

# Anybus<sup>®</sup> CompactCom<sup>™</sup> 40

## BACnet/IP w. IT Functionality

### NETWORK GUIDE

SCM-1202-040 1.4 en-US ENGLISH

---

# Important User Information

## Liability

Every care has been taken in the preparation of this document. Please inform HMS Industrial Networks of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks. HMS Industrial Networks assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks cannot assume responsibility for actual use based on these examples and illustrations.

## Intellectual Property Rights

HMS Industrial Networks has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the USA and other countries.

---

# Table of Contents

Page

<b>1</b>	<b>Preface .....</b>	<b>5</b>
1.1	About this document .....	5
1.2	Related Documents .....	5
1.3	Document History .....	5
1.4	Document Conventions .....	5
1.5	Document Specific Conventions .....	6
1.6	Trademark Information .....	6
<b>2</b>	<b>About the Anybus CompactCom 40 BACnet/IP .....</b>	<b>7</b>
2.1	General .....	7
2.2	Features .....	7
2.3	Fieldbus Conformance Notes .....	7
2.4	Certification .....	8
<b>3</b>	<b>Basic Operation .....</b>	<b>9</b>
3.1	General Information .....	9
3.2	Device Customization .....	9
3.3	BACnet/IP Implementation .....	10
3.4	Communication Settings .....	22
3.5	Diagnostics .....	22
3.6	Network Data Exchange .....	22
3.7	File System .....	25
<b>4</b>	<b>COV Notifications, Alarms and Events .....</b>	<b>27</b>
4.1	General .....	27
4.2	COV (Change of Value) Notifications .....	27
4.3	Alarm/Event Functionality .....	27
4.4	Setup of Alarm and Events .....	28
<b>5</b>	<b>FTP Server .....</b>	<b>32</b>
5.1	General Information .....	32
5.2	User Accounts .....	32
5.3	Session Example .....	33
<b>6</b>	<b>Web Server .....</b>	<b>34</b>
6.1	General Information .....	34
6.2	Default Web Pages .....	34
6.3	Server Configuration .....	38

<b>7</b>	<b>E-mail Client.....</b>	<b>41</b>
7.1	General Information .....	41
7.2	How to Send E-mail Messages .....	41
<b>8</b>	<b>Server Side Include (SSI) .....</b>	<b>42</b>
8.1	General Information .....	42
8.2	Include File .....	42
8.3	Command Functions.....	42
8.4	Argument Functions .....	57
8.5	SSI Output Configuration .....	61
<b>9</b>	<b>JSON .....</b>	<b>62</b>
9.1	General Information .....	62
9.2	JSON Objects.....	63
9.3	Example .....	81
<b>10</b>	<b>Anybus Module Objects.....</b>	<b>82</b>
10.1	General Information .....	82
10.2	Anybus Object (01h) .....	83
10.3	Diagnostic Object (02h) .....	84
10.4	Network Object (03h) .....	85
10.5	Network Configuration Object (04h) .....	87
10.6	Socket Interface Object (07h) .....	98
10.7	SMTP Client Object (09h).....	115
10.8	File System Interface Object (0Ah) .....	120
10.9	Network Ethernet Object (0Ch) .....	121
<b>11</b>	<b>Host Application Objects .....</b>	<b>122</b>
11.1	General Information .....	122
11.2	BACnet Host Object (EFh) .....	123
11.3	Ethernet Host Object (F9h) .....	129
<b>A</b>	<b>Categorization of Functionality .....</b>	<b>133</b>
A.1	Basic .....	133
A.2	Extended .....	133
<b>B</b>	<b>Implementation Details .....</b>	<b>134</b>
B.1	SUP-Bit Definition .....	134
B.2	Anybus State Machine .....	134
B.3	Application Watchdog Timeout Handling.....	134
B.4	Implemented BACnet BIBBs .....	134

---

<b>C</b>	<b>Secure HICP (Secure Host IP Configuration Protocol)</b>	<b>136</b>
C.1	General	136
C.2	Operation	136
<b>D</b>	<b>Technical Specification</b>	<b>137</b>
D.1	Front View	137
D.2	Functional Earth (FE) Requirements	138
D.3	Power Supply	138
D.4	Environmental Specification	138
D.5	EMC Compliance	138
<b>E</b>	<b>Backward Compatibility</b>	<b>139</b>
E.1	Initial Considerations	139
E.2	Hardware Compatibility	139
E.3	General Software	145
E.4	Network Specific — BACnet/IP	147
<b>F</b>	<b>Copyright Notices</b>	<b>148</b>

**This page intentionally left blank**

# 1 Preface

## 1.1 About this document

This document is intended to provide a good understanding of the functionality offered by the Anybus CompactCom 40 BACnet/IP. The document describes the features that are specific to Anybus CompactCom 40 BACnet/IP. For general information regarding Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design and communication systems in general. The information in this network guide should normally be sufficient to implement a design. However if advanced BACnet/IP specific functionality is to be used, in-depth knowledge of BACnet/IP networking internals and/or information from the official BACnet/IP specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the BACnet/IP specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

For additional related documentation and file downloads, please visit the support website at [www.anybus.com/support](http://www.anybus.com/support).

## 1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
BACnet specification	ASHRAE	Doc. Id. 135, 2008

## 1.3 Document History

Version	Date	Description
1.0	2017-06-14	First release
1.1	2017-07-10	Added appendix on backwards compatibility BACnet objects updated
1.2	2018-05-08	Minor update
1.3	2018-10-17	Minor correction to Network Configuration Object
1.4	2019-02-27	Rebranding Minor update

## 1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

- Itemized information
- Instructions that can be carried out in any order

...and for action-result type instructions:

- This action...
  - leads to this result

**Bold typeface** indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

Monospaced text is used to indicate program code and other kinds of data input/output such as configuration scripts.

This is a cross-reference within this document: [Document Conventions, p. 5](#)

This is an external link (URL): [www.hms-networks.com](http://www.hms-networks.com)



*This is additional information which may facilitate installation and/or operation.*



This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.



### Caution

This instruction must be followed to avoid a risk of personal injury.



### WARNING

This instruction must be followed to avoid a risk of death or serious injury.

## 1.5 Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.
- The terms “basic” and “extended” are used to classify objects, instances and attributes.

## 1.6 Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks.

All other trademarks are the property of their respective holders.



## 2 About the Anybus CompactCom 40 BACnet/IP

### 2.1 General

The Anybus CompactCom 40 BACnet/IP communication module provides instant BACnet and BACnet/IP connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

This product conforms to all aspects of the host interface for Anybus CompactCom 40 modules defined in the Anybus CompactCom 40 Hardware and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

### 2.2 Features

- Fulfills all requirements for a BACnet/IP device
- Two BACnet/IP ports
- Data sharing
- Linear network topology supported
- 100 Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- E-mail client
- JSON
- Server Side Include (SSI) functionality
- Customizable Identity Information
- 256 ADIs available in simple mode for mapping to BACnet objects
- A total of 256 ADIs per BACnet object type available in advanced mode for mapping to BACnet objects
- Transparent Socket Interface
- Change Of Value (COV) notification and Alarm/Event functionality supported (max 64 ADIs available)
- Support for Foreign Device Registration functionality

### 2.3 Fieldbus Conformance Notes

- BACnet International does not require a certification for the use of BACnet products.
- The Anybus CompactCom 40 BACnet/IP has not been tested by BACnet International. However it is recommended to test the final product for conformance with BACnet/IP.
- To enable the product to appear as a vendor specific implementation rather than a generic Anybus module, customize the information in the BACnet Object.

## 2.4 Certification

HMS Industrial Networks will not certify the Anybus CompactCom 40 BACnet/IP.

The module is implemented as a BACnet Application Specific Controller (B-ASC). Even though HMS Industrial Networks will not certify the module, the implementation fulfills the requirements for certification as a B-ASC. See [Implemented BACnet BIBBs, p. 134](#) for a complete list of BACnet BIBBs implemented in the module.

## 3 Basic Operation

### 3.1 General Information

#### 3.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 40 BACnet/IP, however due to the nature of the BACnet/IP networking system certain restrictions must be taken into account:

- There is no support for arrays of data elements in the ADIs as all data on BACnet is represented as single units without any possibility to access data in any other way.
- Data types UINT64, SINT64, BIT2-BIT7, BITSx, OCTET and PADx cannot be represented on BACnet. PADx can be used for explicit padding of process data, but can not be translated to any BACnet object.
- It is not possible to map read process data.

For in depth information regarding the Anybus CompactCom software interface, consult the general Anybus CompactCom 40 Software Design Guide.

See also ...

- Anybus-CompactCom 40 Software Design Guide, Application Data Object (FEh)
- [Network Object \(03h\), p. 85](#)

### 3.2 Device Customization

#### 3.2.1 Network Identity

By default, the module uses the following identity settings:

<b>Vendor Name:</b>	"HMS Industrial Networks"
<b>Vendor ID:</b>	01E6h (HMS Industrial Networks)
<b>Model Name</b>	"CompactCom 40 BACnet/IP"
<b>Object Name</b>	"CompactCom 40 BACnet/IP"
<b>Network Type</b>	009Ah("BACnet/IP")
<b>Product Name:</b>	CompactCom 40 BACnet/IP

Optionally, it is possible to customize the identity of the module by implementing the corresponding instance attributes in the BACnet Host Object.

See also...

- [BACnet Host Object \(EFh\), p. 123](#) (Host Application Object)
- [Network Object \(03h\), p. 85](#)

### 3.2.2 Web Interface

The web interface can be fully customized to suit a particular application. Data and web pages are stored in a FLASH-based file system, which can be accessed using any standard FTP-client.

See also...

- [File System, p. 25](#)
- [FTP Server, p. 32](#)
- [Web Server, p. 34](#)

### 3.2.3 Socket Interface (Advanced Users Only)

The built-in socket interface allows additional protocols to be implemented on top of TCP/IP.

See also...

- [Socket Interface Object \(07h\), p. 98](#) (Anybus Module Object)
- [Message Segmentation, p. 113](#)

## 3.3 BACnet/IP Implementation

It is recommended to enable attribute 7 (Support advanced mapping) in the BACnet Host Object, to fully take advantage of the functionality and flexibility of the module.

The module is implemented as a B-ASC (BACnet Application Specific Controller). It supports the following BACnet objects:

Object Name	Class
Device object	8
Analog Value object	2
Binary Value object	5
Multi-state Value object	19
Notification Class object	15

Each Anybus CompactCom 40 BACnet/IP contains one Device object and six Notification Class objects. These objects are fixed and can not be changed by the application.

The Analog Value, Binary Value, and Multi-State Value objects and their data are mapped against the ADIs in the Application Data object.

The BACnet Interoperability Building Blocks (BIBBs), that are implemented in the module, are listed in appendix [B](#).

See also...

- [Network Data Exchange, p. 22](#)
- Application Data Object (see Anybus-CompactCom Software Design Guide)
- [Implemented BACnet BIBBs, p. 134](#)
- [Application Data \(ADIs\), p. 22](#)
- [Mapping of BACnet Objects to Anybus CompactCom, p. 23](#)
- [BACnet Host Object \(EFh\), p. 123](#)

### 3.3.1 Device Object

The BACnet device object contains information about the module as a node on a BACnet network. Apart from the value of the Object\_Identifier, the values of the properties in the object can not be changed by the application directly. Some values can be changed by setting the corresponding attributes in the BACnet Host object.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off.

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R/W		The instance number portion (device instance) of this property is affected when the value attribute of Instance 3 in the Network Configuration Object is changed.
Object_Name	"CompactCom 40 BACnet/IP"	R/W		This property can be written to only if the Object Name in the BACnet Host object can be set. This property can be changed by setting the corresponding attribute in the BACnet Host object
Object_Type	DEVICE	R		
System_Status	NON_OPERATIONAL OPERATIONAL	R		The status of the system is reported as NON_OPERATIONAL if the Anybus CompactCom module has entered the ERROR state. Otherwise the state is reported as OPERATIONAL.
Vendor_Name	"HMS Industrial Networks"	R		This property can be changed by setting the corresponding attribute in the BACnet Host object
Vendor_Identifier	486	R		HMS Industrial Networks This property can be changed by setting the corresponding attribute in the BACnet Host object
Model_Name	"CompactCom 40 BACnet/IP"	R		This property can be changed by setting the corresponding attribute in the BACnet Host object
Firmware_Revision	N/A	R		Firmware revision of the Anybus CompactCom 40 BACnet/IP as a string. This property can be changed by setting the corresponding attribute in the BACnet Host object
Application_Software_Revision	N/A	R		
Protocol_Version	1	R		
Protocol_Revision	14	R		
Protocol_Service_Supported	1001 0100 0000 1011 1100 1000 0010 0000 1110 0001	R		Bit map stating what protocol services are supported in the object, see <a href="#">Supported BACnet Services, p. 20</a> .
Protocol_Object_Types_Supported	0010 0100 1000 0001 0001 0000	R		Bit map stating what object types are supported in the object (analog-value, binary-value, multi-state value, device, and notification-class objects).
Object_List		R		This list is filled based on the ADIs that are implemented in the application.
Max_APDU_Length_Accepted	1024	R		
Segmentation_Supported	SUPPORTED_BOTH	R		Segmentation supported for both Rx and Tx APDU.

Property Identifier	Value	R/W	NV	Description/Comment
Max_Segments_Accepted	5	R		Supporting 5 segments Number of maximum length APDUs that can be received in a segmented message. This is the maximum APDU payload in a request after segmentation.
Local_Time	N/A	R		The local time is synchronized from the application at power up or set from the BACnet network via the TimeSynchronization service.
Local_Date	N/A	R		The local date is synchronized from the application at power up or set from the BACnet network via the TimeSynchronization service.
APDU_Timeout	10000 (default) Valid range: 0 - 65535	R/W	NV	APDU transaction timeout (ms). The value 0 is only valid if Number_Of_APDU_Retries = 0.
Number_Of_APDU_Retries	3 (default) Valid range: 0 - 255	R/W	NV	Number of APDU transaction and/or segment retransmission retries.
Device_Address_Binding	N/A	R		Managed by the APL layer and contains the list of all device address bindings of active client processes inside the Anybus CompactCom BACnet/IP module.
Database_Revision	N/A	R	NV	Incremented by 1 each time an object identifier is changed or the name of a BACnet object is changed.
APDU_Segment_Timeout	5000 (default) Valid range: 0 - 65535	R/W	NV	APDU segment timeout (ms). The value 0 is only valid if Number_Of_APDU_Retries = 0.
Active_COV_Subscriptions		R		Populated based on active COV subscription (max. 60) entries in AE module. Registered with SubscribeCOV service.
Property_List	See description	R		Array containing all properties supported by the device object, except Object_Name, Object_Type, Object_Identifier and Property_List: [ System_Status, Vendor_Name, Vendor_Identifier, Model_Name, Firmware_Revision, Application_Software_Version, Protocol_Version, Protocol_Revision, Protocol_Service_Supported, Protocol_Object_Types_Supported, Object_List, Max_APDU_Length_Accepted, Segmentation_Supported, Max_Segments_Accepted, Local_Time, Local_Date, APDU_Timeout, Number_Of_APDU_Retries, Device_Address_Binding, Database_Revision, APDU_Segment_Timeout, Active_COV_Subscriptions ]
Serial_Number	Serial number of the module assigned during production.	R		This property can be changed via the <a href="#">BACnet Host Object (EFh)</a> , p. 123.

### 3.3.2 Analog Value Object

The analog value object is mapped to ADIs of data types that represent analog values, e.g. UINT16.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- [Mapping of BACnet Objects to Anybus CompactCom, p. 23](#)
- [Communication Settings, p. 22](#)
- [Alarm/Event Functionality, p. 27](#)

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		In simple mode: Analog_Value_# (# = instance number). The property is not writable. In advanced mode: Corresponding ADI name. The property is writable if the Set_Attribute service, to update the Name attribute of the Application Data Instances, is supported. If the host application for any reason returns an error code when the ADI name is read, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Object_Type	ANALOG_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value converted to Real. If the host application for any reason returns an error code, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Status_Flags	F, F, F, F	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bits 1 - 3: not used. Set to FALSE.
Event_State	NORMAL (0)	R		Valid states: 0: NORMAL 3: HIGH_LIMIT 4: LOW_LIMIT
Out_Of_Service	FALSE	R		Always FALSE
Units	NO_UNITS	R		
COV_Increment	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value.
Time_Delay	0	R/W	NV	Time delay for an event to be triggered after occurrence (s) Min. value: 0 Max. value: UINT32max/1000
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
High_Limit	0	R/W	NV	Min. value: Min. value of the ADI's data type
Low_Limit	0	R/W	NV	Max. value: Max. value of the ADI's data type
Deadband	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value

Property Identifier	Value	R/W	NV	Description/Comment
Limit_Enable	F, F	R/W	NV	Bit string that determines what TO event limits are enabled Bit 0: LOW_LIMIT_ENABLE Bit 1: HIGH_LIMIT_ENABLE
Event_Enable	F, F, F	R/W	NV	Bit string that determines what TO events that are enabled Bit 0: TO-OFFNORMAL Bit 1: not used. Set to FALSE. Bit 2: TO-NORMAL
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT
Acked_Transitions	T, T, T	R		Bit string that determines what TO events has been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. Only available if the corresponding ADI is mapped on the write process data channel. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. <a href="#">See Setup of Alarm and Events, p. 28</a> for more details regarding this property.
Property_List	See description	R		Array containing all properties supported by the analog value object, except Object_Name, Object_Type, Object_Identifier and Property_List: <b>Corresponding ADI not mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service, Units ] <b>Corresponding ADI mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service, Units, COV_Increment, Time_Delay, Notification_Class, High_Limit, Low_Limit, Deadband, Limit_Enable, Event_Enable, Acked_Transitions, Notify_Type, Event_Time_Stamps, Event_Detection_Enable]

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present\_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet Real value and returned to the network. If an error is returned from the application, the BACnet device class error code OPERATIONAL\_PROBLEM is returned to the network.

When the Present\_Value property is written from the network, the BACnet Real value is converted to the data type of the corresponding ADI. For all data types, except FLOAT, all



decimal precision is lost. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL\_PROBLEM.

### 3.3.3 Binary Value Object

The binary value object is mapped to ADIs of data type BOOL.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- [Mapping of BACnet Objects to Anybus CompactCom, p. 23](#)
- [Communication Settings, p. 22](#)
- [Alarm/Event Functionality, p. 27](#)

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		In simple mode: Binary_Value_# (# = instance number). The property is not writable. In advanced mode: Corresponding ADI name. The property is writable if the Set_Attribute service, to update the Name attribute of the Application Data Instances, is supported. If read request for any reason returns a error code from the application, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Object_Type	BINARY_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value
Status_Flags	F, F, F, F	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bits 1 - 3: not used. Set to FALSE.
Event_State	NORMAL	R		Valid states: 0: NORMAL 2: OFF_NORMAL
Out_Of_Service	FALSE	R		Always FALSE
Time_Delay	0	R/W	NV	Time delay for an event to be triggered after occurrence (s)
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
Alarm_Value	INACTIVE (0)	R/W	NV	
Event_Enable	N/A	R/W	NV	Bit string that determines what TO event is enabled Bit 0: TO-OFFNORMAL Bit 1: not used. Set to FALSE. Bit 2: TO-NORMAL
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT

Property Identifier	Value	R/W	NV	Description/Comment
Acked_Transitions	T, T, T	R		Bit string that determines what TO events have been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. See <a href="#">Setup of Alarm and Events, p. 28</a> for more details regarding this property.
Property_List	See description	R		Array containing all properties supported by the binary value object, except Object_Name, Object_Type, Object_Identifier and Property_List: <b>Corresponding ADI not mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service ] <b>Corresponding ADI mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service, Time_Delay, Notification_Class, Alarm_Value, Event_Enable, Acked_Transitions, Notify_Type, Event_Time_Stamps, Event_Detection_Enable]

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present\_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet BinaryPV value and returned to the network. If an error is returned from the application, the BACnet device class error code OPERATIONAL\_PROBLEM is returned to the network.

When the Present\_Value property is written from the network, the BACnet BinaryPV value is converted to Bool. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL\_PROBLEM.

### 3.3.4 Multi-State Value Object

The multi-state value object is mapped to ADIs of data type ENUM.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- [Mapping of BACnet Objects to Anybus CompactCom, p. 23](#)
- [Communication Settings, p. 22](#)
- [Alarm/Event Functionality, p. 27](#)

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		In simple mode: Multistate_Value_# (# = instance number). The property is not writable. In advanced mode: Corresponding ADI name. The property is writable if the Set_Attribute service, to update the Name attribute of the Application Data Instances, is supported. If a read request returns an error code from the application, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Object_Type	MULTISTATE_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value
Status_Flags	F, F, F, F	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bit 1: FAULT Bits 2 - 3: not used. Set to FALSE.
Event_State	NORMAL	R		Valid states: 0: NORMAL 1: FAULT 2: OFF_NORMAL
Out_Of_Service	FALSE	R		Always FALSE
Number_Of_States	N/A	R		Corresponding ADI Max_Value + 1. If an error occurs when reading the Max_value, this property will be set to 256.
Time_Delay	0	R/W	NV	Time delay for an event to be triggered after occurrence (s)
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
Alarm_Values	Empty list	R/W	NV	
Fault_Values	Empty list	R/W	NV	
Event_Enable	N/A	R/W	NV	Bit string that determines what TO event is enabled Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT

Property Identifier	Value	R/W	NV	Description/Comment
Acked_Transitions	T, T, T	R		Bit string that determines the TO events that have been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. Only available if the corresponding ADI is mapped on the write process data channel. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. <a href="#">See Setup of Alarm and Events, p. 28</a> for more details regarding this property.
State_Text	N/A	R		This property is mapped against the module's Get_Enum_String service
Reliability	NO_FAULT_DETECTED	R		Object reliability Only available if the corresponding ADI is mapped on the write process data channel. 0: NO_FAULT_DETECTED 9: MULTI_STATE_FAULT
Property_List	See description	R		Array containing all properties supported by the multistate value object, except Object_Name, Object_Type, Object_Identifier and Property_List: <b>Corresponding ADI not mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service, Number_Of_States, State_Text, Reliability ] <b>Corresponding ADI mapped on write PD:</b> [ Present_Value, Status_Flags, Event_State, Out_Of_Service, Number_Of_States, Time_Delay, Notification_Class, Alarm_Values, Fault_Values, Event_Enable, Acked_Transitions, Notify_Type, Event_Time_Stamps, Event_Detection_Enable, State_Text, Reliability ]

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present\_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet Unsigned value, incremented by one and returned to the network. (The ENUM ADI is zero based in the Anybus CompactCom, but the first state of a multistate value object on BACnet is one.) If an error is returned from the application, a BACnet device class error code OPERATIONAL\_PROBLEM is returned to the network.

When the Present\_Value property is written from the network, the BACnet Unsigned value is decremented by one. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL\_PROBLEM.

The State\_Text property is linked to the Get\_Enum\_String service of the ADI. The property is an array property, where array index 0 returns the number of states (see property Number\_Of\_States). All other indices return the corresponding state text. Multistate states begin at 1, so the value is decremented by 1 and a Get\_Enum\_String command is sent to the corresponding ADI. If an error is returned from the application, a BACnet device class error code OPERATIONAL\_PROBLEM is returned to the network.

### 3.3.5 Notification Class Object

An Anybus CompactCom 40 BACnet/IP always contain 6 (0 - 5) instances of this object. Each instance contains a list of devices that need to be informed about certain events and alarms.

The default values of properties, stored in the non-volatile memory, are assigned by the Anybus CompactCom module the first time the module is started. When a BACnet user requests to write any of these properties the data is saved to non volatile memory directly after the validation of the service write request.

See also...

- [COV Notifications, Alarms and Events, p. 27](#)

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	Default: Notification_Class_# (# = instance number)	RW	NV	BACnet Char string (only ANSI34 supported). Max. 30 characters
Object_Type	NOTIFICATION_CLASS	R		
Notification_Class	N/A	R		Equal to instance number
Priority	Default: {3, 0, 0, 0} Values 0 - 255 are allowed, a lower value has higher priority than a higher value.	Index 0: R Index 1-3: R/W	NV	BACnet array of Unsigned values: 0: Number of Elements 1: TO-OFFNORMAL 2: TO-FAULT 3: TO-NORMAL
Ack_Required	Default: {0, 0, 0} (0 = not set)	R/W	NV	BACnet Bit string of 3 bits: Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Recipient_List	N/A	R/W	NV	BACnet list of BACnetDestination primitives. There are 18 recipient list entries available to be configured among the 6 Notification Class instances.
Property_List	See description	R		Array containing all properties supported by the notification class object, except Object_Name, Object_Type, Object_Identifier and Property_List: [ Notification_Class, Priority, Ack_Required, Recipient_List, Recipient_Error_Field ]
Vendor property 600: Recipient_Error_Field	Default: {0, 0, 0, ...} (0 = not set)	R		BACnet Bit string of a maximum of 60 bits. The string has one bit per element present in the Recipient_List. If a bit is set an event notification error is present for the corresponding recipient entry in the Recipient_List property.

The Recipient\_List specifies one or more recipients that a notification event should be sent to. In total there are 18 list entries available to be divided among the 6 Notification Class object instances. These are managed on a “pool” basis, and when all have been assigned to the Notification object instances, an error is issued (CLASS\_SERVICES, CODE\_NO\_SPACE\_TO\_ADD\_LIST\_ELEMENT).

### 3.3.6 Supported BACnet Services

The following services are supported by the Anybus CompactCom 40 BACnet/IP

Service	Description
ReadProperty	Reads a single property from a BACnet object
ReadPropertyMultiple	Reads multiple properties from a BACnet object
WriteProperty	Writes a single property to a BACnet object
WritePropertyMultiple	Writes multiple properties to a BACnet object
Who-Is	Upon receipt of a Who-Is request the module will return an I-Am response containing its BACnet device object instance number. The service is used to find out which devices are present on the network, and their addresses.
Who-Has	This service is used to find out what devices in a network are implementing a specific object. The service uses either the object identifier or the object name for lookup. Upon a receipt of a Who-Has request the module will return an I-Have response, if it implements the requested object.
I-Am	This service is sent by the module in response to a matching Who-Is request. It is used to locate BACnet devices based on the device object instance number.
DeviceCommunication-Control	Turns off the BACnet communication of the module. Once communication is disabled the module will only respond to DeviceCommunicationControl requests and ReinitializeDevice messages. If only initiation instead of all communication is disabled, the module will also respond to Who-Is requests.
ReinitializeDevice	Resets devices over BACnet. A cold start results in a hardware reset. A warm start results in a restart of the BACnet stack and data mapping will not be affected.
TimeSynchronization	Synchronizes the current time and date of the module.
SubscribeCOV	Subscribes for changes of value with a BACnet object. The ADI corresponding to the object has to be mapped on write process data.
GetAlarmSummary	Responds with a list containing information about all active alarms.
GetEventSummary	Responds with a list containing information about all active alarms and events.
AcknowledgeAlarm	Acknowledges an active alarm.

### 3.3.7 BACnet Error Codes

#### Device Class Error Codes

Error Code	Indicates
OPERATIONAL_PROBLEM	The application has generated a not expected response, e.g. reading the value attribute of an ADI failed when performing ReadProperty on a value object.

#### Property Class Error Codes

Error Code	Indicates
WRITE_ACCESS_DENIED	Trying to write a non-writable property.
READ_ACCESS_DENIED	Trying to read a non-readable property.
PROPERTY_IS_NOT_AN_ARRAY	Trying to read or write an array index of a property that is not an array.
INVALID_DATA_TYPE	Trying to write a property using the wrong data type.
VALUE_OUT_OF_RANGE	The value written to a property is either too large or too small.
UNKNOWN_PROPERTY	Trying to read a non-existent property.
INVALID_ARRAY_INDEX	Trying to read or write an array index that does not exist for the property.
CHARACTER_SET_NOT_SUPPORTED	Writing a char string property with a character set not supported by the Anybus CompactCom module.
DUPLICATE_NAME	The value written to an Object_Name property already exists as an object name for another object.
OTHER	The Anybus CompactCom module encountered a general error.

**Service Class Error Codes**

Error Code	Indicates
INCONSISTENT_PARAMETERS	Something went wrong while parsing the data in a WriteProperty request.
INVALID_EVENT_STATE	The event state supplied in an AcknowledgeAlarm request is not correct.
INVALID_TIME_STAMP	The time stamp supplied in an AcknowledgeAlarm request is not correct.
SERVICE_REQUEST_DENIED	Generic error occurred for an AcknowledgeAlarm request
CHARACTER_SET_NOT_SUPPORTED	Wrong character set used for the password supplied with a ReinitializeDevice or DeviceCommunicationControl request.
COMMUNICATION_DISABLED	Received an invalid request for the current DeviceCommunicationControl state.
COV_SUBSCRIPTION_FAILED	a SubscribeCOV request could not be completed due to lack of resources or it is not possible to set up the object for COV notification due to properties not being mapped on process-write data.

**Object Class Error Codes**

Error Code	Indicates
UNKNOWN_OBJECT	The requested object does not exist in the Anybus CompactCom module.
NO_ALARM_CONFIGURED	There is no alarm or event to acknowledge for the event state supplied with the AcknowledgeAlarm request.
OTHER	Generic error during object request
ABORT_BUFFER_OVERFLOW	Response APDU is too large.
UNKNOWN_OBJECT	The requested object does not exist in the Anybus CompactCom module.
OTHER	The Anybus CompactCom module encountered a general error.

**Security Class Error Codes**

Error Code	Indicates
PASSWORD_FAILURE	Wrong password for ReinitializeDevice or DeviceCommunicationControl service.

## 3.4 Communication Settings

As with other Anybus-CompactCom products, network related communication settings are grouped in the Network Configuration Object (04h).

In this case, this includes...

<b>UDP</b>	All data to and from the module is transported via UDP
<b>TCP/IP settings</b>	<p>These settings must be set properly in order for the module to be able to participate on the network.</p> <p>The module supports DHCP, which may be used to retrieve the TCP/IP settings from a DHCP-server automatically. DHCP is enabled by default, but can be disabled if necessary.</p>
<b>Physical Link Settings</b>	By default, the module uses auto-negotiation to establish the physical link settings, however it is possible to force a specific setting if necessary.

The parameters in the Network Configuration Object (04h) are available from the network through the built-in web server.

See also...

- [Web Server, p. 34](#)
- [Network Configuration Object \(04h\), p. 87](#)
- [Secure HICP \(Secure Host IP Configuration Protocol\), p. 136](#)

## 3.5 Diagnostics

Major unrecoverable faults are reported in the Diagnostic Object

See also...

- [Diagnostic Object \(02h\), p. 84](#)

## 3.6 Network Data Exchange

### 3.6.1 Application Data (ADIs)

ADIs are represented through the BACnet objects. The properties of the BACnet objects are mapped to instances in the Application Data Object on the host application side.

There are two mapping schemes, one simple and one advanced. The application decides what scheme to use by implementing an attribute (#7, Support advanced mapping) in the BACnet Host Object. Accessible range of ADIs for the simple mapping scheme is 1 to 256. In the advanced mapping scheme, there are 256 analog value objects, 256 binary value objects and 256 multi-state value objects that can be mapped to ADIs in any order. It is recommended to use advanced mapping to fully take advantage of the functionality and flexibility of the module.

See also...

- [Mapping of BACnet Objects to Anybus CompactCom, p. 23](#)
- [BACnet Host Object \(EFh\), p. 123](#)



### 3.6.2 Translation of Data Types

The Anybus data types are translated to BACnet standard and vice versa as follows:

Anybus Data Type	BACnet Object Name	Comments
BOOL	Binary Value Object	Each ADI element of this type occupies one byte.
ENUM	Multi-state Value Object	
SINT8	Analog Value Object	
UINT8	Analog Value Object	
SINT16	Analog Value Object	Each ADI element of this type occupies two bytes.
UINT16	Analog Value Object	
SINT32	Analog Value Object	Each ADI element of this type occupies four bytes.
UINT32	Analog Value Object	
FLOAT	Analog Value Object	
CHAR	Analog Value Object	Each ADI element of this type occupies one byte.
PADx	-	
		Explicit padding of process data.

If an ADI of a size that doesn't match the size of the Present Value property in a Value Object, is mapped to a Value object, it will result in FAULT or in OPERATIONAL\_PROBLEM when accessed.

Anybus CompactCom data types BIT1 and BOOL1 are not implemented.

### 3.6.3 Mapping of BACnet Objects to Anybus CompactCom

#### Simple Mapping

In the simple mapping schema, that is implemented by default, the module will scan the first 256 implemented instances of the application data object during initialization. It will read the data type attribute and based on the data type BACnet objects will be created in a sequential order, starting with 0. Anybus CompactCom data types will be mapped to BACnet objects according to the table above. This mapping will stay fixed. For an example see the figure below:

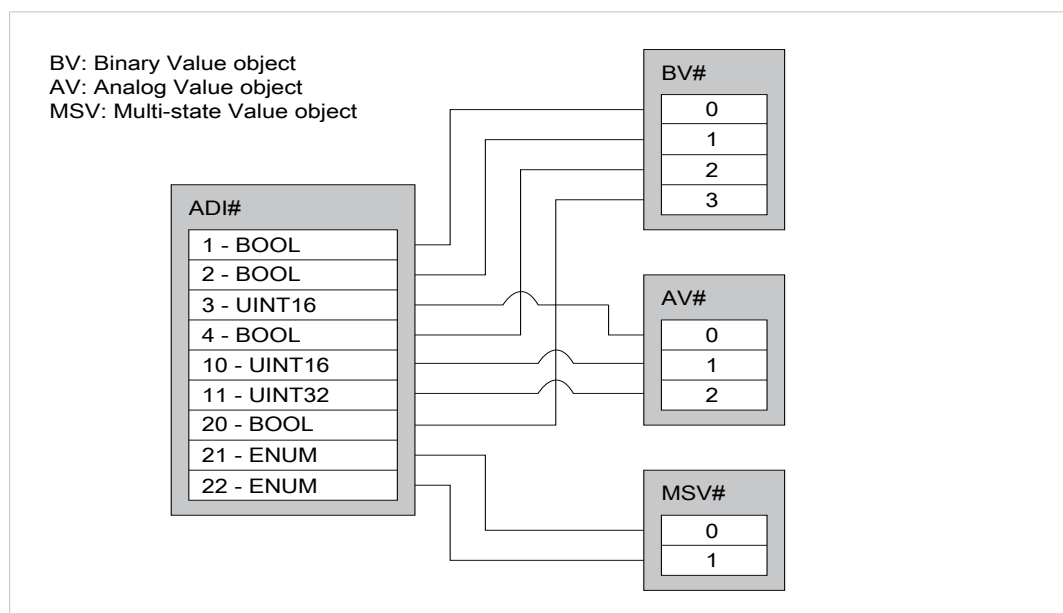


Fig. 1

### Advanced Mapping

If the attribute “Advanced mapping supported” in the BACnet Host object is true, the user can create almost any type of mapping of BACnet objects to Anybus CompactCom ADIs. This mapping can be designed to closely match the application and it is up to the application to keep track of the ADI - BACnet object relationships. The services of the BACnet host object are then used to implement this mapping on BACnet. Valid range of object identifier number is 0 - 2039 for each value object type. For an example see the figure below:

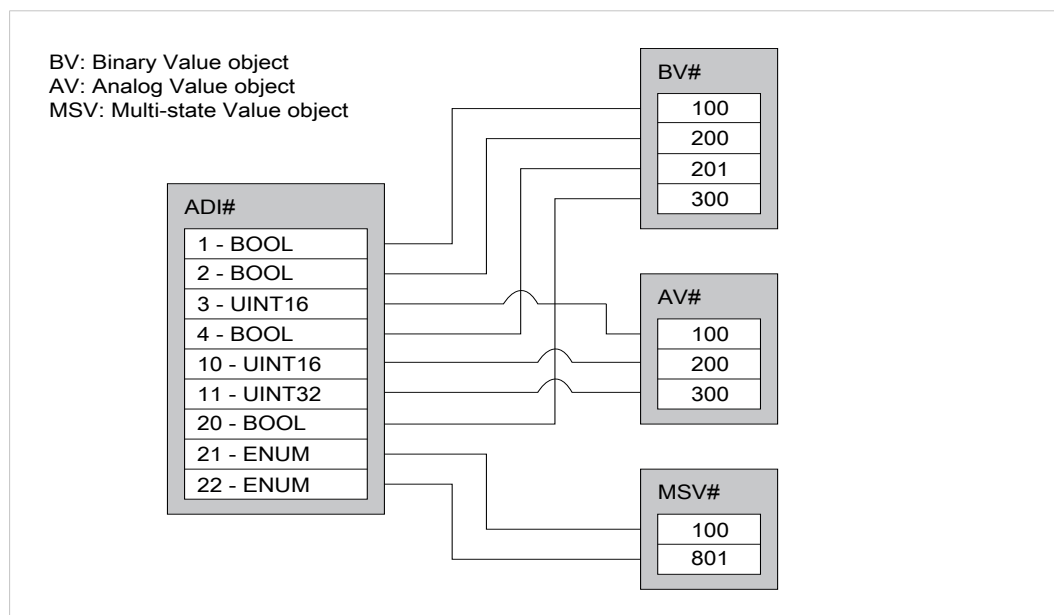


Fig. 2

### 3.6.4 Process Data

The Anybus CompactCom 40 BACnet/IP supports COV (Change Of Value) notification and Alarm/Event functionality, see [COV Notifications, Alarms and Events, p. 27](#). To be able to use these features for a BACnet object, the corresponding ADI must be mapped on write process data. If not, the module has no way of detecting any changes in the value of the ADI. Also, if an ADI is mapped on write process data, the properties used for COV notifications and Alarms/Events will be accessible to the corresponding BACnet object. If the mapping is changed, all COV and Alarm/Event information, that has been stored in non volatile memory, will be cleared.

Up to 64 ADIs can be mapped to write process data. It is possible to map multiple elements of the PADx data type.

No read process data is passed from the module to the application.

## 3.7 File System

### 3.7.1 Overview

The Anybus CompactCom 40 BACnet/IP has a built-in file system, that can be accessed from the application and from the network. Three directories are predefined:

<b>VFS</b>	The virtual file system that e.g. holds the web pages of the module.
<b>Application</b>	This directory provides access to the application file system through the Application File System Interface Object (EAh) (optional).
<b>Firmware</b>	This directory provides access to the application file system through the Application File System Interface Object (EAh) (optional).



*In the firmware folder, it is not possible to use append mode when writing a file. Be sure to use write mode only.*

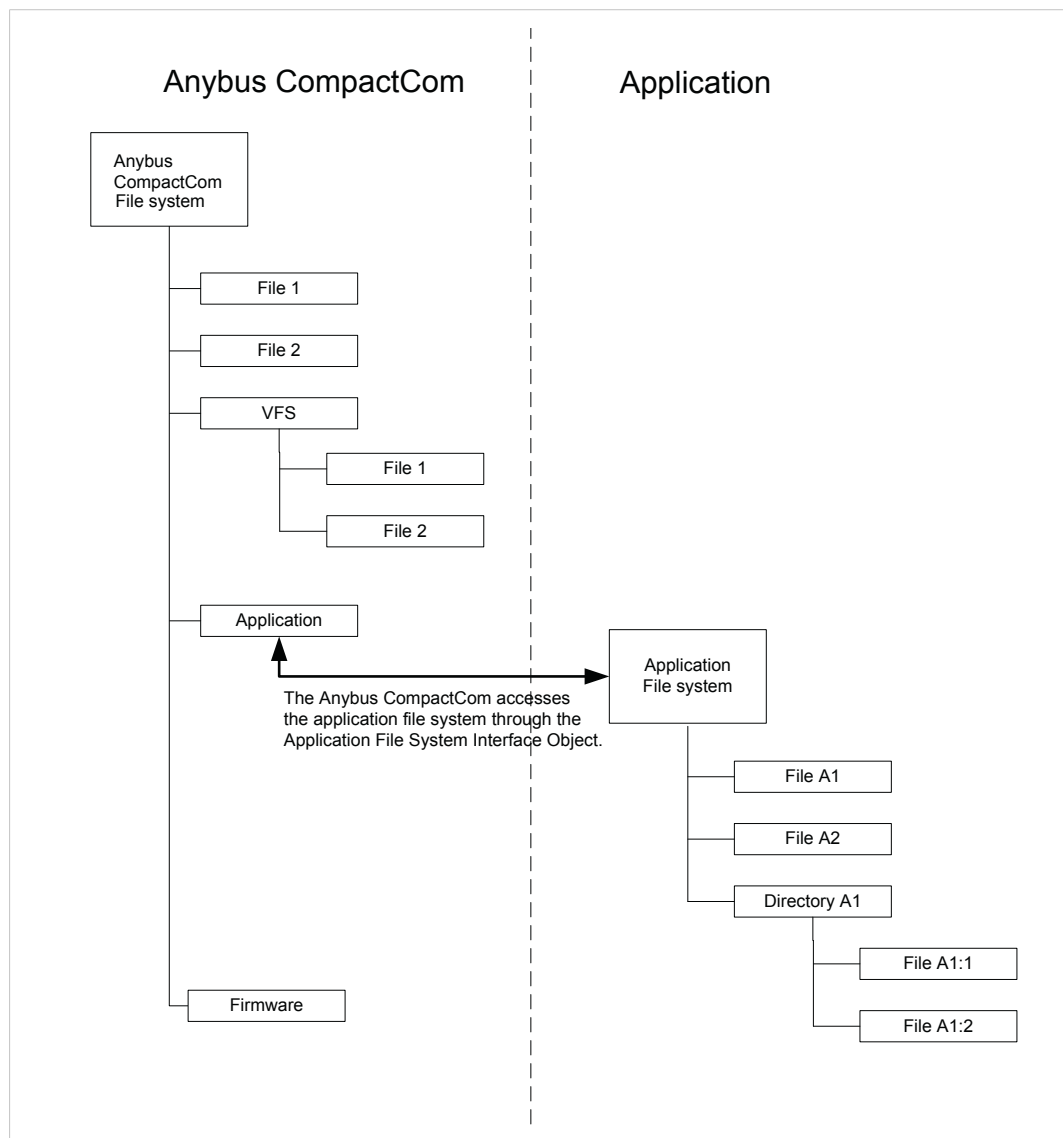


Fig. 3

### 3.7.2 General Information

The built-in file system hosts 28 Mb of nonvolatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application (through the Anybus File System Interface Object (0Ah).

Maximum number of directories and files that can be stored in the root directory is 511, if only short filenames are used (8 bytes name + 3 bytes extension). If longer filenames are used, less than 511 directories/files can be stored. This limitation does not apply to other directories in the file system.

The file system uses the following conventions:

- \ (backslash) is used as a path separator
- Names may contain spaces, but must not begin or end with one.
- Valid characters in names are ASCII character numbers less than 127, excluding the following characters: \ / : \* ? " < > |
- Names cannot be longer than 48 characters
- A path cannot be longer than 225 characters (filename included)

See also...

- [FTP Server, p. 32](#)
- [Web Server, p. 34](#)
- [E-mail Client, p. 41](#)
- [Server Side Include \(SSI\), p. 42](#)
- [File System Interface Object \(0Ah\), p. 120](#)



The file system is located in flash memory. Due to technical reasons, each flash segment can be erased approximately 100000 times before failure, making it unsuitable for random access storage.

The following operations will erase one or more flash segments:

Deleting, moving or renaming a file or directory

Writing or appending data to an existing file

Formatting the file system

### 3.7.3 System Files

The file system contains a set of files used for system configuration. These files, known as “system files”, are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). The format of these files are, with some exceptions, based on the concept of keys, where each keys can be assigned a value, see below.

#### Example 1:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

## 4 COV Notifications, Alarms and Events

### 4.1 General

The Anybus CompactCom 40 BACnet/IP supports COV (Change Of Value) notification and Alarm/Event functionality. These features can only be used for a BACnet object if the corresponding ADI is mapped on write process data.



*If the mapping of the ADIs is changed, all COV and Alarm/Event information, that has been stored in non volatile memory, will be cleared.*

### 4.2 COV (Change of Value) Notifications

A BACnet device can subscribe for a COV (Change of value) notification from another BACnet device on the network. The first device uses the service SubscribeCOV to send the identifier of the desired object to the second device to activate the notification. A change in the Present\_value property will then trigger a COV notification to the first BACnet device.

Each Value object is mapped to an ADI in the Anybus-CompactCom module, and the property Present\_value corresponds to the Value attribute in the ADI. If the ADI is not mapped on the write process data area any change in the value will go unnoticed and a subscribeCOV service will return the error code COV\_SUBSCRIPTION\_FAILED.

In Binary Value and Multi-State Value objects any change in the present value will cause a notification to be sent. In Analog Value objects a property, COV\_increment, is used to decide how much a present value needs to change in order for a notification to be sent.

### 4.3 Alarm/Event Functionality

A change in the present value of an object can also be used to trigger an alarm or an event.

Each value object is associated to a Notification Class object instance. Each instance contains a list of all recipients of a specific Alarm/Event notification. Alarms and events are handled in the same way by the value objects. If an alarm or an event is to be issued is decided by the application in the property Notify\_Type in the value object and the choice indicates the severity of the notification to the recipient.

Alarms/events are issued every time a value object changes state. The reasons for changing states are specific to each of the three value objects as described below. The possibility to issue an alarm or an event has to be enabled in each object. It is also possible to delay an alarm/event using the Time\_Delay property of the object. As previously mentioned this functionality is only available if the ADIs are mapped to write process data.

#### 4.3.1 Analog Value Object Alarm/Event Functionality

An Analog Value object has three possible states: NORMAL, LOW\_LIMIT and HIGH\_LIMIT. The LOW\_LIMIT state is entered when the present value, object property Present\_value, falls below the value in the object property Low\_limit. The HIGH\_LIMIT state is entered when the present value rises above the value in the property High\_limit. When the present value returns above or below the given limit, the object returns to the NORMAL state. The OFF\_NORMAL bit in the property Event\_enable must be set, to allow transitions to other states than the NORMAL state. The LOW\_LIMIT and HIGH\_LIMIT bits in the property Limit\_Enable must be set to allow transitions to the corresponding state.

A change of state will result in either a TO-NORMAL event or a TO-OFFNORMAL event issued to the Notification Class object instance. A TO-OFFNORMAL event indicates that the value of the

property value has increased above the defined high limit or decreased below the defined low limit. A TO-NORMAL event indicates a return to the NORMAL state.

### 4.3.2 Binary Value Object Alarm/Event Functionality

A Binary Value Object has two states, NORMAL and OFF\_NORMAL. An alarm value can be set and when the property Present\_Value is changed to this value, the object changes from the NORMAL state to the OFF\_NORMAL state. The corresponding bit(s) must be set in the property Event\_Enable for this functionality to be available.

A change of state will result in either a TO-NORMAL event or a TO-OFFNORMAL event issued to the Notification Class object instance. A TO-OFFNORMAL event indicates that the value of the property value has changed to equal the alarm value. A TO-NORMAL event indicates a return to the NORMAL state.

### 4.3.3 Multi-State Value Object Alarm/Event Functionality

The Multi-State Value object contains one list for alarm values and one for fault values. When the property Present\_Value is changed to a value that is present in the alarm value list, the object enters the OFF\_NORMAL state and a TO-OFFNORMAL event is issued. If the value is present in the fault values list, the object enters the FAULT state and a TO-FAULT event is issued. A TO-NORMAL event is issued when the object returns to the NORMAL state from any of the other two states. The corresponding bit(s) must be set in the property Event\_Enable for the functionality to be available.

### 4.3.4 Summary of States and Events for the Value Objects

Object	State	Possible events
Analog Value Object	NORMAL	TO-OFFNORMAL
	HIGH_LIMIT	TO-NORMAL
	LOW_LIMIT	
Binary Value Object	NORMAL	TO-OFFNORMAL
	OFF_NORMAL	TO-NORMAL
Multi State Value Object	NORMAL	TO-FAULT TO-OFFNORMAL
	FAULT	TO-NORMAL TO-OFFNORMAL
	OFF_NORMAL	TO-NORMAL TO-FAULT

## 4.4 Setup of Alarm and Events

This section gives a short guide to what properties in the BACnet objects need to be setup and monitored when implementing the BACnet alarm/event functionality.

### 4.4.1 Notification Class Object

BACnet object notifications are classified as either TO-OFFNORMAL, TO-FAULT, or TO-NORMAL. Any event triggered by a BACnet value object is interpreted as one of these three.

See also...

- [Notification Class Object, p. 19](#)

### Priority

This property defines how to prioritize the three different kinds of notifications/events that can be issued by an object. A lower value has priority over a higher value, e.g. setting Priority arrayidx 2 to 1 and the other entries in the array to 5 and 76 will give TO-FAULT priority over the other notifications.

If all three events are given the same priority in the property, they will receive the following priority by default:

1. TO-NORMAL
2. TO-FAULT
3. TO-OFFNORMAL

I.e. TO-NORMAL has the highest priority.

### Ack Required

This property specifies if an acknowledgement is required from the recipients of a specific TO event. Every TO event sent is flagged to tell the recipient whether an acknowledgement is required or not, based on the corresponding TO bit in the bit string in this property.

### Recipient List

This list specifies the BACnet recipients that shall receive a notification of an alarm/event associated with the notification class. The list is a sequence of BACnetDestination primitives, where each entry consists of the following elements:

Element	Description
Transitions	Bit string that determines what TO events shall be sent to this recipient.
IssueConfirmedNotifications	Boolean that specifies if a ConfirmedEventNotification or UnConfirmedEventNotification request shall be sent to the recipient.
ProcessIdentifier	Unsigned32 value that specifies the process identifier that is linked to the notification event. If an acknowledgement is required for the specific event (see Ack_Required) the acknowledgement source has to have a valid value of a recipient process identifier to successfully acknowledge the alarm/event.
Recipient	BACnetRecipient primitive that can either be a DeviceInstanceNumber or a BACnetMAC address. The primitive will be added to the DAB (Device Address Binding) lookup mechanism of the BACnet APL layer and who-is/I-am BACnet UnConfirmedServices will be used to probe for the recipient device on BACnet. When a valid lookup is found the event notifications will be sent to that device. If no valid probe is found the corresponding bit in VendorProperty: 600, BACnetRecipientErrorField will be set in the notification class.
ToTime	BACnetTime primitive that specifies the ending time for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.
FromTime	BACnetTime primitive that specifies the starting time for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.
ValidDays	BACnetDaysOfWeek that specifies the valid days of the week for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.

### Vendor Property: 600, Recipient\_Error\_Field

This vendor specific property, added by HMS Industrial Networks, holds a bit string that is used for diagnostics to determine if a recipient has notification errors present or not. Bit 0 in the bit string represents recipient 1 in the RecipientList property, bit 1 represents recipient 2, and so on.

If the bit for a recipient is set, one of the following error events are present:

- The recipient device was not found on the BACnet network.

- A ConfirmedEventNotification request sent to the recipient returns a negative acknowledge response or an internal timeout.

If a bit has been set, as a result of an error, the bit will be cleared when a successful notification event has been sent or when the Recipient\_List property has been updated with a new list of recipients.

#### 4.4.2 Analog Value Object

Available TO events are TO-NORMAL and TO-OFFNORMAL. The object has three states, NORMAL, LOW\_LIMIT and HIGH\_LIMIT, but the TO events only reflect transitions to and from a normal state to a not normal state.

##### Object Properties to Setup

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Limit Enable	Bit string that enables/disables HIGH_LIMIT and LOW_LIMIT guarding.
High Limit	BACnet real value (float) that specifies the upper limit of the normal value span where a TO_OFFNORMAL event will be issued.
Low Limit	BACnet real value (float) that specifies the lower limit of the normal value span where a TO_OFFNORMAL event will be issued.
Deadband	BACnet real value (float) that specifies the deadband for the high and low limits
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.

##### Object Properties to Monitor

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the 3 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set.

#### 4.4.3 Binary Value Object

Available TO events are TO-NORMAL and TO-OFFNORMAL.

##### Object Properties to Setup

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Alarm Value	BACnet BinaryPV enumeration that specifies when Present_Value is OFF_NORMAL, either 0 (inactive) or 1 (active).
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.



**Object Properties to Monitor**

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the 2 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set.

**4.4.4 Multi-state Value Object**

Available TO events are TO-NORMAL, TO-OFFNORMAL, and TO-FAULT.

**Object Properties to Setup**

If the same value is specified for the Alarm Values property as for the Fault Values property, both a TO\_OFFNORMAL and a TO\_FAULT will be triggered.

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Alarm Values	BACnet list of unsigned values that will trigger a TO-OFFNORMAL event.
Fault Values	BACnet list of unsigned values that will trigger a TO-FAULT event.
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.



*Any valid value, that is not listed in the Alarm Values and Fault Values properties, can generate a TO-NORMAL event.*

**Object Properties to Monitor:**

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the 3 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set. If a TO-FAULT event is active the FAULT bit will be set.
Reliability	BACnet enumeration that is set to 9 (multi_state_fault) if a TO-FAULT event is active and 0 (no_fault_detected) otherwise.

## 5 FTP Server

### 5.1 General Information

The built-in FTP-server makes it easy to manage the file system using a standard FTP client. It can be disabled using attribute #6 in the Ethernet Host Object (F9h).

By default, the following port numbers are used for FTP communication:

- TCP, port 20 (FTP data port)
- TCP, port 21 (FTP command port)

The FTP server supports up to two concurrent clients.

### 5.2 User Accounts

User accounts are stored in the configuration file \ftp.cfg. This file holds the usernames, passwords, and home directory for all users. Users are not able to access files outside of their home directory.

File Format:

```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
User3:Password3:Homedirectory3
```

Optionally, the UserN:PasswordN-section can be replaced by a path to a file containing a list of users as follows:

File Format (\ftp.cfg):

```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
.
.
UserN:PasswordN:HomedirectoryN
\path\userlistA:HomedirectoryA
\path\userlistB:HomedirectoryB
```

The files containing the user lists shall have the following format:

File Format:

```
User1:Password1
User2:Password2
User3:Password3
.
.
UserN:PasswordN
```

Notes:

- Usernames must not exceed 16 characters in length.
- Passwords must not exceed 16 characters in length.
- Usernames and passwords must only contain alphanumeric characters.
- If \ftp.cfg is missing or cannot be interpreted, all username/password combinations will be accepted and the home directory will be the FTP root (i.e. \ftp\).

- The home directory for a user must also exist in the file system, if the user shall be able to log in. It is not enough just to add the user information to the ftp.cfg file.
- If Admin Mode has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i. e. the home directory will be the system root). The vfs folder is read-only.
- It is strongly recommended to have at least one user with root access (\) permission. If not, Admin Mode must be enabled each time a system file needs to be altered (including \ftp.cfg).

### 5.3 Session Example

The Windows Explorer features a built-in FTP client which can easily be used to access the file system as follows:

1. Open the Windows Explorer.
2. In the address field, type FTP://<user>:<password>@<address>
  - - Substitute <address> with the IP address of the Anybus module
  - - Substitute <user> with the username
  - - Substitute <password> with the password
3. Press **Enter**. The Explorer will now attempt to connect to the Anybus module using the specified settings. If successful, the file system will be displayed in the Explorer window.

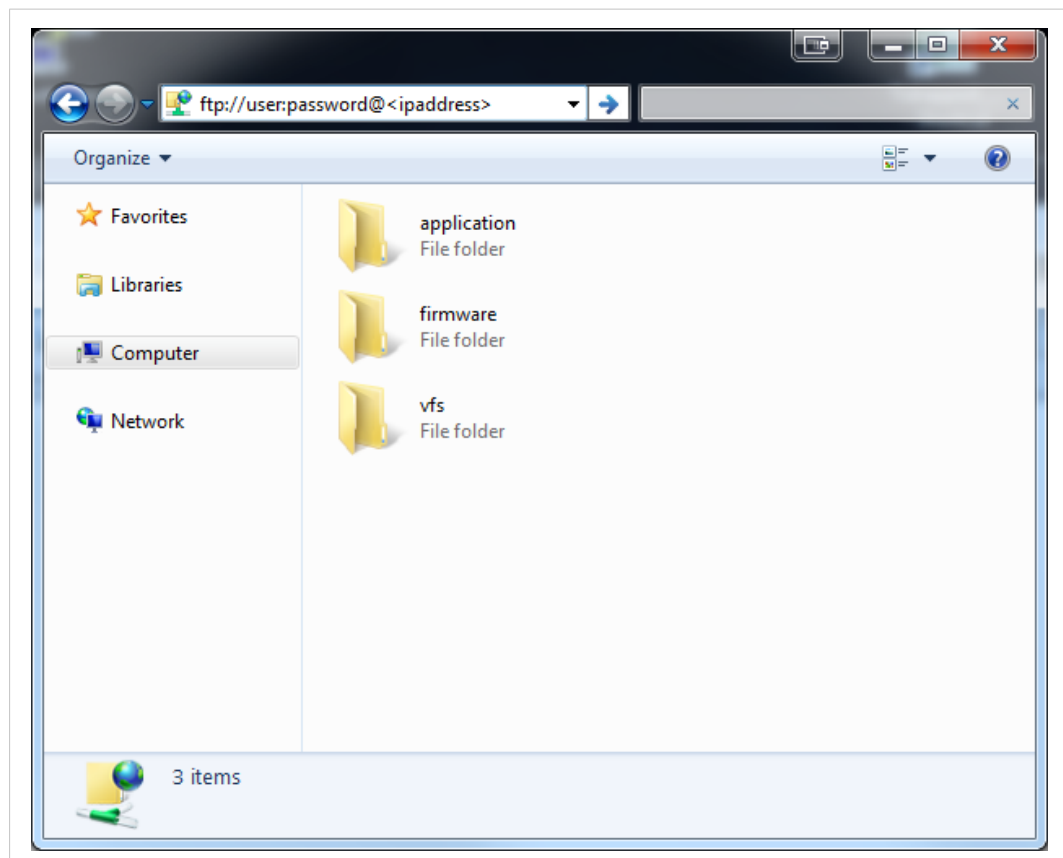


Fig. 4

## 6 Web Server

### 6.1 General Information

The built-in web server provides a flexible environment for end-user interaction and configuration purposes. JSON, SSI and client-side scripting allow access to objects and file system data, enabling the creation of advanced graphical user interfaces.

The web interfaces are stored in the file system, which can be accessed through the FTP server. If necessary, the web server can be completely disabled in the Ethernet Host Object (F9h).

The web server supports up to 20 concurrent connections and communicates through port 80.

See also...

- [FTP Server, p. 32](#)
- [Server Side Include \(SSI\), p. 42](#)
- [JSON, p. 62](#)
- [Ethernet Host Object \(F9h\), p. 129](#)

### 6.2 Default Web Pages

The default web pages provide access to:

- Network configuration parameters
- Network status information
- Access to the host application ADIs

The default web pages are built of files stored in a virtual file system accessible through the vfs folder. These files are read only and cannot be deleted or overwritten. The web server will first look for a file in the web root folder. If not found it will look for the file in the vfs folder, making it appear as the files are located in the web root folder. By loading files in the web root folder with exactly the same names as the default files in the vfs folder, it is possible to customize the web pages, replacing such as pictures, logos and style sheets.

If a complete customized web system is designed and no files in the vfs folder are to be used, it is recommended to turn off the virtual file system completely, see the File System Interface Object.

See also...

- [File System, p. 25](#)
- [File System Interface Object \(0Ah\), p. 120](#)

### 6.2.1 Network Configuration

The network configuration page provides an interface for changing TCP/IP and SMTP settings in the Network Configuration Object.

## Anybus-CompactCom BACnet/IP

### Network configuration

#### IP Configuration

IP address:	<input type="text" value="10.11.8.243"/>
Subnet mask:	<input type="text" value="255.255.0.0"/>
Gateway:	<input type="text" value="10.11.0.1"/>
Host name:	<input type="text"/>
Domain name:	<input type="text" value="hms.se"/>
DNS1:	<input type="text" value="10.10.20.6"/>
DNS2:	<input type="text" value="10.10.12.12"/>
DHCP:	<input checked="" type="checkbox"/>

Store settings

#### SMTP Settings

SMTP Server:	<input type="text" value="mail.hms.se"/>
SMTP User:	<input type="text" value="anybus.test"/>
SMTP Pswd:	<input type="password" value="••••••••"/>

Store settings

#### Ethernet Configuration

Comm 1:	<input type="text" value="Auto"/>
Comm 2:	<input type="text" value="Auto"/>

Store settings

[► Main](#) [► Network interface](#)

Fig. 5

Available editable settings are explained in the following sections..

**IP Configuration**

Name	Description
IP address	The TCP/IP settings of the module
Subnet mask	Default values: 0.0.0.0Value ranges: 0.0.0.0 - 255.255.255.255
Gateway	The module needs a reset for the changes to take effect
Host name	IP address or name Max 64 characters Changes will take effect immediately
Domain name	IP address or name Max 48 characters Changes will take effect immediately
DNS 1	Primary and secondary DNS server, used to resolve host name
DNS 2	Default values: 0.0.0.0Value ranges: 0.0.0.0 - 255.255.255.255 Changes will take effect immediately
DHCP	Checkbox for enabling or disabling DHCP Default value: enabled The module needs a reset for the changes to take effect

**SMTP Settings**

Changes to these settings will take effect immediately

Name	Description
SMTP Server	IP address or name Max 64 characters
SMTP User	Max 64 characters
SMTP Pswd	Max 64 characters

**Ethernet Configuration**

Changes to these settings will take effect immediately.

Name	Description
Comm 1	Ethernet speed/duplex settings
Comm 2	Default value: auto

**6.2.2 Ethernet Statistics Page**

The Ethernet statistics web page contains the following information:

Ethernet Link		Description
Port 1	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
Port 2	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
BACnet/IP Statistics		Description
SvrUnConfRxReqRecv		Number of UnConfirmed server requests received
SvrUnConfTxReqSent		Number of UnConfirmed server request sent
ICltUnConfTxReqSent		Number of UnConfirmed client request sent
BACnet APL (application layer) Statistics		Description
		(Available for both client and server transactions.=
TrActive		Active server transactions
TrActiveMax		Max active server transactions
TrTxSegSent		Number of tx segments sent

BACnet/IP Statistics	Description
TrTxSegAckRecv	Number of tx ack received
TrTxSegNegAckRecv	Number of negative tx ack received
TrRxSegRecv	Number of rx segments received
TrRxSegAckSent	Number of rx segments ack sent
TrRxSegDupAckSent	Number of rx segments dup ack sent
TrRxSegNegAckSent	Number of rx segments neg ack sent
TrRxAPDURecv	Number of Confirmed trans received
TrTxAPDUSent	Number of Confirmed trans sent
TrTxSegTimeout	Number of tx segment timeouts
TrRxSegTimeout	Number of rx segment timeouts
TrImpDelete	Number of implicit deletes
TrTxTmoDelete	Number of tx timeout deletes
TrRxTmoDelete	Number of rx timeout deletes
TrRxAbortsRecv	Number of rx aborts received
TrTxAbortsRecv	Number of tx aborts received
TrAbortsSent	Number of transaction aborts sent
TrRejectsSent	Number of transaction rejects sent
TrErrorsSent	Number of transaction errors sent
BACnet AE (Alarm and Event) Module statistics.	Description
iActiveCOVEntry	Number of active COV subscriptions
iMaxActiveCOVEntry	The maximum number of active COV subscriptions ever present since startup.
ICOVLifeEntryRefCnt	Number of COV subscriptions that have a lifetime enabled. When the lifetime elapses they will be automatically deleted,
ICOVTrAllocResumes	APL client transactions (confirmed requests) resource allocation resumes. A needed resource was unavailable and a resume of the COV update process was scheduled.
ICOVNwMsgAllocResumes	NWL client message resource allocation resumes. A needed resource was unavailable and a resume of the COV update process was scheduled.
ICOVConfNotifys	Number of confirmed COV notifications sent.
iCOVUnConfNotifys	Number of unconfirmed COV notifications sent
iCOVConfNotifyErrors	Number of confirmed COV notification errors. A negative acknowledgement or/and in timeout was returned to a confirmed COV to notify request.
iActiveAEEvents	Number of currently active AE module events
iActiveNCRecip	Number of active NC recipients assigned to a notification class
IAETrAllocResumes	APL client transactions (confirmed requests) resource allocation resumes. A needed resource was unavailable and a resume of the AE update process was scheduled.
IAENwMsgAllocResumes	NWL client message resource allocation resumes.
IAEConfNotifys	Number of confirmed AE notifications sent
IAEUnConfNotifys	Number of unconfirmed AE notifications sent
IAEConfNotifyErrors	Number of confirmed AE notification errors. A negative acknowledgement or/and an internal timeout was returned to a confirmed AE notify request.
IAEDABLookupErrors	Number of DAB (Device Address Binding) lookup errors. When an AE event is about to be sent to a BACnet recipient and we don't have an active binding against it (managed by who-is/i-am negotiations by the APL layer) this error is issued every time a notification could not be sent.
Interface Counters	Description
In Octets:	Received bytes.
In Ucast Packets:	Received unicast packets.
In NUcast packets:	Received non-unicast packets (broadcast and multicast).
In Discards:	Received packets discarded due to no available memory buffers.

Interface Counters	Description
In Errors:	Received packets discarded due to reception error.
In Unknown Protos:	Received packets with unsupported protocol type.
Out Octets:	Sent bytes.
Out Ucast packets:	Sent unicast packets.
Out NUcast packets:	Sent non-unicast packets (broadcast and multicast).
Out Discards:	Outgoing packets discarded due to no available memory buffers.
Out Errors:	Transmission errors.

## 6.3 Server Configuration

### 6.3.1 General Information

Basic web server configuration settings are stored in the system file \http.cfg. This file holds the web server name, root directory for the web interface, content types, and a list of file types which shall be scanned for SSI.

```
File Format:
[ServerName]
WebServerName
[WebRoot]
\web

[FileTypes]
FileType1:ContentType1
FileType2:ContentType2
...
FileTypeN:ContentTypeN

[SSIFileTypes]
FileType1
FileType2
...
FileTypeN
```

<b>Web Server Name</b> [ServerName]	Configures the web server name included in the HTTP header of the responses from the module.
<b>Web Root Directory</b> [WebRoot]	The web server cannot access files outside this directory.
<b>Content Types</b> [FileTypes]	A list of file extensions and their reported content types. See also... Default content types below
<b>SSI File Types</b> [SSIFileTypes]	By default, only files with the extension "shtm" are scanned for SSI. Additional SSI file types can be added here as necessary.

The web root directory determines the location of all files related to the web interface. Files outside of this directory and its subdirectories *cannot* be accessed by the web server.



### 6.3.2 Index page

The module searches for possible index pages in the following order:

1. <WebRoot>\index.htm
2. <WebRoot>\index.html
3. <WebRoot>\index.shtm
4. <WebRoot>\index.wml



*Substitute <WebRoot> with the web root directory specified in \http.cfg.*

*If no index page is found, the module will default to the virtual index file (if enabled).*

---

See also ...

- Default web pages

### 6.3.3 Default Content Types

By default, the following content types are recognized by their file extension:

File Extension	Reported Content Type
htm, html, shtm	text/html
gif	image/gif
jpeg, jpg, jpe	image/jpeg
png	image/x-png
js	application/x-javascript
bat, txt, c, h, cpp, hpp	text/plain
zip	application/x-zip-compressed
exe, com	application/octet-stream
wml	text/vnd.wap.wml
wmlc	application/vnd.wap.wmlc
wbmp	image/vnd.wap.wbmp
wmls	text/vnd.wap.wmlscript
wmlsc	application/vnd.wap.wmlscriptc
xml	text/xml
pdf	application/pdf
css	text/css

Content types can be added or redefined by adding them to the server configuration file.

### 6.3.4 Authorization

Directories can be protected from web access by placing a file called “web\_accs.cfg” in the directory to protect. This file shall contain a list of users that are allowed to access the directory and its subdirectories.

Optionally, a login message can be specified by including the key [AuthName]. This message will be displayed by the web browser upon accessing the protected directory.

```
File Format:
  Username1:Password1
  Username2:Password2
  ...
  UsernameN:PasswordN

  [AuthName]
  (message goes here)
```

The list of approved users can optionally be redirected to one or several other files.



*If the list of approved users is put in another file, be aware that this file can be accessed and read from the network.*

In the following example, the list of approved users will be loaded from here.cfg and too.cfg.

```
[File path]
\i\put\some\over\here.cfg
\i\actually\put\some\of\it\here\too.cfg

[AuthType]
Basic

[AuthName]
Howdy. Password, please.
```

The field “AuthType” is used to identify the authentication scheme.

Value	Description
Basic	Web authentication method using plaintext passwords.
Digest	More secure method using challenge-response authentication. Used as default if no [AuthType] field is specified.

## **7 E-mail Client**

### **7.1 General Information**

The built-in e-mail client allows the application to send e-mail messages through an SMTP-server. Messages can either be specified directly in the SMTP Client Object (04h), or retrieved from the file system. The latter may contain SSI, however note that for technical reasons, certain commands cannot be used (specified separately for each SSI command).

The client supports authentication using the 'LOGIN' method. Account settings etc. are stored in the Network Configuration Object (04h).

### **7.2 How to Send E-mail Messages**

To be able to send e-mail messages, the SMTP-account settings must be specified.

This includes:

- A valid SMTP-server address
- A valid username
- A valid password

To send an e-mail message, perform the following steps:

1. Create a new e-mail instance using the Create command (03h)
2. Specify the sender, recipient, topic and message body in the e-mail instance
3. Issue the Send Instance Email command (10h) towards the e-mail instance
4. Optionally, delete the e-mail instance using the Delete command (04h)

Sending a message based on a file in the file system is achieved using the Send Email from File command. This command is described in the SMTP Client Object (04h).

## 8 Server Side Include (SSI)

### 8.1 General Information

Server Side Include functionality, or SSI, allows data from files and objects to be represented on web pages and in e-mail messages.

SSI are special commands embedded within the source document. When the Anybus CompactCom module encounters such a command, it will execute it, and replace it with the result (if applicable).

By default, only files with the extension 'shtm' are scanned for SSI.

### 8.2 Include File

This function includes the contents of a file. The content is scanned for SSI.



*This function cannot be used in e-mail messages.*

Syntax:

```
<?--#include file="filename"-->
```

filename:                      Source file

Scenario	Default Output
Success	(contents of file)

### 8.3 Command Functions

#### 8.3.1 General Information

Command functions executes commands and includes the result.

**General Syntax**

```
<?--#exec cmd_argument='command'-->
```

command:                      Command function, see below



*"command" is limited to a maximum of 500 characters.*

#### Command Functions

Command	Valid for E-mail Messages
GetConfigItem()	Yes
SetConfigItem()	No
SsiOutput()	Yes
DisplayRemoteUser	No
ChangeLanguage()	No
IncludeFile()	Yes
SaveDataToFile()	No

Command	Valid for E-mail Messages
printf()	Yes
scanf()	No

### 8.3.2 GetConfigItem()

This command returns specific information from a file in the file system.

#### File Format

The source file must have the following format:

```
[key1]
value1

[key2]
value2
...
[keyN]
valueN
```

#### Syntax:

```
<?--exec cmd_argument='GetConfigItem("filename", "key" [, "separator"])'-->
```

filename:               Source file to read from  
key:                    Source [key] in file.  
separator:              Optional; specifies line separation characters (e.g. "<br>").  
                         (default is CRLF).

#### Default Output

Scenario	Default Output
Success	<i>(value of specified key)</i>
Authentication Error	"Authentication error"
File open error	"Failed to open file 'filename'"
Key not found	"Tag (key) not found"

**Example**

The following SSI...

```
<?--exec cmd_argument='GetConfigItem("\example.cnf", "B")'-->
```

... in combination with the following file ('example.cnf')...

```
[A]  
First  
[B]  
Second  
[C]  
Third
```

... returns the string 'Third'.

### 8.3.3 SetConfigItem()

This function stores an HTML-form as a file in the file system.



*This function cannot be used in e-mail messages.*

#### File Format

Each form object is stored as a [tag], followed by the actual value.

```
[form object name 1]
form object value 1

[form object name 2]
form object value 2

[form object name 3]
form object value 3

...
[form object name N]
form object value N
```



*Form objects with names starting with underscore will not be stored.*

#### Syntax:

```
<?--exec cmd_argument='SetConfigItem("filename",[ Overwrite])'-->
```

filename: Destination file. If the specified file does not exist, it will be created (provided that the path is valid).

Overwrite: Optional; forces the module to create a new file each time the command is issued. The default behavior is to modify the existing file.

#### Default Output

Scenario	Default Output
Success	"Configuration stored to 'filename'"
Authentication Error	"Authentication error"
File open error	"Failed to open file 'filename'"
File write error	"Could not store configuration to 'filename'"

### Example

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SetConfigItem command.

```
<HTML>
<HEAD><TITLE>SetConfigItem Test</TITLE></HEAD>
<BODY>

<!--#exec cmd_argument='SetConfigItem("\food.txt")'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Name">Name: </LABEL><BR>
    <INPUT type="text" name="Name"><BR><BR>

    <LABEL for="_Age">Age: </LABEL><BR>
    <INPUT type="text" name="_Age"><BR><BR>

    <LABEL for="Food">Food: </LABEL><BR>
    <INPUT type="radio" name="Food" value="Cheese"> Cheese<BR>
    <INPUT type="radio" name="Food" value="Sausage"> Sausage<BR><BR>

    <LABEL for="Drink">Drink: </LABEL><BR>
    <INPUT type="radio" name="Drink" value="Wine"> Wine<BR>
    <INPUT type="radio" name="Drink" value="Beer"> Beer<BR><BR>

    <INPUT type="submit" name="_submit">
    <INPUT type="reset" name="_reset">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file ('food.txt') may look somewhat as follows:

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```



*In order for this example to work, the HTML file must be named "test.shtm".*



### 8.3.4 SsiOutput()

This command temporarily modifies the SSI output of the following command function.

**Syntax:**

```
<?--#exec cmd_argument='SsiOutput("success", "failure")'-->
```

success:                      String to use in case of success  
failure:                      String to use in case of failure

**Default Output**

(this command produces no output on its own)

**Example**

The following example illustrates how to use this command.

```
<?--#exec cmd_argument='SsiOutput ("Parameter stored", "Error")'-->  
<?--#exec cmd_argument='SetConfigItem("File.cfg", Overwrite)'-->
```

See also...

- [SSI Output Configuration, p. 61](#)

### 8.3.5 DisplayRemoteUser

This command stores returns the username on an authentication session.



*This command cannot be used in e-mail messages.*

**Syntax:**

```
<?--#exec cmd_argument='DisplayRemoteUser'-->
```

**Default Output**

Scenario	Default Output
Success	(current user)

### 8.3.6 ChangeLanguage()

This command changes the language setting based on an HTML form object.



*This function cannot be used in e-mail messages.*

#### Syntax:

```
<!--#exec cmd_argument='ChangeLanguage ( "source" ) '-->
```

source:                      Name of form object which contains the new language setting.

The passed value must be a single digit as follows:

Form value	Language
"0"	English
"1"	German
"2"	Spanish
"3"	Italian
"4"	French

#### Default Output

Scenario	Default Output
Success	"Language changed"
Error	"Failed to change language"

#### Example

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the ChangeLanguage() command.

```
<HTML>
<HEAD><TITLE>ChangeLanguage Test</TITLE></HEAD>
<BODY>

<!--#exec cmd_argument='ChangeLanguage ("lang") '-->

<FORM action="test.shtm">
  <P>
    <LABEL for="lang">Language (0-4) : </LABEL><BR>
    <INPUT type="text" name="lang"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```



*In order for this example to work, the HTML file must be named "test.shtm".*

### 8.3.7 IncludeFile()

This command includes the content of a file. Note that the content is not scanned for SSI.

#### Syntax:

```
<?--#exec cmd_argument='IncludeFile("filename" [, separator])'-->
```

filename: Source file  
separator: Optional; specifies line separation characters (e.g. "<br>").

#### Default Output

Scenario	Default Output
Success	<i>(file contents)</i>
Authentication Error	"Authentication error"
File Open Error	"Failed to open file 'filename'"

#### Example

The following example demonstrates how to use this function.

```
<HTML>
<HEAD><TITLE>IncludeFile Test</TITLE></HEAD>
<BODY>
  <H1> Contents of 'info.txt':</H1>
  <P>
    <?--#exec cmd_argument='IncludeFile("info.txt")'-->.
  </P>
</BODY>
</HTML>
```

Contents of 'info.txt':

```
Neque porro quisquam est qui dolorem ipsum quia dolor sit
amet,consectetur, adipisci velit...
```

When viewed in a browser, the resulting page should look somewhat as follows:

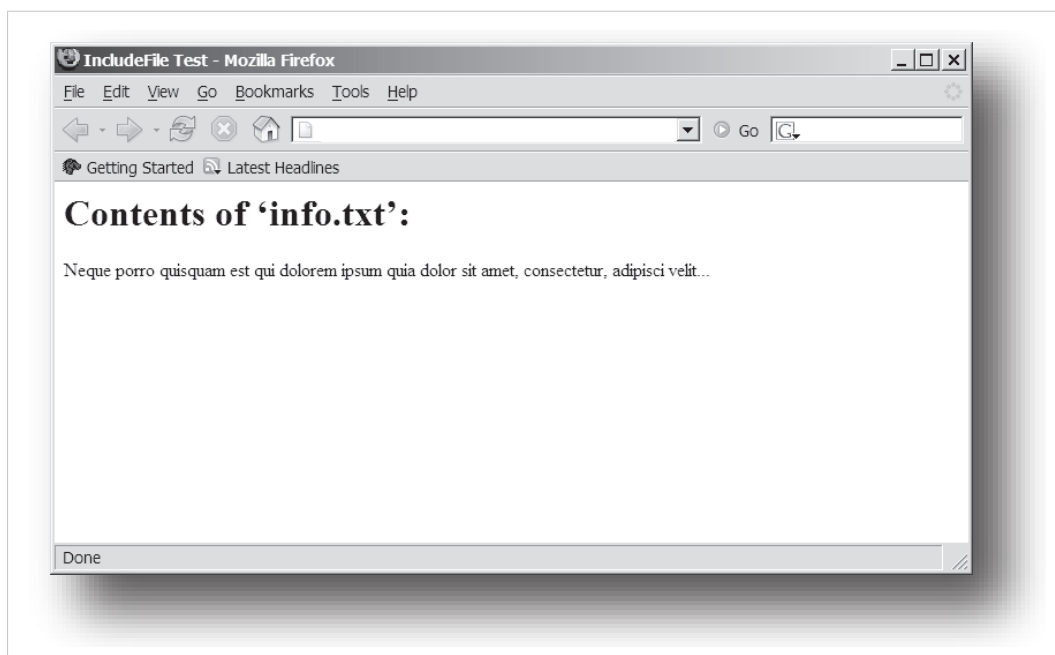


Fig. 6

See also...

- [Include File, p. 42](#)

### 8.3.8 SaveDataToFile()

This command stores data from an HTML form as a file in the file system. Content from the different form objects are separated by a blank line (2\*CRLF).



*This function cannot be used in e-mail messages.*

#### Syntax:

```
<!--#exec cmd_argument='SaveDataToFile("filename" [, "source"],  
Overwrite|Append) '-->
```

filename	Destination file. If the specified file does not exist, it will be created (provided that the path is valid).
source:	Optional; by specifying a form object, only data from that particular form object will be stored. Default behavior is to store data from all form objects except the ones where the name starts with underscore.
Overwrite Append	Specifies whether to overwrite or append data to existing files.

#### Default Output

Scenario	Default Output
Success	"Configuration stored to ' <i>filename</i> '"
Authentication Error	"Authentication error"
File Write Error	"Could not store configuration to ' <i>filename</i> '"

### Example

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SaveDataToFile command.

```
<HTML>
<HEAD><TITLE>SaveDataToFile Test</TITLE></HEAD>
<BODY>

<!--#exec cmd_argument='SaveDataToFile("\stuff.txt", "Meat", Overwrite) '-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Fruit">Fruit: </LABEL><BR>
    <INPUT type="text" name="Fruit"><BR><BR>

    <LABEL for="Meat">Meat: </LABEL><BR>
    <INPUT type="text" name="Meat"><BR><BR>

    <LABEL for="Meat">Bread: </LABEL><BR>
    <INPUT type="text" name="Bread"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file (\stuff.txt) will contain the value specified for the form object called “Meat”.



*In order for this example to work, the HTML file must be named “test.shtm”.*

### 8.3.9 printf()

This function returns a formatted string which may contain data from the Anybus CompactCom module and/or application. The formatting syntax used is similar to that of the standard C-function printf().

The function accepts a template string containing zero or more formatting tags, followed by a number of arguments. Each formatting tag corresponds to a single argument, and determines how that argument shall be converted to human readable form.

#### Syntax:

```
<?--#exec cmd_argument='printf("template" [, argument1, ..., argumentN])'-->
```

- |           |  |
|-----------|--|
| template: | Template which determines how the arguments shall be represented. May contain any number of formatting tags which are substituted by subsequent arguments and formatted as requested. The number of format tags must match the number of arguments; if not, the result is undefined.<br>See section "Formatting Tags" below for more information.  |
| argument: | Source arguments; optional parameters which specify the actual source of the data that shall be inserted in the template string. The number of arguments must match the number of formatting tags; if not, the result is undefined.<br>At the time of writing, the only allowed argument is ABCCMessage().<br>See also... <ul style="list-style-type: none"> <li>• <a href="#">ABCCMessage(), p. 57</a></li> </ul> |

#### Default Output

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string ( <a href="#">Errors, p. 60</a> )

#### Example

See ..

- [ABCCMessage\(\), p. 57](#)
- [Example \(Get\\_Attribute\);, p. 59](#)

#### Formatting Tags

Formatting tags are written as follows:

```
%[Flags][Width][.Precision][Modifier]type
```

- **Type (Required)**

The Type-character is required and determines the basic representation as follows:

Type Character	Representation	Example
c	Single character	b
d, i	Signed decimal integer.	565
e, E	Floating-point number in exponential notation.	5.6538e2
f	Floating-point number in normal, fixed-point notation.	565.38
g, G	%e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeroes/decimal point are not printed.	565.38
o	Unsigned octal notation	1065
s	String of characters	Text
u	Unsigned decimal integer	4242
x, X	Hexadecimal integer	4e7f
%	Literal %; no assignment is made	%

- **Flags (Optional)**

Flag Character	Meaning
-	Left-justify the result within the give width (default is right justification)
+	Always include a + or - to indicate whether the number is positive or negative
(space)	If the number does not start with a + or -, prefix it with a space character instead.
0 (zero)	Pad the field with zeroes instead of spaces
#	For %e, %E, and %f, forces the number to include a decimal point, even if no digits follow. For %x and %X, prefixes 0x or 0X, respectively.

- **Width (Optional)**

Width	Meaning
number	Specifies the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded to make up the field width. The result is never truncated even if the result is larger.

- **Precision (Optional)**

The exact meaning of this field depends on the type character:

Type Character	Meaning
d, i, o, u, x, X	Specifies the minimum no. of decimal digits to be printed. If the value to be printed is shorter than this number, the result is padded with space. Note that the result is never truncated, even if the result is larger.
e, E, f	Specifies the no. of digits to be printed after the decimal point (default is 6).
g, G	Specifies the max. no. of significant numbers to be printed.
s	Specifies the max. no. of characters to be printed
c	(no effect)

- **Modifier**

Modifier Character	Meaning
hh	Argument is interpreted as SINT8 or UINT8
h	Argument is interpreted as SINT16 or UINT16
L	Argument is interpreted as SINT32 or UINT32



### 8.3.10 scanf()

This function is very similar to the printf() function described earlier, except that it is used for input rather than output. The function reads a string passed from an HTML form object, parses the string as specified by a template string, and sends the resulting data to the specified argument. The formatting syntax used is similar to that of the standard C-function scanf().

The function accepts a source, a template string containing zero or more formatting tags, followed by a number of arguments. Each argument corresponds to a formatting tag, which determines how the data read from the HTML form shall be interpreted prior sending it to the destination argument.



*This command cannot be used in e-mail messages.*

#### Syntax:

```
<?--#exec cmd_argument='scanf("source", "template" [,
                                argument1, ..., argumentN])'-->
```

source	Name of the HTML form object from which the string shall be extracted.
template:	Template which specifies how to parse and interpret the data. May contain any number of formatting tags which determine the conversion prior to sending the data to subsequent arguments. The number of formatting tags must match the number of arguments; if not, the result is undefined. See section "Formatting Tags" below for more information.
argument:	Destination argument(s) specifying where to send the interpreted data. The number of arguments must match the number of formatting tags; if not, the result is undefined. At the time of writing, the only allowed argument is ABCCMessage(). See also... <ul style="list-style-type: none"> <li><a href="#">ABCCMessage(), p. 57</a></li> </ul>

#### Default Output

Scenario	Default Output
Success	"Success"
Parsing error	"Incorrect data format"
Too much data for argument	"Too much data"
ABCCMessage error	ABCCMessage error string ( <a href="#">Errors, p. 60</a> )

#### Example

See also...

[ABCCMessage\(\), p. 57](#)

[Example \(Set\\_Attribute\);, p. 59](#)

#### Formatting Tags

Formatting tags are written as follows:

```
%[*][Width][Modifier]type
```

- **Type (Required)**

The Type-character is required and determines the basic representation as follows:

Type	Input	Argument Data Type
c	Single character	CHAR
d	Accepts a signed decimal integer	SINT8 SINT16 SINT32
i	Accepts a signed or unsigned decimal integer. May be given as decimal, hexadecimal or octal, determined by the initial characters of the input data: Initial Characters: Format: 0x Hexadecimal 0: Octal 1... 9: Decimal	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
u	Accepts an unsigned decimal integer.	UINT8 UINT16 UINT32
o	Accepts an optionally signed octal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
x, X	Accepts an optionally signed hexadecimal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
e, E, f, g, G	Accepts an optionally signed floating point number. The input format for floating-point numbers is a string of digits, with some optional characteristics:  – It can be a signed value  – It can be an exponential value, containing a decimal rational number followed by an exponent field, which consists of an 'E' or an 'e' followed by an integer.	FLOAT
n	Consumes no input; the corresponding argument is an integer into which scanf writes the number of characters read from the object input.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
s	Accepts a sequence of nonwhitespace characters	STRING
[scanset]	Accepts a sequence of nonwhitespace characters from a set of expected bytes specified by the scanlist (e.g '[0123456789ABCDEF]') A literal '[' character can be specified as the first character of the set. A caret character (^) immediately following the initial '[' inverts the scanlist, i.e. allows all characters except the ones that are listed.	STRING
%	Accepts a single %input at this point; no assignment or conversion is done. The complete conversion specification should be %%. -	-

- **\* (Optional)**

Data is read but ignored. It is not assigned to the corresponding argument.

- **Width (Optional)**

Specifies the maximum number of characters to be read

- **Modifier (Optional)**

Specifies a different data size.

Modifier	Meaning
h	SINT8, SINT16, UINT8 or UINT16
l	SINT32 or UINT32

## 8.4 Argument Functions

### 8.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

#### General Syntax:

(Syntax depends on context)

#### Argument Functions:

Function	Description
ABCCMessage()	-

### 8.4.2 ABCCMessage()

This function issues an object request towards an object in the module or in the host application.

#### Syntax

```
ABCCMessage(object, instance, command, ce0, ce1,
            msgdata, c_type, r_type)
```

object	Specifies the Destination Object
instance	Specifies the Destination Instance
command	Specifies the Command Number
ce0	Specifies CmdExt[0] for the command message
ce1	Specifies CmdExt[1] for the command message
msgdata	Specifies the actual contents of the MsgData[] subfield in the command <ul style="list-style-type: none"> <li>Data can be supplied in direct form (format depends on c_type)</li> <li>The keyword "ARG" is used when data is supplied by the parent command (e.g. scanf()).</li> </ul>
c_type:	Specifies the data type in the command (msgdata), see below.
r_type:	Specifies the data type in the response (msgdata), see below.

Numeric input can be supplied in the following formats:

Decimal (e.g. 50)	(no prefix)
Octal (e.g. 043)	Prefix 0 (zero)
Hex (e.g. 0x1f)	Prefix 0x

- Command Data Types (c\_type)

For types which support arrays, the number of elements can be specified using the suffix [n], where n specifies the number of elements. Each data element must be separated by space.

Type	Supports Arrays	Data format (as supplied in msgdata)
BOOL	Yes	1
SINT8	Yes	-25
SINT16	Yes	2345
SINT32	Yes	-2569
UINT8	Yes	245
UINT16	Yes	40000
UINT32	Yes	32
CHAR	Yes	A
STRING	No	"abcde" <b>Note:</b> Quotes can be included in the string if preceded by backslash ("") Example: "We usually refer to it as \'the Egg\'"
FLOAT	Yes	5.6538e2
NONE	No	Command holds no data, hence no data type

- Response Data Types (r\_type)

For types which support arrays, the number of elements can be specified using the suffix [n], where n specifies the number of elements.

Type	Supports Arrays	Data format (as supplied in msgdata)
BOOL	Yes	Optionally, it is possible to exchange the BOOL data with a message based on the value (true or false). In such case, the actual data type returned from the function will be STRING. Syntax: BOOL<true><false> For arrays, the format will be BOOL[n]<true><false>.
SINT8	Yes	-
SINT16	Yes	-
SINT32	Yes	-
UINT8	Yes	This type can also be used when reading ENUM data types from an object. In such case, the actual ENUM value will be returned.
UINT16	Yes	-
UINT32	Yes	-
CHAR	Yes	-
STRING	No	-
ENUM	No	When using this data type, the ABCMessage() function will first read the ENUM value. It will then issue a 'Get Enum String'-command to retrieve the actual enumeration string. The actual data type in the response will be STRING.
FLOAT	Yes	-
NONE	No	Response holds no data, hence no data type



*It is important to note that the message will be passed transparently to the addressed object. The SSI engine performs no checks for violations of the object addressing scheme, e.g. a malformed Get\_Attribute request which (wrongfully) includes message data will be passed unmodified to the object, even though this is obviously wrong. Failure to observe this may cause loss of data or other undesired side effects.*

**Example (Get\_Attribute):**

This example shows how to retrieve the IP address using printf() and ABCCMessage().

```
<?--#exec cmd_argument='printf( "%u.%u.%u.%u",
    ABCCMessage(4,3,1,5,0,0,NONE,UINT8[4] ) )'-->
```

Variable	Value	Comments
object	4	Network Configuration Object (04h)
instance	3	Instance #3 (IP address)
command	1	Get_attribute
ce0	5	Attribute #5
ce1	0	-
msgdata	0	-
c_type	NONE	Command message holds no data
r_type	UINT8[4]	Array of 4 unsigned 8-bit integers

**Example (Set\_Attribute):**

This example shows how to set the IP address using scanf() and ABCCMessage(). Note the special parameter value “ARG”, which instructs the module to use the passed form data (parsed by scanf() ).

```
<?--#exec cmd_argument='scanf("IP", "%u.%u.%u.%u",
    ABCCMessage(4,3,2,5,0,ARG,UINT8[4],NONE ) )'-->
```

Variable	Value	Comments
object	4	Network Configuration Object (04h)
instance	3	Instance #3 (IP address)
command	2	Set_attribute
ce0	5	Attribute #5
ce1	0	-
msgdata	ARG	Use data parsed by scanf() call
c_type	UINT8[4]	Array of 4 unsigned 8-bit integers
r_type	NONE	Response message holds no data

## Errors

In case an object request results in an error, the error code in the response will be evaluated and translated to readable form as follows:

Error Code	Output
0	"Unknown error"
1	"Unknown error"
2	"Invalid message format"
3	"Unsupported object"
4	"Unsupported instance"
5	"Unsupported command"
6	"Invalid CmdExt[0]"
7	"Invalid CmdExt[1]"
8	"Attribute access is not set-able"
9	"Attribute access is not get-able"
10	"Too much data in msg data field"
11	"Not enough data in msg data field"
12	"Out of range"
13	"Invalid state"
14	"Out of resources"
15	"Segmentation failure"
16	"Segmentation buffer overflow"
17... 255	"Unknown error"

See also...

[SSI Output Configuration, p. 61](#)

## 8.5 SSI Output Configuration

Optionally, the SSI output can be permanently changed by adding the file \output.cfg.

File format:

<pre>[ABCCMessage_X] 0:"Success string" 1:"Error string 1" 2:"Error string 2" ... 16:"Error string 16"</pre>	Each error code corresponds to a dedicated output string, labelled from 1 to 16. See <a href="#">Errors, p. 60</a>
<pre>[GetConfigItem_X] 0:"Success string" 1:"Authentication error string" 2:"File open error string" 3:"Tag not found string"</pre>	Use "%s" to include the name of the file.
<pre>[SetConfigItem_X] 0:"Success string" 1:"Authentication error string" 2:"File open error string" 3:"File write error string"</pre>	Use "%s" to include the name of the file.
<pre>[IncludeFile_X] 0:"Success string" 1:"Authentication error string" 2:"File read error string"</pre>	Use "%s" to include the name of the file.
<pre>[scanf_X] 0:"Success string" 1:"Parsing error string"</pre>	-
<pre>[ChangeLanguage_X] 0:"Success string" 1:"Change error string"</pre>	-

All content above can be included in the file multiple times changing the value "X" in each tag for different languages. The module will then select the correct output string based on the language settings. If no information for the selected language is found, it will use the default SSI output.

Value of X	Language
0	English
1	German
2	Spanish
3	Italian
4	French

See also...

- 

[SsiOutput\(\), p. 47](#)

## 9 JSON

### 9.1 General Information

JSON is an acronym for JavaScript Object Notation and an open standard format for storing and exchanging data in an organized and intuitive way. In Anybus CompactCom, it is used to transmit data objects consisting of name - value pairs between the webserver in the Anybus CompactCom and a web application. The object members are unordered, thus the value pairs can appear in any order. JavaScripts are used to create dynamic web pages to present the values. Optionally, a callback may be passed to the GET-request for JSONP output.

JSON is more versatile than SSI in that you not only can read and write, but also change the size and the look of the web page dynamically. A simple example of how to create a web page is added at the end of this chapter.

#### 9.1.1 Encoding

JSON requests shall be UTF-8 encoded. The module will interpret JSON requests as UTF-8 encoded, while all other HTTP requests will be interpreted as ISO-8859-1 encoded. All JSON responses, sent by the module, are UTF-8 encoded, while all other files sent by the web server are encoded as stored in the file system.

#### 9.1.2 Access

It is recommended to password protect the JSON resources. Add password protection by adding a file called web\_accs.cfg in the root directory (all web content will be protected). The file is described in the “Web Server” section in this document.

#### 9.1.3 Error Response

If the module fails to parse or process a request, the response will contain an error object with an Anybus error code:

```
{
  "error"      : 02
}
```

The Anybus error codes are listed in the Anybus CompactCom 40 Software Design Guide.



## 9.2 JSON Objects

### 9.2.1 ADI

#### info.json

```
GET adi/info.json[?callback=<function>]
```

This object holds information about the ADI JSON interface. This data is static during runtime.

Name	Data Type	Note
dataformat	Number	0 = Little endian 1 = Big endian (Affects value, min and max representations)
numadis	Number	Total number of ADIs
webversion	Number	Web/JSON API version

JSON response example:

```
{
  "dataformat": 0,
  "numadis": 123,
  "webversion": 1
}
```

#### data.json

```
GET adi/data.json?offset=<offset>&count=<count>[&callback=<function>]
GET adi/data.json?inst=<instance>&count=<count>[&callback=<function>]
```

These object calls fetch a sorted list of up to <count> ADIs values, starting from <offset> or <instance>. The returned values may change at any time during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the "order number" of the first requested ADI. The first implemented ADI will always get order number 0. <count> number of existing ADI values will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. <count> number of ADI values is returned. A null value will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
—	Array of Strings	Sorted list of string representations of the ADI value attributes

JSON response example (using offset):

```
[
  "FF",
  "A201",
  "01FAC105"
]
```

JSON response example (using inst):

```
[
  "FF",
  "A201",
  null,
  null,
  "01FAC105"
]
```

### metadata.json

```
GET adi/metadata.json?offset=<offset>&count=<count>[&callback=<function>]
GET adi/metadata.json?inst=<instance>&count=<count>[&callback=<function>]
```

These object calls fetch a sorted list of metadata objects for up to <count> ADIs, starting from <offset> or <instance>.

The returned information provided is a transparent representation of the attributes available in the host Application Data object (FEh). See the Anybus CompactCom 40 Software Design Guide for more information about the content of each attribute.

The ADI metadata is static during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the "order number" of the first requested ADI. The first implemented ADI will always get order number 0. Metadata objects for <count> number of existing ADI will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. Metadata objects for <count> number of ADI values are returned. A null object will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
instance	Number	-
name	String	May be NULL if no name is present.
numelements	Number	-
datatype	Number	-
min	String	Hex formatted string, see <a href="#">Hex Format Explained, p. 80</a> for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see <a href="#">Hex Format Explained, p. 80</a> for more information. May be NULL if no maximum value is present.
access	Number	Bit 0: Read access Bit 1: Write access

## JSON response example (using offset):

```
[
{
  "instance": 1,
  "name": "Temperature threshold",
  "numelements": 1,
  "datatype": 0,
  "min": "00",
  "max": "FF",
  "access": 0x03
},
{
  ...
}
]
```

## JSON response example (using inst):

```
[
{
  "instance": 1,
  "name": "Temperature threshold",
  "numelements": 1,
  "datatype": 0,
  "min": "00",
  "max": "FF",
  "access": 0x03
},
null,
null
{
  ...
}
]
```

**metadata2.json**

```
GET adi/metadata2.json?offset=<offset>&count=<count>[&callback=<function>]
GET adi/metadata2.json?inst=<instance>&count=<count>[&callback=<function>]
```

This is an extended version of the metadata function that provides complete information about the ADIs. This extended version is needed to describe more complex data types such as Structures.

The information provided is a transparent representation of the attributes available in the host Application Data object (FEh). See the Anybus CompactCom 40 Software Design Guide for more information about the content of each attribute.

The ADI metadata is static during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the “order number” of the first requested ADI. The first implemented ADI will always get order number 0. Metadata objects for <count> number of existing ADI will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. Metadata objects for <count> number of ADI values are returned. A null object will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
instance	Number	-
numelements	Array of umbers	-
datatype	Array of Numbers	Array of datatypes. For Structures and Variables, each array element defines the data type of the corresponding element of the instance value. For Arrays, one array element defines the data type for all elements of the instance value.
descriptor		Array of descriptors. For Structures and Variables, each array element defines the descriptor of the corresponding element of the instance value. For Arrays, one array element defines the descriptor for all elements of the instance value.
name		May be NULL if no name is present.
min	String	Hex formatted string, see <a href="#">Hex Format Explained, p. 80</a> for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see <a href="#">Hex Format Explained, p. 80</a> for more information. May be NULL of no maximum value is present.
default	String	Hex formatted string, see <a href="#">Hex Format Explained, p. 80</a> for more information. May be NULL of no default value is present.
numsubelements	Array of Numbers	For Structures and Variables each array element defines the number of subelements of the corresponding element of the instance value. May be NULL if not present.
elementname	Array of Strings	Array of names, one for each instance value element. May be NULL if not present.

## JSON response example (using offset):

```
[
{
  "instance": 1,
  "numelements": 1,
  "datatype": [0 ],
  "descriptor": [9 ],
  "name": "Temperature threshold",
  "max": "FF",
  "min": "00",
  "default": "00",
  "numsubelements": null
  "elementname": null
},
{
  ...
}
]
```

## JSON response example (instance):

```
[
{
  "instance": 1,
  "numelements": 1,
  "datatype": [0 ],
  "descriptor": [9 ],
  "name": "Temperature threshold",
  "max": "FF",
  "min": "00",
  "default": "00",
  "numsubelements": null
  "elementname": null
},
null,
null
{
  ...
}
]
```

**enum.json**

```
GET adi/enum.json?inst=<instance>[&value=<element>][&callback=<function>]
```

This object call fetches a list of enumeration strings for a specific instance.

The ADI enum strings are static during runtime.

Request data:

Name	Data Type	Description
inst	Number	Instance number of the ADI to get enum string for.
value	Number	Optional. If given only the enumstring for the requested <value> is returned.
callback	String	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
string	String	String representation for the corresponding value.
value	Number	Value corresponding to the string representation.

JSON response example:

```
[
  {
    "string": "String for value 1",
    "value": 1
  },
  {
    "string": "String for value 2",
    "value": 2
  },
  {
    ...
  }
]
```

**update.json**

```
POST adi/update.json
```

**Form data:**

```
inst=<instance>&value=<data>[&elem=<element>][&callback=<function>]
```

Updates the value attribute of an ADI.

**Request data:**

Name	Data Type	Description
inst	Number	Instance number of the ADI
value	String	Value to set. If the value attribute is a number it shall be hex formatted, see <a href="#">Hex Format Explained, p. 80</a> for more information.
elem	Number	Optional. If specified only a single element of the ADI value is set. Then <data> shall only contain the value of the specified <element>.
callback	String	Optional. A callback function for JSONP output.

**Response data:**

Name	Data Type	Note
result	Number	0 = success The Anybus CompactCom error codes are used. Please see the Anybus CompactCom 40 Software Design Guide.

```
{  
  "result" : 0  
}
```

## 9.2.2 Module

### info.json

```
GET module/info.json
```

Response data:

Name	Data Type	Description
modulename	String	-
serial	String	32 bit hex ASCII
fwver	Array of Number	(major, minor, build)
uptime	Array of Number	[high, low] milliseconds (ms)
cpuload	Number	CPU load in %
fwvertext	String	Firmware version in text
vendorname	String	Vender name (Application Object (FFh), instance attribute #8)
hwvertext	String	Hardware version in text
networktype	Number	Network type (Network Object (03h), instance attribute #1)

JSON response example:

```
{
  "modulename": "ABCC M40",
  "serial": "ABCDEF00",
  "fwver": [ 1, 5, 0 ],
  "uptime": [ 5, 123456 ],
  "cpuload": 55
  "fwvertext": "1.05.02",
  "vendorname": "HMS Industrial Networks",
  "hwvertext": "2",
  "networktype": "0085",
}
```



### 9.2.3 Network

#### ethstatus.json

```
GET network/ethstatus.json.
```

Name	Data Type	Description
mac	String	6 byte hex
comm1	Object	See object definition in the table below
comm2	Object	See object definition in the table below

#### Comm Object Definition:

Name	Data Type	Description
link	Number	0: No link 1: Link
speed	Number	0: 10 Mbit 1: 100 Mbit
duplex	Number	0: Half 1: Full

JSON response example:

```
{
  "mac":      "003011FF0201",
  "comm1":    {
    "link":    1,
    "speed":   1,
    "duplex":  1
  },
  "comm2":    {
    "link":    1,
    "speed":   1,
    "duplex":  1
  }
}
```

**ipstatus.json & ipconf.json**

These two object share the same data format. The object ipconf.json returns the configured IP settings, and ipstatus.json returns the actual values that are currently used. ipconf.json can also be used to alter the IP settings.

```
GET network/ipstatus.json
```

or

```
GET network/ipconf.json
```

Name	Data Type	Note
dhcp	Number	-
addr	String	-
subnet	String	-
gateway	String	-
dns1	String	-
dns2	String	-
hostname	String	-
domainname	String	-

```
{
  "dhcp":      0,
  "addr":      "192.168.0.55",
  "subnet":    "255.255.255.0",
  "gateway":   "192.168.0.1",
  "dns1":      "10.10.55.1",
  "dns2":      "10.10.55.2",
  "hostname":  "abcc123",
  "domainname": "hms.se"
}
```

To change IP settings, use network/ipconf.json. It accepts any number of arguments from the list above. Values should be in the same format.

Example:

```
GET ipconf.json?dhcp=0&addr=10.11.32.2&hostname=abcc123&domainname=hms.se
```

**ethconf.json**

```
GET network/ethconf.json
```

Name	Data Type	Note
mac	String	-
comm1	Number	-
comm2	Number	Only present if two Ethernet ports are activated in the module.

The values of “comm1” and “comm2” are read from the Network Configuration object, instances #7 and #8.

```
{
  "mac":      [00, 30, 11, FF, 02, 01],
  "comm1":    0,
  "comm2":    4
}
```

The parameters “comm1” and “comm2” are configurable by adding them as arguments to the GET request:

```
GET network/ethconf.json?comm1=0&comm2=4
```

The parameters “comm1” and “comm2” may hold an error object with Anybus error code if the module fails processing the request:

```
{
  "mac":      [00, 30, 11, FF, 02, 01],
  "comm1":    0,
  "comm2":    { error: 14 },
}
```

The Anybus CompactCom error codes are used. Please see the Anybus CompactCom 40 Software Design Guide.

**ifcounters.json**

```
GET network/ifcounters.json?port=<port>
```

Valid values for the argument <port> are 0, 1, and 2.

- Valid values for the argument <port> are 0, 1, and 2.
- Port number 0 option refers to the internal port (CPU port).
- Port number 2 option is only valid if two Ethernet ports are activated in the module.

Name	Data Type	Description
inotets	Number	IN: bytes
inucast	Number	IN: unicast packets
innucast	Number	IN: broadcast and multicast packets
indiscards	Number	IN: discarded packets
inerrors	Number	IN: errors
inunknown	Number	IN: unsupported protocol type
outotets	Number	OUT: bytes
outucast	Number	OUT: unicast packets
outnucast	Number	OUT: broadcast and multicast packets
outdiscards	Number	OUT: discarded packets
outerrors	Number	OUT: errors

**mediacounters.json**

```
GET network/mediacounters.json?port=<port>
```

The argument <port> is either 1 or 2.

Port number 2 option is only valid if two Ethernet ports are activated in the module.

Name	Data Type	Description
align	Number	Frames received that are not an integral number of octets in length
fcs	Number	Frames received that do not pass the FCS check
singlecoll	Number	Successfully transmitted frames which experienced exactly one collision
multicoll	Number	Successfully transmitted frames which experienced more than one collision
latecoll	Number	Number of collisions detected later than 512 bit times into the transmission of a packet
excesscoll	Number	Frames for which transmissions fail due to excessive collisions
sqetest	Number	Number of times SQE test error is generated
deferredtrans	Number	Frames for which the first transmission attempt is delayed because the medium is busy
macrecerr	Number	Frames for which reception fails due to an internal MAC sublayer receive error
mactranserr	Number	Frames for which transmission fails due to an internal MAC sublayer transmit error
cserr	Number	Times that the carrier sense was lost or never asserted when attempting to transmit a frame
toolong	Number	Frames received that exceed the maximum permitted frame size

**nwstats.json**

```
GET network/nwstats.json
```

This object lists available statistics data. The data available depends on the product.

Example output:

```
[
  or
  [ { "identifier": "eipstats", "title": "EtherNet/IP Statistics" } ]
  or
  [ { "identifier": "eitstats", "title": "Modbus TCP Statistics" } ]
  or
  [
    { "identifier": "bacnetipstats",
      "title": "BACnet/IP Statistics" },
    { "identifier": "bacnetaplserverstats",
      "title": "BACnet Application Layer Server Statistics" },
    { "identifier": "bacnetaplclientstats",
      "title": "BACnet Application Layer Client Statistics" },
    { "identifier": "bacnetalarmstats",
      "title": "BACnet Alarm and Event Module Statistics" }
  ]
  or
  [ { "identifier": "eplifcounters", "title": "IT Interface Counters" } ]
  or
  [
    { "identifier": "ectstats", "title": "EtherCAT Statistics" },
    { "identifier": "eoeifcounters", "title": "EoE Interface Counters" },
  ]
  or
  [ { "identifier": "pnpopf", "title": "Fiber Optical Statistics" } ]
```

Get network specific statistics (<ID> is an “identifier” value returned from the previous command):

```
GET network/nwstats.json?get=<ID>
```

“eipstats”

```
[
  { "name": "Established Class1 Connections", "value": 0 },
  { "name": "Established Class3 Connections", "value": 1 },
  { "name": "Connection Open Request", "value": 0 },
  { "name": "Connection Open Format Rejects", "value": 0 },
  { "name": "Connection Open Resource Rejects", "value": 0 },
  { "name": "Connection Open Other Rejects", "value": 0 },
  { "name": "Connection Close Requests", "value": 0 },
  { "name": "Connection Close Format Rejects", "value": 0 },
  { "name": "Connection Other Rejects", "value": 0 },
  { "name": "Connection Timeouts", "value": 0 },
]
```

**“eitstats”**

```
[
  { "name": "Modbus Connections", "value": 0 },
  { "name": "Connection ACKs", "value": 1 },
  { "name": "Connection NACKs", "value": 0 },
  { "name": "Connection Timeouts", "value": 0 },
  { "name": "Process Active Timeouts", "value": 0 },
  { "name": "Processed messages", "value": 0 },
  { "name": "Incorrect messages", "value": 0 },
]
```

**“bacnetipstats”**

```
[
  { "name": "Unconfirmed server requests received", "value": 0 },
  { "name": "Unconfirmed server requests sent", "value": 1 },
  { "name": "Unconfirmed client requests sent", "value": 0 },
]
```

**“bacnetaplserverstats”**

```
[
  { "name": "Active transactions", "value": 0 },
  { "name": "Max Active transactions", "value": 1 },
  { "name": "Tx segments sent", "value": 0 },
  { "name": "Tx segment ACKs received", "value": 0 },
  { "name": "Tx segment NAKs received", "value": 0 },
  { "name": "Rx segments received", "value": 0 },
  { "name": "Rx segment ACKs sent", "value": 0 },
  { "name": "Duplicate Rx segment ACKs sent", "value": 0 },
  { "name": "Rx segment NAKs sent", "value": 0 },
  { "name": "Confirmed transactions sent", "value": 0 },
  { "name": "Confirmed transactions received", "value": 0 },
  { "name": "Tx segment timeouts", "value": 0 },
  { "name": "Rx segment timeouts", "value": 0 },
  { "name": "Implicit deletes", "value": 0 },
  { "name": "Tx timeout deletes", "value": 0 },
  { "name": "Rx timeout deletes", "value": 0 },
  { "name": "Tx aborts received", "value": 0 },
  { "name": "Rx aborts received", "value": 0 },
  { "name": "Transaction aborts sent", "value": 0 },
  { "name": "Transaction rejects sent", "value": 0 },
  { "name": "Transaction errors sent", "value": 0 },
]
```

**“bacnetapclientstats”**

```
[
  { "name": "Active transactions", "value": 0 },
  { "name": "Max Active transactions", "value": 1 },
  { "name": "Tx segments sent", "value": 0 },
  { "name": "Tx segment ACKs received", "value": 0 },
  { "name": "Tx segment NAKs received", "value": 0 },
  { "name": "Rx segments received", "value": 0 },
  { "name": "Rx segment ACKs sent", "value": 0 },
  { "name": "Duplicate Rx segment ACKs sent", "value": 0 },
  { "name": "Rx segment NAKs sent", "value": 0 },
  { "name": "Confirmed transactions sent", "value": 0 },
  { "name": "Confirmed transactions received", "value": 0 },
  { "name": "Tx segment timeouts", "value": 0 },
  { "name": "Rx segment timeouts", "value": 0 },
  { "name": "Implicit deletes", "value": 0 },
  { "name": "Tx timeout deletes", "value": 0 },
  { "name": "Rx timeout deletes", "value": 0 },
  { "name": "Tx aborts received", "value": 0 },
  { "name": "Rx aborts received", "value": 0 },
  { "name": "Transaction aborts sent", "value": 0 },
  { "name": "Transaction rejects sent", "value": 0 },
  { "name": "Transaction errors sent", "value": 0 },
]
```

**“bacnetalarmstats”**

```
[
  { "name": "COV Active subscriptions", "value": 0 },
  { "name": "COV Max active subscriptions", "value": 1 },
  { "name": "COV Lifetime subscriptions", "value": 0 },
  { "name": "COV Confirmed resumes", "value": 0 },
  { "name": "COV Unconfirmed resumes", "value": 0 },
  { "name": "COV Confirmed notifications sent", "value": 0 },
  { "name": "COV Unconfirmed notifications sent", "value": 0 },
  { "name": "COV Confirmed notification errors", "value": 0 },
  { "name": "AE Active events", "value": 0 },
  { "name": "AE Active NC recipients", "value": 0 },
  { "name": "AE Confirmed resumes", "value": 0 },
  { "name": "AE UnConfirmed resumes", "value": 0 },
  { "name": "AE Confirmed notifications sent", "value": 0 },
  { "name": "AE UnConfirmed notifications sent", "value": 0 },
  { "name": "AE Confirmed notification errors", "value": 0 },
  { "name": "AE DAB lookup errors", "value": 0 },
]
```



## “eplifcounters”

```
[
  { "name": "In Octets", "value": 22967 },
  { "name": "In Ucast Packets", "value": 121 },
  { "name": "In NUCast Packets", "value": 31 },
  { "name": "In Discards", "value": 0 },
  { "name": "In Errors", "value": 0 },
  { "name": "In Unknown Protos", "value": 0 },
  { "name": "Out Octets", "value": 169323 },
  { "name": "Out Ucast Packets", "value": 168 },
  { "name": "Out NUCast Packets", "value": 16 },
  { "name": "Out Discards", "value": 0 },
  { "name": "Out Errors", "value": 0 },
]
```

## “ectstats”

```
[
  { "name": "Logical EoE port link", "value": "Yes" },
  { "name": "Invalid frame counter IN port", "value": 1 },
  { "name": "Rx error counter IN port", "value": 1 },
  { "name": "Forwarded error counter IN port", "value": 1 },
  { "name": "Lost link counter IN port", "value": 1 },
  { "name": "Invalid frame counter OUT port", "value": 1 },
  { "name": "Rx error counter OUT port", "value": 1 },
  { "name": "Forwarded error counter OUT port", "value": 1 },
  { "name": "Lost link counter OUT port", "value": 1 },
]
```

## “eoeifcounters”

```
[
  { "name": "In Octets", "value": 22967 },
  { "name": "In Ucast Packets", "value": 121 },
  { "name": "In NUCast Packets", "value": 31 },
  { "name": "In Discards", "value": 0 },
  { "name": "In Errors", "value": 0 },
  { "name": "In Unknown Protos", "value": 0 },
  { "name": "Out Octets", "value": 169323 },
  { "name": "Out Ucast Packets", "value": 168 },
  { "name": "Out NUCast Packets", "value": 16 },
  { "name": "Out Discards", "value": 0 },
  { "name": "Out Errors", "value": 0 },
]
```

## “pnprof”

```
[
  { "name" : "Port 1 Temperature (C)", "value" : "41.37" },
  { "name" : "Port 1 Power Budget (dB)", "value" : "23.0" },
  { "name" : "Port 1 Power Budget Status", "value" : "OK" },
  { "name" : "Port 2 Temperature (C)", "value" : "40.57" },
  { "name" : "Port 2 Power Budget (dB)", "value" : "0.0" },
  { "name" : "Port 2 Power Budget Status", "value" : "OK" }
]
```

## 9.2.4 Services

### smtp.json

```
GET services/smtp.json
```



*Password is not returned when retrieving the settings.*

Name	Data Type	Note
server	String	IP address or name of mail server, e.g. "mail.hms.se"
user	String	-

```
[
  { "server": "192.168.0.55" },
  { "user": "test" }
]
```

Set:

Form data:

```
[
  [server=192.168.0.56]&[user=test2]&[password=secret],
]
```

## 9.2.5 Hex Format Explained

The metadata max, min, and default fields and the ADI values are ASCII hex encoded binary data. If the data type is an integer, the endianness used is determined by the dataformat field found in adi/info.json.

Examples:

The value 5 encoded as a UINT16, with dataformat = 0 (little endian):

```
0500
```

The character array "ABC" encoded as CHAR[3] (dataformat is not relevant for CHAR):

```
414243
```

## 9.3 Example

This example shows how to create a web page that fetches Module Name and CPU load from the module and presents it on the web page. The file, containing this code, has to be stored in the built-in file system, and the result can be seen in a common browser.

```
<html>
  <head>
    <title>Anybus CompactCom</title>

    <!-- Imported libs -->
    <script type="text/javascript" src="vfs/js/jquery-1.9.1.js"></script>
    <script type="text/javascript" src="vfs/js/tmpl.js"></script>
  </head>
  <body>
    <div id="info-content"></div>
    <script type="text/x-tmpl" id="tmpl-info">
      <b>From info.json</b><br>
      Module name:
      {%=o.modulename%}<br>

      CPU Load:
      {%=o.cpuload%}%<br>
    </script>
    <script type="text/javascript">
      $.getJSON( "/module/info.json", null, function(data){
        $("#info-content").html( tmpl("tmpl-info", data ) );
      });
    </script>
  </body>
</html>
```

## 10 Anybus Module Objects

### 10.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 40 BACnet/IP.

Standard Objects:

- [Anybus Object \(01h\), p. 83](#)
- [Diagnostic Object \(02h\), p. 84](#)
- [Network Object \(03h\), p. 85](#)
- [Network Configuration Object \(04h\), p. 87](#)

Network Specific Objects:

- [Socket Interface Object \(07h\), p. 98](#)
- [SMTP Client Object \(09h\), p. 115](#)
- [File System Interface Object \(0Ah\), p. 120](#)
- [Network Ethernet Object \(0Ch\), p. 121](#)

## 10.2 Anybus Object (01h)

### Category

Basic, extended

### Object Description

This object assembles all common Anybus data, and is described thoroughly in the general *Anybus CompactCom 40 Software Design Guide*.

### Supported Commands

<b>Object:</b>	Get_Attribute
<b>Instance:</b>	Get_Attribute
	Set_Attribute
	Get_Enum_String

### Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

### Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Standard Anybus CompactCom 40)
2... 11	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	<u>Value:</u> <u>Color:</u> 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.

## Extended

#	Name	Access	Type	Value
17	Virtual attributes	Get/Set	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
18	Black list/White list	Get/Set		
19	Network time	Get	UINT64	Value of the internal Real Time Clock (RTC). Can be based on the host application RTC or time synchronization from the network. Format is specified below.

## Network Time Format

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Year – 1900 (e.g year 2016 is represented as 116.)	Month (January = 1)	Day of month	Day of week (Monday = 1)	Hour (24 hour system)	Minutes of the hour	Seconds of the minute	Hundredths of the second

### 10.3 Diagnostic Object (02h)

Consult the Anybus CompactCom 40 Software Design Guide for information.

## 10.4 Network Object (03h)

### Category

Basic

### Object Description

For more information regarding this object, consult the general *Anybus CompactCom 40 Software Design Guide*.

### Supported Commands

<b>Object:</b>	Get_Attribute
<b>Instance:</b>	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area
	Map_ADI_Write_Ext_Area
	Map_ADI_Read_Ext_Area

### Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

## Instance Attributes (Instance #1)

### Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	009Ah
2	Network type string	Get	Array of CHAR	"BACnet/IP"
3	Data format	Get	ENUM	01h (MSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area. (Consult the general <i>Anybus CompactCom 40 Software Design Guide</i> for further information.)
6	Read process data size	Get	UINT16	0. (The Anybus CompactCom 40 BACnet/IP does not support read process data.)
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state.  <div> <div>Value:</div> <div>Meaning:</div> </div> 00h      No information available 01h      The Get_All_BACnet_Object_Instances service request for analog value objects failed. 02h      The Get_All_BACnet_Object_Instances service request for binary value objects failed. 03h      The Get_All_BACnet_Object_Instances service request for multistate value objects failed. 04h      An ADI mapped on process data could not be resolved as a BACnet object during start up



## 10.5 Network Configuration Object (04h)

### Category

Extended

### Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

If the settings in this object do not match the configuration used, the Module Status LED will flash red to indicate a minor error.

The object is described in further detail in the Anybus CompactCom 40 Software Design Guide.

As soon as the used combination of IP address, Subnet mask and Gateway is changed, the module informs the application by writing the new set to instance #1, attribute #16 in the Ethernet Host Object (F9h).

See also...

- [Communication Settings, p. 22](#)
- [E-mail Client, p. 41](#)
- [Ethernet Host Object \(F9h\), p. 129](#)

### Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Network Configuration"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0015h (21)
4	Highest instance number	Get	UINT16	0019h (25)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

### Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"IP address" (Multilingual, see page 97)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Subnet mask" (Multilingual, see page 97)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #5, Gateway Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Gateway" (Multilingual, see page 97)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

## Instance Attributes (Instance #6, DHCP Enable)

Value is used after module reset.

#	Name	Access	Data Type	Description									
1	Name	Get	Array of CHAR	“DHCP” (Multilingual, see page 97)									
2	Data type	Get	UINT8	08h (= ENUM)									
3	Number of elements	Get	UINT8	01h (one element)									
4	Descriptor	Get	UINT8	07h (read/write/shared access)									
5	Value	Get/Set	ENUM	Any change is valid after reset. (Multilingual, see page 97)  <table><tr><th><u>Value</u></th><th><u>String</u></th><th><u>Meaning</u></th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled (default)</td></tr></table>	<u>Value</u>	<u>String</u>	<u>Meaning</u>	00h	“Disable”	DHCP disabled	01h	“Enable”	DHCP enabled (default)
<u>Value</u>	<u>String</u>	<u>Meaning</u>											
00h	“Disable”	DHCP disabled											
01h	“Enable”	DHCP enabled (default)											
6	Value	Get/Set	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset.  <table><tr><th><u>Value</u></th><th><u>String</u></th><th><u>Meaning</u></th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled (default)</td></tr></table>	<u>Value</u>	<u>String</u>	<u>Meaning</u>	00h	“Disable”	DHCP disabled	01h	“Enable”	DHCP enabled (default)
<u>Value</u>	<u>String</u>	<u>Meaning</u>											
00h	“Disable”	DHCP disabled											
01h	“Enable”	DHCP enabled (default)											

## Instance Attributes (Instance #7 Ethernet Communication Settings 1)

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Comm 1” (Multilingual, see page 97)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<u>Value</u> <u>String</u> <u>Meaning</u> (Multilingual, see page 97)
				00h      “Auto”      Auto negotiation (default)
				01h      “10 HDX”      10Mbit, half duplex
				02h      “10 FX”      10Mbit, full duplex
				03h      “100HDX”      100Mbit, half duplex
				04h      “100FX”      100Mbit, full duplex
6	Value	Get/Set	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset.
				<u>Value</u> <u>String</u> <u>Meaning</u> (Multilingual, see page 97)
				00h      “Auto”      Auto negotiation (default)
				01h      “10 HDX”      10Mbit, half duplex
				02h      “10 FX”      10Mbit, full duplex
				03h      “100HDX”      100Mbit, half duplex
04h      “100FX”      100Mbit, full duplex				

## Instance Attributes (Instance #8 Ethernet Communication Settings 2)

Changes have immediate effect.

#	Name	Access	Data Type	Description																					
1	Name	Get	Array of CHAR	“Comm 2” (Multilingual, see page 97)																					
2	Data type	Get	UINT8	08h (= ENUM)																					
3	Number of elements	Get	UINT8	01h (one element)																					
4	Descriptor	Get	UINT8	07h (read/write/shared access)																					
5	Value	Get/Set	ENUM	<table><tr><th>Value</th><th>String</th><th>Meaning</th></tr><tr><td colspan="3">(Multilingual, see page 97)</td></tr><tr><td>00h</td><td>“Auto”</td><td>Auto negotiation (default)</td></tr><tr><td>01h</td><td>“10 HDX”</td><td>10Mbit, half duplex</td></tr><tr><td>02h</td><td>“10 FX”</td><td>10Mbit, full duplex</td></tr><tr><td>03h</td><td>“100HDX”</td><td>100Mbit, half duplex</td></tr><tr><td>04h</td><td>“100FX”</td><td>100Mbit, full duplex</td></tr></table>	Value	String	Meaning	(Multilingual, see page 97)			00h	“Auto”	Auto negotiation (default)	01h	“10 HDX”	10Mbit, half duplex	02h	“10 FX”	10Mbit, full duplex	03h	“100HDX”	100Mbit, half duplex	04h	“100FX”	100Mbit, full duplex
Value	String	Meaning																							
(Multilingual, see page 97)																									
00h	“Auto”	Auto negotiation (default)																							
01h	“10 HDX”	10Mbit, half duplex																							
02h	“10 FX”	10Mbit, full duplex																							
03h	“100HDX”	100Mbit, half duplex																							
04h	“100FX”	100Mbit, full duplex																							
6	Value	Get/Set	ENUM	<p>Holds the configured value, which will be written to attribute #5 after the module has been reset.</p> <table><tr><th>Value</th><th>String</th><th>Meaning</th></tr><tr><td colspan="3">(Multilingual, see page 97)</td></tr><tr><td>00h</td><td>“Auto”</td><td>Auto negotiation (default)</td></tr><tr><td>01h</td><td>“10 HDX”</td><td>10Mbit, half duplex</td></tr><tr><td>02h</td><td>“10 FX”</td><td>10Mbit, full duplex</td></tr><tr><td>03h</td><td>“100HDX”</td><td>100Mbit, half duplex</td></tr><tr><td>04h</td><td>“100FX”</td><td>100Mbit, full duplex</td></tr></table>	Value	String	Meaning	(Multilingual, see page 97)			00h	“Auto”	Auto negotiation (default)	01h	“10 HDX”	10Mbit, half duplex	02h	“10 FX”	10Mbit, full duplex	03h	“100HDX”	100Mbit, half duplex	04h	“100FX”	100Mbit, full duplex
Value	String	Meaning																							
(Multilingual, see page 97)																									
00h	“Auto”	Auto negotiation (default)																							
01h	“10 HDX”	10Mbit, half duplex																							
02h	“10 FX”	10Mbit, full duplex																							
03h	“100HDX”	100Mbit, half duplex																							
04h	“100FX”	100Mbit, full duplex																							

## Instance Attributes (Instance #9, DNS1)

This instance holds the address to the primary DNS server. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS1" (Multilingual, see page 97)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #10, DNS2)

This instance holds the address to the secondary DNS server. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS2" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #11, Host name)

This instance holds the host name of the module. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. Host name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 64 characters

### Instance Attributes (Instance #12, Domain name)

This instance holds the domain name. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h (48 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. Domain name, 48 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Domain name, 48 characters

### Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server address. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP server" (Multilingual, see page 97)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. SMTP server address, 64 characters.
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP server address, 64 characters.

### Instance Attributes (Instance #14, SMTP User)

This instance holds the user name for the SMTP account. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP user" (Multilingual, see page 97)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. SMTP account user name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account user name, 64 characters

### Instance Attributes (Instance #15, SMTP Password)

This instance holds the password for the SMTP account. Changes are valid after reset..

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP Pswd" (Multilingual, see page 97)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. SMTP account password, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account password, 64 characters

### Instance Attributes (Instance #16, MDI 1 Settings )

This instance holds the settings for MDI/MDIX 1. Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"MDI 1"
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<div> <div>Value (ENUM):</div> <div>00h</div> <div>01h</div> <div>02h</div> </div> <div> <div>String: Meaning:</div> <div>"Auto" (default)</div> <div>"MDI"</div> <div>"MDIX"</div> </div>
5	Value	Get/Set	ENUM	<div>Holds the configured value, which will be written to attribute #5 after the module has been reset.</div> <div> <div>Value (ENUM):</div> <div>00h</div> <div>01h</div> <div>02h</div> </div> <div> <div>String: Meaning:</div> <div>"Auto" (default)</div> <div>"MDI"</div> <div>"MDIX"</div> </div>

### Instance Attributes (Instance #17, MDI 2 Settings )

This instance holds the settings for MDI/MDIX 2. Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"MDI 2"
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<div> <div>Value (ENUM):</div> <div>00h</div> <div>01h</div> <div>02h</div> </div> <div> <div>String: Meaning:</div> <div>"Auto" (default)</div> <div>"MDI"</div> <div>"MDIX"</div> </div>
5	Value	Get/Set	ENUM	<div>Holds the configured value, which will be written to attribute #5 after the module has been reset.</div> <div> <div>Value (ENUM):</div> <div>00h</div> <div>01h</div> <div>02h</div> </div> <div> <div>String: Meaning:</div> <div>"Auto" (default)</div> <div>"MDI"</div> <div>"MDIX"</div> </div>

### Instance Attributes (Instances #18 and #19)

These instances are reserved for future attributes.

### Instance Attributes (Instance #20, Device Instance)

This instance maps to the Object\_Identifier property of the Device Object. Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Device Inst” (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	06h (= UINT32)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT32	Min = 0 Max = 3FFFFFFh Default = ( Module serial number & 3FFFFFFh) The value is stored in Non-Volatile storage
6	Configured Value	Get	UINT32	Configured value of attribute 5 stored for use in case of reset.

### Instance Attributes (Instance #21, UDP port)

This instance holds the settings for the UDP port. Valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“UDP port” (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	05h (= uint16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Min = 1, Max = 65535 Default = BAC0h The value is stored in Non-Volatile storage
6	Configured Value	Get	UINT16	Configured value of attribute 5 stored for use in case of reset.

### Instance Attributes (Instance #22, Process Active Timeout)

This instance specifies the Process Active Timeout. Changes take immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Process tmo” (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Default = 0 The value is stored in Non-Volatile storage
6	Configured Value	Get	UINT16	Since the value is used directly when changed this attribute always has the same value as attribute #5.



### Instance Attributes (Instance #23, Foreign Device Registration IP Address)

This instance gives the IP address for where the BBMD to register as foreign device. Changes take immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"FDR IP address" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements) Byte order: Example – 10.10.12.13 IP address byte #0: 10 IP address byte #1: 10 IP address byte #2 : 12 IP address byte #3: 13
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Valid range: 0.0.0.0 – 255.255.255.255 0.0.0.0 = Default The value is stored in Non-Volatile storage When written with a valid value the module will un-register with the current BBMD (if any) and send a Register-Foreign-Device request to the new BBMD.
6	Configured Value	Get	UINT8	Since the value is used directly when changed this attribute always has the same value as attribute #5.

### Instance Attributes (Instance #24, Foreign Device Registration UDP Port)

This instance gives the UDP port to use for foreign device registration. Changes take immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"FDR UDP port" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Min = 1 Max = 65535 Default = BAC0h The value is stored in Non-Volatile storage When written with a valid value the module will un-register with the current BBMD (if any) and send a Register-Foreign-Device request to the new BBMD. This will only be done if the value attribute of instance #23 is set to a valid value.
6	Configured Value	Get	UINT16	Since the value is used directly when changed this attribute always has the same value as attribute #5.

### Instance Attributes (Instance #25, Foreign Device Registration Time to Live Value)

This instance gives the foreign device registration time to live. Changes take immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"FDR TTL value" (Multilingual, see <a href="#">page 97</a> )
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Min = 0 sec Max = 65535 sec Default = 0 sec The value is stored in Non-Volatile storage When written with a valid value the module will send a Register-Foreign-Device with the new time to live value. This will only be done if the value attribute of instance 23 is set to a valid value.
6	Configured Value	Get	UINT16	Since the value is used directly when changed this attribute always has the same value as attribute #5.

## Multilingual Strings

The instance names and enumeration strings in this object are multilingual, and are translated based on the current language settings as follows:

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
7	Comm 1	Komm 1	Comu 1	Connessione 1	Comm 1
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
8	Comm 2	Komm 2	Comu 2	Connessione 2	Comm 2
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
9	DNS1	DNS 1	DNS Primaria	DNS1	DNS1
10	DNS2	DNS 2	DNS Secundia.	DNS2	DNS2
11	Host name	Host name	Nombre Host	Nome Host	Nom hôte
12	Domain name	Domain name	Nobre Domain	Nome Dominio	Dom Domaine
13	SMTP Server	SMTP Server	Servidor SMTP	Server SMTP	SMTP serveur
14	SMTP User	SMTP User	Usuario SMTP	Utente SMTP	SMTP utiliza.
15	SMTP Pswd	SMTP PSWD	Clave SMTP	Password SMTP	SMTP mt passe
20	Device inst	Geraetenummer	Istanc.Dispos	Dispositivo	Inst produit
21	UDP port	UDP Port	Puerto UDP	Porta UDP	Port UDP
22	Process tmo	Prozess Tmo	Tout Proceso	Tout Processo	Process tmo
23	FDR IP address	FDR IP-Adr.	Dir. IP FDR	Indir. IP FDR	Adresse IP FDR
24	FDR UDP port	FDR UDP-Port	Puerto UDP FDR	Porta UDP FDR	Port UDP FDR
25	FDR TTL value	FDR TTL-Wert	Valor FDR TTL	Val. TTL FDR	Valeur TTL FDR

## 10.6 Socket Interface Object (07h)

### Category

Extended

### Object Description

This object provides direct access to the TCP/IP stack socket interface, enabling custom protocols to be implemented over TCP/UDP.

Note that some of the commands used when accessing this object may require segmentation. A message will be segmented if the amount of data sent or received is larger than the message channel can handle. For more information, see [Message Segmentation, p. 113](#).



*The use of functionality provided by this object should only be attempted by users who are already familiar with socket interface programming and who fully understands the concepts involved in TCP/IP programming.*

### Supported Commands

<b>Object:</b>	Get_Attribute
	Create (See below)
	Delete (See below)
	DNS_Lookup (See below)
<b>Instance:</b>	Get_Attribute
	Set_Attribute
	Bind (See below)
	Shutdown (See below)
	Listen (See below)
	Accept (See below)
	Connect (See below)
	Receive (See below)
	Receive_From (See below)
	Send (See below)
	Send_To (See below)
	P_Add_membership (See below)
	IP_Drop_membership (See below)

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Socket interface"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	Number of opened sockets
4	Highest instance no.	Get	UINT16	Highest created instance number
11	Max. no. of instances	Get	UINT16	0008h (8 instances): BACnet/IP 0014h (20 instances): All other industrial Ethernet networks

## Instance Attributes (Sockets #1...Max. no. of instances)

### Extended

#	Name	Access	Data Type	Description
1	Socket Type	Get	UINT8	<p><u>Value:</u>    <u>Socket Type</u></p> <p>00h        SOCK_STREAM, NONBLOCKING (TCP)</p> <p>01h        SOCK_STREAM, BLOCKING (TCP)</p> <p>02h        SOCK_DGRAM, NONBLOCKING (UDP)</p> <p>03h        SOCK_DGRAM, BLOCKING (UDP)</p>
2	Port	Get	UINT16	Local port that the socket is bound to
3	Host IP	Get	UINT32	Host IP address, or 0 (zero) if not connected
4	Host port	Get	UINT16	Host port number, or 0 (zero) if not connected
5	TCP State	Get	UINT8	<p>State (TCP sockets only):</p> <p><u>Value</u>    <u>State/Description</u></p> <p>00h        CLOSED    Closed</p> <p>01h        LISTEN    Listening for connection</p> <p>02h        SYN_SENT   Active, have sent and received SYN</p> <p>03h        SYN_RECEIVED   Have sent and received SYN</p> <p>04h        ESTABLISHED   Established.</p> <p>05h        CLOSE_WAIT   Received FIN, waiting for close</p> <p>06h        FIN_WAIT_1   Have closed, sent FIN</p> <p>07h        CLOSING    Closed exchanged FIN; await FIN ACK</p> <p>08h        LAST_ACK    Have FIN and close; await FIN ACK</p> <p>09h        FIN_WAIT_2   Have closed, FIN is acknowledged</p> <p>Ah        TIME_WAIT   Quiet wait after close</p>
6	TCP RX bytes	Get	UINT16	Number of bytes in RX buffers (TCP sockets only)
7	TCP TX bytes	Get	UINT16	Number of bytes in TX buffers (TCP sockets only)
8	Reuse address	Get/Set	BOOL	<p>Socket can reuse local address</p> <p><u>Value</u>    <u>Meaning</u></p> <p>1        Enabled</p> <p>0        Disabled (default)</p>
9	Keep alive	Get/Set	BOOL	<p>Protocol probes idle connection (TCP sockets only).</p> <p>If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75s. The connection is terminated after 8 failed probe attempts.</p> <p><u>Value</u>    <u>Meaning</u></p> <p>1        Enabled</p> <p>0        Disabled (default)</p>
10	IP Multicast TTL	Get/Set	UINT8	<p>IP Multicast TTL value (UDP sockets only).</p> <p>Default = 1.</p>
11	IP Multicast Loop	Get/Set	BOOL	<p>IP multicast loop back (UDP sockets only)</p> <p>Must belong to group in order to get the loop backed message</p> <p><u>Value</u>    <u>Meaning</u></p> <p>1        Enabled (default)</p> <p>0        Disabled</p>
12	(reserved)			
13	TCP No Delay	Get/Set	BOOL	<p>Don't delay send to coalesce packets (TCP).</p> <p><u>Value</u>    <u>Meaning</u></p> <p>1        Delay (default)</p> <p>0        Don't delay (turn off Nagle's algorithm on socket)</p>
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)

## Command Details: Create

### Category

Extended

### Details

<b>Command Code</b>	03h
<b>Valid for:</b>	Object Instance

### Description

This command creates a socket.

This command is only allowed in WAIT\_PROCESS, IDLE and PROCESS\_ACTIVE states.

- Command Details

Field	Contents										
CmdExt[0]	(reserved, set to zero)										
CmdExt[1]	<table><tr><td><u>Value:</u></td><td><u>Socket Type:</u></td></tr><tr><td>00h</td><td>SOCK_STREAM, NON-BLOCKING (TCP)</td></tr><tr><td>01h</td><td>SOCK_STREAM, BLOCKING (TCP)</td></tr><tr><td>02h</td><td>SOCK_DGRAM, NON-BLOCKING (UDP)</td></tr><tr><td>03h</td><td>SOCK_DGRAM, BLOCKING (UDP)</td></tr></table>	<u>Value:</u>	<u>Socket Type:</u>	00h	SOCK_STREAM, NON-BLOCKING (TCP)	01h	SOCK_STREAM, BLOCKING (TCP)	02h	SOCK_DGRAM, NON-BLOCKING (UDP)	03h	SOCK_DGRAM, BLOCKING (UDP)
<u>Value:</u>	<u>Socket Type:</u>										
00h	SOCK_STREAM, NON-BLOCKING (TCP)										
01h	SOCK_STREAM, BLOCKING (TCP)										
02h	SOCK_DGRAM, NON-BLOCKING (UDP)										
03h	SOCK_DGRAM, BLOCKING (UDP)										

- Response Details

Field	Contents	Comments
Data[0]	Instance number (low)	Instance number of the created socket.
Data[1]	Instance number (high)	

## Command Details: Delete

### Category

Extended

### Details

**Command Code** 04h  
**Valid for:** Object Instance

### Description

This command deletes a previously created socket and closes the connection (if connected).

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the 'Shutdown'-command (see below) before deleting the socket, causing the connection to be closed with the FIN-flag instead.
- Command Details

Field	Contents	Comments
CmdExt[0]	Instance number to delete (low)	Instance number of socket that shall be deleted.
CmdExt[1]	Instance number to delete (high)	

- Response Details  
(no data)

## Command Details: Bind

### Category

Extended

### Details

**Command Code** 10h  
**Valid for:** Instance

### Description

This command binds a socket to a local port.

- Command Details

Field	Contents	Comments
CmdExt[0]	Requested port number (low)	Set to 0 (zero) to request binding to any free port.
CmdExt[1]	Requested port number (high)	

- Response Details

Field	Contents	Comments
CmdExt[0]	Bound port number (low)	Actual port that the socket was bound to.
CmdExt[1]	Bound port number (high)	

## Command Details: Shutdown

### Category

Extended

### Details

Command Code	11h
Valid for:	Instance

### Description

This command closes a TCP-connection using the FIN-flag. Note that the response does not indicate if the connection actually shut down, which means that this command cannot be used to poll non-blocking sockets, nor will it block for blocking sockets.

- Command Details

Field	Contents		
CmdExt[0]	(reserved, set to zero)		
CmdExt[1]	<table><tr><td><u>Value:</u> 00h 01h 02h</td><td><u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel</td></tr></table>	<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel
<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel		

- Response Details

(no data)

The recommended sequence to gracefully shut down a TCP connection is described below.

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.
2. Receive data on socket until error message Object specific error (EPIPE (13)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.
3. Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

Host initiates shutdown:

1. Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.
2. Try to send any unsent data to the host.
3. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel.
4. Delete the socket instance.



## Command Details: Listen

### Category

Extended

### Details

<b>Command Code</b>	12h
<b>Valid for:</b>	Instance

### Description

This command puts a TCP socket in listening state.

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	(reserved)

- Response Details  
(no data)

## Command Details: Accept

### Category

Extended

### Details

<b>Command Code</b>	13h
<b>Valid for:</b>	Instance

### Description

This command accepts incoming connections on a listening TCP socket. A new socket instance is created for each accepted connection. The new socket is connected with the host and the response returns its instance number.

<b>NONBLOCKING mode</b>	This command must be issued repeatedly (polled) for incoming connections. If no incoming connection request exists, the module will respond with error code 0006h (EWOULDBLOCK).
<b>BLOCKING mode</b>	This command will block until a connection request has been detected.

This command will only be accepted if there is a free instance to use for accepted connections. For blocking connections, this command will reserve an instance.

- Command Details  
(no data)
- Response Details

Field	Contents
Data[0]	Instance number for the connected socket (low byte)
Data[1]	Instance number for the connected socket (high byte)
Data[2]	Host IP address byte 4
Data[3]	Host IP address byte 3
Data[4]	Host IP address byte 2
Data[5]	Host IP address byte 1
Data[6]	Host port number (low byte)
Data[7]	Host port number (high byte)

## Command Details: Connect

### Category

Extended

### Details

<b>Command Code</b>	14h
<b>Valid for:</b>	Instance

### Description

For SOCK\_DGRAM-sockets, this command specifies the peer with which the socket is to be associated (to which datagrams are sent and the only address from which datagrams are received).

For SOCK\_STREAM-sockets, this command attempts to establish a connection to a host.

SOCK\_STREAM-sockets may connect successfully only once, while SOCK\_DGRAM-sockets may use this service multiple times to change their association. SOCK\_DGRAM-sockets may dissolve their association by connecting to IP address 0.0.0.0, port 0 (zero).

<b>NON-BLOCKING mode:</b>	This command must be issued repeatedly (polled) until a connection is connected, rejected or timed out. The first connect-attempt will be accepted, thereafter the command will return error code 22 (EINPROGRESS) on poll requests while attempting to connect.
<b>BLOCKING mode:</b>	This command will block until a connection has been established or the connection request is cancelled due to a timeout or a connection error.

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Host IP address byte 4
Data[1]	Host IP address byte 3
Data[2]	Host IP address byte 2
Data[3]	Host IP address byte 1
Data[4]	Host port number (low byte)
Data[5]	Host port number (high byte)

- Response Details

(no data)

## Command Details: Receive

### Category

Extended

### Details

Command Code	15h
Valid for:	Instance

### Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (for more information, see [Message Segmentation, p. 113](#)).

For SOCK-DGRAM-sockets, the module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

For SOCK\_STREAM-sockets, the module will return the requested number of bytes from the received data stream. If the actual data size is less than requested, all available data will be returned.

<b>NON-BLOCKING mode:</b>	If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.
<b>BLOCKING mode:</b>	The module will not issue a response until the operation has finished.

If the module responds successfully with 0 (zero) bytes of data, it means that the host has closed the connection. The send channel may however still be valid and must be closed using **Shutdown** and/or **Delete**.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see <a href="#">Message Segmentation, p. 113</a>
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 113](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see <a href="#">Message Segmentation, p. 113</a>
Data[0...n]	Received data	-

## Command Details: Receive\_From

### Category

Extended

### Details

<b>Command Code</b>	16h
<b>Valid for:</b>	Instance

### Description

This command receives data from an unconnected SOCK\_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (For more information, see [Message Segmentation, p. 113](#)).

The module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

The response message contains the IP address and port number of the sender.

<b>NON-BLOCKING mode:</b>	If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.
<b>BLOCKING mode:</b>	The module will not issue a response until the operation has finished.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see <a href="#">Message Segmentation, p. 113</a>
Data[0]	Receive data size (low byte)	Only used in the first segment
Data[1]	Receive data size (high byte)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 113](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see <a href="#">Message Segmentation, p. 113</a>
Data[0]	Host IP address byte 4	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Received data	

## Command Details: Send

### Category

Extended

### Details

<b>Command Code</b>	17h
<b>Valid for:</b>	Instance

### Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (For more information, see [Message Segmentation, p. 113](#)).

**NON-BLOCKING mode:** If there isn't enough buffer space available in the send buffers, the module will respond with error code 0006h (EWOULDBLOCK)

**BLOCKING mode:** If there isn't enough buffer space available in the send buffers, the module will block until there is.

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 113](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see <a href="#">Message Segmentation, p. 113</a> )
Data[0...n]	Data to send	-

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
Data[0]	Number of sent bytes (low)	Only valid in the last segment
Data[1]	Number of sent bytes (high)	

## Command Details: Send\_To

### Category

Extended

### Details

Command Code	18h
Valid for:	Instance

### Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (For more information, see appendix For more information, see [Message Segmentation, p. 113](#)).

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 113](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see <a href="#">Message Segmentation, p. 113</a>
Data[0]	Host IP address byte 4	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Data to send	

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
Data[0]	Number of sent bytes (low byte)	Only valid in the last segment
Data[1]	Number of sent bytes (high byte)	

## Command Details: IP\_Add\_Membership

### Category

Extended

### Details

<b>Command Code</b>	19h
<b>Valid for:</b>	Instance

### Description

This command assigns the socket an IP multicast group membership. The module always joins the “All hosts group” automatically, however this command may be used to specify up to 20 additional memberships.

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details  
(no data)

## Command Details: IP\_Drop\_Membership

### Category

Extended

### Details

<b>Command Code</b>	1Ah
<b>Valid for:</b>	Instance



## Description

This command removes the socket from an IP multicast group membership.

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details

(no data)

## Command Details: DNS\_Lookup

### Category

Extended

### Details

Command Code	1Bh
Valid for:	Object

## Description

This command resolves the given host name and returns the IP address.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0... N]	Host name	Host name to resolve

- Response Details (Success)

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	IP address byte 4	IP address of the specified host
Data[1]	IP address byte 3	
Data[2]	IP address byte 2	
Data[3]	IP address byte 1	

## Socket Interface Error Codes (Object Specific)

The following object-specific error codes may be returned by the module when using the socket interface object.

Error Code	Name	Meaning
1	ENOBUFS	No internal buffers available
2	ETIMEDOUT	A timeout event occurred
3	EISCONN	Socket already connected
4	EOPNOTSUPP	Service not supported
5	ECONNABORTED	Connection was aborted
6	EWOULDBLOCK	Socket cannot block because unblocking socket type
7	ECONNREFUSED	Connection refused
8	ECONNRESET	Connection reset
9	ENOTCONN	Socket is not connected
10	EALREADY	Socket is already in requested mode
11	EINVAL	Invalid service data
12	EMSGSIZE	Invalid message size
13	EPIPE	Error in pipe
14	EDESTADDRREQ	Destination address required
15	ESHUTDOWN	Socket has already been shutdown
16	(reserved)	-
17	EHAVEOOB	Out of band data available
18	ENOMEM	No internal memory available
19	EADDRNOTAVAIL	Address is not available
20	EADDRINUSE	Address already in use
21	(reserved)	-
22	EINPROGRESS	Service already in progress
28	ETOOMANYREFS	Too many references
101	Command aborted	If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command.
102	DNS name error	Failed to resolve the host name (name error response from DNS server.
103	DNS timeout	Timeout when performing a DNS lookup.
104	DNS command failed	Other DNS error.

## Message Segmentation

### General

**Category:** Extended

The maximum message size supported by the Anybus CompactCom 40 is normally 1524 bytes. In some applications a maximum message size of 255 bytes is supported, e.g. if an Anybus CompactCom 40 is to replace an Anybus CompactCom 30 without any changes to the application. The maximum socket message size is 1472. To ensure support for socket interface messages larger than 255 bytes a segmentation protocol is used.



*The segmentation bits have to be set for all socket interface messages, in the commands where segmentation can be used, whether the messages have to be segmented or not.*

The segmentation protocol is implemented in the message layer and must not be confused with the fragmentation protocol used on the serial host interface. Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.

The module supports 1 (one) segmented message per instance

### Command Segmentation

When a command message is segmented, the command initiator sends the same command header multiple times. For each message, the data field is exchanged with the next data segment.

Command segmentation is used for the following commands (Socket Interface Object specific commands):

- Send
- Send To

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to the message channel size), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.
- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

### Segmentation Control Bits (Command)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	Set to 0 (zero)

### Segmentation Control Bits (Response)

Bit	Contents	Meaning
0... 7	(reserved)	Ignore

### Response Segmentation

When a response is segmented, the command initiator requests the next segment by sending the same command multiple times. For each response, the data field is exchanged with the next data segment.

Response segmentation is used for responses to the following commands (Socket Interface Object specific commands):

- Receive
- Receive From

When receiving a segmented response, the following rules apply:

- In the first segment, FS is set.
- In all subsequent segment, both FS and LS are cleared.
- In the last segment, LS is set.
- For single segment responses (i.e. size less or equal to the message channel size), both FS and LS are set.
- The command initiator may at any time abort the operation by issuing a message with AB set.

### Segmentation Control bits (Command)

Bit	Contents	Meaning
0	(reserved)	(set to zero)
1		
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	Set to 0 (zero)

### Segmentation Control bits (Response)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2...7	(reserved)	Set to 0 (zero)

## 10.7 SMTP Client Object (09h)

### Category

Extended

### Object Description

This object groups functions related to the SMTP client.

### Supported Commands

<b>Object:</b>	Get_Attribute
	Create
	Delete
	Send e-mail from file (see below)
<b>Instance:</b>	Get_Attribute
	Set_Attribute
	Send e-mail (see below)

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"SMTP Client"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0006h
12	Success count	Get	UINT16	Reflects the no. of successfully sent messages
13	Error count	Get	UINT16	Reflects the no. of messages that could not be delivered

### Instance Attributes (Instance #1)

Instances are created dynamically by the application.

#	Name	Access	Data Type	Description
1	From	Get/Set	Array of CHAR	e.g. "someone@somewhere.com"
2	To	Get/Set	Array of CHAR	e.g. "someone.else@anywhere.net"
3	Subject	Get/Set	Array of CHAR	e.g. "Important notice"
4	Message	Get/Set	Array of CHAR	e.g. "Shut down the system"

## Command Details: Create

### Category

Extended

### Details

**Command Code** 03h  
**Valid for:** Object

### Description

This command creates an e-mail instance.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
Data[0]	Instance number	low byte
Data[1]		high byte

## Command Details: Delete

### Category

Extended

### Details

<b>Command Code</b>	04h
<b>Valid for:</b>	Object

### Description

This command deletes an e-mail instance.

- Command Details

Field	Contents	Comments
CmdExt[0]	E-mail instance number	low byte
CmdExt[1]		high byte

- Response Details  
(no data)

Command Details: Send E-mail From File

Category

Extended

Details

Command Code	11h
Valid for:	Object

Description

This command sends an e-mail based on a file in the file system.

The file must be a plain ASCII-file in the following format:

```
[To]
recipient

[From]
sender

[Subject]
email subject

Se [Headers]
extra headers, optional

[Message]
actual email message
```

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0... n]	Path + filename of message file

- Response Details

(no data)



## Command Details: Send E-mail

### Category

Extended

### Details

Command Code	10h
Valid for:	Instance

### Description

This command sends the specified e-mail instance.

- Command Details  
(no data)
- Response Details  
(no data)

## Object Specific Error Codes

Error Codes	Meaning
1	SMTP server not found
2	SMTP server not ready
3	Authentication error
4	SMTP socket error
5	SSI scan error
6	Unable to interpret e-mail file
255	Unspecified SMTP error
(other)	(reserved)

## 10.8 File System Interface Object (0Ah)

### Category

Extended

### Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations.

This object is thoroughly described in *Anybus CompactCom 40 Software Design Guide*.

## 10.9 Network Ethernet Object (0Ch)

### Category

Extended

### Object Description

This object provides Ethernet-specific information to the application.

### Supported Commands

**Object:** Get\_Attribute

**Instance:** Get\_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Network Ethernet"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-

### Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also "Ethernet Host Object (F9h)"

# 11 Host Application Objects

## 11.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may be implemented within the host application firmware to expand the BACnet/IP implementation.

### Standard Objects

- “Application File System Object (EAh)” (see Anybus CompactCom 40 Software Design Guide)
- “Application Object (FFh)” (see Anybus CompactCom 40 Software Design Guide)
- “Application Data Object (FEh)” (see Anybus CompactCom 40 Software Design Guide)

### Network Specific Objects:

- [BACnet Host Object \(EFh\), p. 123](#)
- [Ethernet Host Object \(F9h\), p. 129](#)

## 11.2 BACnet Host Object (EFh)

### Object Description

This object implements BACnet specific features in the host application. If attribute #7 (Support Advanced Mapping) is enabled, the application can define the ADI mapping of the module to suit the application.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The module will attempt to retrieve the values of these attributes during startup; if an attribute is not implemented in the host application, simply respond with an error message (06h, "Invalid CmdExt[0]"). In such cases, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, "Invalid CmdExt[0]").

See also...

- Anybus CompactCom 40 Software Design Guide, "Error Codes"
- [Device Object, p. 11](#)

### Supported Commands

<b>Object:</b>	Get_Attribute
	Get_ADI_By_BACnet_Object_Instance
	Get_ADI_By_BACnet_Name
	Get_All_BACnet_Object_Instances
	Get_BACnet_Object_Instance_By_ADI
<b>Instance:</b>	Get_Attribute
	Set_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"BACnet"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

**Instance Attributes (Instance #1)**

#	Name	Access	Data Type	Default Value	Comment
1	Object Name	Get/Set	Array of CHAR	"CompactCom 40 BACnet/IP"	Changes the Object_Name property of the BACnet Device Object Max 64 bytes. Both Get and Set are optional. If Set is implemented Get is required.
2	Vendor Name	Get	Array of CHAR	"HMS Industrial Networks"	Changes the Vendor_Name property of the BACnet Device Object Max 64 bytes. Both Get and Set are optional. If Set is implemented Get is required.
3	Vendor Identifier	Get	UINT16	486	Changes the Vendor_Identifier property of the BACnet Device Object.
4	Model Name	Get	Array of CHAR	"CompactCom 40 BACnet/IP"	Changes the Model_Name property of the BACnet Device Object and the product name, displayed on the web pages and in HICP responses.
5	Firmware Revision	Get	Array of CHAR	The modules firmware revision	Changes the Firmware_Revision property of the BACnet Device Object Max 16 bytes.
6	Application_Software_Version	Get	Array of CHAR	The modules firmware revision	Changes the Application_Software_Version property of the BACnet Device Object Max 16 bytes.
7	Support advanced mapping	Get	BOOL	False	If true, the application supports advanced BACnet to ADI mapping schema and must support all related commands.
8	Current date and time	Get/Set	Struct of:  UINT16 UINT8 UINT8 UINT8 UINT8 UINT8	  0 0 0 0 0 0	Sanity checks will be done when this value is read. If invalid values are detected the whole struct will be set to 0. If the Anybus CompactCom 40 BACnet/IP receives a date/time update from the network it will write updated date/time to this attribute. Current year Current month Current day Current hour Current minute Current second
9	Password	Get	Array of CHAR	"Admin"	Password used for the ReinitializeDevice and DeviceCommunicationControl commands. Max 20 bytes.

## Command Details: Get\_ADI\_By\_BACnet\_Object\_Instance

### Category

Extended

### Details

<b>Command Code</b>	10h
<b>Valid for:</b>	Object

### Description

By setting the attribute Support advanced mapping (#7), all requests to BACnet data objects will be translated to ADIs using this service.

The service is used to translate from BACnet addressing to Anybus CompactCom addressing. Request to supported BACnet object classes are forwarded to the application, so that it can return the corresponding ADI.

For information about BACnet object classes, see [BACnet/IP Implementation, p. 10](#).

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	BACnet Object	BACnet object class
MsgData[2... 5]	BACnet Instance	-

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	ADI	ADI that corresponds to the requested BACnet Object Instance

## Command Details: Get\_ADI\_By\_BACnet\_Name

### Category

Extended

### Details

<b>Command Code</b>	11h
<b>Valid for:</b>	Object

### Description

By setting the attribute Support advanced mapping (#7), all requests to BACnet data objects by name will be translated to ADIs using this service.

The service is used to translate from BACnet addressing to Anybus CompactCom addressing. Request to supported BACnet object classes are forwarded to the application, so that it can return the corresponding ADI.

For information about BACnet object classes, see [BACnet/IP Implementation, p. 10](#).

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0... N]	BACnet Object_Name	This field holds a string containing the BACnet object name

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	ADI	ADI that corresponds to the requested BACnet Object Name
MsgData[2,3]	BACnet object class	The BACnet object class that corresponds to the BACnet Object_Name requested
MsgData[4... 7]	BACnet Instance	The BACnet Instance that corresponds to the BACnet Object_Name requested.



## Command Details: Get\_All\_BACnet\_Object\_Instances

### Category

Extended

### Details

<b>Command Code</b>	12h
<b>Valid for:</b>	Object

### Description

If the attribute Support advanced mapping (#7) is set, the Object\_List attribute in the Device Object will be populated during initialization using this service. The response returns a bit array of 2040 entries. A set bit indicates that the corresponding instance is implemented within the application.

The example in the table below, shows the first two bytes in the application response. Instances 1, 4, 10, 12, and 13 are implemented in the application.

Byte	Value
0	12h (0001 0010b)
1	34h (0011 0100b)

For information about BACnet object classes, see [BACnet/IP Implementation, p. 10](#).

By setting the attribute Support advanced mapping (#7), this service must be implemented. When the mapping of write process data has been performed, it is not possible for the module to know which ADI corresponds to which BACnet object identifier. This service finds that information by translating from an ADI to a BACnet object identifier.

For information about BACnet object classes, see [BACnet/IP Implementation, p. 10](#).

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	BACnet Object	BACnet object class

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0... 254]	Object List	Bit array indicating implemented BACnet objects corresponding to the requested BACnet object class.

## Command Details: Get\_BACnet\_Object\_Instance\_By\_ADI

### Category

Extended

### Details

<b>Command Code</b>	13h
<b>Valid for:</b>	Object

### Description

If the attribute Support advanced mapping (#7) is set, this service must be implemented. When the mapping of write process data has been performed, it is not possible for the module to know which ADI corresponds to which BACnet object identifier. This service finds that information by translating from an ADI to a BACnet object identifier.

For information about BACnet object classes, see [BACnet/IP Implementation, p. 10](#).

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	ADI	The ADI for which the application wants to find the BACnet object identifier.

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	BACnet object class	The BACnet object class used for the ADI supplied in the command. (UINT16)
MsgData[2... 5]	BACnet Instance	The BACnet instance corresponding to the ADI supplied in the command. (UINT32)

## 11.3 Ethernet Host Object (F9h)

### Object Description

This object implements Ethernet features in the host application.

### Supported Commands

**Object:** Get\_Attribute

**Instance:** Get\_Attribute

Set\_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Ethernet"
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

### Instance Attributes (Instance #1)

- If an attribute is not implemented, the default value will be used.
- The module is preprogrammed with a valid MAC address. To use that address, do not implement attribute #1.
- Do not implement attributes #9 and #10, only used for PROFINET devices, if the module shall use the preprogrammed MAC addresses.
- If new MAC addresses are assigned to a PROFINET device, these addresses (in attributes #1, #9, and #10) have to be consecutive, e.g. (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03) with the first five octets not changing.

#	Name	Access	Data Type	Default Value	Comment
1	MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Note that the new Mac address value must be obtained from the IEEE. Do not implement this attribute if the preprogrammed Mac address is to be used.
2	Enable HICP	Get	BOOL	True (Enabled)	Enable/Disable HICP
3	Enable Web Server	Get	BOOL	True (Enabled)	Enable/Disable Web Server (Not used if Transparent Ethernet is enabled.)
4	(reserved)				Reserved for Anybus CompactCom 30 applications.
5	Enable Web ADI access	Get	BOOL	True (Enabled)	Enable/Disable Web ADI access (Not used if Transparent Ethernet is enabled.)
6	Enable FTP server	Get	BOOL	True (Enabled)	Enable/Disable FTP server (Not used if Transparent Ethernet is enabled.)
7	Enable admin mode	Get	BOOL	False (Disabled)	Enable/Disable FTP admin mode (Not used if Transparent Ethernet is enabled.)
8	Network Status	Set	UINT16	-	See below.

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	<b>Note:</b> This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
10	Port 2 MAC address	Get	Array of UINT8	-	<b>Note:</b> This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
11	Enable ACD	Get	BOOL	True (Enabled)	Enable/Disable ACD protocol. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.
12	Port 1 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 1. <ul style="list-style-type: none"> <li>This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 1 is always enabled.</li> <li>This attribute is not used by PROFINET and Ethernet POWERLINK</li> </ul> 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.
13	Port 2 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 2. <ul style="list-style-type: none"> <li>This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 2 is always enabled.</li> <li>This attribute is not used by PROFINET</li> </ul> 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website. 02h: Inactive. The attribute is set to this value for a device that only has one physical port. All two-port functionality is disabled. No references can be made to this port. <b>Note:</b> This functionality is available for Ethernet/IP and Modbus-TCP devices.
14	(reserved)				
15	Enable reset from HICP	Get	BOOL	0 = False	Enables the option to reset the module from HICP.
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	Whenever the configuration is assigned or changed, the Anybus CompactCom module will update this attribute.

#	Name	Access	Data Type	Default Value	Comment
17	IP address byte 0–2	Get	Array of UINT8 [3]	[0] = 192 [1] = 168 [2] = 0	First three bytes in IP address. Used in standalone shift register mode if the configuration switch value is set to 1-245. In that case the IP address will be set to: Y[0].Y[1].Y[2].X Where Y0-2 is configured by this attribute and the last byte X by the configuration switch.
18	Ethernet PHY Configuration	Get	Array of BITS16	0x0000 for each port	Ethernet PHY configuration bit field. The length of the array shall equal the number of Ethernet ports of the product. Each element represents the configuration of one Ethernet port (element #0 maps to Ethernet port #1, element #1 maps to Ethernet port #2 and so on). <b>Note:</b> Only valid for EtherNet/IP and Modbus-TCP devices.  Bit 0:        Auto negotiation fallback duplex 0 = Half duplex 1 = Full duplex  Bit 1–15:    Reserved
20	SNMP read-only community string	Get	Array of CHAR	“public”	<b>Note:</b> This attribute is only valid for PROFINET devices. Sets the SNMP read-only community string. Max length is 32.
21	SNMP read-write community string	Get	Array of CHAR	“private”	<b>Note:</b> This attribute is only valid for PROFINET devices. Sets the SNMP read-write community string. Max length is 32.
22	DHCP Option 61 source	Get	ENUM	0 (Disabled)	<b>Note:</b> This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
23	DHCP Option 61 generic string	Get	Array of UINT8	N/A	<b>Note:</b> This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
24	Enable DHCP Client	Get	BOOL	1 = True	<b>Note:</b> This attribute is currently valid for Ethernet/IP and PROFINET devices. Enable/disable DHCP Client functionality 0:            DHCP Client functionality is disabled 1:            DHCP Client functionality is enabled

## Network Status

This attribute holds a bit field which indicates the overall network status as follows:

Bit	Contents	Description	Comment
0	Link	Current global link status 1= Link sensed 0= No link	
1	IP established	1 = IP address established 0 = IP address not established	
2	(reserved)	(mask off and ignore)	
3	Link port 1	Current link status for port 1 1 = Link sensed 0 = No link	EtherCAT only: This link status indicates whether the Anybus CompactCom is able to communicate using Ethernet over EtherCAT (EoE) or not. That is, it indicates the status of the logical EoE port link and is not related to the link status on the physical EtherCAT ports.
4	Link port 2	Current link status for port 2 1 = Link sensed 0 = No link	Not used for EtherCAT
5... 15	(reserved)	(mask off and ignore)	

## DHCP Option 61 (Client Identifier)



*Only valid for EtherNet/IP devices*

The DHCP Option 61 (Client Identifier) allow the end-user to specify a unique identifier, which has to be unique within the DHCP domain.

Attribute #22 (DHCP Option 61 source) is used to configure the source of the Client Identifier. The table below shows the definition for the Client identifier for different sources and their description.

Value	Source	Description
0	Disable	The DHCP Option 61 is disabled. This is the default value if the attribute is not implemented in the application.
1	MACID	The MACID will be used as the Client Identifier
2	Host Name	The configured Host Name will be used as the Client Identifier
3	Generic String	Attribute #23 will be used as the Client Identifier

Attribute #23 (DHCP Option 61 generic string) is used to set the Client Identifier when Attribute #22 has been set to 3 (Generic String). Attribute #23 contains the Type field and Client Identifier and shall comply with the definitions in RFC 2132. The allowed max length that can be passed to the module via attribute #23 is 64 octets.

Example:

If Attribute #22 has been set to 3 (Generic String) and Attribute #23 contains 0x01, 0x00, 0x30, 0x11, 0x33, 0x44, 0x55, the Client Identifier will be represented as an Ethernet Media Type with MACID 00:30:11:33:44:55.

Example 2:

If Attribute #22 has been set to 2 (Host Name) Attribute #23 will be ignored and the Client Identifier will be the same as the configured Host Name.

## **A Categorization of Functionality**

The objects, including attributes and services, of the Anybus CompactCom and the application are divided into two categories: basic and extended.

### **A.1 Basic**

This category includes objects, attributes and services that are mandatory to implement or to use. They will be enough for starting up the Anybus CompactCom and sending/receiving data with the chosen network protocol. The basic functions of the industrial network are used.

Additional objects etc, that will make it possible to certify the product also belong to this category.

### **A.2 Extended**

Use of the objects in this category extends the functionality of the application. Access is given to the more specific characteristics of the industrial network, not only the basic moving of data to and from the network. Extra value is given to the application.

Some of the functionality offered may be specialized and/or seldom used. As most of the available network functionality is enabled and accessible, access to the specification of the industrial network may be required.

## B Implementation Details

### B.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of BACnet/IP, this means that the SUP-bit is set when the time (ms) elapsed since the last BACnet request is less than the value of the parameter “Process Active Timeout”, if this parameter value is greater than zero.

### B.2 Anybus State Machine

The table below describes how the Anybus state machine relates to the BACnet/IP network

Anybus State	Implementation	Comment
WAIT_PROCESS	The module stays in this state until a BACnet request arrives.	-
ERROR	IP conflict	-
PROCESS_ACTIVE	BACnet request(s) addressed to this module have been received within the last “Process Active Timeout” time.	<ul style="list-style-type: none"> <li>If no process active timeout value is specified (i.e. the parameter is set to 0), the module will remain in this state after the first BACnet request has been received.</li> <li>The supervised bit is set when the module is in this state.</li> </ul>
IDLE	N/A	-
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

### B.3 Application Watchdog Timeout Handling

Upon detection of an application watchdog timeout, the module will cease network participation and shift to state EXCEPTION. No other network specific actions are performed.

### B.4 Implemented BACnet BIBBs

The Anybus CompactCom 40 BACnet/IP is implemented as a BACnet Application Specific Controller (B-ASC). To make the module eligible for certification as a B-ASC, the following BIBBs (BACnet Interoperability Building Blocks) are implemented:

BIBB	Corresponding BACnet Service(s)
Data Sharing-ReadProperty-B (DS-RP-B)	ReadProperty (Execute)
Data Sharing-ReadPropertyMultiple-B (DS-RPM-B)	ReadPropertyMultiple (Execute)
Data Sharing-WriteProperty-B (DS-WP-B)	WriteProperty (Execute)
Data Sharing-WritePropertyMultiple-B (DS-WPM-B)	WritePropertyMultiple (Execute)
Data Sharing-COV-B (DS-COV-B)	SubscribeCOV (Execute) ConfirmedCOVNotification (Initiate) UnConfirmedCOVNotification (Initiate)
Alarm and Event-Notification Internal-B (AE-N-I-B)	ConfirmedEventNotification (Initiate) UnConfirmedEventNotification (Initiate)
Alarm and Event-ACK-B (AE-ACK-B)	AcknowledgeAlarm (Execute)
Alarm and Event-Alarm Summary-B (AE-ASUM-B)	GetAlarmSummary (Execute)
Alarm and Event-Information-B (AE-INFO-B)	GetEventInformation (Execute)
Device Management-Dynamic Device Binding-A (DM-DDB-A)	Who-Is (Initiate)



<b>BIBB</b>	<b>Corresponding BACnet Service(s)</b>
	I-Am (Execute)
Device Management-Dynamic Device Binding-B (DM-DDB-B)	Who-Is (Execute) I-Am (Initiate)
Device Management-Dynamic Object Binding-B (DM-DDB-B)	Who-Has (Execute) I-Have (Initiate)
Device Management-Device Communication Control-B (DM-DCC-B)	DeviceCommunicationControl (Execute)
Device Management-TimeSynchronization-B (DM-TS-B)	TimeSynchronization (Execute)
Device Management-ReinitializeDevice-B (DM-RD-B)	ReinitializeDevice (Execute)

## C Secure HICP (Secure Host IP Configuration Protocol)

### C.1 General

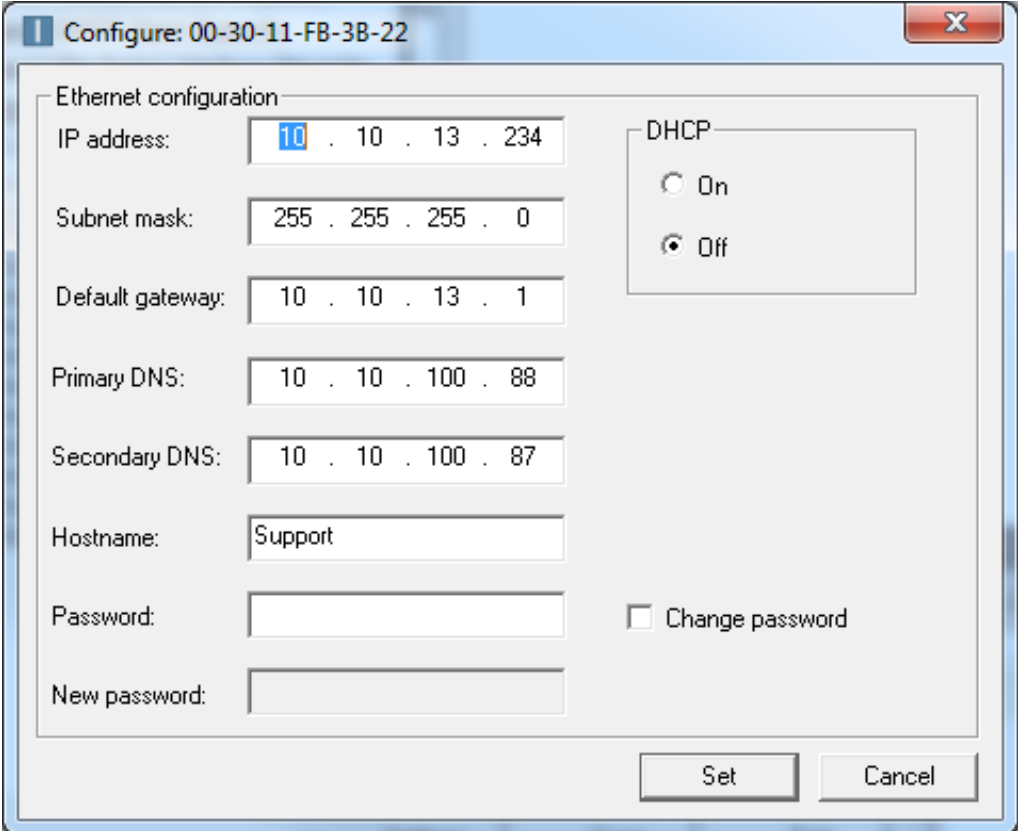
The Anybus CompactCom 40 BACnet/IP supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the HMS website, [www.anybus.com](http://www.anybus.com). This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

The protocol offers secure authentication and the ability to restart/reboot the device(s).

### C.2 Operation

When the application is started, the network is automatically scanned for Anybus products. The network can be rescanned at any time by clicking **Scan**.

To alter the network settings of a module, double-click on its entry in the list. A window will appear, containing the settings for the module.



The screenshot shows a configuration window titled "Configure: 00-30-11-FB-3B-22". It contains the following fields and controls:

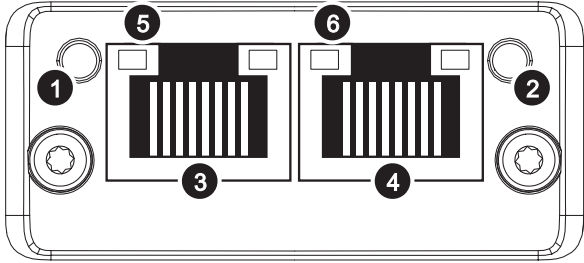
- Ethernet configuration** (grouped box):
  - IP address: 10 . 10 . 13 . 234
  - Subnet mask: 255 . 255 . 255 . 0
  - Default gateway: 10 . 10 . 13 . 1
  - Primary DNS: 10 . 10 . 100 . 88
  - Secondary DNS: 10 . 10 . 100 . 87
  - Hostname: Support
  - Password: (empty text field)
  - New password: (empty text field)
- DHCP** (grouped box):
  - ☐ On
  - ☒ Off
- ☐ Change password
- Set** button
- Cancel** button

Fig. 7

Validate the new settings by clicking **Set**, or click **Cancel** to cancel all changes. Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, check the **Change password** checkbox and enter the password in the **New password** text field.

## D Technical Specification

### D.1 Front View

#	Item	
1	Network Status LED	
2	Module Status LED	
3	Ethernet Interface, Port 1	
4	Ethernet Interface, Port 2	
5	Link/Activity Port 1	
6	Link/Activity Port 2	

#### D.1.1 Network Status LED

LED State	Indication/Description
Off	No power or no IP address
Green	<ul style="list-style-type: none"> <li>On-line, one or more BACnet messages have arrived</li> <li>Module has active COV subscriptions</li> <li>At least one value object has one or more events enabled</li> </ul>
Green, flashing	On-line, waiting for first BACnet message.
Red	Duplicate IP address, FATAL error
Red, flashing	<ul style="list-style-type: none"> <li>Connection timeout. No BACnet message has been received within the configured "process active timeout" time.</li> <li>A COV or Alarm/Event notification could not be sent to its recipient.</li> </ul>

A test sequence is performed on this LED during startup.

#### D.1.2 Module Status LED

LED State	Indication/Description
Off	No power
Green	Normal operation
Red/green, alternating	Firmware update from file system in progress
Red	Major fault (state EXCEPTION, FATAL error etc.)
Red, flashing	Recoverable fault(s)

A test sequence is performed on this LED during startup.

#### D.1.3 Link/Activity LED 5/6

LED State	Indication/Description
Off	No link, no activity
Green	Link (100 Mbit/s) established
Green, flickering	Activity (100 Mbit/s)
Yellow	Link (10 Mbit/s) established
Yellow, flickering	Activity (10 Mbit/s)

#### D.1.4 Ethernet Interface

The Ethernet interface supports autonegotiation and Auto MDI-X, with 10/100Mbit, full or half duplex operation.

### D.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the Anybus CompactCom 40 BACnet/IP must be properly connected to functional earth via the FE pad / FE mechanism described in the general *Anybus CompactCom M40 Hardware Design Guide*. If the brick version is used, please make sure that the hardware is properly connected to FE.

HMS Industrial Networks does not guarantee proper EMC behavior unless these FE requirements are fulfilled.

### D.3 Power Supply

#### D.3.1 Supply Voltage

The module requires a regulated 3.3V power source as specified in the general *Anybus CompactCom M40 Hardware Design Guide*.

#### D.3.2 Power Consumption

The Anybus CompactCom 40 BACnet/IP is designed to fulfil the requirements of a Class B module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general *Anybus CompactCom M40 Hardware Design Guide*.

The current hardware design consumes up to 380 mA (RMS).



*It is strongly advised to design the power supply in the host application based on the power consumption classifications described in the general Anybus CompactCom M40 Hardware Design Guide, and not on the exact power requirements of a single product.*

*In line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification.*

---

### D.4 Environmental Specification

Consult the Anybus CompactCom M40 Hardware Design Guide for further information.

### D.5 EMC Compliance

Consult the Anybus CompactCom M40 Hardware Design Guide for further information.

## E Backward Compatibility

The Anybus CompactCom M40 series of industrial network modules have significantly better performance and include more functionality than the modules in the Anybus CompactCom 30 series. The 40 series is backward compatible with the 30 series in that an application developed for the 30 series should be possible to use with the 40 series, without any major changes. Also it is possible to mix 30 and 40 series modules in the same application.

This appendix presents the backwards compatibility issues that have to be considered for Anybus CompactCom 40 BACnet/IP, when designing with both series in one application, or when adapting a 30 series application for the 40 series.

### E.1 Initial Considerations

There are two options to consider when starting the work to modify a host application developed for Anybus CompactCom 30-series modules to also be compatible with the 40-series modules:

- Add support with as little work as possible i.e. reuse as much as possible of the current design.
  - This is the fastest and easiest solution but with the drawback that many of the new features available in the 40-series will not be enabled (e.g. enhanced and faster communication interfaces, larger memory areas, and faster communication protocols).
  - You have to check the hardware and software differences below to make sure the host application is compatible with the 40-series modules. Small modifications to your current design may be needed.
- Make a redesign and take advantage of all new features presented in the 40-series.
  - A new driver and host application example code are available at [www.anybus.com/starterkit40](http://www.anybus.com/starterkit40) to support the new communication protocol. This driver supports both 30-series and 40-series modules.
  - You have to check the hardware differences below and make sure the host application is compatible with the 40-series modules.



*This information only deals with differences between the 30-series and the 40-series.*

---

Link to support page: [www.anybus.com/support](http://www.anybus.com/support).

### E.2 Hardware Compatibility

Anybus CompactCom is available in three hardware formats; Module, Chip, and Brick.

#### E.2.1 Module

The modules in the 30-series and the 40-series share physical characteristics, like dimensions, outline, connectors, LED indicators, mounting parts etc. They are also available as modules without housing.



**Fig. 8**      **Anybus CompactCom M30/M40**

## E.2.2      **Chip**

The chip (C30/C40) versions of the Anybus CompactCom differ completely when it comes to physical dimensions.



There is no way to migrate a chip solution from the 30-series to the 40-series without a major hardware update.

### E.2.3 Brick

The Anybus CompactCom B40-1 does not share dimensions with the Anybus CompactCom B30. The B40-1 is thus not suitable for migration. However HMS Industrial Networks has developed a separate brick version in the 40-series, that can be used for migration. This product, B40-2, shares dimensions etc. with the B30. Please contact HMS Industrial Networks for more information on the Anybus CompactCom B40-2.

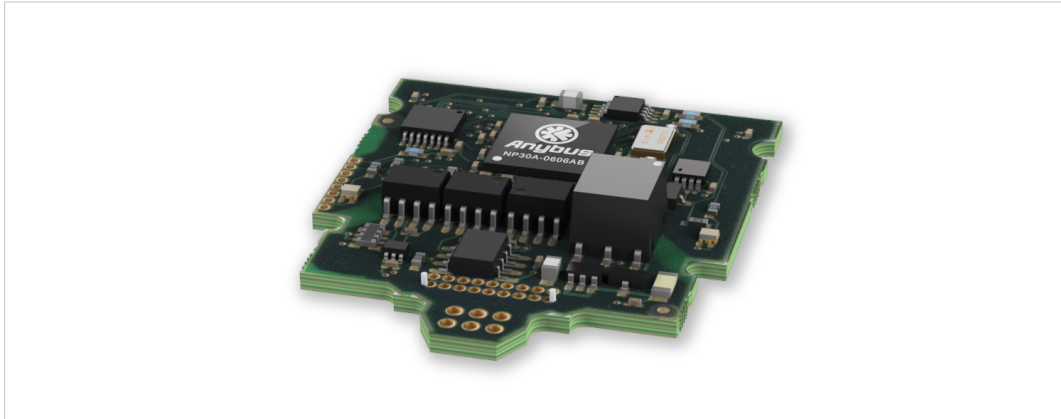


Fig. 9 Anybus CompactCom B30

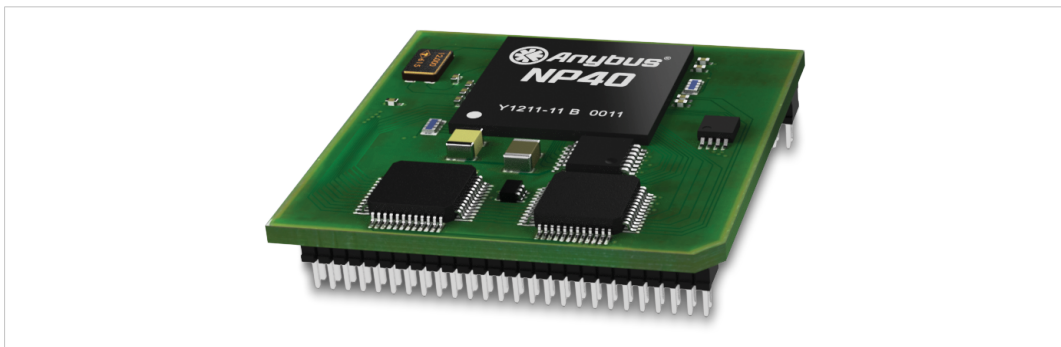


Fig. 10 Anybus CompactCom B40-1 (not for migration)

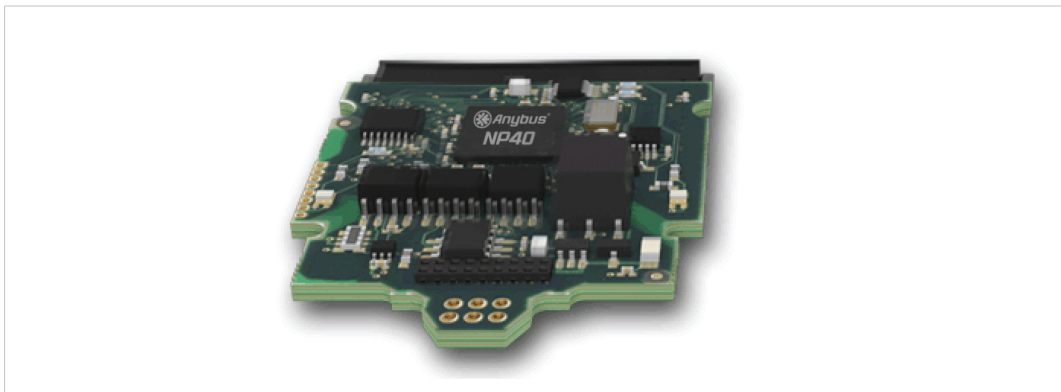


Fig. 11 Anybus CompactCom B40-2





**GIP[0..1]/LED3[A..B]**

These pins are tri-stated inputs by default in the 30-series. In the 40-series, these pins are tri-stated until the state NW\_INIT. After that they become open-drain, active low LED outputs (LED3A/LED3B).

No modification of the hardware is needed, if your current design has

- tied these pins to GND
- pulled up the pins
- pulled down the pins
- left the pins unconnected

However, if the application drive the pins high, a short circuit will occur.

If you connect the pins to LEDs, a pull-up is required.

In the 40-series, there is a possibility to set the GIP[0..1] and GOP[0..1] in high impedance state (tri-state) by using attribute #16 (GPIO configuration) in the Anybus object (01h). I.e. if it is not possible to change the host application hardware, this attribute can be configured for high impedance state of GIP and GOP before leaving NW\_INIT state.

Related Information: *Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)*, Section “LED Interface/D8-D15 (Data Bus)”.

**GOP[0..1]/LED4[A..B]**

These pins are outputs (high state) by default in the 30-series. In the 40-series, these pins are tri-stated until the state NW\_INIT, and after that they become push-pull, active low LED outputs (LED4A/LED4B).

This change should not affect your product.

Related Information: *Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)*, Section 3.2.3, “LED Interface/D8-D15 (Data Bus)”.

**Address Pins A[11..13]**

The address pins 11, 12, and 13 are ignored by the 30-series. These pins must be high when accessing the 40-series module in backwards compatible 8-bit parallel mode. If you have left these pins unconnected or connected to GND, you need to make a hardware modification to tie them high.

**Max Input Signal Level ( $V_{IH}$ )**

The max input signal level for the 30-series is specified as  $V_{IH}=V_{DD}+0,2\text{ V}$ , and for the 40-series as  $V_{IH}=3.45\text{ V}$ . Make sure that you do not exceed 3.45 V for a logic high level.

**RMII Compatibility**

If the RMII mode is being used on an Anybus CompactCom 40 module and it is desired to remain compatible with the 30 series, it is important to disable this connection when switching to an Anybus CompactCom 30 module due to pin conflicts. The RMII port of the host processor should be set to tristate by default, and only be enabled if an RMII capable Anybus CompactCom 40 is detected. In case the RMII connection cannot be disabled through an internal hardware control on the host processor, it will be necessary to design in external hardware (i.e. a FET bus switch) to prevent short circuits

Related Information: *Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)*, Section 3.2.5, "RMII — Reduced Media-Independent Interface".

## E.3 General Software

### E.3.1 Extended Memory Areas

The memory areas have been extended in the 40-series, and it is now possible to access larger sizes of process data (up to 4096 bytes instead of former maximum 256 bytes) and message data (up to 1524 bytes instead of former maximum 255 bytes). The 30-series has reserved memory ranges that the application should not use. The 40-series implements new functionality in some of these memory areas.



*To use the extended memory areas you need to implement a new communication protocol which is not part of this document.*

*Memory areas not supported by the specific network cannot be used. Make sure you do not access these areas, e.g. for doing read/write memory tests.*

Related Information: *Anybus CompactCom 40 Software Design Guide (HMSI-216-125)*, Section “Memory Map”

### E.3.2 Faster Ping-Pong Protocol

The ping-pong protocol (the protocol used in the 30-series) is faster in the 40-series. A 30-series module typically responds to a so called ping within 10-100 µs. The 40-series typically responds to a ping within 2 µs.

Interrupt-driven applications (parallel operating mode) may see increased CPU load due to the increased speed.

### E.3.3 Requests from Anybus CompactCom to Host Application During Startup

All requests to software objects in the host application must be handled and responded to (even if the object does not exist). This applies for both the 30-series and the 40-series. The 40-series introduces additional objects for new functionality.

There may also be additional commands in existing objects added to the 40-series that must be responded to (even if it is not supported).

If your implementation already responds to all commands it cannot process, which is the expected behavior, you do not need to change anything.

### E.3.4 Anybus Object (01h)

Attribute	30-series	40-series	Change/Action/Comment
#1, Module Type	0401h	0403h	Make sure the host application accepts the new module type value for the 40-series.
#15, Auxiliary Bit	Available	Removed	It is not possible to turn off the “Changed Data Indication” in the 40-series. Also see “Control Register CTRL_AUX-bit” and “Status Register STAT_AUX-bit” below.
#16, GPIO Configuration	Default: General input and output pins	Default: LED3 and LED4 outputs	See also .. <ul style="list-style-type: none"> <li><a href="#">GIP[0..1]/LED3[A..B], p. 143</a></li> <li><a href="#">GOP[0..1]/LED4[A..B], p. 143</a></li> </ul>

### E.3.5 Control Register CTRL\_AUX-bit

**30-series** The CTRL\_AUX bit in the control register indicates to the Anybus CompactCom if the process data in the current telegram has changed compared to the previous one.

**40-series** The value of the CTRL\_AUX bit is always ignored. Process data is always accepted.

All released Anybus CompactCom 30 example drivers from Anybus CompactCom comply with this difference.

Related Information: *Anybus CompactCom 40 Software Design Guide (HMSI-216-125)*, section “Control Register”.

### E.3.6 Status Register STAT\_AUX-bit

**30-series** The STAT\_AUX bit in the status register indicates if the output process data in the current telegram has changed compared to the previous one. This functionality must be enabled in the Anybus object (01h), Attribute #15. By default, the STAT\_AUX bit functionality is disabled.

**40-series** The STAT\_AUX bit indicates updated output process data (not necessarily changed data) from the network compared to the previous telegram. The functionality is always enabled.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

Related Information: *Anybus CompactCom 40 Software Design Guide (HMSI-216-125)*, section “Status Register”.

### E.3.7 Control Register CTRL\_R-bit

**30-series** The application may change this bit at any time.

**40-series** For the 8-bit parallel operating mode, the bit is only allowed to transition from 1 to 0 when the STAT\_M-bit is set in the status register. When using the serial operating modes, it is also allowed to transition from 1 to 0 in the telegram immediately after the finalizing empty fragment.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

Related Information: *Anybus CompactCom 40 Software Design Guide (HMSI-216-125)*, section “Control Register”.

### E.3.8 Modifications of Status Register, Process Data Read Area, and Message Data Read Area

In the 40-series, the Status Register, the Process Data Read Area, and the Message Data Read Area are write protected in hardware (parallel interface). If the software for some reason writes to any of those areas, a change is needed.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

## E.4 Network Specific — BACnet/IP

### E.4.1 Network Configuration Object (04h)

The instances in the Network Configuration Object have been rearranged for the Ethernet based modules for consistency. Network specific instances are moved to instance number 20 and onwards. This is done to increase the number of instances in the section that is not network specific.

If the host application is using any of the parameters below, the software must be updated to use the new instance numbers.

Parameter Name	30-series Instance #	40-series Instance #
Device Instance	3	20
UDP Port	4	21
Process Active Timeout	5	22
IP Address	6	3
Subnet Mask	7	4
Gateway Address	8	5
DHCP Enable	9	6
Comm 1 Settings	10	7
Comm 2 Settings	11	8
DNS1	12	9
DNS2	13	10
Host Name	14	11
Domain Name	15	12
SMTP Server	16	13
SMTP User	17	14
SMTP Password	18	15
Foreign Device Registration IP	19	23
Foreign Device Registration UDP Port	20	24
Foreign Device Registration Time to Live Value	21	25

### E.4.2 Reduced Network Resources Due to Memory Constraints

The Anybus CompactCom 40 BACnet/IP will have reduced network resources compared to the Anybus CompactCom 30 due to memory constraints.

Network Resource	30-series	40-series
Maximum size of BACnet NPDU	1476	1024
Maximum number of active server requests	10	5
Number of supported COV server subscriptions	60	60
Maximum number of Network Configuration object recipients supported	60	18
Number of client requests	120	78
Number of supported Network Configuration events	256	64
Maximum size of APDU service payload with segmentation included	32 kB	5 kB
Number of BACnet objects (advanced mode )	6120	768
Number of BAPL DeviceAddressBindings supported	18	18

## F Copyright Notices

This product includes software developed by Carnegie Mellon, the Massachusetts Institute of Technology, the University of California, and RSA Data Security:

\*\*\*\*\*

Copyright 1986 by Carnegie Mellon.

\*\*\*\*\*

Copyright 1983,1984,1985 by the Massachusetts Institute of Technology

\*\*\*\*\*

Copyright (c) 1988 Stephen Deering.

Copyright (c) 1982, 1985, 1986, 1992, 1993

The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Stephen Deering of Stanford University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*\*\*\*\*

Copyright (C) 1990-2, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

\*\*\*\*\*

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

