be chosen freely. Several of the tokens listed above can also be used together. A few examples:

**New format**

- When evaluated, "\{\&evtime\[TIME=%u%R\]} Operator (\{\&op\}) generated an event." could look like this: 2014-12-05 15:35:20.234 Operator Andy generated an event.
- When evaluated, "Event at \{\&evtime\[TIME=%Y-%m-%d %H:%M\]} local time" could look like this: Event at 1990-25-03 12:35 local time.

**Old format**

- When evaluated, "\[%\[TIME=%u%R\]] Operator [%op] generated an event." could look like this: 2014-12-05 15:35:20.234 Operator Andy generated an event.
- When evaluated, "Event at [%TIME=%Y-%m-%d %H:%M] local time" could look like this: Event at 1990-25-03 12:35 local time.’
The event text visible in the HMI application using structure `MpAuditTrailUIOutputType` is limited to 100 characters.

If VC4 or the text system is chosen as the text source, and the event list is exported, then texts defined in the text system or HMI application can be read out up to a maximum length of 200 characters. The individually added events ([user], [message], [ev], etc.) can additionally have a length of 200. The timestamp ([time]) is limited to 80 characters.

Example in VC4:

![Image showing event text limits](image)

The resulting string can have any length in the export file.

![Exported event text without length limit](image)

If more than 200 characters are entered for an individual text (see image above), then `{Err:Overflow}` will be displayed in the exported file:
1.1.3 Configuration during runtime

The configuration parameters can be modified at runtime using \textit{MpAuditTrailConfig}. Nevertheless, it is important to note the state that the component must be in for the different parameters to allow the change to take place. The following states are possible:

- ![Icon](image)
  
  This icon means that this parameter can be changed in any state.

- ![Icon](image)

  This icon means that this parameter is applied when a new rising edge occurs on input "Enable" of \textit{MpAuditTrail}.

Each parameter in the configuration structure (\textit{MpAuditTrailConfigType}) is shown with one of these icons.

1.1.4 OPC UA events

What is an OPC UA event?

Here we are talking about the OPC UA information model. Within this information model, events are generated to exchange information between devices. An event consists of a time of receipt, message and severity. MpAudit reads these types of OPC UA events and lists them in the audit trail. If a variable on the controller is modified via an OPC UA client, for example, then an OPC UA event is generated. This is detected by MpAudit and entered in the event list.

Necessary MpAudit settings

At runtime, OPC UA events are recorded using \textit{MpAuditTrail}. In the audit configuration, value changes (16) or OPC UA events (19) must be enabled:

Necessary OPC UA settings

The OPC UA server must be active. It can be enabled (green) from the controller's configuration.

In addition, a role must be assigned (red) to \textit{Security admin} in the same place. In addition, the recording of events must be enabled on the server. This is done by enabling \textit{Auditing server facet} (violet). This prepares the OPC UA server for recording OPC UA events.
It is now time to decide which variables can be accessed via OPC UA. To do this, switch to the Configuration View. The "OPC UA default view file" is added in the Connectivity - OpcUA folder:

In this view, the wide range of properties for the individual variables can be modified (unit, access rights, etc.). The following properties are important in order to record OPC UA events:

- Auditing must be enabled.

- All variables for which events should be logged must be enabled.

### 1.1.5 VC4 event system under MpAudit

MpAudit makes it possible to retain events in the HMI application. The VC4 event system is used for this.

A VC object running on a terminal (see MpAuditTrail configuration) is not currently supported as a text source by MpAudit.

This must be enabled via shared resources:

All data points whose events should be logged must be assigned a UserID:

TheMpAuditVC4Event function block can be used to specify a range of several UserIDs for which events can be logged. The following values can be used for the example above:

- MinUserID: 100
- MaxUserID: 103

This logs all events that affect the TestNumFloat, TestNumFloatUnit, TestNumInt and TestNumIntUnit data points as long as MpAuditVC4Event is active.

The same UserID can also be used multiple times in VC4. If a UserID is used more than once, then a warning will be output during compilation.

### 1.1.6 Using the new format

The new format makes it possible to display units for a variable. For more information, see Displaying units for variables.

To use the new format, "TextSystem" must be selected as the text source in the MpAuditTrail configuration:
Only one format can be used. Combining the old and new format is not permitted!

The new format is specified as follows, for example:

- **Parameter has changed from {&old} to {&new}**
  Information is specified in curly brackets.
  The “&” character is used to indicate a token.

Compared to the old format:

- **Parameter has changed from [%old] to [%new]**
  Information is specified in brackets.
  The “%” character is used to indicate a token.

### 1.1.7 Displaying units for variables

In order to display the unit for variables, the new format must be used. For more information about the new format, see Using the new format.

To determine which variable should have which unit, see Unit management.

Parameter "Language" for **MpAuditExport**, **MpAuditTrail** and **MpAuditTrailUI** can be used to define the language or unit to be used when displaying events.

#### Defining the language and unit for display and export

How a unit is displayed is defined under "Displaying the unit within a text". For export or display, however, it is necessary to specify both the language as well as the unit system. This is done for the components listed above using input/parameter "Language". In order to define the language and unit system for export or display, input "Language" can be used in the following way:

'Language|UnitSystem'

A few examples:

- 'de': Language is German, values are shown with the engineering unit
- 'en|imperial-us': Language is English, unit system is imperial-US
- 'de|metric': Language is German, unit system is metric
- 'imperial': Default language, unit system is imperial

For details about defining display units, see section **MpComUnit configuration**.

#### Displaying the unit within a text

In addition to displaying and converting process values that are part of a text (e.g. alarms, audit events), it is also possible to define how a unit should be displayed:

- **Symbol %s**: This defines that the unit symbol for the current display unit will be displayed (e.g. "mm").
- **Short text %s**: This defines that the full name of the current display unit will be displayed (e.g. "millimeters").
- **Description %d**: This defines that the description for the current display unit will be displayed (e.g. "Distance measured in millimeters"). A description does not necessarily exist for each unit.

Conversion takes place automatically; how the unit is displayed is defined with the following syntax:
• `{&key}`: Defines the process variable whose value should be displayed. This value will be converted automatically to the current display unit. If no display unit is defined, then the engineering unit for the value is output.

• `{&key[UNIT=xx]}`: Displays the unit text for the unit assigned to the value. The parameters mentioned above (%s, %n, %d) can be specified in place of "xx".

A few examples:

• "Parameter changed from {&old} to {&new} {&new[UNIT= %n]}" evaluated: "Parameter changed from 210 to 580 millimeter".

### 1.2 Guides

#### 1.2.1 Getting started

##### 1.2.1.1 Recording a user event

This section will demonstrate how to record a user event using MpAudit. The mapp configuration defines which information is included in the event. The recorded user events are then exported and checked in the generated log file.

##### 1.2.1.1.1 Creating a project

You must first create a new project in Automation Studio.
1.2.1.1.2 Add mapp component

Adding the MpAudit configuration

The next step is to select the desired mapp component in the Toolbox and then drag-and-drop the configuration into the "mapp" folder.

Auditing can be configured in this configuration.

**Editing the auditing configuration**

As the "Text source", we will use "None". This means that if an event is logged, the text for the event comes directly from the "Output format" parameter in the configuration.

The code \[%[TIME=%c]\]: returns the time of the event. \[%msg\] can be used to specify a user-defined message, which can be defined later.

More information about these codes can be found in section Creating entries of Automation Help.

Parameter "Encrypt" is set to **FALSE**. As a result, the exported file will not be encrypted and can be opened in any editor.
1.2.1.3 Adding a program

The next step is to add a program. In this case, we will add a Ladder Diagram program. We could use any of the other programming languages, however.

1.2.1.4 Adding MpAuditTrail to the program

Searching for MpAudit in the Ladder Diagram Catalog

We can search for MpAudit in the Ladder Diagram Catalog. All MpAudit components are listed.
Selecting a function block

MpAuditTrail forms the core of auditing. The function block is added to the program using drag-and-drop. A variable of type MpAuditTrail is then declared.
1.2.1.1.5 Configuring MpAuditTrail

Connecting MpLink

First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.

Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".
Specifying the data storage device

In order for us to be able to export the recorded events, we will specify a data storage device named CF as an address. In the controller configuration, the data storage device will be specified under "FileDevices".
1.2.1.1.6 Adding MpAuditCustomEvent to the program

Searching for MpAudit in the Ladder Diagram Catalog

We can search for MpAudit in the Ladder Diagram Catalog. All MpAudit components are then listed.

Adding the function

Function MpAuditCustomEvent is added using drag-and-drop. This component generates user-defined events. To prevent the function from being called cyclically, we will right-click on the function to add the 2 parameters "EN/ENO" to it.
1.2.1.1.7 Configuring MpAuditCustomEvent

Enabling the component

In order to ensure that the function is only called once and not cyclically, a switch is used that only reacts to a rising edge of "TriggerEvent". It is positioned before input "EN". An event is thus only recorded if variable "TriggerEvent" is TRUE.

Connecting MpLink

Now connect input "MpLink" to the MpLink previously created in the Configuration View.
Specifying additional information

Parameter "Message" will be used to transfer the message that has been inserted in the text in place of the [%msg] code. In our example, the message is linked with variable "MessageVar". Parameters "Type" and "Comment" can be used to transfer additional details. Variables "MessageVar", "TypeVar" and "CommentVar" must be of data type WSTRING[100].

MpAuditCustomEvent Status

Variable "CustomEventStatus" of data type DINT is created to see the current status of the function.
1.2.1.1.8 Generating the file structure

Now generate a file structure for the memory card for ARsim.

1.2.1.1.9 Testing the user event in the program

Recording the user event

Monitor mode must be active to record a user event.
Function block MpAuditTrail_0 and variables "MessageVar" and "TriggerEvent" are added to the Watch window. "Hello World" is specified for "MessageVar". If variable "TriggerEvent" is then set to TRUE, an event is recorded.

"CurrentRecord" indicates that an event has been recorded. Variable "TriggerEvent" is set back to FALSE.
A second event with the message "This is another event" is now specified. This will be recorded as well on a rising edge of "TriggerEvent".

"CurrentRecord" now has value 2.
Exporting events

Events are exported to the specified data storage device in order to check whether they were recorded with the correct information. This is done using command "Export" on the input of MpAuditTrail.

Recorded events can be viewed by opening the exported file.
1.3 Configuration

1.3.1 MpAuditTrail configuration

The configuration can be modified at runtime using function block MpAuditTrailConfig.

**id_Audit_TextSource_VC4_OutputFormatSource**

Designated as "OutputFormat" in the MpAuditTextSourceType data type

**id_Audit_TextSource_VC4_OutputFormat**

Designated as "OutputFormatSource" in the MpAuditTextSourceType data type

**id_Audit_Export_PassPhrase**

Specifies the key used to encode the file. A 256-bit AES key is used. The user *cannot* independently decrypt the file. "PassPhrase" is only needed to provide the application developer with the possibility of affecting the degree of security. "PassPhrase" **must** be changed in order to ensure encryption takes places sensibly.

**id_Audit_Export_PDF_Header**

Header information can be specified here. The text source is the same as for the other event texts, so it is defined in the configuration. It might look something like this:

- **Configuration as source**: "Enter text directly"
- **Text system as source**: "\{Namespace/TextID\}"
- **VC4 as source**: "group/index"

**id_Audit_Export_PDF**

Library MpReport is needed to export the event list as a PDF file. Since the library does not contain any components, it is sufficient to simply add it to the project.

**Memory storage settings**

Which events should be recorded can be defined under "Events". The option can be enabled in the advanced parameters ("Show advanced parameters").

Memory storage settings can be defined in the first section of the configuration. It is possible to define how much storage space should be reserved for events and how much memory should be available for events in SRAM.
Storage location

Parameter "Record memory" defines the way in which recorded data is saved. The following modes can be selected:

UserROM: If memory type UserROM is used, then the save interval must also be specified ("SaveInterval"). Recorded data is first saved temporarily to DRAM. After each save interval, the data is then saved to UserROM.

SRAM: When using battery-backed memory, there must be sufficient memory reserved in the controller configuration. Can only be used if the hardware being used is equipped with SRAM memory.

Buffered UserROM: If memory type "Buffered UserROM" is used, then the data is first saved temporarily to SRAM before being transferred to UserROM.

Temporary: Data is saved to DRAM. It is important to note that this data is not protected in the event of power failure or restart!

Text source

The text source can be defined in section "Text source". It is possible to define where the description text for the various events should come from. For more information, see Creating entries.

Archiving

When an archive should be created from the current events can be defined in section "Archive". If the option is enabled, how often archiving should take place must be defined in addition to how large the archive is permitted to be.

Export settings

Export settings can be defined in section "Export". The filename ("Filename pattern") and archive name ("Archive name pattern") can be defined by enabling the advanced parameters.

"Raw data" adds raw data to the export file. Which file type should be used for the export is defined under "File type". The exported file can be encrypted with "Encrypt". "Passphrase" is the value of the key used to encrypt the file. A 256-bit AES key is used. The user cannot independently decrypt the file. "PassPhrase" is only needed to provide the application developer with the possibility of affecting the degree of security. "PassPhrase" must be changed in order to ensure encryption takes places sensibly!
If "Encrypt" = TRUE, then the encrypted data is saved as a ZIP file.

Additional settings can be made for file type PDF:

- **Default font**: Standard font. Used if no other font was defined for the current language.
- **Page format**: The advanced parameters must be enabled in order to modify this parameter ("Show advanced parameters"). Defines the Page format.
- **Page direction**: The advanced parameters must be enabled in order to modify this parameter ("Show advanced parameters"). Defines the orientation of the pages.
- **Page margins**: The advanced parameters must be enabled in order to modify this parameter ("Show advanced parameters"). Defines the distance between the text and edge of page.
- **Owner password**: It is only possible to change permissions for copying, printing and editing if this password is known.
- **User password**: The user will be asked to enter this password when opening the PDF file. If a password is not specified here, then a password will not be required to open the PDF.
- **Allow print**: Defines whether the PDF file can be printed
- **Allow copy**: Defines whether the PDF file can be copied
- **Allow edit**: Defines whether the PDF file can be edited
Variable monitor

The variable monitor can be used instead of function block MpAuditRegPar. Process variables specified here are monitored automatically by MpAudit. An event is recorded as soon as a variable changes.

Query

Queries can be created under "Data queries". These queries can be used to filter or search for specific events in the event list. A unique name for the query must be specified under "Name". If the request is to be started via MpAuditQuery, the name defined here must be specified on input parameter "Name". The value of the variable specified on "Update count" is automatically increased by 1 as soon as new information is available. This means that as soon as new data corresponding to the specified filter criteria (WHERE) is available, the counter value of the variable on "Update count" changes. This can be used as a trigger to start command "Execute" on MpAuditQuery.

The information to be included in the query is defined via "Select" and "Where":

SELECT

"Select" determines which information is to be requested and where it is to be copied. A variable must be specified on "Process variable". If the query is started via MpAuditQuery, the information is displayed on this process variable. An array variable can also be specified. The array variable must be specified as follows: "::MyVar[]".

"Column" determines which information is to be queried. For more information about individual types, see Event types.

WHERE

Under "Where", the type of event from which the information is to be queried ("Column") is specified. For more information about individual types, see Event types. Using "Operator" and "Compare to", the query can be provided with additional filters, for example events with a certain event type or events from a specific point in time. This means that certain values can be queried, for example. "Compare to" can be a fixed value ("Value") or process variable ("PV"). If "LIKE" is used as an operator, similar entries can be searched for. Wildcard symbol "*" must be used. If "**Test**" is used as "Value", for example, entries "Test01", "MyTest" and "MyTest01" can be searched for.

Example

All alarm names that have occurred are to be searched for in the event list. All alarm events have event ID 33. For more information, see here. This means it can be filtered by event ID 33 ("WHERE"). Since the alarm name is to be displayed, "Alarm name" is selected ("SELECT"). The information is displayed in variable Text, which is of data type STRING[200][99]. This means that the variable is an array variable with 100 elements. Since it is an array variable, the variable must be specified with ::Text[].

This displays all alarm names of the alarm events that have occurred in variable Text.
1.4 Use cases

This section outlines several possible use cases for MpAudit components.

1.4.1 Use case 1: Auditing without HMI

Requirement

An application should record value changes coming from an external (3rd-party) HMI device. It should be possible to log in as a user and change various parameters from this HMI device. Switching users and changes to parameters should be recorded continuously (old and new values). The raw data should be exported in XML format (values without text, however). This allows the texts to be put together later from the raw data.

Solution

Component list

- *MpAuditTrail* (own MpLink): Records audit events
- *MpAuditValueChange* (MpLink from MpAuditTrail): Creates a generic value change event that can originate from various sources (in this case from the MpAuditValueChange component)
- *MpUserLogin* (own MpLink): Component from library MpUserLogin for logging users in/out

Connection diagram

Using the mapp components

*MpAuditTrail* implements the main event recorder. All events are recorded in a ring buffer. Archives should be created in order to prevent data loss. The data can be exported with the "Export" command.

The *MpAuditValueChange* component is used to generate a generic value change event. The old and new parameters are transferred to this function for this.
The **MpUserLogin** component is necessary to link the value change event to the logged in user. If a user is logged in, all events that occur are linked to this user (the connection is established through the use of the same MpLink).

**Setup/Settings**

Both the output format and export of raw data are defined in the configuration:

```
[TIME=%c] value changed by %op: %id from %old to %new
```

(The syntax is explained in more detail in section Creating entries.)

The output text looks like this in the file after the export (example):

"2015-02-03 10:54:32.546 Value changed by User1: Speed from 20.000000 to 25.000000".

Additional raw data (<Data>) is added to the entry in the file and looks like this:

```
<Event ID="0x950010" Record="3">
  <Text>2015-03-02 11:05:03.720 Value changed from User1: Velocity from 20.000000 to 25.000000</Text>
  <Data>
    <time>1425294303.720 +00:00</time>
    <map>Audit</map>
    <old>20.000000</old>
    <new>25.000000</new>
    <dpid>ExtNumeric</dpid>
    <ev>16</ev>
    <op></op>
  </Data>
</Event>
```

The time is based on "Unix time" (seconds, 1425294303) with milliseconds counter (.720) and UTC adjustment (+0:00).

**1.4.2 Use case 2: Auditing with local VC4-based HMI application**

**Requirement**

The following actions should be checked and recorded in the application:

- User entries/changes via the VC4-based HMI application that affect critical parameters (e.g. change in speed).
- User logins (including failed attempts), logouts and all changes to user management (creation of new users, editing of existing users, e.g. if a user is locked due to too many failed login attempts)
- Changes to the machine hardware (mechanical and electrical, e.g. if an X20 module is removed from the bus)

The newest events should be displayed in the HMI application. In addition, events are not permitted to be lost.

**Solution**

**Component list**

- **MpAuditTrail** (own MpLink): Records audit events
- **MpAuditTrailUI** (MpLink from MpAuditTrail): Displays the latest audit events in the VC4-based HMI application
- **MpAuditVC4Event** (MpLink from MpUserLogin): Reads value, changed events in the VC4-based HMI application (e.g. somebody presses a key)
- **MpAuditCustomEvent** (MpLink from MpUserLogin): Records customer-specific events (e.g. changes to machine hardware)
- **MpUserLogin** (own MpLink): Component from the MpUser library for logging users in/out, connecting audit events to the currently logged in user; in addition, all events involving user management can be recorded
Connection diagram

Using the mapp components

**MpAuditTrail** implements the main event recorder. All events are recorded in a ring buffer (data object). Archives should be created in order to prevent data loss. This archive must be exported by the application or user. The export is stored in an XML file.

**MpAuditTrailUI** can be used to display the recorded events in the current language (optimized for VC4-based HMI applications). All events in the recording ring buffer can be displayed with this component.

**MpAuditTrailVC4Event** is the connection to the VC4 event system. It reads all events from the HMI application and filters them so that only relevant and selected events are recorded by the audit trail (for changes to parameters tagged with a user ID, see section **Settings for detecting VC4 events** below). VC4 events contain the user, old value and new value of the changed parameter.

The **MpAuditCustomEvent** component is used to record any type of event. It is up to the application to use this function for the necessary purposes.

**MpUserLogin** is necessary to link value change events to the logged in user. If a user is logged in, all events that occur are linked to this user (the connections are established through the use of the same MpLink).

Configuration

The configuration for this use case looks like this:

The previously mentioned archive function is defined in the configuration. Also specified here is that the texts for the individual events come from VC4 text groups.

Settings for detecting VC4 events

The VC4 event system must be enabled to receive value change events in the VC4-based HMI application:

Each parameter in the VC4-based HMI application whose value changes should be detected receives a separate user ID:

After each parameter has a separate user ID, a description must be created that is added when the parameter changes. This description is added to a separate global text group that is then specified in the shared resources as a source for texts for VC4 changes.
This allows changes to be determined in the HMI application using the user ID. If a parameter tagged with a user ID is changed, an audit event is created. The text specified for this audit event comes from the global text group.

**Text for individual events**

The entry in the audit trail will appear differently depending on the event. For example, the entries for when a user logs in in the morning and when the speed is increased from 100 mm/s to 200 mm/s will not look the same.

To get readable information for each event, it is necessary to configure an appropriate source for these texts in the configuration (see above). The texts are read in this use case from VC4 text groups, starting with text index 0 in text group 0 according to the configuration. From the text at position 0, other text groups and their contents are referenced. For an exact explanation of the syntax, see section Creating entries.

The image above shows the text group with index 0, which contains the text for the main event (index 0). This is the starting point for all event texts. The time (\[% TIME = uR\]) and username (\[% op\]) are displayed first. An individual text can be specified for each event (\[%ev $ [TEXT = 0]\]) refers to text group 0 and a text index that depends on %ev after the user name. For simplicity's sake, the individual event texts are stored with the main event texts in the same text group.

Data is stored differently for each event.

- **2: User-defined actions (login, logout, etc.)** - A defined string ("user action") followed by a detailed text depending on what happened (\[%Act $ [TEXT = 5]\]). This text is then taken from another text group (text group 5, text index depends on %Act).

- **16: Value change** - Shows a defined string ("Changed DP") followed by a descriptive text data point (\[%dpid $ [TEXT = g2]\]). Parameter "g2" is the global text group "DpDescription", which was mentioned in connection with VC4 events. The old value (\[%old\]) and new value (\[%new\]) are displayed. If the entries are changed using a dropdown element in VC4, then the raw data is replaced by the selection texts (\$ [TEXT =% sel]).

- **18: User-specific event** - Redirect to another text group in which the individual texts are saved depending on the type of event (\[% typ $ [TEXT = 4]\]) (text group 4, text index depends on %typ)

### 1.4.3 Use case 3: Logging OPC UA variables

**Requirement**

An application is connected to an HMI application via OPC UA. Changed values for process variables that are connected to an HMI application should be logged by MpAudit.

**Solution**

**Component list**

- **MpAuditTrail** (own MpLink): Records events

**Connection diagram**
Configuration

The MpAuditTrail configuration is added. Value changes of OPC UA variables are logged using event type "Value-changes". It is possible to view value changes immediately as with the other PVs or just VC4 value changes.

If OPC UA variables should be logged specifically, then the event type "Opc value-changes" or simply "Value changes" can be used. This makes it possible to differentiate OPC UA value changes from other value changes. The expanded view of the configuration shows which events should be used (show advanced parameters).

The OPC UA system is enabled under "OPC-UA system" in the hardware configuration. The "Auditing server facet" option is enabled. A user from the AS user role system is specified under "Security admin". This user is able to receive the logged events.

To declare a process variable as an OPC UA tag, right-click the desired variable and select "Enable tag". For more information, see section OPC UA.
The function for logging variables ("Audit events") must be enabled in the OPC UA configuration.

Using the mapp components

After the OPC UA configuration, the **MpAuditTrail** component is added. This component implements the main event recorder. All events (OPC UA variable value change, VC4 value change, etc.) are recorded in a ring buffer (data object).

1.4.4 Use case 4: Displaying a temperature value with unit in the event list

**Requirement**

The temperature value of a zone should be recorded in a heating coil. As soon as the temperature changes, it should be clear what the old and new temperature values are. The event should be saved to a list. The event could look something like this: "The temperature setpoint of the heating coil increased from $128^\circ C$ to $136^\circ C$".

The list should be stored on the CompactFlash card in both English and German. The English event list should use the imperial unit system; the German should use metric. The temperature should be converted automatically depending on the unit.
The unit system can be modified in the machine's HMI application. It is possible to switch between the metric, imperial and imperial-US systems. When using the metric system, the temperature value should be displayed in degrees Celsius; in imperial and imperial-US, degrees Fahrenheit should be displayed.

Degrees Celsius are used when working in the software.

Solution

Component list

- MpAuditTrail (own MpLink): Records audit events

Connection diagram

![Connection diagram](image)

Preparation for using the unit system

In order to use the unit system, the OPC UA server must be enabled in the target system configuration:

![Configuring OPC UA System](image)

The OPC UA default view configuration must then be added. The variables used in the project can be enabled as OPC UA variables in the file. For more information, see View.

Variable **TempVar** is enabled and **degrees Celsius** is used as the unit since degrees Celsius is used when working in the application.

![Variable Configuration](image)

The user role system must be added in order to use the unit system. User "Anonymous" must be assigned to role "Everyone".
MpAudit configuration

The MpAudit configuration must be added in order to record events. In order for the unit to be logged along with the variable being recorded, the text system must be used with the default formatter. This means that "Text system" is used as the text source ("Text source").

"Format text source" defines the position in the text system at which the event text is searched for. The namespace and text ID are used for identification. For more information, see "Text system as the text source" in section Creating entries.

Variable TempVar is specified under "Variable monitor". If the variable is modified, the value change is recorded.
Text system configuration

The text used to output the event is defined by the text system. A TMX file, the project languages to be used and the configuration of the text system are added. The TMX file must be specified in the configuration under "TMX files for target". For more information about the text system, see Text system.

If the value of variable TempVar changes, the text specified under text ID "Root" is used for the event text. The original value of variable TempVar is displayed in token old. The new value is displayed in token new. The unit (e.g. °C) of the variable is displayed with [UNIT=%s].
For more information about tokens, see Event types.
For more information about specifying the unit, see Displaying units for variables.

MpComUnit configuration

The definition of which unit should be displayed in which unit system takes place in the MpComUnit configuration. Variable TempVar to be recorded is specified under "Data points. Which display unit should be used per unit system is defined in section "DisplayUnits". Which engineering unit should be used per unit system is defined in section "Units". For more information, see Unit management.

Degrees Celsius should be used in the metric system. Degrees Fahrenheit should be used for the imperial and imperial-US systems. Unit abbreviations CEL and FAH are therefore used.

For which token is used for which unit, see here.
Using the mapp components

Component **MpAuditTrail** is added after the configuration. This component implements the main event recorder. All events are recorded in a ring buffer (data object).

Input parameter "Language" defines the language or unit used when displaying or exporting events. It can be specified like this, for example: "Language = 'en|imperial'".

This means that the imperial system and English will be used. For more information, see Displaying units for variables.

The event list can be exported by setting parameter "Export" to **TRUE**.

The imperial system should be used for the English event list. "Language" is therefore set to "en|imperial".

The exported file looks something like this: 
The metric system should be used for the German event list. "Language" is therefore set to "de|metric".

The exported file looks something like this:

1.5 Libraries

1.5.1 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpAuditCustomEvent</td>
<td>This function allows a user-defined event.</td>
</tr>
<tr>
<td>MpAuditClearBuffer</td>
<td>This function resets the buffer containing all currently logged events.</td>
</tr>
<tr>
<td>MpAuditRegPar</td>
<td>This function block allows the user to register a PV that is monitored for value changes.</td>
</tr>
<tr>
<td>MpAuditStringChange</td>
<td>This function logs the changes made to an ASCII string.</td>
</tr>
<tr>
<td>MpAuditTrail</td>
<td>This function block manages event logging.</td>
</tr>
<tr>
<td>MpAuditTrailConfig</td>
<td>This function block configures event logging.</td>
</tr>
<tr>
<td>MpAuditTrailUI</td>
<td>This function block allows access to the current event via the HMI application.</td>
</tr>
<tr>
<td>MpAuditValueChange</td>
<td>This function logs the changes made to a value.</td>
</tr>
<tr>
<td>MpAuditVC4Event</td>
<td>This function block logs events in the HMI application.</td>
</tr>
<tr>
<td>MpAuditWStringChange</td>
<td>This function logs the changes made to a Unicode string.</td>
</tr>
<tr>
<td>MpAuditStartBatch</td>
<td>Calling this function starts a new batch.</td>
</tr>
<tr>
<td>MpAuditExport</td>
<td>This function block can export (filtered) data.</td>
</tr>
<tr>
<td>MpAuditQuery</td>
<td>A query can be started using MpAuditQuery. This allows you to search for certain events in the event list.</td>
</tr>
</tbody>
</table>

1.5.1.1 MpAuditClearBuffer

This function resets the buffer containing all currently logged events.
Function block

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 MpAuditTrail configuration).</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).

1.5.1.1.1 Description

The same MpLink used by MpAuditTrail is used for this component as well.

This function clears all events currently logged in the buffer. The act of clearing the buffer generates an event to log the resetting of the buffer and the user who is currently logged in. This event will appear as the first event in the now-empty buffer.

For information about how data is recorded, see section Event types under "Clear buffer event".

1.5.1.2 MpAuditCustomEvent

This function allows a user-defined event.

Function

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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Type</td>
<td>WSTRING[100]</td>
<td>Entered when creating the entry under %typ (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>Message</td>
<td>WSTRING[100]</td>
<td>Entered when creating the entry under %msg (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>Comment</td>
<td>WSTRING[100]</td>
<td>Entered when creating the entry under %cmt (see Creating entries).</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>DINT</td>
<td>Status information of function.</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

The MpLink used here comes from the MpAuditTrail configuration to which this event is assigned.

This function makes it possible to create user events. The type of event can be defined with parameter "Type". Parameter "Message" can be used to record the actual event, while an additional comment can be added with "Comment".
For information about which data in this event can be included when recording, see "User-defined event" in section Event types.

### 1.5.1.3 MpAuditExport

This function block can export (filtered) data.

**Function block**

#### Optional parameters

<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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>Filter</td>
<td>MpAuditExportFilterType</td>
<td>Used to filter the events.</td>
</tr>
<tr>
<td>ToRecord</td>
<td>UDINT</td>
<td>Defines the event up to which data should be exported.</td>
</tr>
<tr>
<td>Language</td>
<td>STRING[20]</td>
<td>Defines the language or unit used to display the events.</td>
</tr>
<tr>
<td>DeviceName</td>
<td>STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>Export</td>
<td>BOOL</td>
<td>Saves the (filtered) data from memory to a file on the specified data storage device (&quot;DeviceName&quot;).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>Record</td>
<td>UDINT</td>
<td>Number of the last exported event.</td>
</tr>
<tr>
<td>Info</td>
<td>MpAuditInfoType</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 same MpLink used by MpAuditTrail is used for this component as well. Filtered events can be exported using MpAuditExport. Input "Language" defines the language or unit used when exporting events. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.

Filtering events

Input parameter "Filter" makes it possible to filter according to certain events. The following filtering options are available:

- **From**: Filters from this time
- **Until**: Filters until this time
- **Event**: Filters according to certain event types
- **Operator**: Filters according to a certain user
- **Pattern**: Filters according to a certain event text Wildcard "*" can be used when searching. If "*Test" is specified, for example, events "Test01", "MyTest" and "MyText01" will be displayed. If the "*" wildcard is not used, then it will only search for event "Text".
- **Batch**: Filters according to a certain batch

Exporting events

Events are exported to the specified data storage device ("DeviceName") using parameter "Export". Using a filter exports the filtered events.

If no filter is specified, all current entries are exported from memory.

After an export, the ID of the last exported event is indicated on output "Record".

"ToRecord" can be used to specify the event from which the export should take place.

If "CurrentRecord = 5200" is currently on the output of MpAuditTrail when data is exported, then events 0 to 5200 are exported.

After the export, MpAuditExport indicates the ID of the last exported event on "Record". This ID can be specified on input "ToRecord". If new events are available on MpAuditTrail ("CurrentRecord = 8920"), then MpAuditExport can be used to export events 5200 to 8920. "ToRecord" always defines the event from which the export is started.
1.5.1.4 MpAuditQuery

A query can be started using MpAuditQuery. This allows you to search for certain events in the event list.

**Function block**

## Optional parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpComIdentType</td>
<td>BOOL</td>
<td>MpLink</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Active</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td>Mode</td>
<td>MpAuditQueryModeEnum</td>
<td>StatusID</td>
</tr>
<tr>
<td>Name</td>
<td>&amp;STRING</td>
<td>CommandDone</td>
</tr>
<tr>
<td>Language</td>
<td>&amp;STRING</td>
<td>Info</td>
</tr>
<tr>
<td>Execute</td>
<td>BOOL</td>
<td>MpAuditQueryInfoType</td>
</tr>
<tr>
<td>Next</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
The same MpLink used by MpAuditTrail is used for this function block. With the help of MpAuditQuery, a query defined in the MpAuditTrail configuration can be started. The desired query is specified on input parameter "Name".

"Mode" defines whether all existing entries should be searched or only the newest ones. In this case, the most recent events are those generated since the last "Update count" (see MpAuditTrail configuration, section "Query"). Input parameter "Language" can define the language or unit in which the filtered events are displayed. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Displaying units for variables.

"Execute = TRUE" starts the request. The requested information can be seen in the process variable defined in the MpAuditTrail configuration in section "Queries".

The following information can be taken from structure "Info":
- **AvailableSpace**: Specifies how much space the process variable defined in the `MpAuditTrail` configuration in section "Queries" offers. If the process variable is a STRING array of 100 elements, 100 is displayed.
- **RowsRead**: Indicates how many entries have been read by the query.
- **HasMoreRows**: With "HasMoreRows = FALSE", all entries of the query are visible in the process variable. With "HasMoreRows = TRUE", not all entries in the process variable could be displayed. If a STRING array of 6 elements was used as the process variable but there are 9 entries, "HasMoreRows = TRUE" is displayed. To get the last 3 entries, command "Next = TRUE" of `MpAuditQuery` must be used. The last 3 entries are written to the variable. The information is written to the first elements of the process variable:

1.5.1.5 **MpAuditRegPar**

This function block allows the user to register a PV that is monitored for value changes.

**Function block**

## Optional parameters

```
MpAuditRegPar
&MpComIdentType MpLink
BOOL Enable
BOOL ErrorReset
&STRING PVName
&STRING Identifier
BOOL Active
BOOL Error
DINT StatusID
MpAuditTrailInfoType
```
### 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 MpAuditTrail 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>PVName</td>
<td>Pointer to STRING[100]</td>
<td>Name of the PV that should be checked for value changes.</td>
</tr>
<tr>
<td>IN</td>
<td>Identifier</td>
<td>Pointer to STRING[100]</td>
<td>Entered when creating the entry under %dpid (see Creating entries).</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>Info</td>
<td>MpAuditInfoType</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).

#### 1.5.1.5.1 Description

The MpLink of an MpAuditTrail configuration is used.

This function block automatically monitors the value changes of a process variable. The validity of the PV is evaluated on a rising edge on the enable input. The PV is registered as soon as the component is active ("Active" = TRUE). If the component is no longer active, then the PV is also no longer registered. A separate MpAuditRegPar must be used for each PV that should be checked for value changes. If the value of one of these PVs changes, a value change event is generated and recorded in the event list by MpAuditTrail. For information about which data in this event can be included when recording, see "Value change" in section Event types.

A new PV can be registered at any time (specifying the name of process variables). The following format is used:

- **Global PV**: `<PV_name>` (e.g. "gTestVariable", "gControl.Parameter.Distance")
- **Local PV**: `<task_name>:<PV_name>` (e.g. "Task1:LocalTestVariable", "AxisCtrl:Basic.Command.MoveVelocity")

**Registration limitations:**

- Arrays that do not start with index 0 cannot be registered.
- Multidimensional arrays cannot be registered.
- Arrays with more than 65535 elements cannot be registered.
- Derived data types
- TIME_OF_DAY
- BYTE
- WORD, DWORD, LWORD
- WSTRING
- LINT

#### 1.5.1.6 MpAuditStartBatch

Calling this function starts a new batch.

### Function block

```
MpComIdentType
WSTRING[50]

MpAuditStartBatch
MpLink
Name
DINT
Return value
```
mapp Services

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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>WSTRING[50]</td>
<td>Name or unique identifier of the batch.</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>DINT</td>
<td>Status information of function.</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).

1.5.1.6.1 Description

The same MpLink used by MpAuditTrail is used for this component as well.

A new batch can be started with this function. When a new batch is created, it is shared with MpAuditTrail and therefore recorded. For information about which data for this event is recorded, see "Batch event" in section Event types.

The name of the batch is defined using input parameter "Name".

A batch can be compared to an order. For example, a machine is first used to produce pepperoni pizza. A batch event called "Pepperoni pizza" is generated. Once enough pepperoni pizzas are produced, a new batch – "Cheese pizza" – is generated.

The filter function from MpAuditTrailUI or MpAuditExport can be used to filter according to events that occurred in a batch:

If the production of cheese pizza is interrupted by changing shifts or a break, the "No production" batch can be started.

1.5.1.7 MpAuditStringChange

This function logs the changes made to an ASCII string.
Function

```
MpAuditStringChange
MpComIdentType  MpLink  Return value  DINT
STRING[100]   Old
STRING[100]   New
STRING[100]   Identifier
```

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 MpAuditTrail configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Old</td>
<td>STRING[100]</td>
<td>Entered when creating the entry under %old (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>New</td>
<td>STRING[100]</td>
<td>Entered when creating the entry under %new (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>Identifier</td>
<td>STRING[100]</td>
<td>Entered when creating the entry under %dpid (see Creating entries).</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>DINT</td>
<td>Status information of function.</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).

1.5.1.7.1 Description

The MpLink used here comes from the MpAuditTrail configuration to which this event is assigned.

The function is supplied with the old and new value of the ASCII string via inputs "Old" and "New". Parameter "Identifier" is used to identify the information source (e.g. the name of the ASCII string where the value changed).

For information about which data in this event can be included when recording, see "Value change" in section Event types.

1.5.1.8 MpAuditTrail

This function block manages event logging.

On our YouTube channel, the tutorials mapp Audit - Part 1 and mapp Audit - Part 2 explain how event management can be implemented.

Our B&R online tutorial platform includes tutorials about mapp Audit.

Function block

```
MpAuditTrail
&MpComIdentType MpLink
BOOL Enable
BOOL ErrorReset
&STRING Language
&STRING DeviceName
BOOL Export
BOOL ExportArchive
BOOL Active
BOOL Error
DINT StatusID
BOOL CommandDone
UDINT CurrentRecord
BOOL ArchiveAvailable
MpAuditTrailInfoType Info
```
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 MpAuditTrail 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>Language</td>
<td>Pointer to STRING[20]</td>
<td>Defines the language or unit used to display the events.</td>
</tr>
<tr>
<td>IN</td>
<td>DeviceName</td>
<td>Pointer to STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>IN</td>
<td>Export</td>
<td>BOOL</td>
<td>Saves the currently logged data from memory to a file on the specified data storage device (“DeviceName”).</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>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>CurrentRecord</td>
<td>UDINT</td>
<td>Counts all events.</td>
</tr>
<tr>
<td>OUT</td>
<td>ArchiveAvailable</td>
<td>BOOL</td>
<td>Indicates that an archive is currently available and can be exported.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpAuditTrailInfoType</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 carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.8.1 Description

This function block forms the core of event management and requires the MpLink of an MpAuditTrail configuration.

As soon as MpAuditTrail is active, an entry is stored in memory for each new event. These entries are stored in a ring buffer. As soon as memory is full, a new event will overwrite the oldest entry. To prevent entries from being lost, the current entries must be archived regularly. Archiving can be configured using MpAuditTrailConfig. If an archive is created, only entries that are new since the last archive are relevant. A new archive is displayed via output "ArchiveAvailable". As soon as a new archive is available, it should be exported. The "ExportArchive" command can be used for this. The archive is transferred to the data storage medium specified by "DeviceName". If several archives are saved on the Automation Runtime system, then the oldest will always be exported. The exact number of available archives can be determined with parameter "NumberArchives" in structure "Info". Command "Export" exports a file with the current entries from memory, regardless of archiving. Output "CurrentRecord" contains the total number of entries that were processed by this MpAudit.

Input parameter "Language" defines the language or unit used with events. In order to define the language and unit system, input "Language" can be used in the following way: ‘Language|UnitSystem’. Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.

The export format is set in the configuration (e.g. PDF, XML). Library MpReport is needed to export the event list as a PDF file. Since the library does not contain any components, it is sufficient to simply add it to the project.

Multiple MpAuditTrail instances can be used to sort events according to type. The configuration can be used to determine which event types should be logged. In this way, each MpAuditTrail instance can log different events.
The event text visible in the HMI application using structure **MpAuditTrailUIOutputType** is limited to 100 characters.

If VC4 or the text system is chosen as the text source, and the event list is exported, then texts defined in the text system or HMI application can be read out up to a maximum length of 200 characters. The individually added events ([user], [message], [ev], etc.) can additionally have a length of 200. The timestamp ([time]) is limited to 80 characters.

Example in VC4:

The resulting string can have any length in the export file.

If more than 200 characters are entered for an individual text (see image above), then {Err:Overflow} will be displayed in the exported file:

---

**Data sheet V**

61
1.5.1.9 MpAuditTrailConfig

This function block configures event logging.

Function block

## Optional parameters

### Block parameters

- **&MpComIdentType**
- **MpLink**: Connection to mapp (MpLink of an MpAuditTrail configuration).
- **BOOL Enable**: The function block is active as long as this input is set.
- **BOOL ErrorReset**: Resets function block errors.
- **BOOL Configuration**: Structure used to specify the configuration.
- **BOOL Load**: Loads the configuration of the component.
- **BOOL Save**: Saves the configuration of the component.
- **BOOL Active**: Function block active.
- **BOOL Error**: Error occurred during execution.
- **DINT StatusID**: Status information.
- **BOOL CommandBusy**: Function block currently executing command.
- **BOOL CommandDone**: Execution successful. Function block is finished.
- **MpAuditTrailInfoType Info**: Additional information about the component.

### 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 Mp-</td>
<td>Connection to mapp (MpLink of an MpAuditTrail 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>Configuration</td>
<td>Pointer to MpAu-</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</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>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpAuditInfoType</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).

The section Configuring instead of programming also explains how to work with the configuration.

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

1.5.1.9.1 Description

The same MpLink used by MpAuditTrail is used for this component as well.

This function block reads and writes the configuration. The complete configuration takes place using the configuration structure on input "Configuration".
Command "Load" is used to read out the currently configured parameters. The currently active configuration on the controller is written to the PV on input "Configuration". This overwrites the data in the structure. Values can then be edited as needed.

Command "Save" applies the values from structure "Configuration" to the active configuration; they are then applied immediately, if possible.

The respective configuration data types for the function block explains when parameters in the configuration are applied.

Configuring structure MpAuditTrailConfigType

This configuration can be used to define which events should be evaluated by MpAuditTrail and which format should be used to store the entry.

Input "Events" is used to define which event types result in an entry. Each element of this array defines an event type. The different event types possible are listed in section Creating entries. The event type must be specified by its ID. MpAuditTrail can log up to ten different event types.

One entry is created for each new event. An entry takes up about 150 bytes in memory. The size of an entry can vary depending on the type and the specified parameters. Where this is saved can be defined with "RecordMemory".

The following options are available:

- **mpAUDIT_MEM_SRAM**: The first choice, whenever possible. If sufficient SRAM is available, all entries can be stored here directly, with the size determined by "RecordingSize".
- **mpAUDIT_MEM_BUFFERD_ROM**: Use if not enough memory is available in SRAM to store all entries there. Entries are stored in SRAM until "SramBufferSize" is reached. Only then are the entries transferred from SRAM to USER ROM. The ring buffer for the entries resides in USER ROM and is configured using parameter "RecordingSize".
- **mpAUDIT_MEM_ROM**: If there is not enough memory in RAM, then this setting can be used to transfer each entry to USER ROM, with the size defined by "RecordingSize". The entries are retained if the controller is switched off, but USER ROM is accessed after every entry (slow).
- **mpAUDIT_MEM_TEMP**: If sufficient DRAM is available, all entries can be stored here directly, with the size determined by "RecordingSize". Disadvantage: Data is lost when the controller is switched off.

**SRAM: When using battery-backed memory, there must be sufficient memory reserved in the controller configuration. Can only be used if the hardware being used is equipped with SRAM memory.**

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%u</td>
<td>Use UTC time (otherwise, local time will be used)</td>
</tr>
<tr>
<td>%H</td>
<td>Hour in 24-hour format. Leading zero if necessary: 00 - 23</td>
</tr>
<tr>
<td>%I</td>
<td>Hour in 12-hour format. Leading zero if necessary: 01 - 12</td>
</tr>
<tr>
<td>%k</td>
<td>Hour in 24-hour format: 0 - 23</td>
</tr>
<tr>
<td>%M</td>
<td>Minute within an hour. Leading zero if necessary: 00 - 59</td>
</tr>
</tbody>
</table>

If the current entries or an archive is exported, the system proceeds after the values configured with "Export" (MpAuditExportType). The name of the file with the current entries is defined using "FileNamePattern":

- **Default format**: Audit_%Y_%m_%d_%H_%M (e.g. "CurrentUserEvents_1990_03_25_01_13")
- **Modified format (example)**: Audit_%d_%m_%Y (e.g. "CurrentUserEvents_25_03_1990")

The same applies when exporting an archive:

- **Default format**: Audit_%Y_%m_%d_%H_%M (e.g. "ArchivedUserEvent_1990_03_25_01_13")
- **Modified format (example)**: Archive_%d_%m_%Y (e.g. "ArchivedUserEvent_11_12_1983")

The following are additional options for adjusting the format:
Token | Description
--- | ---
%S | Seconds within a minute. Leading zero if necessary: 00 - 59
%L | Milliseconds within a second. Leading zero if necessary: 000 - 999
%z | Local shift to UTC time
%s | Seconds since 1 January 1970
%C | Four-digit year divided by 100. Displayed with two digits. Leading zero if necessary: 00 - 99
%Y | Year as a four-digit number. Zero at the beginning if necessary
%y | The last two digits of the year. Leading zero if necessary: 00 - 99
%m | Month. Leading zero if necessary: 01 - 31
%d | Day of the month. Leading zero if necessary: 01 - 31
%e | Day of the month: 1 - 31
%H | Time in 24-hour format: "%H:%M"
%M | Time in 24-hour format: "%H:%M:%S"
%D | Date in the following format: "%m/%d/%y"
%F | Date in ISO 8601 format: "%Y-%m-%d"
%c | Date and time in the following format: "%F %T.%L"

Data can be encrypted during the export. "Encrypt" must be selected for this. Decryption takes place using "PassPhrase". The longer the password, the more secure the data. If "Encrypt" = TRUE, then the encrypted data is saved as a ZIP file.

The decompression program used to unzip the data must support at least .zip format 5.1. 7zip and PKWare have been tested.

Each new entry is stored without text in memory. Only the IDs that reference the text sources are specified. The texts are only copied to the entry from the text source during the export. The "RawData" option can be used if the raw data should be exported. The raw data is appended to the end of the file/archive. Raw data is never appended to a PDF export (post-processing not possible).

Parameter "FileType" defines the file format used to export the data. There are several ways to influence the formatting of exported files (for a PDF, MpAuditExportPdfType can be used to define the font, for example).

Each entry requires a text source. This is where the text is taken from when creating the file/archive. This source is defined with "TextSource" (MpAuditTextSourceType). The source for the text in the entry can basically be the configuration or a VC4 text group. "Type" defines which source should be used.

- **Configuration:** If the configuration is used as text source, then the text from "OutputFormat" is used.
- **VC4 text group:** If VC4 is the source, the visualization object must be defined with "Name". The name of the visualization object must be specified. The text group is defined with "OutputFormatSource". The index of the text group must be specified. If the text group can be found in the shared resources, then a "g" must precede the index (e.g. "g3" - text group with index 3 in the shared resources). The text within the text group is specified with "OutputFormat". The text index is also specified again here (e.g. 0 - text with index 0 within the text group).
- **Text system:** Here, the text system is referenced. Both the namespace and the TextID must be specified.

Refer to section Creating entries for information about how entries can look and which options are available for including data in them. If the requested text is not found (incorrect text group or index specified), then the text defined with "ErrorMessage" (MpAuditErrorMessageType) is displayed. The same applies if a token was not specified correctly (see section Creating entries):

- **Invalid text group / index:** Text from "TextNotFound" used.
- **Invalid token:** Text from "SpecifierInvalid" used.

Archiving is configured using "Archive" (MpAuditArchiveType). "Enable" can be used to allow the automatic creation of archives. Input "Mode" defines the days on which archives are created. Everyday or just Monday through Friday are options. Parameter "Time" determines the time when archives are created. If a large amount of new entries occurs on a day and lost data is possible as a result, then an additional archive can also be created. Parameter "MaxSize" can be used to define when an additional archive should be created. If the new entries (those since the last time an archive was created) reach the "MaxSize", a new archive is created regardless of the time setting. If no new entries have occurred since the last archive, then no new archive is created. This example illustrates this procedure:

**Configuration**

- **Enable:** TRUE
- **Mode:** mpAUDIT_ARCHIVE_MO_TO_FR
- **Time:** 6:00 PM
- **MaxSize:** 100 kB
The following table represents a possible week:

<table>
<thead>
<tr>
<th>Day</th>
<th>Used memory for new entries since the last archive</th>
<th>Created archive</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>• 6:00 PM - 63 kB</td>
<td>• 6:00 PM - 1 archive</td>
<td>No new entries have occurred since the last archive; no new archive is created.</td>
</tr>
<tr>
<td>Tuesday</td>
<td>• 6:00 PM - 0 kB</td>
<td>• 6:00 PM - No archive is created</td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>• 6:00 PM - 86 kB</td>
<td>• 6:00 PM - 1 archive</td>
<td>At 3:33 PM, the number of new entries exceeds &quot;MaxSize&quot;; an additional archive is created as a result. At 6:00 PM, another archive is created (as configured).</td>
</tr>
<tr>
<td>Thursday</td>
<td>• 3:33 PM - 101 kB</td>
<td>• 3:33 PM - 1 archive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 6:00 PM - 13 kB</td>
<td>• 6:00 PM - 1 archive</td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>• 6:00 PM - 47 kB</td>
<td>• 6:00 PM - 1 archive</td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>Omitted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sunday</td>
<td>Omitted</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.5.1.10 MpAuditTrailUI

This function block allows access to the current event via the HMI application.

On our YouTube channel, the tutorials mapp Audit - Part 1 and mapp Audit - Part 2 explain how event management can be implemented.

Our B&R online tutorial platform includes tutorials about mapp Audit.

Function block

```
& MpComIdentType & MpAuditTrailUI
  MpLink
  BOOL Enable
  BOOL ErrorReset
  MpAuditTrailUISetupType UISetup
  BOOL Refresh
  & STRING Language
  & MpAuditTrailUIConnectType UIConnect

MpAuditInfoType Info

<table>
<thead>
<tr>
<th>Interface</th>
<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 MpAuditTrail configuration).</td>
<td></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>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>UISetup</td>
<td>MpAuditTrailUISetupType</td>
<td>Used to configure the elements connected to the HMI application.</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>Refresh</td>
<td>BOOL</td>
<td>A rising edge on this command updates the event list (e.g. after the language has been changed).</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>Language</td>
<td>Pointer to STRING[20]</td>
<td>Defines the language or unit used to display the events.</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>UIConnect</td>
<td>Pointer to MpAuditTrailUIConnectType</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpAuditInfoType</td>
<td>Additional information about the component.</td>
<td></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).
1.5.1.10.1 Description

The MpLink for this function block is the same that is used for function block MpAuditTrail. All events logged by MpAuditTrail are then displayed by MpAuditTrailUI. MpAuditTrailUI represents a connection between the currently stored events and a Visual Components 4 HMI application. The "Refresh" command updates the display of events in the HMI application. Input parameter "Language" defines the language or unit used when exporting events. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.

Data is exchanged between MpAuditTrailUI and the HMI application using structure MpAuditTrailUIConnectType.

MpAuditTrailUIConnectType

Structure "UIConnect" is divided into the following areas:

- **Status**: Parameter "Status" provides information about the current activity of event management.
- **Output**: Represents the current events in the HMI application
- **Filter**: Offers the option of filtering events according to certain criteria

MpAuditTrailUISetupType

"EventListSize" defines in this structure how many entries from the file list should be displayed on one page of the HMI application. "ScrollWindow" determines how many entries from the list are initially displayed when scrolling up and down.

1.5.1.11 MpAuditValueChange

This function logs the changes made to a value.

**Function**

```
MpComIdentType
MpAuditValueChange
MpLink
LREAL Old
LREAL New
STRING[100] Identifier
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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Old</td>
<td>LREAL</td>
<td>Entered when creating the entry under %old (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>New</td>
<td>LREAL</td>
<td>Entered when creating the entry under %new (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>Identifier</td>
<td>STRING[100]</td>
<td>Entered when creating the entry under %dpid (see Creating entries).</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>DINT</td>
<td>Status information of function.</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).

1.5.1.11.1 Description

The MpLink used here comes from the MpAuditTrail configuration to which this event is assigned.

The function is supplied with the old and new value of a LREAL variable via inputs "Old" and "New". Parameter "Identifier" is used to identify the information source (e.g. the name of the PV whose value changed).

For information about which data can be incorporated into this event, see "Value change" in section Event types.
1.5.1.12 MpAuditVC4Event

This function block logs events in the HMI application.

**Function**

<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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>BOOL</td>
<td>Enable</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>BOOL</td>
<td>ErrorReset</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>BOOL</td>
<td>Localize</td>
<td>Defines whether the user ID or description of the user ID is used to create an entry (see VC4 event system under MpAudit).</td>
</tr>
<tr>
<td>UINT</td>
<td>MinUserID</td>
<td>First ID from VC4 (see VC4 event system under MpAudit) for which events are logged.</td>
</tr>
<tr>
<td>UINT</td>
<td>MaxUserID</td>
<td>Last ID from VC4 (see VC4 event system under MpAudit) for which events are logged.</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 MpAuditTrail 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>Localize</td>
<td>BOOL</td>
<td>Defines whether the user ID or description of the user ID is used to create an entry (see VC4 event system under MpAudit).</td>
</tr>
<tr>
<td>IN</td>
<td>MinUserID</td>
<td>UINT</td>
<td>First ID from VC4 (see VC4 event system under MpAudit) for which events are logged.</td>
</tr>
<tr>
<td>IN</td>
<td>MaxUserID</td>
<td>UINT</td>
<td>Last ID from VC4 (see VC4 event system under MpAudit) for which events are logged.</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>
</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).

1.5.1.12.1 Description

The MpLink used here comes from the MpAuditTrail configuration to which this event is assigned.

This function block logs events that have been defined using the VC4 event system. All UserID value changes between "MinUserID" and "MaxUserID" are recorded, and an entry is created.

Input "Localize" is used to decide whether the user ID or the description of the user ID is used when creating an entry if %dpid is used to put the exact data of the events in the entry.

For information about which data in this event can be included when recording, see "VC4 value change" in section Event types.

1.5.1.13 MpAuditWStringChange

This function logs the changes made to a Unicode string.
Function block

```
MpAuditWStringChange
MpComIdentType MpLink
WSTRING[100] Old
WSTRING[100] New
STRING[100] Identifier
DINT ReturnValue
```

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 MpAuditTrail configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Old</td>
<td>WSTRING[100]</td>
<td>Entered when creating the entry under %old (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>New</td>
<td>WSTRING[100]</td>
<td>Entered when creating the entry under %new (see Creating entries).</td>
</tr>
<tr>
<td>IN</td>
<td>Identifier</td>
<td>STRING[100]</td>
<td>Entered when creating the entry under %dpid (see Creating entries).</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>DINT</td>
<td>Status information of function.</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).

1.5.1.13.1 Description

The MpLink used here comes from the MpAuditTrail configuration to which this event is assigned.

The function is supplied with the old and new value of the Unicode string via inputs "Old" and "New". Parameter "Identifier" is used to identify the information source (e.g. the name of the Unicode string where the value changed).

For information about which data in this event can be included when recording, see "Value change" in section Event types.

1.5.2 Data types and enumerators

1.5.2.1 Data types

1.5.2.1.1 MpAuditArchiveType

Settings for archiving events

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Switches the archive function on/off</td>
</tr>
<tr>
<td>Mode</td>
<td>MpAuditArchiveModeEnum</td>
<td>mpAUDIT_ARCHIVE_DAILY</td>
<td>Defines which weekdays archiving takes place</td>
</tr>
<tr>
<td>Time</td>
<td>TIME_OF_DAY</td>
<td></td>
<td>Defines which time archiving takes place</td>
</tr>
<tr>
<td>MaxSize</td>
<td>UDINT</td>
<td>100</td>
<td>If the number of events reaches this size, an archive is created immediately regardless of the day/time [kB].</td>
</tr>
</tbody>
</table>

1.5.2.1.2 MpAuditDiagType

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>MpAuditStatusIDType</td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>
### 1.5.2.1.3 MpAuditErrorMessageType

This data type defines how a missing text is replaced. If MpAudit cannot find the desired text, this text will be output in its place.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextNotFound</td>
<td>STRING[20]</td>
<td><em>(Err:Text)</em></td>
<td>Text that is displayed if the desired index was not found</td>
</tr>
<tr>
<td>SpecifierInvalid</td>
<td>STRING[20]</td>
<td><em>(Err:Specifier)</em></td>
<td>Text that is displayed if the desired icon was not found or is not available for this event</td>
</tr>
</tbody>
</table>

### 1.5.2.1.4 MpAuditExportFilterType

This data type indicates how events are currently filtered.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>MpAuditTrailUICurrDTFilterType</td>
<td>Time from which the export files are displayed</td>
</tr>
<tr>
<td>Until</td>
<td>MpAuditTrailUICurrDTFilterType</td>
<td>Time until which the export files are displayed</td>
</tr>
<tr>
<td>Event</td>
<td>MpAuditTrailUIEventFilterType</td>
<td>Defines which events are displayed</td>
</tr>
<tr>
<td>Operator</td>
<td>MpAuditTrailUIOpFilterType</td>
<td>Determines from which user events are displayed</td>
</tr>
<tr>
<td>Pattern</td>
<td>MpAuditTrailUIPatternFilterType</td>
<td>Defines the event text to be used for filtering</td>
</tr>
<tr>
<td>Batch</td>
<td>MpAuditTrailUIBatchFilterType</td>
<td>Defines which batch should be displayed</td>
</tr>
</tbody>
</table>

### 1.5.2.1.5 MpAuditExportPdfType

This data type provides information about how the PDF export of the audit trail should look.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultFont</td>
<td>MpAuditFontTypeEnum</td>
<td>Standard font. Used if no other font was defined for the current language.</td>
</tr>
<tr>
<td>Format</td>
<td>MpAuditPageFormatEnum</td>
<td>Page format</td>
</tr>
<tr>
<td>Direction</td>
<td>MpAuditPageDirectionEnum</td>
<td>Defines the orientation: landscape or portrait.</td>
</tr>
<tr>
<td>Margins</td>
<td>MpAuditPageMarginEnum</td>
<td>Defines the distance between the text and edge of page.</td>
</tr>
<tr>
<td>OwnerPassword</td>
<td>STRING[50]</td>
<td>It is only possible to change permissions for copying, printing and editing if this password is known.</td>
</tr>
<tr>
<td>UserPassword</td>
<td>STRING[50]</td>
<td>The user will be asked to enter this password when opening the PDF file. If a password is not specified here, then a password will not be required to open the PDF.</td>
</tr>
<tr>
<td>AllowPrint</td>
<td>BOOL</td>
<td>Defines whether the PDF file can be printed</td>
</tr>
<tr>
<td>AllowCopy</td>
<td>BOOL</td>
<td>Defines whether the PDF file can be copied</td>
</tr>
<tr>
<td>AllowEdit</td>
<td>BOOL</td>
<td>Defines whether the PDF file can be edited</td>
</tr>
<tr>
<td>Icon</td>
<td>STRING[100]</td>
<td>An icon can be integrated in the header, specify the path here. The only format supported is .jpg. The icon is always displayed in its original size. Format: DeviceName/FolderName/FolderName/FileName.jpg</td>
</tr>
</tbody>
</table>
| Header             | STRING[100]   | Header information can be specified here. The text source is the same as for the other event texts, so it is defined in the configuration. It might look something like this:  

- Configuration as source: "Enter text directly"  
- Text system as source: "{$Namespace/TextID}"  
- VC4 as source: "group/index"  

<table>
<thead>
<tr>
<th>Fonts</th>
<th>Array[0..4] of MpAuditLanguageFontType</th>
<th>Defines which font should be used for which language</th>
</tr>
</thead>
</table>
### 1.5.2.1.6 MpAuditExportType

Describes how the export of files/archives is proceeding.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileNamePattern</td>
<td>STRING[50]</td>
<td>&quot;Audit_%Y_%m_%d_%H_%M&quot;</td>
<td>Filename including timestamp. This format can be changed.</td>
</tr>
<tr>
<td>ArchiveNamePattern</td>
<td>STRING[50]</td>
<td>&quot;Archive_%Y_%m_%d_%H_%M&quot;</td>
<td>Archive name including timestamp. This format can be changed.</td>
</tr>
<tr>
<td>PassPhrase</td>
<td>STRING[100]</td>
<td>&quot;DefaultPassword1&quot;</td>
<td>Password (or phrase) for the encrypted files / archive.</td>
</tr>
<tr>
<td>Encrypt</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Encrypted export files</td>
</tr>
<tr>
<td>RawData</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Attaches raw data to the export file</td>
</tr>
<tr>
<td>FileType</td>
<td>MpAuditFileTypeEnum</td>
<td>mpAUDIT_FILE_TYPE_TXT</td>
<td>File type for the export file</td>
</tr>
<tr>
<td>PDF</td>
<td>MpAuditExportPdfType</td>
<td>PDF export settings</td>
<td>File type for the export file</td>
</tr>
</tbody>
</table>

### 1.5.2.1.7 MpAuditInfoType

This data type provides additional information for the MpAudit component.

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

### 1.5.2.1.8 MpAuditLanguageFontType

This data type defines the export language and the font to be used.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>STRING[20]</td>
<td>Language for PDF export</td>
</tr>
<tr>
<td>Font</td>
<td>MpAuditFontTypeEnum</td>
<td>Font for PDF export</td>
</tr>
</tbody>
</table>

### 1.5.2.1.9 MpAuditQueryInfoType

This data type provides additional information for function block MpAuditQuery.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableSpace</td>
<td>UDINT</td>
<td>Maximum number of entries that can be specified in the PV.</td>
</tr>
<tr>
<td>RowsRead</td>
<td>UDINT</td>
<td>Number of entries read.</td>
</tr>
<tr>
<td>HasMoreRows</td>
<td>BOOL</td>
<td>Specifies whether more information is available.</td>
</tr>
<tr>
<td>Diag</td>
<td>MpAuditDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

### 1.5.2.1.10 MpAuditStatusIDType

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>MpAuditErrorEnum</td>
<td>Error code for mapp component</td>
</tr>
<tr>
<td>Severity</td>
<td>MpComSeveritiesEnum</td>
<td>Describes the type of information supplied by the status ID (success, information, warning, error)</td>
</tr>
<tr>
<td>Code</td>
<td>UINT</td>
<td>Code for the status ID. This error number can be used to search for additional information in the help system.</td>
</tr>
</tbody>
</table>

### 1.5.2.1.11 MpAuditTextSourceType

This data type defines which text will be used for the individual events when creating an archive/file.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>MpAuditTextSourceEnum</td>
<td>mpAUDIT_TEXT_SOURCE_NONE</td>
<td>Type of text source</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>STRING[50]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OutputFormat</strong></td>
<td>STRING[255]</td>
<td>&quot;[%[TIME=%c]] [%ev] [%op]&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>OutputFormatSource</strong></td>
<td>STRING[100]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ErrorMessage</strong></td>
<td>MpAuditErrorMessageType</td>
<td></td>
<td>Replacement text that is added if the specified ID is not found</td>
</tr>
</tbody>
</table>

1.5.2.1.12 MpAuditTrailConfigType

This data type is used to configure event logging.

**SRAM:** When using battery-backed memory, there must be sufficient memory reserved in the controller configuration. Can only be used if the hardware being used is equipped with SRAM memory.

![Memory configuration diagram](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>Array[0..9] of UINT</td>
<td>[16#10,16#12,8(0)]</td>
<td>Defines which events are logged (see Creating entries). Different Event types can be split up between multiple MpAuditTrail instances. The following events are logged by default:</td>
</tr>
<tr>
<td>RecordingSize</td>
<td>UDINT</td>
<td>200</td>
<td>Reserved memory for events [kB]</td>
</tr>
<tr>
<td>SramBufferSize</td>
<td>UDINT</td>
<td>20</td>
<td>Reserved memory for events in SRAM [kB] (this parameter is only used if &quot;RecordMemory&quot; is set to mpAUDIT_MEM_BUFFERED_ROM)</td>
</tr>
</tbody>
</table>
### 1.5.2.1.13 MpAuditTrailInfoType

This data type provides additional information for the `MpAuditTrail` component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberArchives</td>
<td>UINT</td>
<td>Number of available archives</td>
</tr>
<tr>
<td>Diag</td>
<td><code>MpAuditDiagType</code></td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

### 1.5.2.1.14 MpAuditTrailUIBatchFilterType

This data type defines the filter for a batch.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter</td>
</tr>
<tr>
<td>Name</td>
<td><code>WSTRING[50]</code></td>
<td>Specifies a batch</td>
</tr>
</tbody>
</table>

### 1.5.2.1.15 MpAuditTrailUIConnectType

This data type contains all of the information necessary to establish a connection to the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td><code>MpAuditTrailUIStatusEnum</code></td>
<td>Current operation</td>
</tr>
<tr>
<td>Output</td>
<td><code>MpAuditTrailUIOutputType</code></td>
<td>Indicates the filtered events</td>
</tr>
<tr>
<td>Filter</td>
<td><code>MpAuditUIFilterType</code></td>
<td>Allows events to be filtered</td>
</tr>
<tr>
<td>Export</td>
<td>BOOL</td>
<td>Command for exporting (filtered) data</td>
</tr>
<tr>
<td>Message box</td>
<td><code>MpAuditUIMessageBoxType</code></td>
<td>Controls dialog boxes</td>
</tr>
</tbody>
</table>

### 1.5.2.1.16 MpAuditTrailUICurrDTFilterType

Indicates from/to which date events should be sorted

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter</td>
</tr>
<tr>
<td>DateTime</td>
<td><code>DATE_AND_TIME</code></td>
<td>Time from/to which filtering should take place</td>
</tr>
</tbody>
</table>

### 1.5.2.1.17 MpAuditTrailUICurrentFilterType

This data type indicates how events are currently filtered.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>MpAuditTrailUICurrDTFilterType</td>
<td>Time from which events are displayed</td>
</tr>
<tr>
<td>Until</td>
<td>MpAuditTrailUICurrDTFilterType</td>
<td>Time until which events are displayed</td>
</tr>
<tr>
<td>Event</td>
<td>MpAuditTrailUIEventFilterType</td>
<td>Defines which events are displayed</td>
</tr>
<tr>
<td>Operator</td>
<td>MpAuditTrailUIOpFilterType</td>
<td>Determines from which user events are displayed</td>
</tr>
<tr>
<td>Pattern</td>
<td>MpAuditTrailUIPatternFilterType</td>
<td>Defines the event messages to be used for filtering</td>
</tr>
<tr>
<td>Batch</td>
<td>MpAuditTrailUIBatchFilterType</td>
<td>Defines which batch should be displayed</td>
</tr>
</tbody>
</table>

### 1.5.2.1.18 MpAuditTrailUIEventFilterType

Filtered according to the type of event.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Exclude</td>
<td>BOOL</td>
<td>Either only this event type (FALSE) or all but this event type (TRUE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Event</td>
<td>ARRAY[0..4] of UDINT</td>
<td>Events that should be filtered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Numeric</td>
</tr>
</tbody>
</table>

### 1.5.2.1.19 MpAuditTrailUIFilterDialogType

This data type can be used to filter events.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer of the visualization page where the filter is configured. VC4 connection: StatusDatapoint from Layer</td>
</tr>
<tr>
<td>From</td>
<td>MpAuditTrailUISetDTFilterType</td>
<td>Time from which events are displayed</td>
</tr>
<tr>
<td>Until</td>
<td>MpAuditTrailUISetDTFilterType</td>
<td>Time until which events are displayed</td>
</tr>
<tr>
<td>Event</td>
<td>MpAuditTrailUIEventFilterType</td>
<td>Defines which events are displayed</td>
</tr>
<tr>
<td>Operator</td>
<td>MpAuditTrailUIOpFilterType</td>
<td>Determines from which user events are displayed</td>
</tr>
<tr>
<td>Confirm</td>
<td>BOOL</td>
<td>Confirms the operation</td>
</tr>
<tr>
<td>Cancel</td>
<td>BOOL</td>
<td>Cancels the operation</td>
</tr>
<tr>
<td>Pattern</td>
<td>MpAuditTrailUIPatternFilterType</td>
<td>Filter for the event messages</td>
</tr>
<tr>
<td>Batch</td>
<td>MpAuditTrailUIBatchFilterType</td>
<td>Defines which batch should be displayed</td>
</tr>
</tbody>
</table>
1.5.2.1.20 MpAuditTrailUIOpFilterType

Filtered by user.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter. VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Exclude</td>
<td>BOOL</td>
<td>Only this user (FALSE) / All other users (TRUE). VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Name</td>
<td>STRING[50]</td>
<td>Operator name. VC4 connection: Datapoint from String.</td>
</tr>
</tbody>
</table>

1.5.2.1.21 MpAuditTrailUIOutputType

List with events and associated navigation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventID</td>
<td>Array[0..19] of UINT</td>
<td>List of IDs from all events meeting the current filter criteria. VC4 connection: BitmapIndexDatapoint from Button (bitmap group for various events icons), see information</td>
</tr>
<tr>
<td>EventText</td>
<td>Array[0..19] of WSTRING</td>
<td>List of all events meeting the current filter criteria. VC4 connection: TextIndexOffset from Button (text group for events), see information</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. The size of the page is defined with parameter &quot;PointListSize&quot; in structure MpAuditTrailUISetupType. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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. The size of the page is defined with parameter “PointListSize” in structureMpAuditTrailUISetupType. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>StepUp</td>
<td>BOOL</td>
<td>Selects the previous entry in the list VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>StepDown</td>
<td>BOOL</td>
<td>Selects the next entry in the list VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</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 “Range” for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. “Range” should correspond to the number of entries that are displayed on one page. VC4 connection: StartDatapoint from Scale</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 “Range” for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. “Range” should correspond to the number of entries that are displayed on one page. VC4 connection: EndDatapoint from Scale</td>
</tr>
</tbody>
</table>

**none**

Info:

Buttons are used to display events. The text an event has is defined by a text snippet in a text group. The size of the text group results from the number of maximum events that can be displayed at one time (in this case 0-19). A text snippet is assigned to each element in the text group with a different index. (TextSnippet[0] connected withMpAuditTrailUIConnect.Output.EventText[0])

**Figure 3: Text snippet**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name3</td>
<td>String</td>
<td>DatapointEvent[3]</td>
</tr>
<tr>
<td>Name2</td>
<td>String</td>
<td>DatapointEvent[2]</td>
</tr>
<tr>
<td>Name1</td>
<td>String</td>
<td>DatapointEvent[1]</td>
</tr>
<tr>
<td>Name0</td>
<td>String</td>
<td>DatapointEvent[0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Detailed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name3</td>
<td>String</td>
<td>DatapointEvent[3]</td>
</tr>
<tr>
<td>Name2</td>
<td>String</td>
<td>DatapointEvent[2]</td>
</tr>
<tr>
<td>Name1</td>
<td>String</td>
<td>DatapointEvent[1]</td>
</tr>
<tr>
<td>Name0</td>
<td>String</td>
<td>DatapointEvent[0]</td>
</tr>
</tbody>
</table>

**Figure 4: Events text group**
none

Display event icons:

To easily tell the difference between different events, it is a good idea to assign an icon for each event type. This can be done using a bitmap group. The bitmap group is created and a bitmap added for each event type. (e.g. VC4 event, user event, etc.)

![Bitmap group for events](image)

**Important!**

The index of the bitmap in the bitmap group must match the event IDs (e.g. VC4 events have index 17).

1.5.2.1.22 MpAuditTrailUIPatternFilterType

This data type defines the filter for the event messages.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter</td>
</tr>
<tr>
<td>Exclude</td>
<td>BOOL</td>
<td>Either only this event message (FALSE) or all but this event message (TRUE)</td>
</tr>
<tr>
<td>Compare</td>
<td>WSTRING[50]</td>
<td>Specifies the event message. Wildcard <code>*</code> can be used.</td>
</tr>
</tbody>
</table>

1.5.2.1.23 MpAuditTrailUISetDTFilterType

Time from or to which events should be filtered
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Year</td>
<td>UINT</td>
<td>Year from/to which filtering should take place</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Numeric</td>
</tr>
<tr>
<td>Month</td>
<td>USINT</td>
<td>Month from/to which filtering should take place</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Numeric</td>
</tr>
<tr>
<td>Day</td>
<td>USINT</td>
<td>Day from/to which filtering should take place</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Numeric</td>
</tr>
<tr>
<td>Hour</td>
<td>USINT</td>
<td>Hour from/to which filtering should take place</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Numeric</td>
</tr>
<tr>
<td>Minute</td>
<td>USINT</td>
<td>Minute from/to which filtering should take place</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Numeric</td>
</tr>
</tbody>
</table>

### 1.5.2.1.24 MpAuditTrailUISetupType

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>EventListSize</td>
<td>UINT</td>
<td>20</td>
<td>Number of events to be displayed on one page of the list in the HMI application. Up to 20 events can be displayed.</td>
</tr>
<tr>
<td>ScrollWindow</td>
<td>USINT</td>
<td>0</td>
<td>Determines how many entries from the list are initially displayed when scrolling up and down</td>
</tr>
</tbody>
</table>

**EventListSize**

EventListSize specifies how many events are displayed on one side of a VC4-based HMI application. The number of buttons in the HMI application that are used to show the events must be equal to EventListSize. (See image above.)

**ScrollWindow**

ScrollWindow specifies how many event entries in the current list should be displayed on the next page when pressing the PageUp and PageDown buttons. (See example.)

**Example:**

![Example Image]

*Note:* The LineUp and LineDown buttons move the entire list one event up or down. (See image below.)
1.5.2.1.25 MpAuditUIFilterType

Makes it possible to filter events.

<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></td>
<td></td>
<td>VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
<tr>
<td>Dialog box</td>
<td>MpAuditTrailUIGenericDialogType</td>
<td>Window for selecting an event filter</td>
</tr>
<tr>
<td>Current</td>
<td>MpAuditTrailUICurrentFilterType</td>
<td>Displays the current filter</td>
</tr>
<tr>
<td>DefaultLayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer on the visualization page on which the filter options are available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: StatusDatapoint from Layer</td>
</tr>
</tbody>
</table>

1.5.2.1.26 MpAuditUIMessageBoxType

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>MpAuditUIEnum</td>
<td>Type of dialog box</td>
</tr>
</tbody>
</table>
| ErrorNumber             | UINT          | Current error number to be displayed (corresponds to the contents of "Dia-
|                         |               | g.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                                                       |
|                         |               | VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1    |
| Cancel                  | BOOL          | Cancels the operation                                                        |
|                         |               | VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1    |

1.5.2.2 Enumerators

1.5.2.2.1 MpAuditArchiveModeEnum

This enumerated data type defines how often an archive is created.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_ARCHIVE_DAILY</td>
<td>Creates a new archive daily</td>
</tr>
<tr>
<td>mpAUDIT_ARCHIVE_MO_TO_FR</td>
<td>Creates an archive from Monday to Friday</td>
</tr>
<tr>
<td>mpAUDIT_ARCHIVE_BATCH</td>
<td>Creates an archive as soon as a new batch is started</td>
</tr>
</tbody>
</table>

1.5.2.2.2 MpAuditFileTypeEnum

This enumerator is used to define the file type for the export.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_FILE_TYPE_TXT</td>
<td>Export as text file</td>
</tr>
<tr>
<td>mpAUDIT_FILE_TYPE_XML</td>
<td>Export as .xml file</td>
</tr>
<tr>
<td>mpAUDIT_FILE_TYPE_PDF</td>
<td>Export as PDF file</td>
</tr>
</tbody>
</table>

1.5.2.2.3 MpAuditFontTypeEnum

This enumerated data type specifies all the fonts that can be used when generating a PDF.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_FONT_COURIER</td>
<td>Courier</td>
</tr>
<tr>
<td>mpAUDIT_FONT HELVETICA</td>
<td>Helvetica</td>
</tr>
<tr>
<td>mpAUDIT_FONT_TIMESROMAN</td>
<td>Times Roman</td>
</tr>
</tbody>
</table>
1.5.2.2.4MpAuditMemoryEnum
This enumerated data type specifies where the logged data is stored in a ring buffer.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_MEM_TEMP</td>
<td>Memory location: DRAM (temporary)</td>
</tr>
<tr>
<td>mpAUDIT_MEM_ROM</td>
<td>Memory location: USER ROM (memory card)</td>
</tr>
<tr>
<td>mpAUDIT_MEM_SRAM</td>
<td>Memory location: SRAM (battery-backed)</td>
</tr>
<tr>
<td>mpAUDIT_MEM_BUFFERD_ROM</td>
<td>Memory location: SRAM (USER ROM buffered)</td>
</tr>
</tbody>
</table>

1.5.2.2.5MpAuditPageDirectionEnum
The user can define the format in which the PDF files should be exported.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_PAGE_DIRECTION_UNDEFINED</td>
<td>Format undefined</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_DIRECTION_PORTRAIT</td>
<td>Portrait</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_DIRECTION_LANDSCAPE</td>
<td>Landscape</td>
</tr>
</tbody>
</table>

1.5.2.2.6MpAuditPageFormatEnum
The user can define the format in which the PDF files should be exported.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_PAGE_FORMAT_UNDEFINED</td>
<td>Format undefined</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_LETTER</td>
<td>Format: Letter</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_LEGAL</td>
<td>Format: Legal</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_A3</td>
<td>Format: A3</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_A4</td>
<td>Format: A4</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_A5</td>
<td>Format: A5</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_B4</td>
<td>Format: B4</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_FORMAT_B5</td>
<td>Format: B5</td>
</tr>
</tbody>
</table>

1.5.2.2.7MpAuditPageMarginEnum
The user can define the format in which the PDF files should be exported.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_PAGE_MARGIN_NORMAL</td>
<td>Normal distance between content and edge of page</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_MARGIN_NARROW</td>
<td>Reduced distance between content and edge of page</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_MARGIN_MODERATE</td>
<td>Increased distance between content and edge of page</td>
</tr>
<tr>
<td>mpAUDIT_PAGE_MARGIN_WIDE</td>
<td>Much increased distance between content and edge of page</td>
</tr>
</tbody>
</table>

1.5.2.2.8MpAuditQueryModeEnum
This enumerated data type specifies how the event list is to be searched.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_QUERY_MODE_ALL</td>
<td>All entries in the event list are searched.</td>
</tr>
<tr>
<td>mpAUDIT_QUERY_MODE_NEW</td>
<td>Only the newest entries in the event list are searched.</td>
</tr>
</tbody>
</table>

1.5.2.2.9MpAuditTextSourceEnum
This enumerated data type specifies the source of the text for archiving or creating a file.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpAUDIT_TEXT_SOURCE_NONE</td>
<td>Event text from the configuration</td>
</tr>
<tr>
<td>mpAUDIT_TEXT_SOURCE_VC4</td>
<td>Event text taken from VC4</td>
</tr>
</tbody>
</table>
### 1.5.2.2.10 MpAuditTrailUIStatusEnum

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>mpAUDIT_UI_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpAUDIT_UI_STATUS_UPDATE</td>
<td>Performs an update</td>
</tr>
<tr>
<td>mpAUDIT_UI_STATUS_FILTER</td>
<td>Displays a window for filtering the results</td>
</tr>
<tr>
<td>mpAUDIT_UI_STATUS_EXPORT</td>
<td>Performs an export</td>
</tr>
<tr>
<td>mpAUDIT_UI_STATUS_ERROR</td>
<td>The last operation generated an error.</td>
</tr>
</tbody>
</table>

### 1.5.2.2.11 MpAuditUIMessageEnum

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>mpAUDIT_UI_MSG_NONE</td>
<td>No dialog box</td>
</tr>
<tr>
<td>mpAUDIT_UI_MSG_ERROR</td>
<td>Dialog box: Error</td>
</tr>
</tbody>
</table>

### 1.5.3 Status numbers

#### 1.5.3.1 1083359157: MpAuditTrail not yet active

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

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

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

**Constant:**
mpAUDIT_INF_WAIT_AUDIT_FB

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

#### 1.5.3.2 1083359152: No entries in archive

**Description:**
The archive has no entries.

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

**Cause/Solution:**
- Archive not created since no entries exist at the defined point in time (MpAuditArchiveType).

**Constant:**
mpAUDIT_INF_ARCHIVE_NO_ENTRIES

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

#### 1.5.3.3 -2138040062: Owner password changed

**Description:**
The password of the owner was changed because it was identical to the user password (2:PasswordMask).
Additional information
- {2:PasswordMask}: Masked password, e.g. (***1). Only changed characters are displayed.

Cause/Solution:
- The password of the owner was identical to the user password.

Constant:
mpAUDIT_WRN_OWNER_PASSWORD_CHANGED

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

1.5.3.4 -2138040060: OPC server disabled

Description:
The OPC listener could not be created. Error source: {1:ErrorNumber}
Value changes should be recorded, but the OPC server needed for this is not enabled.

Additional information
- {1:ErrorNumber}: Reason for the error

Constant:
mpAUDIT_INT_WRN_OPC_CREATE

Cause/Solution:
- Check whether the OPC server is enabled.
- Check the properties of the OPC server.
- See section OPC UA events.

1.5.3.5 -2137980917: Warning loading configuration

Description:
A warning occurred while loading the configuration file (in the PV). 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:
- Configuration file damaged
- Difference between the contents of the configuration file and the PV (e.g. more data in the configuration file than in the PV)

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

Constant:
mpAUDIT_WRN_CONFIG_LOAD

1.5.3.6 -2137866328: Fallback language being used

Description:
The fallback language is being used.
Reaction:
The function block indicates an active warning on output "StatusID".

Cause/Solution:
- The specified language was not found; the fallback language is being used instead.
- The fallback language is defined with "TextSource" (MpAuditTextSourceType).

Constant:
mpAUDIT_WRN_LANGUAGE_FALLBACK

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport

1.5.3.7 -2137866327: Input "Language" not evaluated

Description:
The "Language" language is not evaluated in VC4.

Reaction:
The function block indicates an active warning on output "StatusID".

Cause/Solution:
- The current language is being used instead of the language set with input "Language".

Constant:
mpAUDIT_WRN_LANGUAGE_IGNORED

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport

1.5.3.8 -2137866313: OPC auditing not enabled

Description:
An error occurred while logging OPC UA variables.

Reaction:
The function block indicates an active warning on output "StatusID".

Cause/Solution:
- Enable "Auditing server facet" in the hardware configuration under OPC UA.

Constant:
mpAUDIT_WRN_OPC_AUDIT_DISABLED

These function blocks / functions can report this error:
- MpAuditTrail
1.5.3.9 -1064298240: Could not create OPC listener

Description:
The OPC listener could not be created. Error source: {1:ErrorNumber}

Additional information

• {1:ErrorNumber}: Reason for the error

Cause/Solution:

• Check whether the OPC server is enabled.
• Check the properties of the OPC server.
• See section OPC UA events.

Constant:

mpAUDIT_ERR_OPC_CREATE

These function blocks / functions can report this error:

• MpAuditTrail

1.5.3.10 -1064298239: Error creating archive server

Description:
Error creating archive server {2:ServerName}. Error source: {1:ErrorNumber}

Additional information

• {1:ErrorNumber}: Reason for the error
• {2:ServerName}: Name of the server

Cause/Solution:

• If SRAM was specified as the memory location, check whether enough memory space has been reserved.
• Specify another memory location, e.g. UserROM.

Constant:

mpAUDIT_ERR_ARCHIVE_SERVER_CREATE

These function blocks / functions can report this error:

• MpAuditTrail

1.5.3.11 -1064298237: PDF functionality not available

Description:
PDF functionality is not available. The necessary library MpReport is missing.

Cause/Solution:

• Add library MpReport.

Constant:

mpAUDIT_ERR_PDF_NOT_AVAILABLE

These function blocks / functions can report this error:

• MpAuditTrail
• MpAuditExport
1.5.3.12 -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:
- MpAuditTrail
- MpAuditTrailConfig
- MpAuditTrailUI
- MpAuditVC4Event
- MpAuditExport

Constant:
mpAUDIT_ERR_ACTIVATION

1.5.3.13 -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:
- MpAuditTrail
- MpAuditTrailConfig
- MpAuditTrailUI
- MpAuditVC4Event
- MpAuditCustomEvent
- MpAuditValueChange
- MpAuditStringChange
- MpAuditWStringChange
- MpAuditExport

Constant:
mpAUDIT_ERR_MPLINK_NULL

1.5.3.14 -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:

• MpAuditTrail
• MpAuditTrailConfig
• MpAuditTrailUI
• MpAuditVC4Event
• MpAuditCustomEvent
• MpAuditValueChange
• MpAuditStringChange
• MpAuditWStringChange
• MpAuditExport

Constant:

mpAUDIT_ERR_MPLINK_INVALID

1.5.3.15 -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:

• MpAuditTrail
• MpAuditTrailConfig
• MpAuditTrailUI
• MpAuditVC4Event
• MpAuditCustomEvent
• MpAuditValueChange
• MpAuditStringChange
• MpAuditWStringChange
• MpAuditExport

Constant:

mpAUDIT_ERR_MPLINK_CORRUPT

1.5.3.16 -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:
• MpAuditTrail
• MpAuditTrailConfig
• MpAuditTrailUI
• MpAuditVC4Event
• MpAuditCustomEvent
• MpAuditValueChange
• MpAuditStringChange
• MpAuditWStringChange
• MpAuditExport

Constant:
mpAUDIT_ERR_MPLINK_CHANGED

1.5.3.17 -1064239096: Configuration structure is a null pointer

Description:
Input "Configuration" 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 "Configuration" on the function block.

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

Constant:
mpAUDIT_ERR_CONFIG_NULL

1.5.3.18 -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:

- MpAuditTrail
- MpAuditTrailConfig
- MpAuditTrailUI
- MpAuditVC4Event
- MpAuditCustomEvent
- MpAuditValueChange
- MpAuditStringChange
- MpAuditWStringChange

Constant:

mpAUDIT_ERR_MPLINK_IN_USE

1.5.3.19 -1064239095: Configuration pointer not a PV

Description:
The specified pointer is not a pointer to a valid PV. 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:
- Check the value on input "Configuration".

These function blocks / functions can report this error:

- MpAuditTrailConfig

Constant:

mpAUDIT_ERR_CONFIG_NO_PV

1.5.3.20 -1064239094: Error loading configuration

Description:
An error occurred while loading the configuration file (in the PV). 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:
- Configuration file damaged
- Difference between the contents of the configuration file and the PV (e.g. different data types, data outside valid limits, etc.)

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

mpAUDIT_ERR_CONFIG_LOAD

1.5.3.21 -1064239092: Error saving configuration

Description:
An error occurred while saving to the configuration file (from the PV). 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:

• Configuration file damaged
• Difference between the contents of the configuration file and the PV (e.g. different data types, data outside valid limits, etc.)

These function blocks / functions can report this error:

• MpAuditTrailConfig

Constant:

mpAUDIT_ERR_CONFIG_SAVE

1.5.3.22 -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:

• MpAuditTrail

Constant:

mpAUDIT_ERR_CONFIG_INVALID

1.5.3.23 -1064124508: Text source not found

Description:
The specified text source could not be found.

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

- Visualization object not found
- Check the "TextSource" configuration parameter (MpAuditTextSourceType).

Constant:

mpAUDIT_ERR_TEXT_SOURCE

These function blocks / functions can report this error:

- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport

1.5.3.24 -1064124507: Text not found in text source

Description:
The requested text was not found in the text source.

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

Cause/Solution:

- The requested text was not found in the text source.

Constant:

mpAUDIT_ERR_TEXT_NAMESPACE

These function blocks / functions can report this error:

- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport

1.5.3.25 -1064124506: Text ID not found

Description:
The specified text ID could not be found.

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

Cause/Solution:

- Text group or text index was not found.
- Check whether the text file was added in the text configuration.

Constant:

mpAUDIT_ERR_TEXT_IDENT

These function blocks / functions can report this error:

- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport
1.5.3.26 -1064124501: Error writing archive

Description:
An error occurred while writing the archive. 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:
- Not enough memory
- See Logger for additional information.

Constant:
mpAUDIT_ERR_ARCHIVE_WRITE

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

1.5.3.27 -1064124502: Archive not found

Description:
The specified archive was not found.

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

Cause/Solution:
- An archive that should have been exported can no longer be found on the controller.

Constant:
mpAUDIT_ERR_ARCHIVE_NOT_FOUND

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

1.5.3.28 -1064124505: Language not found

Description:
The specified language was not found.

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 language ID.

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

- MpAuditTrail
- MpAuditTrailUI
- MpAuditExport

1.5.3.29 -1064124500: Text too long

Description:
The text for this event is too long.

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

Cause/Solution:

- The text from the text group in VC4 is too long.
- The text is truncated or empty.

Constant:
mpAUDIT_ERR_TEXT_OVERFLOW

These function blocks / functions can report this error:

- MpAuditTrail
- MpAuditExport

1.5.3.30 -1064124499: Could not allocate memory

Description:
The memory needed for logging event could not be reserved. 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:

- Not enough memory available
- Change the type of memory ("RecordMemory" in MpAuditTrailConfigType).
- Check whether sufficient UserRAM memory was allocated for the "Memory configuration" entry in the hardware configuration. Set the memory to its maximum size.
- Check whether the hardware being used has SRAM memory.
- See Logger for additional information.

Constant:
mpAUDIT_ERR_BUFFER_CREATE

These function blocks / functions can report this error:

- MpAuditTrail

1.5.3.31 -1064124498: Error writing export file

Description:
An error occurred while writing the export file. 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:
- Check input "DeviceName".
- See Logger for additional information.

Constant:
mpAUDIT_ERR_WRITE_EXPORT_FILE

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditExport

1.5.3.32 -1064124497: Error reading events

Description:
An error occurred while reading the events from memory/archive.

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

Cause/Solution:
- Archive damaged
- Current event memory damaged

Constant:
mpAUDIT_ERR_READ_BUFFER_ENTRIES

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditExport

1.5.3.33 -1064124495: Event not recorded

Description:
An event could not be recorded. 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:
- Event logging is canceled until the error is reset via input "ErrorReset".
- See the logger for additional information.

Constant:
mpAUDIT_ERR_EVENT_RECORDER

These function blocks / functions can report this error:
- MpAuditTrail
1.5.3.34 -1064124494: Invalid file device

Description:
The specified file device on input "DeviceName" 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:
- Nothing is attached to input "DeviceName".
- The attached file device does not exist.
- A connection to the desired file device could not be established.

Constant:
mpAUDIT_ERR_INVALID_FILE_DEV

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditExport

1.5.3.35 -1064124493: Maximum number of archives reached

Description:
The maximum number of archives has been reached. Additional archives are not possible.

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

Cause/Solution:
- Archives should be exported as soon as they are available.

Constant:
mpAUDIT_ERR_MAX_ARCHIVES

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

1.5.3.36 -1064124492: 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:
mpAUDIT_ERR_MISSING_UICONNECT
These function blocks / functions can report this error:

- MpAuditTrailUI

1.5.3.37 -1064124490: Error reading VC4 events

Description:
Errors occurred while reading VC4 events 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 VC4-based HMI application is not yet started (event system not ready).
- See Logger for additional information.

Constant:
mpAUDIT_ERR_READ_VC_EVENTS

These function blocks / functions can report this error:
- MpAuditVC4Event

1.5.3.38 -1064124488: PV registration failed

Description:
Registration of a PV via the configuration or function block MpAuditRegPar failed.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time. The PV is re-evaluated on a rising edge on the enable input.

Cause/Solution:
- PV was not correctly specified on the input of MpAuditRegPar. Details about how the PV must be specified can be found here.
- The PV cannot be registered if MpAuditTrail is not active ("Active" = FALSE).
- PV name is NULL.
- PV with this name not used by the system; check the name/address of the PV.
- The data type of the PV cannot be registered. See Logger for additional information.

Constant:
mpAUDIT_ERR_MONITOR_PV

These function blocks / functions can report this error:
- MpAuditTrail
- MpAuditRegPar

1.5.3.39 -1064124487: Batch ID missing

Description:
No name was assigned to the batch.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time. The PV is re-evaluated on a rising edge on the enable input.
**Cause/Solution:**
- Specify a batch name.

**Constant:**

```cpp
mpAUDIT_ERR_NO_BATCH_ID
```

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

**1.5.3.40 -1064124486: Invalid unit system**

**Description:**
An invalid unit system was specified.

**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 unit system being used.

**Constant:**

```cpp
mpAUDIT_ERR_MEASSYSTEM
```

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

**1.5.3.41 -1064124485: Request not found**

**Description:**
The specified request could not be found.

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

**Cause/Solution:**
- Check if the request exists in the MpAuditTrail configuration.

**Constant:**

```cpp
mpAUDIT_ERR_QUERY_NOT_FOUND
```

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

**1.5.4 Alarms**

**1.5.4.1 mpAUDIT_ALM_ARCHIVE_AVAILABLE: Archive available**

**Description:**
Archive ready for export

**Reaction:**
A mapp alarm is triggered if a MpAlarmX component is active.
Cause/Solution:

- New archive ready for export

Behavior

Persistent alarm

1.5.4.2 mpAUDIT_ALM_ARCHIVE_OVERFLOW: Archiving stopped

Description:
The maximum number of archives has been reached. Additional archives are not possible.

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

Cause/Solution:

- Create memory space for new archives.
- Remove old archives.

Behavior

Edge alarm