Library Description

WAGO Software
CODESYS Library
WagoLibNetSnmpManager.lib
for implementing SNMP Manager functions

Version 1.1.0
Every conceivable measure has been taken to ensure the accuracy and completeness of this documentation. However, as errors can never be fully excluded, we always appreciate any information or suggestions for improving the documentation.

E-Mail: documentation@wago.com

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally protected by trademark or patent.
# Table of Contents

1 Notes about this Documentation ................................................................. 4  
1.1 Copyright ................................................................................................... 4  
1.2 Symbols ..................................................................................................... 5  
1.3 Number Notation ....................................................................................... 7  
1.4 Font Conventions ...................................................................................... 7  

2 WagoLibNetSnmpManager.lib................................................................... 8  
2.1 SMNPM_Version ...................................................................................... 9  
2.2 SNMPM_DINT_TO_TLV ...................................................................... 10  
2.3 SNMPM_NULL_TO_TLV ..................................................................... 12  
2.4 SNMPM_UDINT_TO_TLV ................................................................... 13  
2.5 SNMPM_STRING_TO_TLV ................................................................. 15  
2.6 SNMPM_TLV_TO_DINT ...................................................................... 17  
2.7 SNMPM_TLV_TO_UDINT ................................................................... 19  
2.8 SNMPM_TLV_TO_STRING .................................................................. 21  
2.9 SNMPM_GET ......................................................................................... 23  
2.10 SNMPM_GET_V3 .................................................................................. 25  
2.11 SNMPM_SET ......................................................................................... 27  
2.12 SNMPM_SET_V3 .................................................................................. 29  
2.13 SNMPT_SEND_ENT_TRAP ................................................................. 31  
2.14 SNMPT_SEND_TRAP .......................................................................... 32  
2.15 SNMPT_SEND_TRAP_TO_ADR_V1 .................................................. 33  
2.16 SNMPT_SEND_TRAP_TO_ADR_V2c ................................................ 34  
2.17 SNMPT_SEND_TRAP_TO_ADR_V3 .................................................. 35  
2.18 SNMPT_SEND_INFORM_ADR ........................................................... 37  
2.19 SNMPT_SEND_INFORM_ADR_V3 .................................................... 39  
2.20 tSNMPM_TLV ....................................................................................... 41  
2.21 tSNMPM_TLV_DATATYPE ................................................................ 42  
2.22 tSNMPM_SecurityLevel ......................................................................... 43  
2.23 tSNMPM_AuthenticationProtocol .......................................................... 44  
2.24 tSNMPM_PrivacyProtocol ..................................................................... 45  
2.25 tSNMPT_TRAP_TYPE .......................................................................... 46  

List of Tables ........................................................................................................ 47
1 Notes about this Documentation

1.1 Copyright

This Manual, including all figures and illustrations, is copyright-protected. Any further use of this Manual by third parties that violate pertinent copyright provisions is prohibited. Reproduction, translation, electronic and phototechnical filing/archiving (e.g., photocopying) as well as any amendments require the written consent of WAGO Kontakttechnik GmbH & Co. KG, Minden, Germany. Non-observance will involve the right to assert damage claims.
1.2 Symbols

**DANGER**

Personal Injury!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

**DANGER**

Personal Injury Caused by Electric Current!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

**WARNING**

Personal Injury!
Indicates a moderate-risk, potentially hazardous situation which, if not avoided, could result in death or serious injury.

**CAUTION**

Personal Injury!
Indicates a low-risk, potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

**NOTICE**

Damage to Property!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

**NOTICE**

Damage to Property Caused by Electrostatic Discharge (ESD)!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

**Note**

Important Note!
Indicates a potential malfunction which, if not avoided, however, will not result in damage to property.
**Information**

Additional Information:
Refers to additional information which is not an integral part of this documentation (e.g., the Internet).
1.3 **Number Notation**

Table 1: Number Notation

<table>
<thead>
<tr>
<th>Number Code</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>100</td>
<td>Normal notation</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>0x64</td>
<td>C notation</td>
</tr>
<tr>
<td>Binary</td>
<td>'100'</td>
<td>In quotation marks, nibble separated with</td>
</tr>
<tr>
<td></td>
<td>'0110.0100'</td>
<td>dots (.)</td>
</tr>
</tbody>
</table>

1.4 **Font Conventions**

Table 2: Font Conventions

<table>
<thead>
<tr>
<th>Font Type</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>italic</em></td>
<td>Names of paths and data files are marked in italic-type.</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>C:\Program Files\WAGO Software</em></td>
</tr>
<tr>
<td>Menu</td>
<td>Menu items are marked in bold letters.</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>Save</em></td>
</tr>
<tr>
<td>&gt;</td>
<td>A greater-than sign between two names means the selection of a</td>
</tr>
<tr>
<td></td>
<td>menu item from a menu.</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>File &gt; New</em></td>
</tr>
<tr>
<td>Input</td>
<td>Designation of input or optional fields are marked in bold letters,</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>Start of measurement range</em></td>
</tr>
<tr>
<td>“Value”</td>
<td>Input or selective values are marked in inverted commas.</td>
</tr>
<tr>
<td></td>
<td>e.g.: Enter the value “4 mA” under <em>Start of measurement range</em>.</td>
</tr>
<tr>
<td>[Button]</td>
<td>Pushbuttons in dialog boxes are marked with bold letters in square</td>
</tr>
<tr>
<td></td>
<td>brackets.</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>[Input]</em></td>
</tr>
<tr>
<td>[Key]</td>
<td>Keys are marked with bold letters in square brackets.</td>
</tr>
<tr>
<td></td>
<td>e.g.: <em>[F5]</em></td>
</tr>
</tbody>
</table>
2 WagoLibNetSnmpManager.lib

The “WagoLibNetSnmpManager.lib” library is used to implement SNMP manager functions. This makes it possible to access other SNMP agents as well as call and modify OIDs via an IEC program.
2.1 SMNPM_Version

The “SNMPM_Version” function is used to determine the version of the library implemented.

<table>
<thead>
<tr>
<th>Category</th>
<th>Query version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_Version</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_Version</td>
<td>WORD</td>
<td>Library version</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPM_VERSION
SNMPM_Version : WORD
```

**Description**

This function returns the version with which the library was implemented in the firmware.
2.2 **SNMPM_DINT_TO_TLV**

Using the “SNMPM_DINT_TO_TLV” function you can initialize a “tSNMPM_TLV” entity with a signed 32-bit integer value. A DINT entity is transferred for this as the initialization value, and the target data type is also selected.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_DINT_TO_TLV</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diInput</td>
<td>DINT</td>
<td>Value to be converted in SNMP-TLV</td>
</tr>
<tr>
<td>stDatatype</td>
<td>tSNMPM_TLV_ DATATYPE</td>
<td>Specification of the SNMP target data type</td>
</tr>
</tbody>
</table>

**Input/output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container for SNMP data</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_DINT_TO_TLV</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPM_DINT_TO_TLV
---diInput : DINT
---stDatatype : tSNMPM_TLV_DATATYPE
---siData : tSNMPM_TLV (VAR_IN_OUT)
```
**Description**

This function is used to initialize a `tSNMPM_TLV` entity with the value given by `diInput` and the data type specified by `stDatatype`.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 5 indicates that conversion is not possible. This can occur, for example, when an attempt is made to convert a signed value (DINT) to an unsigned value (`SNMPM_TLV_GAUGE`). The valid SNMP data type `SNMP_TLV_INTEGER` exists for this function.

A return value of 7 indicates that there is insufficient memory in the device, meaning that `stData` could not be correctly initialized.
### 2.3 SNMPM_NULL_TO_TLV

The “SNMPM_NULL_TO_TLV” function is used to clear a “tSNMPM_TLV” entity. This instance can then be used, for example, to send traps without appending any additional information.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_NULL_TO_TLV</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

#### Input/output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container for SNMP data</td>
</tr>
</tbody>
</table>

#### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_NULL_TO_TLV</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

#### Graphical illustration

```
SNMPM_NULL_TO_TLV

stData : tSNMPM_TLV (VAR_IN_OUT) SNMPM_NULL_TO_TLV : INT
stData : tSNMPM_TLV (VAR_IN_OUT)
```

#### Description

This function is used to initialize a tSNMPM_TLV entity with ZERO.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 7 indicates that there is insufficient memory in the device, meaning that stData could not be correctly initialized.
2.4 **SNMPM_UDINT_TO_TLV**

The “SNMPM_UDINT_TO_TLV” function is used to initialize a “tSNMPM_TLV” entity with a signed 32-bit integer value. In this process, a UDINT entity is transferred as the initialization value and the target data type is also selected.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_UDINT_TO_TLV</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>udiInput</td>
<td>UDINT</td>
<td>Value to be converted in SNMP-TLV</td>
</tr>
<tr>
<td>stDatatype</td>
<td>tSNMPM_TLV_DATATYPE</td>
<td>Specification of the SNMP target data type</td>
</tr>
</tbody>
</table>

### Input/output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container for SNMP data</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_UDINT_TO_TLV</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
-- SNMPM_UDINT_TO_TLV
  udiInput: UDINT               SNMPM_UDINT_TO_TLV: INT
  stDatatype : tSNMPM_TLV_DATATYPE
  sData : tSNMPM_TLV (VAR_IN_OUT)
  sData : tSNMPM_TLV (VAR_IN_OUT)
```
This function is used to initialize a tSNMPM_TLV entity with the value given in udiInput and the data type specified by stDatatype.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 5 indicates that conversion is not possible. This can occur, for example, when an attempt is made to convert a positive value (DINT) to a signed value (SNMPM_TLV_INTEGER).

The valid SNMP data types SNMP_TLV_GAUGE, SNMPM_TLV_TIMETICKS, SNMPM_TLV_COUNTER and SNMPM_TLV_UINTEGER exist for this function.

A return value of 7 indicates that there is insufficient memory in the device, meaning that stData could not be correctly initialized.
2.5 SNMPM_STRING_TO_TLV

Using the “SNMPM_STRING_TO_TLV” function you can initialize a “tSNMPM_TLV” entity with a string (STRING) of up to 255 characters. In this process, a STRING(255) entity is transferred as the initialization value and the target data type also selected. Strings can also be converted to numerical values.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_STRING_TO_TLV</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sInput</td>
<td>STRING(255)</td>
<td>String to be converted in SNMP-TLV</td>
</tr>
<tr>
<td>stDatatype</td>
<td>tSNMPM_TLV_DA</td>
<td>Specification of the SNMP target data type</td>
</tr>
</tbody>
</table>

### Input/output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container for SNMP data</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_STRING_TO_TLV</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
SNMPM_STRING_TO_TLV

sInput : STRING(255)          SNMPM_STRING_TO_TLV : INT
stDatatype : tSNMPM_TLV_DA   stData : tSNMPM_TLV (VAR_IN_OUT)
stData : tSNMPM_TLV (VAR_IN_OUT)
```
Description
Using this function, a tSNMPM_TLV entity is initialized with the string transferred in sInput and the data type specified by stDatatype. An attempt is made to convert the string into numerical data types when required, meaning that this function can convert to nearly all types of data.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 5 indicates that conversion is not possible.

A return value of 7 indicates that there is insufficient memory in the device.

A return value of 8 indicates that it is not possible to convert stData to a string.
2.6 **SNMPM_TLV_TO_DINT**

Using the “SNMPM_TLV_TO_DINT” function you can extract the associated value from a “tSNMPM_TLV” entity that has been initialized using a signed 32-bit integer value. A DINT variable is transferred as the target entity in this process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_TLV_TO_DINT</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container from which the DINT value is to be extracted</td>
</tr>
</tbody>
</table>

**Input/output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diValue</td>
<td>DINT</td>
<td>Target for DINT value</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_TLV_TO_DINT</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPM_TLV_TO_DINT

stData : tSNMPM_TLV           SNMPM_TLV_TO_DINT : INT

diValue : DINT (VAR_IN_OUT)  diValue : DINT (VAR_IN_OUT)
```
**Description**

This function is used to extract the value from a tSNMPM_TLV (stData) entity that was previously initialized with a signed 32-bit value and write this value to the target variable diValue.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 5 indicates that conversion is not possible. This can occur, for example when stData does not contain a signed 32-bit value (e.g., SNMPM_TLV_GAUGE).
### 2.7 SNMPM_TLV_TO_UDINT

The “SNMPM_TLV_TO_UDINT” function is used to extract the associated value from a “tSNMPM_TLV” entity that has been initialized using a signed 32-bit integer value. A UDINT variable is transferred as the target entity in this process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_TLV_TO_UDINT</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

#### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container from which the UDINT value is to be extracted</td>
</tr>
</tbody>
</table>

#### Input/output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>udiValue</td>
<td>UDINT</td>
<td>Target for the UDINT value</td>
</tr>
</tbody>
</table>

#### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_TLV_TO_UDINT</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

#### Graphical illustration

```plaintext
SNMPM_TLV_TO_UDINT
  stData: tSNMPM_TLV
  SNMPM_TLV_TO_UDINT: INT
  udiValue: UDINT (VAR_IN_OUT) udiValue: UDINT (VAR_IN_OUT)
```
This function is used to extract the associated value from a tSNMPM_TLV (stData) entity that was previously initialized with an unsigned 32-bit value and to write this value to the target variable diValue.

A 0 (zero) is returned when this action is successful.

In the event of an error, the return value is not equal to 0.

A return value of 5 indicates that conversion is not possible. This can occur, for example, when stData contains a signed 32-bit value (e.g., SNMPM_TLV_INTEGER).
2.8 SNMPM_TLV_TO_STRING

Using the SNMPM_TLV_TO_STRING function, the respective value can be extracted as a STRING from a tSNMPM_TLV entity that has been initialized with any arbitrary value. A STRING(255) variable is transferred as the target entity in this process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Conversion of data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_TLV_TO_STRING</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container from which the value is to be extracted</td>
</tr>
</tbody>
</table>

### Input/output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sValue</td>
<td>STRING(255)</td>
<td>STRING value target</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_TLV_TO_STRING</td>
<td>INT</td>
<td>Error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 Successful</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Conversion not possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 String in stData is too long and has been cut off at 255 characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 Insufficient memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 String conversion not possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 stData has not been initialized.</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
SNMPM_TLV_TO_STRING

stData : tSNMPM_TLV
sValue : STRING(255) (VAR_IN_OUT)
```
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This function is used to extract the respective value as a string from a tSNMPM_TLV (stData) entity that was previously initialized and write this value to the target variable sValue.</td>
</tr>
<tr>
<td>A 0 (zero) is returned when this action is successful.</td>
</tr>
<tr>
<td>In the event of an error, the return value is not equal to 0.</td>
</tr>
<tr>
<td>A return value of 5 indicates that conversion is not possible.</td>
</tr>
<tr>
<td>A return value of 6 indicates that the string contained in stData is too long and that the return string has been cut off.</td>
</tr>
<tr>
<td>A return value of 9 indicates that stData has not been initialized.</td>
</tr>
</tbody>
</table>
2.9 SNMMPM_GET

The SNMMPM_GET function block calls OID data from a specified host, using SNMPv1 in the process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Query OIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMMPM_GET</td>
</tr>
<tr>
<td>Type</td>
<td>Function block</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sCommunity</td>
<td>STRING(63)</td>
<td>Indicates the SNMP community</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMMPM_TLV</td>
<td>Container for the results of a query</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wError</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
SNMMPM_GET
bEnable : BOOL       stData : tSNMMPM_TLV
sHost : STRING(127) bBusy : BOOL
sOID : STRING(127)  bDone : BOOL
sCommunity : STRING(63) wResult : WORD
```
**Description**

Using the SNMPM_GET function block, an individual “SNMP v1 GET” query can be transmitted and an OID queried from a remote or local host.

sHost is used here to indicate the target. The string “localhost” or IP 127.0.0.1 can be used for this to query data from the local host. Use of the “localhost” string is, however, recommended.

If the query has been completed successfully, the value for the queried OID is saved in stData. If an error occurs, the variable wResult receives a value not equal to 0.

The following error codes can be issued for this:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It was not possible to properly initialize the SNMP session.</td>
</tr>
<tr>
<td>2</td>
<td>An internal error has occurred.</td>
</tr>
<tr>
<td>3</td>
<td>Error while analyzing the OID string</td>
</tr>
<tr>
<td>4</td>
<td>Error during data exchange with the target host</td>
</tr>
<tr>
<td>10</td>
<td>Timeout: No response received from the target host.</td>
</tr>
<tr>
<td>11</td>
<td>The SNMP packet that was received contains corrupt data.</td>
</tr>
<tr>
<td>12</td>
<td>Critical error: The function block has not been properly initialized.</td>
</tr>
</tbody>
</table>
2.10 **SNMPM\_GET\_V3**

The SNMPM\_GET\_V3 function block calls OID data from the specified host, using SNMPv3 in the process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Query OIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_GET_V3</td>
</tr>
<tr>
<td>Type</td>
<td>Function block</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sUsername</td>
<td>STRING(127)</td>
<td>Indicates the SNMP user name</td>
</tr>
<tr>
<td>sAuthLevel</td>
<td>tSNMPM_SecurityLevel</td>
<td>Indicates the security level</td>
</tr>
<tr>
<td>sAuthProt</td>
<td>tSNMPM_AuthenticationProtocol</td>
<td>Indicates the protocol to be used for authentication</td>
</tr>
<tr>
<td>sAuthPass</td>
<td>STRING(127)</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>sPrivProt</td>
<td>tSNMPM_PrivacyProtocol</td>
<td>Indicates the encryption method to be used</td>
</tr>
<tr>
<td>sPrivPass</td>
<td>STRING(127)</td>
<td>Pass phrase for encryption</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container for the results of a query</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wError</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>
Graphical illustration

```
SNMPM_GET_V3

bEnable: BOOL
sHost: STRING(127)
sOID: STRING(127)
sUsername: STRING(127)
sAuthLevel: tSNMPM_SecurityLevel
sAuthProt: tSNMPM_AuthenticationProtocol
sAuthPass: STRING(127)
sPrivProt: tSNMPM_PrivacyProtocol
sPrivPass: STRING(127)

stData: tSNMPM_TLV
bBusy: BOOL
bDone: BOOL
wResult: WORD
```

Description

Using the SNMPM_GET_V3 function block, an individual “SNMP v3 GET” query can be transmitted and an OID queried from a remote or local host. sHost is used here to indicate the target. The string “localhost” or IP 127.0.0.1 can be used for this to query data from the local host. Use of the “localhost” string is, however, recommended.

In addition, the user name, the security level and, where applicable, the encryption protocols to be used must be indicated here. More information about this is given in the descriptions for the specific data types.

If the query has been completed successfully, the value for the queried OID is saved in stData. If an error occurs, the variable wResult receives a value not equal to 0.

The following error codes can be issued for this:

1. It was not possible to properly initialize the SNMP session.
2. An internal error has occurred.
3. Error while analyzing the OID string
4. Error during data exchange with the target host
10. Timeout: No response received from the target host
11. The SNMP packet that was received contains corrupt data.
12. Critical error: The function block has not been properly initialized.
13. The authentication pass phrase could not be generated.
14. The encryption pass phrase could not be generated.
2.11 **SNMPM_SET**

The SNMPM_SET function block sets OID data at the specified host, using SNMPv1 in the process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Set OIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPM_SET</td>
</tr>
<tr>
<td>Type</td>
<td>Function block</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sCommunity</td>
<td>STRING(63)</td>
<td>Indicates the SNMP community</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with the data item to be set</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wError</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPM_SET
---
 bEnable : BOOL
 sHost : STRING(127)
 sOID : STRING(127)
 sCommunity : STRING(63)
 stData : tSNMPM_TLV
```
### Description

Using the SNMPM_SET function block, an individual “SNMP v1 SET” query can be sent, allowing an OID to be set at a remote or local host. sHost is used here to indicate the target. The string “localhost” or IP 127.0.0.1 can be used for this to set data at the local host. Use of the “localhost” string is, however, recommended.

If an error occurs, the variable wResult receives a value not equal to 0.

The following error codes can be issued for this:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It was not possible to properly initialize the SNMP session.</td>
</tr>
<tr>
<td>2</td>
<td>An internal error has occurred.</td>
</tr>
<tr>
<td>3</td>
<td>Error while analyzing the OID string</td>
</tr>
<tr>
<td>4</td>
<td>Error during data exchange with the target host</td>
</tr>
<tr>
<td>10</td>
<td>Timeout: No response received from the target host</td>
</tr>
<tr>
<td>11</td>
<td>The SNMP packet that was received contains corrupt data.</td>
</tr>
<tr>
<td>12</td>
<td>Critical error: The function block has not been properly initialized.</td>
</tr>
</tbody>
</table>
2.12 **SNMPM_SET_V3**

The SNMPM_SET_V3 function block sets OID data at the specified host, using SNMPv3 in the process.

<table>
<thead>
<tr>
<th>Category</th>
<th>Set OIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>SNMPM_SET_V3</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Function block</td>
</tr>
<tr>
<td><strong>Name of library</strong></td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td><strong>Required libraries</strong></td>
<td>---</td>
</tr>
<tr>
<td><strong>Applicable to</strong></td>
<td>750-82xx</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started.</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sUsername</td>
<td>STRING(127)</td>
<td>Indicates the SNMP user name</td>
</tr>
<tr>
<td>sAuthLevel</td>
<td>tSNMPM_SecurityLevel</td>
<td>Indicates the security level</td>
</tr>
<tr>
<td>sAuthProt</td>
<td>tSNMPM_AuthenticationProtocol</td>
<td>Indicates the protocol to be used for authentication</td>
</tr>
<tr>
<td>sAuthPass</td>
<td>STRING(127)</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>sPrivProt</td>
<td>tSNMPM_PrivacyProtocol</td>
<td>Indicates the encryption method to be used</td>
</tr>
<tr>
<td>sPrivPass</td>
<td>STRING(127)</td>
<td>Pass phrase for encryption</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with the data item to be set</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wError</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>
**Graphical illustration**

```
SNMPM_SET_V3
```

- bEnable : BOOL
- sHost : STRING(127)
- OID : STRING(127)
- sUserName : STRING(127)
- sAuthLevel : tSNMPM_SecurityLevel
- sAuthProt : tSNMPM_AuthenticationProtocol
- sAuthPass : STRING(127)
- sPrivacy : tSNMPM_PrivacyProtocol
- sPrivPass : STRING(127)
- stData : tSNMPM_TLV
- bBusy : BOOL
- bDone : BOOL
- wResult : WORD

**Description**

Using the “SNMPM_SET_V3” function block, individual “SNMP v3 SET” queries can be sent, allowing an OID to be set at a remote or local host. 

sHost is used here to indicate the target. The string “localhost” or IP 127.0.0.1 can be used for this to set data at the local host. Use of the “localhost” string is, however, recommended.

In addition, the user name, the security level and the encryption protocols to be used must be indicated here. More information about this is given in the descriptions for the specific data types.

If the query has been completed successfully, the value for the queried OID is saved in stData. If an error occurs, the variable wResult receives a value not equal to 0.

The following error codes can be issued for this:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It was not possible to properly initialize the SNMP session.</td>
</tr>
<tr>
<td>2</td>
<td>An internal error has occurred.</td>
</tr>
<tr>
<td>3</td>
<td>Error while analyzing the OID string</td>
</tr>
<tr>
<td>4</td>
<td>Error during data exchange with the target host</td>
</tr>
<tr>
<td>10</td>
<td>Timeout: No response received from the target host</td>
</tr>
<tr>
<td>11</td>
<td>The SNMP packet that was received contains corrupt data.</td>
</tr>
<tr>
<td>12</td>
<td>Critical error: The function block has not been properly initialized.</td>
</tr>
<tr>
<td>13</td>
<td>The authentication pass phrase could not be generated.</td>
</tr>
<tr>
<td>14</td>
<td>The encryption pass phrase could not be generated.</td>
</tr>
</tbody>
</table>
The “SNMPT_SEND_ENT_TRAP” function sends a single user-specific trap (enterprise-trap). The targets for this trap can be configured via the WBM and CBM.

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wTrap</td>
<td>WORD</td>
<td>Trap number</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPT_SEND_ENT_TRAP</td>
<td>BOOL</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPT_SEND_ENT_TRAP
---------
wTrap : WORD SNMPT.Send.Ent.TRAP : BOOL
```

**Description**

Using the “SNMPT_SEND_ENT_TRAP” it is possible to send a so-called “enterprise”, i.e., a user-defined trap. The recipients entered in the SNMP configuration are used as the target or targets for this trap.
2.14 SNMPT_SEND_TRAP

The “SNMPT_SEND_TRAP” sends a single user-specific trap. The targets for this trap can be configured via the WBM and CBM.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPT_SEND_TRAP</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>stTrapType</td>
<td>tSNMPT_TRAP_TYPE</td>
<td>Type of trap</td>
</tr>
<tr>
<td>wSpecificType</td>
<td>WORD</td>
<td>Specific trap type</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPT_SEND_TRAP</td>
<td>INT</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPT_SEND_TRAP
   sEnterprise : STRING(127)  SNMPT_SEND_TRAP : INT
   stTrapType : tSNMPT_TRAP_TYPE
   wSpecificType : WORD
   stData : tSNMPM_TLV
```

**Description**

Using the “SNMPT_SEND_TRAP” function, it is possible to send an SNMP trap. The recipients entered in the SNMP configuration are used as the target or targets for this trap.
The “SNMPT_SEND_TRAP_TO_ADR_V1” sends a single user-specific trap in the SNMP protocol version 1 variant. The target is transferred to the function as a parameter.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sCommunity</td>
<td>STRING(63)</td>
<td>Indicates the SNMP community</td>
</tr>
<tr>
<td>stTrapType</td>
<td>tSNMPT_TRAP_TYPE</td>
<td>Type of trap</td>
</tr>
<tr>
<td>wSpecificType</td>
<td>WORD</td>
<td>Specific trap type</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPT_SEND_TRAP_TO_ADR_V1</td>
<td>INT</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
SNMPT_SEND_TRAP_TO_ADR_V1
    sHost : STRING(127)
    sEnterprise : STRING(127)
    sCommunity : STRING(63)
    stTrapType : tSNMPT_TRAP_TYPE
    wSpecificType : WORD
    stData : tSNMPM_TLV
```

### Description

Using the “SNMPT_SEND_TRAP_TO_ADR_V1” function, it is possible to send an SNMP trap. An IP address is transferred as the target or targets to the function for this trap.
2.16 SNMPT_SEND_TRAP_TO_ADR_V2c

The “SNMPT_SEND_TRAP_TO_ADR_V2c” sends a single user-specific trap in the variant of the SNP protocol version 2c. The target is transferred to the function as a parameter.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NMPT_SEND_TRAP_TO_ADR_V2c</td>
</tr>
<tr>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sCommunity</td>
<td>STRING(63)</td>
<td>Indicates the SNMP community</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID of the trap object</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPT_SEND_TRAP_TO_ADR_V2C</td>
<td>INT</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

### Graphical illustration

```
SNMPT_SEND_TRAP_TO_ADR_V2C
  sHost: STRING(127)          SNMP_T_SEND_TRAP_TO_ADR_V2C: INT
  sEnterprise: STRING(127)
  sCommunity: STRING(63)
  sOID: STRING(127)
  stData: tSNMPM_TLV
```

### Description

Using the “SNMPT_SEND_TRAP_TO_ADR_V2c” function, it is possible to send an SNMP trap.
An IP address is transferred as the target or targets to the function for this trap.
2.17 SNMPT_SEND_TRAP_TO_ADR_V3

The “SNMPT_SEND_TRAP_TO_ADR_V3” function sends a single user-specific trap in the variant of the SNMP protocol version 2c. The target is transferred to the function as a parameter.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Function</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sUsername</td>
<td>STRING(127)</td>
<td>Indicates the SNMP user name</td>
</tr>
<tr>
<td>sAuthLevel</td>
<td>tSNMPM_SecurityLevel</td>
<td>Indicates the security level</td>
</tr>
<tr>
<td>sAuthProt</td>
<td>tSNMPM_AuthenticationProtocol</td>
<td>Indicates the protocol to be used for authentication</td>
</tr>
<tr>
<td>sAuthPass</td>
<td>STRING(127)</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>sPrivProt</td>
<td>tSNMPM_PrivacyProtocol</td>
<td>Indicates the encryption method to be used</td>
</tr>
<tr>
<td>sPrivPass</td>
<td>STRING(127)</td>
<td>Pass phrase for encryption</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID of the trap object</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPT_SEND_TRAP_TO_ADR_V3</td>
<td>INT</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>
Using the “SNMPT_SEND_TRAP_TOADR_V3” function, it is possible to send an SNMP trap. An IP address is transferred as the target or targets to the function for this trap.
2.18 **SNMPT_SEND_INFORM_ADR**

The “SNMPT_SEND_INFORM_ADR” function block sends an “SNMP V2 Inform” message to a recipient defined with the call.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMP SEND_INFORM_ADR</td>
</tr>
<tr>
<td>Type</td>
<td>Function block</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

**Input parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sCommunity</td>
<td>STRING(63)</td>
<td>Indicates the SNMP community</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID of the trap object</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

**Output parameter**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wResult</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>

**Graphical illustration**

```
SNMPT_SEND_INFORM_ADR
    - bEnable : BOOL
    - sHost : STRING(127)
    - sEnterprise : STRING(127)
    - sCommunity : STRING(63)
    - sOID : STRING(127)
    - stData : tSNMPM_TLV
    - bBusy : BOOL
    - bDone : BOOL
    - wResult : WORD
```
**Description**

Using the “SNMPT_SEND_INFORM_ADR” function block it is possible to send an “SNMP v2 inform” message.

sHost is used here to indicate the target.

If an error occurs, the variable wResult receives a value not equal to 0.

The following error codes can be issued for this:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It was not possible to properly initialize the SNMP session.</td>
</tr>
<tr>
<td>2</td>
<td>An internal error has occurred.</td>
</tr>
<tr>
<td>3</td>
<td>Error while analyzing the OID string</td>
</tr>
<tr>
<td>4</td>
<td>Error during data exchange with the target host</td>
</tr>
<tr>
<td>12</td>
<td>Critical error: The function block has not been properly initialized.</td>
</tr>
</tbody>
</table>
2.19 **SNMPT_SEND_INFORM_ADR_V3**

Using the “SNMPT_SEND_INFORM_ADR_V3” function block it is possible to send an “SNMP V3 inform” message to a recipient defined with the call.

<table>
<thead>
<tr>
<th>Category</th>
<th>Send traps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SNMPT_SEND_INFORM_ADR_V3</td>
</tr>
<tr>
<td>Type</td>
<td>Function block</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Required libraries</td>
<td>---</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
</tbody>
</table>

### Input parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Notifies the function block that data exchange is to be started</td>
</tr>
<tr>
<td>sHost</td>
<td>STRING(127)</td>
<td>Indicates the target host name or an IP address</td>
</tr>
<tr>
<td>sEnterprise</td>
<td>STRING(127)</td>
<td>OID to be queried</td>
</tr>
<tr>
<td>sUsername</td>
<td>STRING(127)</td>
<td>Indicates the SNMP user name</td>
</tr>
<tr>
<td>sAuthLevel</td>
<td>tSNMPM_SecurityLevel</td>
<td>Indicates the security level</td>
</tr>
<tr>
<td>sAuthProt</td>
<td>tSNMPM_AuthenticationProtocol</td>
<td>Indicates the protocol to be used for authentication</td>
</tr>
<tr>
<td>sAuthPass</td>
<td>STRING(127)</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>sPrivProt</td>
<td>tSNMPM_PrivacyProtocol</td>
<td>Indicates the encryption method to be used</td>
</tr>
<tr>
<td>sPrivPass</td>
<td>STRING(127)</td>
<td>Pass phrase for encryption</td>
</tr>
<tr>
<td>sOID</td>
<td>STRING(127)</td>
<td>OID of the trap object</td>
</tr>
<tr>
<td>stData</td>
<td>tSNMPM_TLV</td>
<td>Container with a data item to be appended to the trap</td>
</tr>
</tbody>
</table>

### Output parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Indicates that the function block is still busy</td>
</tr>
<tr>
<td>bDone</td>
<td>BOOL</td>
<td>Indicates that the query has been completed</td>
</tr>
<tr>
<td>wResult</td>
<td>WORD</td>
<td>Return value/Error code</td>
</tr>
</tbody>
</table>
### Graphical illustration

```
SNMP.SendInform_ADR_V3
```

- `bEnable` : BOOL
- `sHost` : STRING(127)
- `sEnterprise` : STRING(127)
- `sUsername` : STRING(127)
- `sAuthLevel` : iSNMPM_SecurityLevel
- `sAuthProt` : iSNMPM_AuthenticationProtocol
- `sAuthPass` : STRING(127)
- `sPrivProt` : iSNMPM_PrivacyProtocol
- `sPrivPass` : STRING(127)
- `sOID` : STRING(127)
- `sData` : iSNMPM_TLV
- `wBusy` : BOOL
- `bDone` : BOOL
- `wResult` : WORD

### Description

Using the “SNMP.SendInform_ADR_V3” function block it is possible to send an “SNMP v3 INFORM” message.

`sHost` is used here to indicate the target.

If an error occurs, the variable `wResult` receives a value not equal to 0.

The following error codes can be issued for this:

1. It was not possible to properly initialize the SNMP session.
2. An internal error has occurred.
3. Error while analyzing the OID string
4. Error during data exchange with the target host
12. Critical error: The function block has not been properly initialized.
2.20 tSNMPM_TLV

The data type tSNMPM_TLV contains the OID raw data required for SNMP.

<table>
<thead>
<tr>
<th>Category</th>
<th>SNMP data container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>tSNMPM_TLV</td>
</tr>
<tr>
<td>Type</td>
<td>Data Type</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
<tr>
<td>Structure</td>
<td>STRUCT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net_snmp_variable_list</td>
<td>---</td>
<td>Byte array</td>
</tr>
</tbody>
</table>

Description

tSNMPM_TLV is the central data format for this library. Data that is to be sent or received must be available in this data format. The net-snmp equivalent of a TLV element is reflected in the IEC area, serving as the data basis for conversion functions.
2.21 tSNMPM_TLV_DATATYPE

The data type tSNMPM_TLV_DATATYPE declares SNMP data types.

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_TLV_INTEGER</td>
<td>2</td>
<td>INTEGER value (DINT)</td>
</tr>
<tr>
<td>SNMPM_TLV_BITS</td>
<td>3</td>
<td>BIT_STRING value (STRING)</td>
</tr>
<tr>
<td>SNMPM_TLV_STRING</td>
<td>4</td>
<td>STRING value (STRING)</td>
</tr>
<tr>
<td>SNMPM_TLV_OBJECT_ID</td>
<td>6</td>
<td>OID value (STRING)</td>
</tr>
<tr>
<td>SNMPM_TLV_IP_ADDRESS</td>
<td>64</td>
<td>IP address (STRING)</td>
</tr>
<tr>
<td>SNMPM_TLV_COUNTER</td>
<td>65</td>
<td>Counter value (UDINT)</td>
</tr>
<tr>
<td>SNMPM_TLV_GAUGE</td>
<td>66</td>
<td>UDINT value (UDINT)</td>
</tr>
<tr>
<td>SNMPM_TLV_TIMETICKS</td>
<td>67</td>
<td>Time value (UDINT)</td>
</tr>
<tr>
<td>SNMPM_TLV_UINTEGER</td>
<td>71</td>
<td>UDINT value (UDINT)</td>
</tr>
</tbody>
</table>

Description

tSNMPM_TLV_DATATYPE declares the SNMP data types that are available. SNMP offers a wide range of different data types, which have other designations in IEC, or which do not exist within IEC. The SNMP agents require that the data type be exactly specified in order to set an OID correctly. This enumeration offers the possibility of using some of the most common data types.
2.22 tSNMPM_SecurityLevel

The data type tSNMPM_SecurityLevel defines the available SNMP security levels.

<table>
<thead>
<tr>
<th>Category</th>
<th>SNMP security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>tSNMPM_SecurityLevel</td>
</tr>
<tr>
<td>Type</td>
<td>Data Type</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
<tr>
<td>Structure</td>
<td>ENUM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_SECURITY_NoAuthNoPriv</td>
<td>1</td>
<td>No authentication and no encryption</td>
</tr>
<tr>
<td>SNMPM_SECURITY_AuthNoPriv</td>
<td>2</td>
<td>Authenticate but do not encrypt</td>
</tr>
<tr>
<td>SNMPM_SECURITY_AuthPriv</td>
<td>3</td>
<td>Authenticate and encrypt</td>
</tr>
</tbody>
</table>

**Description**

tSNMPM_SecurityLevel defines the requested security level in the SNMPv3 protocol.
SNMPM_SECURITY_NoAuthNoPriv indicates that neither authentication nor encryption is to be employed (the communication is therefore just as insecure as with v1/v2c).
SNMPM_SECURITY_AuthNoPriv indicates that authentication is to be employed, but not encryption (everything is still transferred in plain text).
SNMPM_SECURITY_AuthPriv is the highest security level, meaning that not only are user name and password to be used, but encryption as well.
2.23 tSNMPM_AuthenticationProtocol

The data type tSNMPM_AuthenticationProtocol defines the algorithms available for generating a password hash for authentication.

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_AUTHP_NONE</td>
<td>1</td>
<td>No password</td>
</tr>
<tr>
<td>SNMPM_AUTHP_MD5</td>
<td>2</td>
<td>MD5 algorithm being used</td>
</tr>
<tr>
<td>SNMPM_AUTHP_SHA</td>
<td>3</td>
<td>SHA1 algorithm being used</td>
</tr>
</tbody>
</table>

**Description**

tSNMPM_AuthenticationProtocol defines the algorithm for generating a hash from the given password.
For communication via SNMP v3, passwords are not transmitted as plain-language words; hashes are generated instead. Different algorithms can also be specified when creating a user for generation of the hash values; the client must likewise select the correct algorithm.
If a security level without password authentication is selected, you can enter SNMPM_AUTHP_NONE here.
2.24 tSNMPM_PrivacyProtocol

The data type tSNMPM_PrivacyProtocol defines the encryption techniques that are available.

<table>
<thead>
<tr>
<th>Category</th>
<th>SNMP security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>tSNMPM_PrivacyProtocol</td>
</tr>
<tr>
<td>Type</td>
<td>Data Type</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx</td>
</tr>
<tr>
<td>Structure</td>
<td>ENUM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPM_PRIVP_NONE</td>
<td>1</td>
<td>No encryption</td>
</tr>
<tr>
<td>SNMPM_PRIVP_DES</td>
<td>2</td>
<td>DES encryption used</td>
</tr>
<tr>
<td>SNMPM_PRIVP_AES</td>
<td>3</td>
<td>AES encryption used</td>
</tr>
</tbody>
</table>

Description

tSNMPM_PrivacyProtocol defines the encryption to be used for “SNMP v3” communication.
If a security level without encryption is selected, you can enter SNMPM_PRIVP_NONE here.
2.25  tSNMPT_TRAP_TYPE

The data type tSNMPT_TRAP_TYPE defines the trap types available.

<table>
<thead>
<tr>
<th>Category</th>
<th>SNMP trap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>tSNMPT_TRAP_TYPE</td>
</tr>
<tr>
<td>Type</td>
<td>Data Type</td>
</tr>
<tr>
<td>Name of library</td>
<td>WagoLibNetSnmpManager.lib</td>
</tr>
<tr>
<td>Applicable to</td>
<td>750-82xx (from FW03)</td>
</tr>
<tr>
<td>Structure</td>
<td>ENUM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP_TRAP_COLDSTART</td>
<td>0</td>
<td>Cold Start</td>
</tr>
<tr>
<td>SNMP_TRAP_WARMSTART</td>
<td>1</td>
<td>Warm restart</td>
</tr>
<tr>
<td>SNMP_TRAP_LINKDOWN</td>
<td>2</td>
<td>ETH link lost</td>
</tr>
<tr>
<td>SNMP_TRAP_LINKUP</td>
<td>3</td>
<td>ETH link connected</td>
</tr>
<tr>
<td>SNMP_TRAP_AUTHFAILS</td>
<td>4</td>
<td>Authentication error</td>
</tr>
<tr>
<td>SNMP_TRAP_EGPNEIGHBORLOSS</td>
<td>5</td>
<td>EGP neighbor lost</td>
</tr>
<tr>
<td>SNMP_TRAP_ENTERPRISE_SPECIFIC</td>
<td>6</td>
<td>Enterprise-specific trap</td>
</tr>
</tbody>
</table>

Description

tSNMPT_TRAP_TYPE defines the types that can be sent with
SNMPT_SEND_TRAP and SNMPT_SEND_TRAP_TO_ADR_V1.
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1: Number Notation</td>
<td>7</td>
</tr>
<tr>
<td>Table 2: Font Conventions</td>
<td>7</td>
</tr>
</tbody>
</table>