Network Interface Appendix

Anybus® CompactCom BACnet/IP w. IT Functionality 2-Port

Doc.Id. HMSI-168-70
Rev. 1.21

Connecting Devices™

HALMSTAD • CHICAGO • KARLSRUHE • TOKYO • BEIJING • MILANO • MULHOUSE • COVENTRY • PUNE • COPENHAGEN

HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

Important User Information

This document is intended to provide a good understanding of the functionality offered by BACnet/IP w. IT. The document only describes the features that are specific to the Anybus CompactCom BACnet/IP w. IT Functionality 2-Port. For general information regarding the 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 use of advanced BACnet/IP w. IT-specific functionality may require in-depth knowledge in BACnet/IP w. IT networking internals and/or information from the official BACnet/IP w. IT specifications. In such cases, the people responsible for the implementation of this product should either obtain the BACnet/IP w. IT specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Liability

Every care has been taken in the preparation of this manual. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, 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 AB. HMS Industrial Networks AB 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 AB 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 AB cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks AB 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 US and other countries.

Trademark Acknowledgements

Anybus ® is a registered trademark of HMS Industrial Networks AB. All other trademarks are the property of their respective holders.

Warning:	This is a class A product. in a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
ESD Note:	This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product.

Anybus CompactCom BACnet/IP w. IT Functionality 2-Port Network Interface Appendix
Rev 1.21
Copyright© HMS Industrial Networks AB
Apr 2013 Doc Id HMSI-168-70

Table of Contents

P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

P.1 Related Documents

Document	Author
Anybus-CompactCom Software Design Guide	HMS
Anybus-CompactCom Hardware Design Guide	HMS
Anybus-CompactCom Software Driver User Guide	HMS
BACnet specification, Doc. Id. 135, 2008	ASHRAE

P.2 Document History

Summary of Recent Changes (1.20 ..... 1.21)

Change	Page(s)
Updated attribute 4 in BACnet Host Object	113
Updated description of HICP	122

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2011-04-18	KeL	All	First release
1.10	2011-06-21	KeL	1, 3, 9, B	Minor corrections and updates
1.11	2011-08-08	KaD	6	Minor addition
1.12	2011-10-26	KeL	3, 9, 10	Minor updates
1.13	2012-04-20	KeL	1	Minor correction
1.20	2012-09-10	KeL	3, 10, E	Update and corrections
1.21	2013-04-04	KeL	10, C	Minor updates

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps.
- Bulleted lists provide information, not procedural steps.
- The terms ‘Anybus’ or ‘module’ refers to the Anybus-CompactCom module.
- The terms ‘host’ or ‘host application’ refers to the device that hosts the Anybus module.
- Hexadecimal values are written in the format NNNNh, where NNNN is the hexadecimal value.

P.4 Sales and Support

Sales		Support	
HMS Sweden (Head Office)			
E-mail:	sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 56	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS North America			
E-mail:	us-sales@hms-networks.com	E-mail:	us-support@hms-networks.com
Phone:	+1-312 - 829 - 0601	Phone:	+1-312-829-0601
Toll Free:	+1-888-8-Anybus	Toll Free:	+1-888-8-Anybus
Fax:	+1-312-629-2869	Fax:	+1-312-629-2869
Online:	www.anybus.com	Online:	www.anybus.com
HMS Germany			
E-mail:	ge-sales@hms-networks.com	E-mail:	ge-support@hms-networks.com
Phone:	+49 (0) 721-989777-000	Phone:	+49 (0) 721-989777-000
Fax:	+49 (0) 721-989777-010	Fax:	+49 (0) 721-989777-010
Online:	www.anybus.de	Online:	www.anybus.de
HMS Japan			
E-mail:	jp-sales@hms-networks.com	E-mail:	jp-support@hms-networks.com
Phone:	+81 (0) 45-478-5340	Phone:	+81 (0) 45-478-5340
Fax:	+81 (0) 45-476-0315	Fax:	+81 (0) 45-476-0315
Online:	www.anybus.jp	Online:	www.anybus.jp
HMS China			
E-mail:	cn-sales@hms-networks.com	E-mail:	cn-support@hms-networks.com
Phone:	+86 (0) 10-8532-3183	Phone:	+86 (0) 10-8532-3023
Fax:	+86 (0) 10-8532-3209	Fax:	+86 (0) 10-8532-3209
Online:	www.anybus.cn	Online:	www.anybus.cn
HMS Italy			
E-mail:	it-sales@hms-networks.com	E-mail:	it-support@hms-networks.com
Phone:	+39 039 59662 27	Phone:	+39 039 59662 27
Fax:	+39 039 59662 31	Fax:	+39 039 59662 31
Online:	www.anybus.it	Online:	www.anybus.it
HMS France			
E-mail:	fr-sales@hms-networks.com	E-mail:	fr-support@hms-networks.com
Phone:	+33 (0) 3 68 368 034	Phone:	+33 (0) 3 68 368 033
Fax:	+33 (0) 3 68 368 031	Fax:	+33 (0) 3 68 368 031
Online:	www.anybus.fr	Online:	www.anybus.fr
HMS UK & Eire			
E-mail:	uk-sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+44 (0) 1926 405599	Phone:	+46 (0) 35 - 17 29 20
Fax:	+44 (0) 1926 405522	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.co.uk	Online:	www.anybus.com
HMS Denmark			
E-mail:	dk-sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+45 (0) 35 38 29 00	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS India			
E-mail:	in-sales@hms-networks.com	E-mail:	in-support@hms-networks.com
Phone:	+91 (0) 20 40111201	Phone:	+91 (0) 20 40111201
Fax:	+91 (0) 20 40111105	Fax:	+91 (0) 20 40111105
Online:	www.anybus.com	Online:	www.anybus.com

1. About the Anybus CompactCom BACnet/IP w. IT Functionality 2-Port

1.1 General

The Anybus-CompactCom BACnet/IP 2-port 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 offered by the module, allowing seamless network integration regardless of network type. The module supports linear network topology.

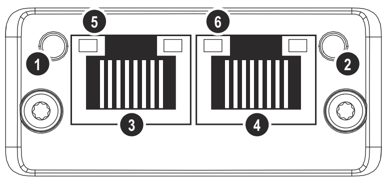
Furthermore, the BACnet Object can be customized, allowing the end product to appear as a vendor-specific implementation rather than a generic Anybus module.

This product conforms to all aspects of the host interface for Active modules defined in the Anybus-CompactCom 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.

1.2 Features

- Fulfills all requirements for a BACnet/IP device
- Two BACnet/IP ports
- Data sharing
- Linear network topology supported
- 10/100Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- Customizable Identity Information
- 256 ADIs available in simple mode for mapping to BACnet objects
- A total of 2040 ADIs available in advanced mode for mapping to BACnet objects
- Transparent Socket Interface
- Change Of Value (COV) notification and Alarm/Event functionality supported (max 256 ADIs available)
- Support for Foreign Device Registration functionality

1.3 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	

Network Status LED

Note: A test sequence is performed on this LED during startup.

LED State	Description
Off	No power or no IP address
Green	• On-line, one or more BACnet messages have arrived • Module has active COV subscriptions • At least one value object has one or more events enabled
Green, flashing	On-line, waiting for first BACnet message.
Red	Duplicate IP address, FATAL error
Red, flashing	• Connection timeout. No BACnet message has been received within the configured 'process active timeout' time. • A COV or Alarm/Event notification could not be sent to its recipient.

Module Status LED

Note: A test sequence is performed on this LED during startup.

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

Link/Activity LED 5/6

LED State	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)

Ethernet Interface

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

2. Tutorial

2.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The tutorial describes and explains a simple example of an implementation with Anybus CompactCom. This chapter includes network specific settings that are needed for a host application to be up and running and possible to test for use on BACnet/IP w. IT networks.

2.2 Implementation

It is recommended to enable attribute 7 (“Support advanced mapping”) to fully take advantage of the functionality and flexibility of the module.

See also ...

- “Application Data (ADIs)” on page 20
- “Mapping of BACnet Objects to Anybus CompactCom” on page 21
- “BACnet Host Object (EFh)” on page 112

2.3 Fieldbus Conformance Notes

- BACnet International does not require a certification for the use of BACnet products.
- The Anybus-CompactCom 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 BACnet/IP w. IT Functionality 2-Port.

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” on page 121 for a complete list of BACnet BIBBs implemented in the module.

3. Basic Operation

3.1 General Information

3.1.1 Software Requirements

Generally, no additional network support code needs to be written in order to support the Anybus-CompactCom 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 and SINT64 cannot be represented on BACnet.
- It is not possible to map read process data.

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

See also...

- Anybus-CompactCom Software Design Guide, ‘Application Data Object (FEh)’
- “Network Object (03h)” on page 61

3.2 Device Customization

3.2.1 Network Identity

By default, the module uses the following identity settings:

• Vendor Name:	"HMS Industrial Networks"
• Vendor ID:	01E6h (HMS Industrial Networks)
• Model Name:	"Anybus CompactCom"
• Object Name:	"Controller"
• Network Type:	009Ah ("BACnet/IP")
• Product Name:	'Anybus-CompactCom 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)" on page 112 (Host Application Object)
- "Network Object (03h)" on page 61 (Anybus Module Object)

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" on page 22
- "FTP Server" on page 31
- "Web Server" on page 33

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)" on page 73 (Anybus Module Object)
- "Message Segmentation" on page 102

3.3 BACnet/IP Implementation

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

Object Name	Class	Described on page
Device object	8	9
Analog Value object	2	11
Binary Value object	5	13
Multi-state Value object	19	14
Notification Class object	15	16

Each Anybus CompactCom BACnet/IP module 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" on page 20
- Application Data Object (see Anybus-CompactCom Software Design Guide)
- "Implemented BACnet BIBBs" on page 121

3.3.1 Device Object

The 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.

Property Identifier	Value	R/W	NV[a]	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[b]	"Controller"	R/W*		*This property can be written to only if the Object Name in the BACnet Host object can be set.
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[b]	"HMS Industrial Networks"	R		
Vendor_Identifier[b]	486	R		HMS Industrial Networks
Model_Name[b]	""Anybus-CompactCom"	R		
Firmware_Revision[b]	N/A	R		Firmware revision of the Anybus CompactCom BACnet/IP as a string.
Application_Software_Revision	N/A	R		
Protocol_Version	1	R		
Protocol_Revision	7	R		

Property Identifier	Value	R/W	NV[a]	Description/Comment
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 “Supported BACnet Services” on page 17.
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	1476	R		
Segmentation_Supported	SUPPORTED_BOTH	R		Segmentation supported for both Rx and Tx APDU.
Max_Segments_Accepted	22	R		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[a]	10000 (default) Valid range: 0 - 65535	R/W	NV	APDU transaction timeout (ms).[c]
Number_Of_APDU_Retries[a]	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[a]	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[a]	5000 (default) Valid range: 0 - 65535	R/W	NV	APDU segment timeout (ms).[c]
Active_COV_Subscriptions		R		Populated based on active COV subscription (max. 60) entries in AE module. Registered with SubscribeCOV service.

a. Properties that are stored in non volatile memory, keep their assigned values when the module is turned off.
b. This property can be changed by setting the corresponding attribute in the BACnet Host object, see page 112.
c. The value 0 is only valid if Number_Of_APDU_Retries = 0.

3.3.2 Analog Value Object

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

See also ...

- "Mapping of BACnet Objects to Anybus CompactCom" on page 21

Property Identifier	Value (default)	R/W	NV[a]	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R		In simple mode: Analog_Value_# (# = instance number) In advanced mode: Corresponding ADI name 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[a]	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value.
Time_Delay[a]	0	R/W	NV	Time delay for an event to be triggered after occur-rence (s) Min. value: 0 Max. value: UINT32max/1000
Notification_Class[a]	0	R/W	NV	Min. value: 0 Max. value: 5
High_Limit[a]	0	R/W	NV	Min. value: Min. value of the ADI's data type Max. value: Max. value of hte ADI's data type
Low_Limit[a]	0	R/W	NV	
Deadband[a]	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value
Limit_Enable[a]	F, F	R/W	NV	Bit string that determines what TO event limits are ena-bled Bit 0: LOW_LIMIT_ENABLE Bit 1: HIGH_LIMIT_ENABLE
Event_Enable[a]	F, F, F	R/W	NV	Bit string that determines what TO events that are ena-bled Bit 0: TO-OFFNORMAL Bit 1: not used. Set to FALSE. Bit 2: TO-NORMAL

Property Identifier	Value (default)	R/W	NV[a]	Description/Comment
Notify_Type[a]	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT
Acked_Transitions[b]	T, T, T	R		Bit string that determines what TO events has been acknowledged by a BACnet recipient. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps[b]	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL

a. 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 "Communication Settings" on page 19 and "Alarm/ Event Functionality" on page 24.
b. Only available if the corresponding ADI is mapped on the write process data channel.

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.

See also ...

- "Mapping of BACnet Objects to Anybus CompactCom" on page 21

Property Identifier	Value (default)	R/W	NV[a]	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R		In simple mode: Binary_Value_# (# = instance number) In advanced mode: Corresponding ADI name 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[a]	0	R/W	NV	Time delay for an event to be triggered after occur-rence (s)
Notification_Class[a]	0	R/W	NV	Min. value: 0 Max. value: 5
Alarm_Value[a]	INACTIVE (0)	R/W	NV	
Event_Enable[a]	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[a]	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT
Acked_Transitions[b]	T, T, T	R		Bit string that determines what TO events have been acknowledged by a BACnet recipient. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps[b]	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL

a. 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 "Communication Settings" on page 19 and "Alarm/ Event Functionality" on page 24.

b. Only available if the corresponding ADI is mapped on the write process data channel.

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.

See also ...

- "Mapping of BACnet Objects to Anybus CompactCom" on page 21

Property Identifier	Value	R/W	NV[a]	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R		In simple mode: Multistate_Value_# (# = instance number) In advanced mode: Corresponding ADI name 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[a]	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[a]	Empty list	R/W	NV	
Fault_Values[a]	Empty list	R/W	NV	
Event_Enable[a]	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[a]	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[a]	Description/Comment
Acked_Transitions[b]	T, T, T	R		Bit string that determines the TO events that have been acknowledged by a BACnet recipient. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps[b]	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
State_Text	N/A	R		This property is mapped against the module's Get_Enum_String service
Reliability[b]	NO_FAULT_DETECTE D	R		Object reliability 0: NO_FAULT_DETECTED 9: MULTI_STATE_FAULT

a. 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 "Communication Settings" on page 19 and "Alarm/ Event Functionality" on page 24.
b. Only available if the corresponding ADI is mapped on the write process data channel.

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[1] and returned to the network. 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.

1. The ENUM ADI is zero based in the Anybus CompactCom, but the first state of a multistate value object on BACnet is one.

3.3.5 Notification Class Object

An Anybus CompactCom BACnet/IP module 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.

See also ...

- "COV Notifications, Alarms and Events" on page 24

Property Identifier	Value	R/W	NV[a]	Description/Comment
Object_Identifier	N/A	R		
Object_Name[a]	Default: Notification_Class_# (# = instance number)	RW	NV	BACnet Char string (only ANSIX34 supported). Max. 30 characters
Object_Type	NOTIFICATION_CLASS	R		
Notification_Class	N/A	R		Equal to instance number
Priority[a]	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[a]	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[a]	N/A	R/W	NV	BACnet list of BACnetDestination primitives. There are 60 list entries available to be configured among the 6 Notification Class instances.
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.

a. Non volatile. The default values of these properties 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.

The Recipient_List specifies one or more recipients that a notification event should be sent to. In total there are 60 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 BACnet/IP w. IT Functionality 2-Port.

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 proper-ties 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 AcknowlegeAlarm 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...

- **UDP**

  All data to and from the module is transported via UDP

- **TCP/IP settings**

  These settings must be set properly in order for the module to be able to participate on the network.

  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.

- **Physical Link Settings**

  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" on page 33
- "Network Configuration Object (04h)" on page 63 (Anybus Module Object)
- "HICP (Host IP Configuration Protocol)" on page 122

3.5 Diagnostics

Major unrecoverable faults are reported in the Diagnostic Object

See also...

- "Diagnostic Object (02h)" on page 64 (Anybus Module Object)

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 2040 analog value objects, 2040 binary value objects and 2040 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...

- "Simple Mapping" on page 21
- "Advanced Mapping" on page 21
- "BACnet Host Object (EFh)" on page 112

3.6.2 Translation of Data Types

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

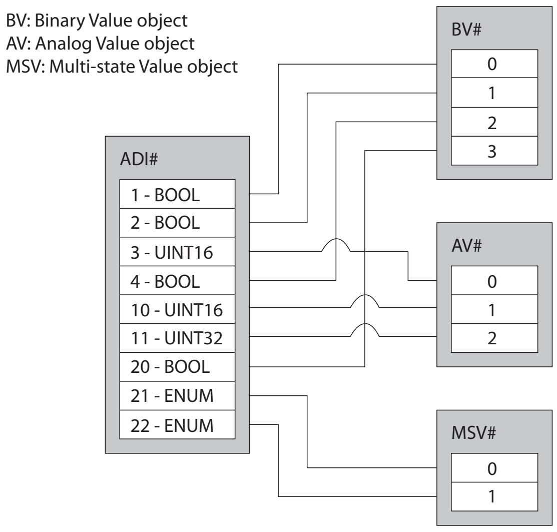
Anybus Data Type	BACnet Object Name	Comments
BOOL	Binary Value Object	Each ADI element of this type occupies one byte.
ENUM	Multi-state Value Object	
SINT8	Analog Value Object	
UINT8	Analog Value Object	
SINT16	Analog Value Object	Each ADI element of this type occupies two bytes.
UINT16	Analog Value 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.

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.

3.6.3 Mapping of BACnet Objects to Anybus CompactCom

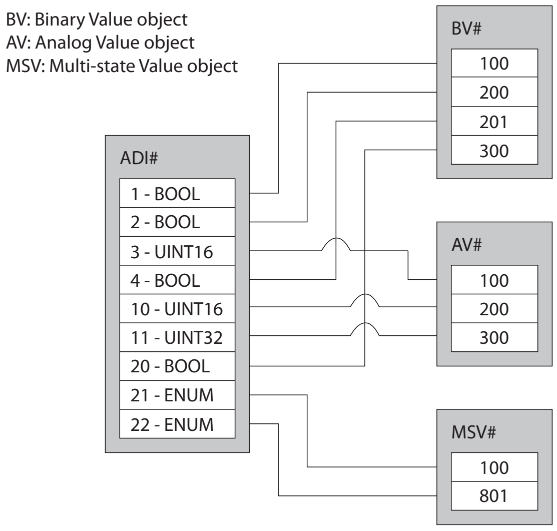
Simple Mapping

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:



Advanced Mapping

If the “Advanced mapping supported” attribute 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:



3.6.4 Process Data

The Anybus CompactCom BACnet/IP supports COV (Change Of Value) notification and Alarm/Event functionality, see "COV Notifications, Alarms and Events" on page 24. 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.

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

3.7 File System

3.7.1 General Information

Category: Extended

The built-in file system hosts 1.43 MByte of non volatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application.

The file system uses the following conventions:

- '\' (backslash) is used as a path separator
- A 'path' originates from the system root and as such must begin with a '\'
- A 'path' must not end with a '\'
- Names may contain spaces (' ') but must not begin or end with one.
- Names must not contain one of the following characters: '\ / : * ? " < > |'
- Names cannot be longer than 48 characters
- A path cannot be longer than 255 characters (filename included)

See also...

- "FTP Server" on page 31
- "Web Server" on page 33
- "E-mail Client" on page 40
- "Server Side Include (SSI)" on page 41
- "File System Interface Object (0Ah)" on page 95

IMPORTANT: *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*

3.7.2 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 Notepad in Microsoft Windows™). The format of these files are, with a few exceptions, based on the concept of 'keys', where each 'key' can be assigned a value, see below.

Example:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

4. COV Notifications, Alarms and Events

4.1 General

The Anybus CompactCom 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. see "General" on page 24.

Note: 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.

Notification Class Object Functionality

The Notification Class object is used to configure certain constraints in the functionality and the recipient list that is common to all objects that are associated to an instance of the Notification Class Object. The Anybus CompactCom BACnet/IP w. IT module supports 6 instances of this object.

There are three object notifications (or events) that each indicate a transition from one state to another: TO-OFFNORMAL, TO-FAULT, and TO-NORMAL. Every alarm/event triggered by a value object, is represented as one of these. The interpretation depends on which value object was the origin of the notification.

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.

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.

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.

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" on page 16

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.

Element	Description
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, 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 trig-gered 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 the2 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:

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[a]	BACnet list of unsigned values that will trigger a TO-OFFNORMAL event.
Fault Values[a]	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.

a. 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.

Note: 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

Category: extended

The built-in FTP-server makes it easy to manage the file system using a standard FTP client.

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 8 concurrent connections.

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:Homedir1
User2:Password2:Homedir2
User3:Password3:Homedir3
```

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:Homedir1
User2:Password2:Homedir2
\path\userlistA:HomedirA
\path\userlistB:HomedirB
```

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

File Format:

```
User1:Password1
User2:Password2
User3:Password3
```

Notes:

- Usernames must not exceed 15 characters in length.
- Passwords must not exceed 15 characters in length.
- Usernames and passwords must only contain alphabetic characters and/or numbers.
- If '\ftp.cfg' is missing or cannot be interpreted, all user name/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 they should be able to log in, just adding the user information to the 'ftp.cfg' file it is not enough.

- If 'Admin Mode' has been enabled in the Ethernet Object, all user name/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).
- 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

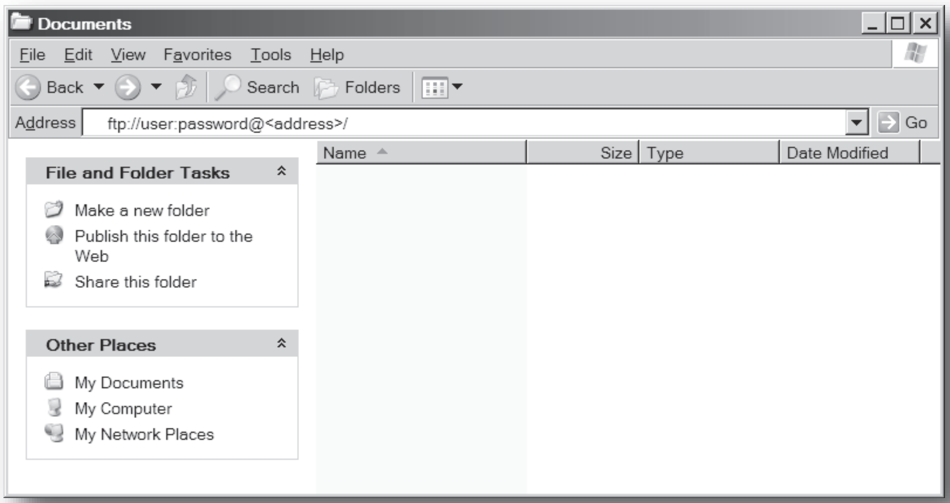
The Windows Explorer features a built-in FTP client which can easily be used to access the file system as follows:

1. Open the Windows Explorer by right-clicking on the 'Start'-button and selecting '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.


Address ftp://user:password@<address>/

6. Web Server

6.1 General Information

Category: extended

The built-in web server provides a flexible environment for end-user interaction and configuration purposes. The powerful combination of SSI and client-side scripting allows 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.

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

See also...

- "FTP Server" on page 31
- "Server Side Include (SSI)" on page 41
- "Ethernet Host Object (F9h)" on page 110

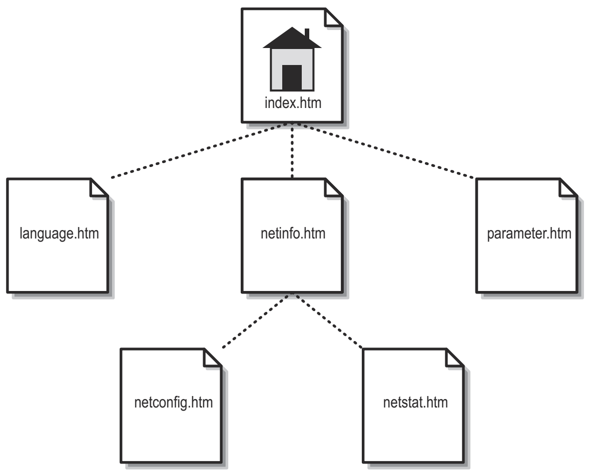
6.2 Default Web Pages

The default web interface consists of a set of virtual files; these virtual files may be replaced, but not permanently erased, by placing files with the same name in the same location (i.e. the web root).

The files can be used as-is or called from a customized web environment.

The files are:

```
<WebRoot>\style.css
<WebRoot>\arrow_red.gif
<WebRoot>\index.htm
<WebRoot>\netinfo.htm
<WebRoot>\netconfig.htm
<WebRoot>\netstat.htm
<WebRoot>\parameter.htm
<WebRoot>\language.htm
```



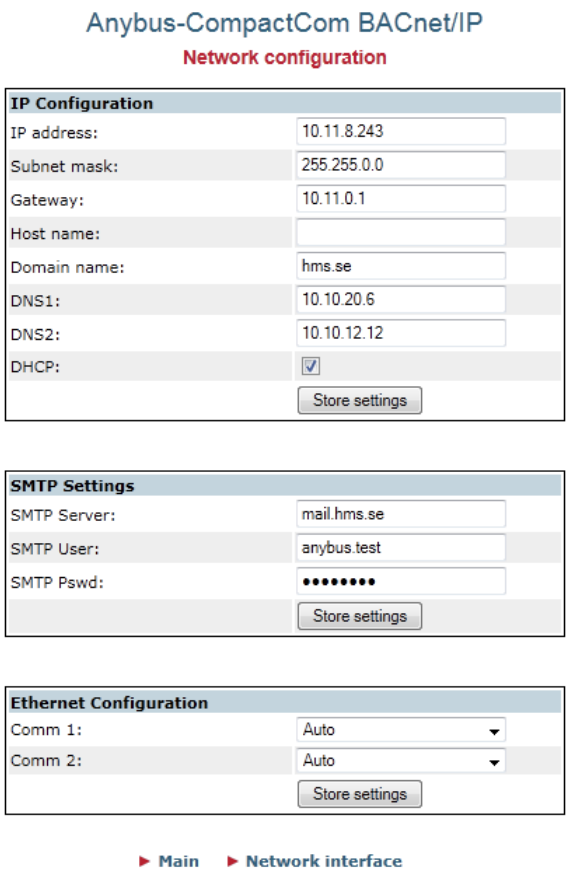
Note: If none of these files are used, it is recommended to completely disable the virtual file system altogether in the File System Interface Object.

See also...

- "File System" on page 22
- "File System Interface Object (0Ah)" on page 95

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:	10.11.8.243
Subnet mask:	255.255.0.0
Gateway:	10.11.0.1
Host name:	
Domain name:	hms.se
DNS1:	10.10.20.6
DNS2:	10.10.12.12
DHCP:	✓
	Store settings

SMTP Settings	
SMTP Server:	mail.hms.se
SMTP User:	anybus.test
SMTP Pswd:	••••••••
	Store settings

Ethernet Configuration	
Comm 1:	Auto
Comm 2:	Auto
	Store settings

Available editable settings will be explained on the next page.

IP configuration

Name	Description
IP address	The TCP/IP settings of the module Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255 The module needs a reset for the changes to take effect
Subnet mask	
Gateway	
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 Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255 Changes will take effect immediately
DNS 2	
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 Default value: auto
Comm 2	

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
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 star-tup.
ICOVLifeEntryRefCnt	Number of COV subscriptions that have a lifetime enabled. When the life-time 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 proc-ess was scheduled.
ICOVNwMsgAllocResums	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 proc-ess was scheduled.

BACnet/IP Statistics	Description
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.
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

Category: advanced

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

File Format:

```
[WebRoot]
\web

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

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

- **Web Root Directory**

  The web server cannot access files outside this directory.

- **Content Types**

  A list of file extensions and their reported content types.

  See also...

  - "Default Content Types" on page 39

- **SSI File Types**

  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

Note 1: Substitute <WebRoot> with the web root directory specified in '\http.cfg'.

Note 2: If no index page is found, the module will default to the virtual index file (if enabled).

See also...

- "Default Web Pages" on page 33

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, see "General Information" on page 38.

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.

File Format:

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

• List of approved users.

```
[AuthName]
(message goes here)
```

• 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.

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

Example:

In this 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

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

7. E-mail Client

7.1 General Information

Category: extended

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, 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.

See also...

- "Network Configuration Object (04h)" on page 63
- "SMTP Client Object (09h)" on page 90

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 user name
- 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 E-mail'-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 E-mail from File'-command. For a description of the file format, see "Command Details: Send E-mail From File" on page 93.

8. Server Side Include (SSI)

8.1 General Information

Category: advanced

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 module encounters such a command, it will execute it, and replace it with the result specified operation (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.

Note: This function cannot be used in e-mail messages.

Syntax:

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

filename-Source file

Default Output:

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 Functions:

Command	Valid for E-mail Messages	Page
GetConfigItem()	Yes	43
SetConfigItem()	No	44
SsiOutput()	Yes	46
DisplayRemoteUser	No	46
ChangeLanguage()	No	47
IncludeFile()	Yes	48
SaveDataToFile()	No	49
printf()	Yes	50
scanf()	No	52

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	*(value of specified key)*
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("\fruit.cnf", "Lemon")'-->
```

... in combination with the following file ('\fruit.cnf')...

```
[Apple]
Green

[Lemon]
Yellow

[Banana]
Blue
```

... returns the string 'Yellow'.

8.3.3 SetConfigItem()

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

Note: This function cannot be used in e-mail messages.

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 behaviour is to modify the existing file.

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
```

Note: Form objects with names starting with underscore ('_') will not be stored.

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
```

Note: 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" on page 58

8.3.5 DisplayRemoteUser

This command stores and returns the user name for an authentication session.

Note: 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.

Note: This command 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>
```

Note: 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	*(file contents)*
Authentication Error	“Authentication error”
File open error	“Failed to open file “*filename*” “

Example:

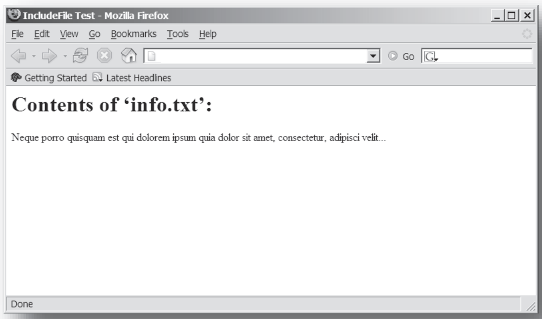
The following example demonstrates how to use this function.

```
<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:


IncludeFile Test - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
Getting Started Latest Headlines
Done

Contents of ‘info.txt’:

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

See also...

- “Include File” on page 41

8.3.8 SaveDataToFile()

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

Note: This command 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 behaviour 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 "*filename*" "
Authentication Error	"Authentication error "
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 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="Bread">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'.

Note: 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 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. Formatting tags are written as follows: `%[Flags][Width][.Precision][Modifier]type` See also... • "Formatting Tags" on page 51
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. At the time of writing, the only allowed argument is ABCCMessage(). See also... • "ABCCMessage()" on page 54

Default Output:

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string ("Errors" on page 57)

Example:

See also...

- "ABCCMessage()" on page 54
- "Example (Get_Attribute):" on page 56

Formatting Tags

- **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.
*	The width is not specified in the format string, it is specified by an integer value preceding the argument that has to be formatted.

- **.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
h	Argument is interpreted as SINT8, SINT16, UINT8 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.

Note: 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. Formatting tags are written as follows: `%[*][Width][Modifier]type` See also... • "Formatting Tags" on page 53
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... • "ABCCMessage()" on page 54

Default Output:

Scenario	Default Output
Success	"Success"
Parsing error	"Incorrect data format "
Too much data for argument	"Too much data "
ABCC Message error	ABCCMessage error string ("Errors" on page 57)

Example:

See also...
- "ABCCMessage()" on page 54
- "Example (Set_Attribute):" on page 56

Formatting Tags

- **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 optionally signed 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 non-whitespace characters	STRING
[scanset]	Accepts a sequence of non-whitespace 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	Page
ABCCMessage()	-	54

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

- Data can be supplied in direct form (format depends on c_type)
- The keyword "ARG" is used when data is supplied by the parent command (e.g. scanf()).

c_type - Specifies the data type in the command (msgdata)

See also...

- "Command Data Types (c_type)" on page 55

r_type - Specifies the data type in the response (msgdata)

See also...

- "Response Data Types (r_type)" on page 55

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

See also...

- "Example (Get_Attribute):" on page 56
- "Example (Set_Attribute):" on page 56

- **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 a 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" **Note:** 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	Comments
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 ABCCMessage() 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

IMPORTANT: *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

See also...

- "printf()" on page 50

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

See also...

- "scanf()" on page 52

Errors

In case an object request results in an error, the error code in the response will be evaluated and translated to the following text strings:

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	“Out of range”
12	“Invalid state”
13	“Out of resources”
14	“Segmentation failure”
15	“Segmentation buffer overflow”
16... 255	“Unknown error”

See also...

- “SSI Output Configuration” on page 58

8.5 SSI Output Configuration

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

File format:

```
[ABCCMessage_X]
0:"Success string"
1:"Error string 1"
2:"Error string 2"
...
16:"Error string 16"
```

Each error code corresponds to a dedicated output string, labelled from 1 to 16.

See also...

- "Errors" on page 57

```
[GetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"Tag not found string"
```

Use "%s" to include the name of the file.

```
[SetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"File write error string"
```

Use "%s" to include the name of the file.

```
[IncludeFile_X]
0:"Success string"
1:"Authentication error string"
2:"File readS error string"
```

Use "%s" to include the name of the file.

```
[scanf_X]
0:"Success string"
1:"Parsing error string"

[ChangeLanguage_X]
0:"Success string"
1:"Change error string"
```

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()" on page 46

9. Anybus Module Objects

9.1 General Information

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

Standard Objects:

- "Anybus Object (01h)" on page 60
- "Network Object (03h)" on page 61
- "Network Configuration Object (04h)" on page 63

Network Specific Objects:

- "Socket Interface Object (07h)" on page 73
- "SMTP Client Object (09h)" on page 90
- "File System Interface Object (0Ah)" on page 95
- "Network Ethernet Object (0Ch)" on page 19

9.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus-CompactCom)
2... 11	-	-	-	Consult the general Anybus-CompactCom Software Design Guide for further information.

12	LED colors	Get	struct of:
			UINT8(LED1A)
			UINT8(LED1B)
			UINT8(LED2A)
			UINT8(LED2B)
				Value:Colour:
				01h Green
				02h Red
				01h Green
				02h Red

13... 15	-	-	-	Consult the general Anybus-CompactCom Software Design Guide for further information.

9.3 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String Map_ADI_Write_Area

Note: It is not possible to map read process data in the Anybus CompactCom BACnet/IP w. IT Functionality 2-Port, thus the standard command Map_ADI_Write_Area is not supported.

Object Attributes (Instance #0)

(Consult the general Anybus-CompactCom 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[a]
6	Read process data size	Get	UINT16	Current read process data size (in bytes). Always zero as it is not possible to map read process data.
7	Exception Information	Get	UINT8	See table below
8... 10	-	-	-	Consult the general Anybus-CompactCom Software Design Guide for further information.

a. Consult the general Anybus-CompactCom Software Design Guide for further information.

Exception Information

Value	Meaning
00h	No information
01h	The Get_All_BACnet_Object_Instances serves request for analog value objects failed
02h	The Get_All_BACnet_Object_Instances serves request for binary value objects failed
03h	The Get_All_BACnet_Object_Instances serves request for multistate value objects failed
04h	An ADI mapped on process data could not be resolved as a BACnet object during start up.

9.4 Network Configuration Object (04h)

Category

Extended, Advanced

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 do not match the configuration used, the Status LED of the module will flash red to indicate a minor error.

See also...

- "Communication Settings" on page 19
- "E-mail Client" on page 40

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #3, Device Instance)

Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Device inst'
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 = 3FFFFEh, default = (module serial number & 3FFFFEh)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #4, Ethernet Port)

Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'UDP port'
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

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #5, Process Active Timeout)

Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Process tmo' (Specifies the process active timeout in milliseconds))
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	The value is stored in non volatile memory (Default = 0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #6, IP Address)

Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'IP address'
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	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #7, Subnet Mask)

Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Subnet mask'
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	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #8, Gateway Address)

Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Gateway'
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	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #9, DHCP)

Changes are valid after reset.

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'DHCP'
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[a]	Get/Set	ENUM	Value:Enum. String:Meaning: 00h 'Disable' DHCP disabled 01h 'Enable' DHCP enabled (default)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #10, Ethernet Communication Settings 1)

Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Comm 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[a]	Get/Set	ENUM	Value:Enum. String:Meaning: 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 HDX' 100Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #11, Ethernet Communication Settings 2)

Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Comm 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[a]	Get/Set	ENUM	Value:Enum. String:Meaning: 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 HDX' 100Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #12, DNS1)

This instance holds the address to the primary DNS server. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'DNS1'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #13, DNS2)

This instance holds the address to the secondary DNS server. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'DNS2'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #14, Host name)

This instance holds the host name of the module. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Host name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with spaces to full length)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #15, Domain name)

This instance holds the domain name. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Domain name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Domain name, 48 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #16, SMTP Server)

This instance holds the SMTP server address. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'SMTP Server'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Dotted decimal format or server name

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #17, SMTP User)

This instance holds user name for the SMTP account. Changes have immediate effect

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'SMTP User'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP account user name, 64 characters (pad with spaces to full length)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #18, SMTP Password)

This instance holds the password for the SMTP account. Changes have immediate effect.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'SMTP Pswd'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP password, 64 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #19, Foreign Device Registration IP Address)

This instance holds the IP address of the BACnet Broadcast Management Device (BBMD) that the module can register as a foreign device on. Changes have immediate effect.

When a valid value is written to this instance, the module will unregister with the current BBMD (if any) and send a request to register a foreign device (Register-Foreign-Device) to the new BBMD.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'FDR IP address'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0). The value is stored in non volatile memory.

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #20, Foreign Device Registration Ethernet Port)

This instance holds the UDP port to use for foreign device registration. Changes have immediate effect.

When a valid value is written to this instance, the module will unregister with the current BBMD (if any) and send a request to register a foreign device (Register-Foreign-Device) to the new BBMD. This will only be done if the value attribute (#5) of instance #19 is set to a valid value.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'FDR UDP port'
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Min = 0, max = FFFFh, default = BAC0h. The value is stored in non volatile memory.

a. Multilingual, see "Multilingual Strings" on page 72.

Instance Attributes (Instance #21, Foreign Device Registration Time to Live Value)

This instance holds the time to live value for the foreign device registration. Changes have immediate effect.

When a valid value is written to this instance, the module will send a send a request to register a foreign device (Register-Foreign-Device) with the new time to live value. This will only be done if the value attribute (#5) of instance #19 is set to a valid value.

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'FDR TTL value'
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Min = 0 s, max = FFFFh s, default = 0 s. The value is stored in non volatile memory.

a. Multilingual, see "Multilingual Strings" on page 72.

Multilingual Strings

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

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

9.5 Socket Interface Object (07h)

Category

Advanced

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. For more information, see "Message Segmentation" on page 102.

IMPORTANT: *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

Object:	Get_Attribute Create (See "Command Details: Create" on page 75) Delete (See "Command Details: Delete" on page 76)
Instance:	Get_Attribute Set_Attribute Bind (See "Command Details: Bind" on page 77) Shutdown (See "Command Details: Shutdown" on page 78) Listen (See "Command Details: Listen" on page 79) Accept (See "Command Details: Accept" on page 80) Connect (See "Command Details: Connect" on page 81) Receive (See "Command Details: Receive" on page 82) Receive_From (See "Command Details: Receive_From" on page 83) Send (See "Command Details: Send" on page 84) Send_To (See "Command Details: Send_To" on page 85) IP_Add_membership (See "Command Details: IP_Add_Membership" on page 86) IP_Drop_membership (See "Command Details: IP_Drop_Membership" on page 87) DNS_Lookup (See "Command Details: DNS_Lookup" on page 88)

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	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0008h

Instance Attributes (Sockets #1...8)

Advanced

#	Name	Access	Type	Description
1	Socket type	Get	UINT8	Value:Socket Type: 00h SOCK_STREAM, NON-BLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NON-BLOCKING (UDP) 03h SOCK_DGRAM, BLOCKING (UDP)
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	State (TCP sockets only): Value:State:Description: 00h CLOSED Closed 01h LISTEN Listening for connection 02h SYN_SENT Active, have sent SYN 03h SYN_RECEIVED Have sent and received SYN 04h ESTABLISHED Established. 05h CLOSE_WAIT Received FIN, waiting for close 06h FIN_WAIT_1 Have closed, sent FIN 07h CLOSING Closed exchanged FIN; await FIN ACK 08h LAST_ACK Have FIN and close; await FIN ACK 09h FIN_WAIT_2 Have closed, FIN is acknowledged 0Ah TIME_WAIT Quiet wait after close
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	Socket can reuse local address Value:Meaning: 1 Enabled 0 Disabled (default)
9	Keep alive	Get/Set	BOOL	Protocol probes idle connection (TCP sockets only) Value:Meaning: 1 Enabled 0 Disabled (default)
10	IP Multicast TTL	Get/Set	UINT8	IP Multicast TTL value (UDP sockets only). Default = 1.
11	IP Multicast Loop	Get/Set	BOOL	IP multicast loop back (UDP sockets only)[a] Value:Meaning: 1 Enable (default) 0 Disable
12	Ack delay time	Get/Set	UINT16	Time for delayed ACKs in ms (TCP sockets only) Default = 200ms[b]
13	TCP No Delay	Get/Set	BOOL	Don’t delay send to coalesce packets (TCP). Value:Meaning: 1 Delay (default) 0 Don’t delay (turn off Nagle’s algorithm on socket)
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)

a. Must belong to group in order to get the loop backed message
b. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.

Command Details: Create

Category

Advanced

Details

Command Code.:	03h
Valid for:	Object Instance

Description

This command creates a socket.

Note: 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]	Value:Socket Type: 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

Advanced

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 "Command Details: Shutdown" on page 78) 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

Advanced

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

Advanced

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]	Value:Mode: 00h Shutdown receive channel 01h Shutdown send channel 02h 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 (EDESTADDRREQ (14)) 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 receive channel.

4. Delete the socket instance.

Command Details: Listen

Category

Advanced

Details

Command Code.: 12h

Valid for: Instance

Description

This command puts a TCP socket in listening state. Backlog queue length is the number of unaccepted connections allowed on the socket. When backlog queue is full, further connections will be refused with RST-flag.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Value:Backlog queue length: 00h 1 01h 2 02h 4	-

- **Response Details**
  (no data)

Command Details: Accept

Category

Advanced

Details

Command Code.:	13h
Valid for:	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.

NON-BLOCKING mode:

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).

BLOCKING mode:

This command will block until a connection request has been detected.

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

Command Details: Connect

Category

Advanced

Details

Command Code.:	14h
Valid for:	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).

NON-BLOCKING mode:

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.

BLOCKING mode:

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

- **Response Details**

(no data)

Command Details: Receive

Category

Advanced

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 (see "Message Segmentation" on page 102).

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.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

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	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Command Segmentation" on page 103
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see "Message Segmentation" on page 102).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Response Segmentation" on page 104
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Advanced

Details

Command Code.:	16h
Valid for:	Instance

Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 102).

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.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

The module will not issue a response until the operation has finished.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Command Segmentation" on page 103
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see "Message Segmentation" on page 102).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Response Segmentation" on page 104
Data[0]	Host IP address byte 3 (low)	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 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Received data	

Command Details: Send

Category

Advanced

Details

Command Code.:	17h
Valid for:	Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 102).

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**

  Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see "Message Segmentation" on page 102).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see "Command Segmentation" on page 103
Data[0...n]	Data to send	-

- **Response Details**

Field	Contents	Notes
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

Advanced

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 (see “Message Segmentation” on page 102).

- **Command Details**

  Note: To allow larger amounts of data (i.e. >255 bytes) to be sent, the command data may be segmented (see “Message Segmentation” on page 102).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see “Command Segmentation” on page 103
Data[0]	Host IP address byte 3 (low)	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 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Data to send	

- **Response Details**

Field	Contents	Notes
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: IP_Add_Membership

Category

Advanced

Details

Command Code.:	19h
Valid for:	Instance

Description

This command assigns to 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	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	Group IP address byte 3 (low)	-
Data[1]	Group IP address byte 2	
Data[2]	Group IP address byte 1	
Data[3]	Group IP address byte 0 (high)	

- **Response Details**

  (no data)

Command Details: IP_Drop_Membership

Category

Advanced

Details

Command Code.:	1Ah
Valid for:	Instance

Description

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

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	Group IP address byte 3 (low)	-
Data[1]	Group IP address byte 2	
Data[2]	Group IP address byte 1	
Data[3]	Group IP address byte 0 (high)	

- **Response Details**
  (no data)

Command Details: DNS_Lookup

Category

Advanced

Details

Command Code.:	1Bh
Valid for:	Object Instance

Description

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

- **Command Details**

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

- **Response Details (Success)**

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

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.

9.6 SMTP Client Object (09h)

Category

Advanced

Object Description

This object groups functions related to the SMTP-client.

See also...

- “File System” on page 22
- “E-mail Client” on page 40
- “Instance Attributes (Instance #16, SMTP Server)” on page 69
- “Instance Attributes (Instance #17, SMTP User)” on page 69
- “Instance Attributes (Instance #18, SMTP Password)” on page 69

Supported Commands

Object:	Get_Attribute Create Delete Send e-mail from file(“Command Details: Send E-mail From File” on page 93)
Instance:	Get_Attribute Set_Attribute Send e-mail(“Command Details: Send E-mail” on page 94)

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

Advanced

Instances are created dynamically by the application.

#	Name	Access	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. "Duck and cover"

Command Details: Create

Category

Advanced

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]		
MsgData[0]	Instance number	low byte
MsgData[1]		high byte

Command Details: Delete

Category

Advanced

Details

Command Code.:	04h
Valid for:	Object

Description

This command deletes an e-mail instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**
  (no data)

Command Details: Send E-mail From File

Category

Advanced

Details

Command Code.:	11h
Valid for:	Object

Description

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

File format:

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

```
[To]
recipient

[From]
sender

[Subject]
e-mail subject

[Headers]
extra headers, optional

[Message]
actual e-mail message
```

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Send E-mail

Category

Advanced

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)

9.7 File System Interface Object (0Ah)

Category

Advanced

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.

Supported Commands

Object:	Get_Attribute
	Create(“Command Details: Create” on page 97)
	Delete(“Command Details: Delete” on page 98)
	Format Disc(“Command Details: Format Disc” on page 107)

Instance:	Get_Attribute File Open(“Command Details: File Open” on page 98) File Close(“Command Details: File Close” on page 99) File Delete(“Command Details: File Delete” on page 99) File Copy(“Command Details: File Copy” on page 100) File Rename(“Command Details: File Rename” on page 101) File Read(“Command Details: File Read” on page 102) File Write(“Command Details: File Write” on page 103) Directory Open(“Command Details: Directory Open” on page 103) Directory Close(“Command Details: Directory Close” on page 104) Directory Delete(“Command Details: Directory Delete” on page 104) Directory Read(“Command Details: Directory Read” on page 105) Directory Create(“Command Details: Directory Create” on page 106) Directory Change(“Command Details: Directory Change” on page 106)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'File System Interface'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0004h
12	Disable virtual file system	Get	BOOL	False
13	Total disc size	Get	Array of UINT32	-
14	Free space	Get	Array of UINT32	-
15	Disc CRC	Get	Array of UINT32	-

Instance Attributes

Advanced

#	Name	Access	Type	Description
1	Instance type	Get	UINT8	Value:Type: 00h Reserved 01h File instance 02h Directory instance
2	File size	Get	UINT32	File size in bytes (zero for directories)
3	Path	Get	Array of CHAR	Path where instance operates

Command Details: Create

Category

Advanced

Details

Command Code.:	03h
Valid for:	Object

Description

This command creates a file operation 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]		
MsgData[0]	Instance number	low byte
MsgData[1]		high byte

Command Details: Delete

Category

Advanced

Details

Command Code.:	04h
Valid for:	Object

Description

This command deletes a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**
  (no data)

Command Details: File Open

Category

Advanced

Details

Command Code.:	10h
Valid for:	Instance

Description

This command opens a file for reading, writing, or appending.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Mode	Value:Mode: 00h Read mode 01h Write mode 02h Append mode
CmdExt[1]	(reserved, set to zero)	-
MsgData[0... n]	Path + filename	Relative to current path

- **Response Details**
  (no data)

Command Details: File Close

Category

Advanced

Details

Command Code.:	11h
Valid for:	Instance

Description

This command closes a previously opened file.

- **Command Details**
  (no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0]	File size	low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		high byte, high word

Command Details: File Delete

Category

Advanced

Details

Command Code.:	12h
Valid for:	Instance

Description

This command permanently deletes a specified file from the file system.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + filename	Relative to current path

- **Response Details**
  (no data)

Command Details: File Copy

Category

Advanced

Details

Command Code.:	13h
Valid for:	Instance

Description

This command makes a copy of a file.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Source path + filename	Relative to current path, separated by NULL
	NULL	
	Destination path + filename	

- **Response Details**
  (no data)

Command Details: File Rename

Category

Advanced

Details

Command Code.:	14h
Valid for:	Instance

Description

This command renames or moves a file.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Old path + filename	Relative to current path, separated by NULL
	NULL	
	New path + filename	

- **Response Details**
  (no data)

Command Details: File Read

Category

Advanced

Details

Command Code.:	15h
Valid for:	Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Bytes	no. of bytes to read
CmdExt[1]	(reserved, set to zero)	-

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0... n]	Data	Data read from file

Command Details: File Write

Category

Advanced

Details

Command Code.:	16h
Valid for:	Instance

Description

Writes data to a file previously opened for writing or appending.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
Data[0... n]	Data	Data to write to file

- **Response Details**

Field	Contents	Comments
CmdExt[0]	Bytes	no. of bytes written
CmdExt[1]	(reserved, ignore)	-

Command Details: Directory Open

Category

Advanced

Details

Command Code.:	20h
Valid for:	Instance

Description

This command opens a directory.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
Data[0... n]	Path + name of directory	Relative to current path

- **Response Details**

  (no data)

Command Details: Directory Close

Category

Advanced

Details

Command Code.:	21h
Valid for:	Instance

Description

This command closes a previously opened directory.

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

Command Details: Directory Delete

Category

Advanced

Details

Command Code.:	22h
Valid for:	Instance

Description

This command permanently deletes an empty directory from the file system.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**
  (no data)

Command Details: Directory Read

Category

Advanced

Details

Command Code.: 23h

Valid for: Instance

Description

This command reads the contents of a directory previously opened for reading.

The command returns information about a single directory entry, which means that the command must be issued multiple times to retrieve the complete contents of a directory. When the last entry has been read, the command returns an "empty" response (i.e. a response where the data size is zero).

- **Command Details**
  (no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0]	Size of entry	Low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		High byte, high word
MsgData[4]	Flags	Bit:Meaning: 0 Entry is a directory 1 Entry is read-only 2 Entry is hidden 3 Entry is a system entry
MsgData[5... n]	Name of entry	-

Command Details: Directory Create

Category

Advanced

Details

Command Code.:	24h
Valid for:	Instance

Description

This command creates a directory.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**
  (no data)

Command Details: Directory Change

Category

Advanced

Details

Command Code.:	25h
Valid for:	Instance

Description

This command changes the current directory/path for an instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**
  (no data)

Command Details: Format Disc

Category

Advanced

Details

Command Code.:	30h
Valid for:	Object

Description

This command formats the file system.

- **Command Details**

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

- **Response Details**

  (no data)

Object Specific Error Codes

Error Codes	Meaning
1	Failed to open file
2	Failed to close file
3	Failed to delete file
4	Failed to open directory
5	Failed to close directory
6	Failed to create directory
7	Failed to delete directory
8	Failed to change directory
9	Copy operation failure (could not open source)
10	Copy operation failure (could not open destination)
11	Copy operation failure (write failed)
12	Unable to rename file

9.8 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)

Extended

#	Name	Access	Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also "Ethernet Host Object (F9h)" on page 110)

10. Host Application Objects

10.1 General Information

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

Standard Objects:

- Application Object (see Anybus-CompactCom Software Design Guide)
- Application Data Object (see Anybus-CompactCom Software Design Guide)

Network Specific Objects:

- "BACnet Host Object (EFh)" on page 112
- "Ethernet Host Object (F9h)" on page 110

10.2 Ethernet Host Object (F9h)

Category

Basic, extended

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	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Default[a]	Comment
1	MAC address[b]	Get	Array of UINT8	-	6 byte physical address value; overrides the pre-programmed MAC address. Note that the new MAC address value must be obtained from the IEEE.

a. If an attribute is not implemented, the module will use this value instead
b. The module is pre-programmed with a valid MAC address. To use that address, do *not* implement this attribute.

Extended

#	Name	Access	Type	Default[a]	Comment
2	Enable HICP	Get	BOOL	True	Value:Meaning: True HICP enabled False HICP disabled (see "HICP (Host IP Configuration Protocol)" on page 122)
3	Enable Web Server	Get	BOOL	True	Value:Meaning: True web server enabled False web server disabled (see "Web Server" on page 33)
5	Enable Web ADI access	Get	BOOL	True	Value:Meaning: True web ADI access enabled False web ADI access disabled (see "Web Server" on page 33)
6	Enable FTP server	Get	BOOL	True	Value:Meaning: True FTP server enabled False FTP server disabled (see "FTP Server" on page 31)
7	Enable admin mode	Get	BOOL	False	Value:Meaning: True FTP Admin mode enabled False FTP Admin mode disabled (see "FTP Server" on page 31)
8	Network Status	Set	UINT16	-	See "Network Status" on page 111

a. If an attribute is not implemented, the module will use this value instead

Network Status

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

Bit	Contents	Description
0	Link	Value:Meaning: True Link sensed False No link
1	IP established	Value:Meaning: True IP address established False IP address not established
2... 15	(reserved)	(mask off and ignore)

10.3 BACnet Host Object (EFh)

Category

Basic, extended, advanced

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 start-up; 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 Software Design Guide, "Error Codes"

Supported Commands

Object:	Get_ADI_By_BACnet_Object_Instance (See "Command Details: Get_ADI_By_BACnet_Object" on page 115)
	Get_ADI_By_BACnet_Name (See "Command Details: Get_ADI_By_BACnet_Name" on page 116)
	Get_All_BACnet_Object_Instances (See "Command Details: Get_All_BACnet_Object_Instances" on page 117)
	Get_BACnet_Object_Instance_By_ADI (See "Command Details: Get_BACnet_Object_Instance_By_ADI" on page 118)
Instance:	-

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)

Basic

#	Name	Access	Type	Default Value	Comment
1	Object Name	Get/Set[a]	Array of CHAR	"Controller"	Any change in the values of these attributes will change the corresponding property of the BACnet Device Object.
2	Vendor Name	Get	Array of CHAR	"HMS Industrial Networks"	
3	Vendor Identifier	Get	UINT16	01E6h	
4	Model Name	Get	Array of CHAR	"Anybus-CompactCom"	Any change in the values of this attribute will change the corresponding property of the BACnet Device Object. A change will also affect the product name displayed on the web pages and in HICP responses (Fieldbus type value).
5	Firmware Revision	Get	Array of CHAR	Anybus CompactCom firmware revision	Any change in the values of these attributes will change the corresponding property of the BACnet Device Object.
6	Application_Software_Version	Get	Array of CHAR	" "	
8	Current date and time:[b]	Get/Set	Struct of:-		
	Year		UINT16	0 (if the attribute is not implemented)	Current year, e.g. 2011
	Month		UINT8		Current month, e.g. 05
	Day		UINT8		Current day, e.g. 14
	Hour		UINT8		Current hour, e.g. 15
	Minute		UINT8		Current minute, e.g. 54
	Second		UINT8		Current second, e.g. 21

a. The application decides whether it will be allowed to set the Object Name attribute or not.

b. This attribute is used if the host application has a real time clock implemented. It is read by the module at startup and written by the module when a TimeSynchronization is received on BACnet to sync the time in the host application. Both get and set access are optional. If set is supported, then get is required. If invalid values are detected at startup, all elements will be set to 0.

Extended

#	Name	Access	Type	Default Value	Comment
7	Support advanced mapping	Get	BOOL	False	If true, the application supports advanced BACnet to ADI mapping schema and must support all related services. See: - "Command Details: Get_ADI_By_BACnet_Object" on page 115 - "Command Details: Get_ADI_By_BACnet_Name" on page 116 - "Command Details: Get_BACnet_Object_Instance_By_ADI" on page 118 - "Command Details: Get_All_BACnet_Object_Instances" on page 117

Advanced

#	Name	Access	Type	Default Value	Comment
9	Password	Get	Array of CHAR	"Admin"	Password used for ReinitializeDevice and DeviceCommunicationControl services. Max 20 bytes.

Command Details: Get_ADI_By_BACnet_Object

Category

Extended

Details

Command Code.:	10h
Valid for:	Object Instance

Description

By setting the 'Support advanced mapping'-attribute (#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" on page 9.

- **Command Details**

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

- **Response Details**

Field	Contents	Notes
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

Command Code.:	11h
Valid for:	Object Instance

Description

By setting the ‘Support advanced mapping’-attribute (#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” on page 9.

- **Command Details**

Field	Contents	Notes
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	Notes
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

Command Code.:	12h
Valid for:	Object Instance

Description

If the 'Support advanced mapping'-attribute (#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.

Example:

```
This example 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)

- **Command Details**

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

- **Response Details**

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

For information about BACnet object classes, see "BACnet/IP Implementation" on page 9.

Command Details: Get_BACnet_Object_Instance_By_ADI

Category

Extended

Details

Command Code.:	13h
Valid for:	Object Instance

Description

If the ‘Support advanced mapping’-attribute (#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” on page 9.

- **Command Details**

Field	Contents	Notes
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	Notes
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)

A. Categorization of Functionality

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

A.3 Advanced

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally 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 parameter "Process Active Timeout" value, if this parameter value is greater than zero.

B.2 Anybus Statemachine

The table below describes how the Anybus Statemachine 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.	• 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. • The supervised bit is set when the module is in this state.
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 BACnet/IP w. IT Functionality 2-Port 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) 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)

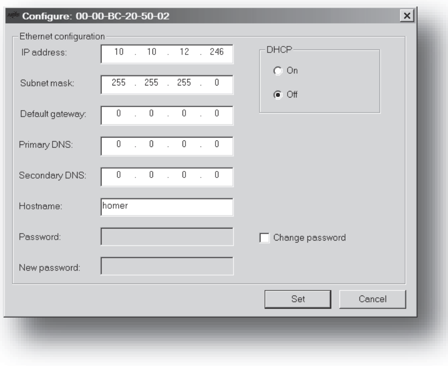
C. HICP (Host IP Configuration Protocol)

C.1 General

The module supports the 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. The utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

C.2 Operation

Upon starting the program, the network is scanned for Anybus products. The network can be re-scanned at any time by clicking 'Scan'.



Configure: 00-00-BC-20-50-02

Ethernet configuration

IP address:	10 . 10 . 12 . 246
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	0 . 0 . 0 . 0
Primary DNS:	0 . 0 . 0 . 0
Secondary DNS:	0 . 0 . 0 . 0
Hostname:	homer
Password:	
New password:	

DHCP: On [✓ Off]

☐ Change password

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

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, click on the 'Change password' checkbox, and enter the password under 'New password'.

D. Technical Specification

D.1 Protective Earth (PE) Requirements

In order to ensure proper EMC behaviour, the module must be properly connected to protective earth via the PE pad / PE mechanism described in the general Anybus-CompactCom Hardware Design Guide.

HMS Industrial Networks does not guarantee proper EMC behaviour unless these PE requirements are fulfilled.

D.2 Power Supply

Supply Voltage

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

Power Consumption

The Anybus-CompactCom 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 Hardware Design Guide.

The current hardware design consumes up to 380 mA[1].

D.3 Environmental Specification

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

D.4 EMC Compliance

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

1. Note that 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. Note however that in any case, the Anybus-CompactCom BACnet/IP will remain a Class B module.

E. Timing & Performance

E.1 General Information

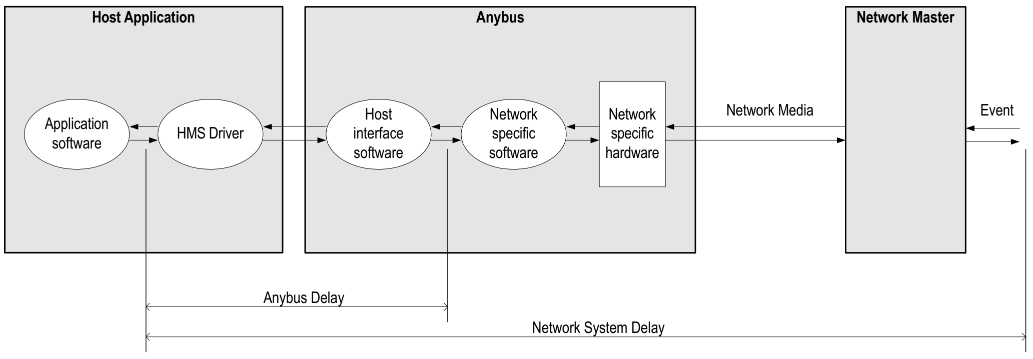
This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom BACnet/IP w. IT Functionality 2-Port.

The following timing aspects are measured:

Category	Parameters	Note
Startup Delay	T1, T2	These parameters are equivalent for all Anybus CompactCom modules. Please consult the Anybus CompactCom Software Design Guide, App. B. for times measured.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	Not supported, see page 125
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	See page 125
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	Not supported, see page 126
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	See page 126

E.2 Process Data

E.2.1 Overview



E.2.2 Anybus Read Process Data Delay (Anybus Delay)

The Read Process Data Delay (labelled 'Anybus delay' in the figure above) is defined as the time measured from just before new data is buffered and available to the Anybus host interface software, to when the data is available to the host application (just after the new data has been read from the driver).

No read process data can be mapped in the Anybus CompactCom BACnet module, thus parameters T6, T7 and T8 are not measured.

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

E.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled 'Anybus delay' in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

In BACnet, this timing parameter measures the time for a COV notification and/or an alarm event to reach the network buffer.

Parameter	Description	Typ.	Min.	Max.	Unit.
T12	Anybus Write Process Data delay, 8 ADIs (single UINT8)	390	260	610	µs
T13	Anybus Write Process Data delay, 16 ADIs (single UINT8)	410	300	660	µs
T14	Anybus Write Process Data delay, 32 ADIs (single UINT8)	520	360	760	µs

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

E.2.4 Network System Read Process Data Delay (Network System Delay)

The Network System Read Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the point where an event is generated at the network master to when the corresponding data is available to the host application (just after the corresponding data has been read from the driver).

No read process data can be mapped in the Anybus CompactCom BACnet module, thus parameters T9, T10 and T11 are not measured.

E.2.5 Network System Write Process Data Delay (Network System Delay)

The Network System Write Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the time after the new data is available from the host application (just before the data is written to the driver) to when this data generates a corresponding event at the network master.

In BACnet, this timing parameter measures the time for a COV notification and/or an alarm event to generate a corresponding event at the network master.

Parameter	Description	Typ.	Min.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	15	8.4	27.2	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	15.6	8	41.6	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	16	8	58	ms

Conditions:

Parameter	Conditions
Application CPU	-
Timer system call interval	1 ms
Driver call interval	0.2... 0.3 ms
No.of ADIs (single UINT8) mapped to Process Data	8, 16 and 32
Communication	Parallel
Telegram types during measurement period	Process Data only
Bus load, no. of nodes, baud rate etc.	Normal

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'' ANDANY 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.