Anybus CompactCom 40

EtherNet/IP IIoT Secure

NETWORK GUIDE

SCM-1202–069 2.6 en-US ENGLISH

Important User Information

Disclaimer

The information in this document is for informational purposes only. Please inform HMS Networks of any inaccuracies or omissions found in this document. HMS Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Networks and is subject to change without notice. HMS Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the 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.

Table of Contents

Page

This page intentionally left blank

1 Preface

1.1 About this document

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

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

For additional related documentation and file downloads, please visit the support website at www.anybus.com/support.

1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
Using OPC UA Application Defined Information Models	HMS	SCM-1202-182
CIP specification, Volumes 1 (CIP Common) and 2 (EtherNet/IP)	ODVA	
OPC UA Specification 1.04	OPC Foundation	

1.3 Document History

Version	Date	Description
1.4	2017-11-28	First release
1.5	2017-12-15	Updated Copyright Appendix Corrected Network Type ID
1.6	2018-05-07	Added Conformance Test Guide Updates to CIP objects ADI and QoS Misc updates
1.7	2018-09-19	Added MQTT functionality Updated Set_Configuration_Data descriptions in EtherNet/IP Host object
1.8	2018-10-23	Minor update, mainly to MQTT section
1.9	2019-06-10	Rebranding Minor updates
2.0	2019-10-28	Secure functionality added
2.1	2019-12-03	New screendumps
2.2	2020-05-11	Updated screenshots in Chapter 3, “Initial Setup and Account Configuration” Expanded and improved OPC UA and MQTT sections Added instance attribute #7 “Limits” to instance #1, OPC UA Object (E3h) Added instance attribute #42 to Network Configuration Object (04h) Updated instance attribute #40, #41, #50, #51, #53, #54, #55 in Network Configuration Object (04h)
2.3	2021–04–16	Added Time Object (13h)
2.4	2021–05–26	Updated CIP Identity Object

Version	Date	Description
		Updated OPC UA and MQTT sections Updated SMTP sections
2.5	2021-09-23	Added OPC UA application defined information models Minor updates
2.6	2021-10-29	Minor updates

1.4 Document Conventions

Numbered lists indicate tasks that should be carried out in sequence:

1. First do this
2. Then do this

Bulleted lists are used for:

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

► An action

→ and a result

User interaction elements (buttons etc.) are indicated with bold text.

```
Program code and script examples
```

Cross-reference within this document: *Document Conventions, p. 8*

External link (URL): www.hms-networks.com

⚠ **WARNING**
Instruction that must be followed to avoid a risk of death or serious injury.

⚠ **Caution**
Instruction that must be followed to avoid a risk of personal injury.

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

ℹ *Additional information which may facilitate installation and/or operation.*

1.5 Document Specific Conventions

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

1.6 Abbreviations

Abbreviation	Meaning
CA	Certificate Authority
API	assigned packet interval
RPI	requested packet interval
T	target (in this case the module)
O	origin (in this case the master)

1.7 Trademarks

Anybus® is a registered trademark of HMS Networks.

EtherNet/IP is a trademark of ODVA, Inc.

All other trademarks are the property of their respective holders.

2 About the Anybus CompactCom 40 EtherNet/IP IIoT Secure

2.1 General

The Anybus CompactCom 40 EtherNet/IP IIoT Secure communication module provides instant EtherNet/IP conformance tested connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type. The module supports both linear and ring network topology.

The modular approach of the Anybus CompactCom 40 platform allows the CIP-object implementation to be extended to fit specific application requirements. Furthermore, the Identity 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 Anybus CompactCom 40 modules defined in the Anybus CompactCom 40 Hardware and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

2.2 Features

- Secure Boot
- TLS support for secure data transfer
- Security chip for secure key storage
- Hardware accelerated cryptographic operations
- Hashed password storage
- Secure web server w. customizable content
- Secure file transfer server (WebDAV)
- Two EtherNet/IP ports
- RJ45 connectors
- Supports OPC UA functionality
- Supports MQTT functionality
- Max. read process data: 1448 bytes
- Max. write process data: 1448 bytes
- Max. process data (read + write, in bytes): 2896 bytes
- Beacon Based DLR (Device Level Ring) and linear network topology supported
- Black channel interface, offering a transparent channel supporting CIP Safety
- 10/100 Mbit, full/half duplex operation
- Email client
- JSON functionality
- Customizable Identity Information
- Up to 65535 ADIs
- CIP Parameter Object support
- Expandable CIP-object implementation
- Supports unconnected CIP routing
- Transparent Socket Interface
- Modular Device functionality
- QuickConnect supported
- Multiple IO assembly instances can be created

2.3 IIoT – Industrial Internet of Things

IIoT gives an application access to the data of a product over the internet. This is, among other things, useful for

- uncovering product failures and deficiencies
- discovering how products are used
- ensuring the quality of products faster

To support IIoT, the Anybus CompactCom 40 EtherNet/IP IIoT Secure supports the protocols OPC UA and MQTT.

See also ...

- *OPC UA, p. 80*
- *MQTT, p. 103*

2.4 Security

Anybus CompactCom 40 EtherNet/IP IIoT Secure provides security features for secure network communication.

In order to secure the network communication the device is equipped with a security chip providing secure key storage together with a hardware accelerated cryptographic engine. The embedded web server as well as the OPC UA and the MQTT communication are secured.

The embedded web server features web pages for security configuration such as certificate installation and user account management. A user can e.g. install its own certificate. This interface can be used as is, or be modified to fit the host product. All web operations are implemented using an JSON API providing the possibility for users to make tools directly accessing this API.

2.5 Certificates

Certificates are a main component in secure communication. They are used to prove the identity of the owner of the certificate. A node will trust a certificate if it trusts the Certificate Authority (CA) that has issued the certificate. Trusted certificates are used to ensure secure communication.

Which certificates to use, depend on the installation. HMS Networks offers a tool that generates device and CA certificates, that can be used during development.

2.5.1 Initial Device Certificate

The Anybus CompactCom 40 EtherNet/IP IIoT Secure comes with a preinstalled initial device certificate. This certificate proves that the device is produced by HMS Networks and will also be used for HTTPS until the device is configured by the end user. Please note that browsers will issue a security warning as long as this certificate is used.

The initial device certificate holds the following identity information.

countryName (C)	SE
stateOrProvinceName (ST)	Halland
Locality (L)	Halmstad
organizationName (O)	HMS Industrial Networks AB
organizationUnitName (OU)	Anybus
commonName (CN)	(module serial number)
serialNumber (SN)	(module serial number)

The certificate is placed in read only storage, and will not be deleted upon factory default reset.

2.5.2 Certificate Authorities

Some protocols, e.g. OPC UA, need to validate the identity of other devices, such as PLCs, that try to connect to the Anybus CompactCom. The user can install CA certificates that are used to validate the certificate provided by the client. The certificates are installed from the internal web pages of the product.

ⓘ *Certificate expire time/date is not validated as Anybus CompactCom does not know time.*

2.5.3 Device Certificates

Device certificates are installed by the end users at configuration time and are used by the various secure protocols to prove the device identity of the Anybus CompactCom 40 EtherNet/IP IIoT Secure to establish secure communication. The certificates and the corresponding private keys are installed using the web interface. It is possible to configure for which protocol each device certificate is to be used.

3 Initial Setup and Account Configuration

This section describes how to set up an application using the default web pages and the default configuration of the Anybus CompactCom 40 EtherNet/IP IIoT Secure.

> ! Secure operation is not available until initial setup and account configuration has been finalized. The steps described in this section have to be followed to ensure secure operation of the device.

An initial device identity certificate is installed in the Anybus CompactCom 40 EtherNet/IP IIoT Secure. This is initially used to setup a connection to the web server and enables the user to access the internal web pages. It is not possible to delete this certificate from the device.

> ! It is recommended to perform this initial configuration offline, with the device connected directly to your computer. This to ensure that anyone who is not authorized, will gain access to the device and configure the first account.

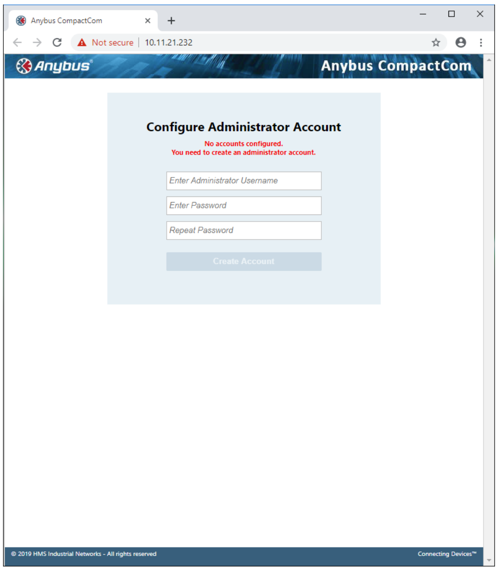
3.1 Set an IP Address

To access the web pages of the Anybus CompactCom, an IP address has to be set for the device. One way to do this is described in *Secure HICP (Secure Host IP Configuration Protocol), p. 240*. An IP address can also be set by the host application.

3.2 Configure First Administrator Account

1. Enter the IP address of Anybus CompactCom 40 EtherNet/IP IIoT Secure in a browser. The web page as shown in the picture will show.

   Create the necessary first administrator account by entering a username and a password.



Configure Administrator Account

No accounts configured.
You need to create an administrator account.

Enter Administrator Username
Enter Password
Repeat Password
Create Account
© 2019 HMS Industrial Networks - All rights reserved
Connecting Devices™

Fig. 1

2. Log in to the device using the newly created administrator account.

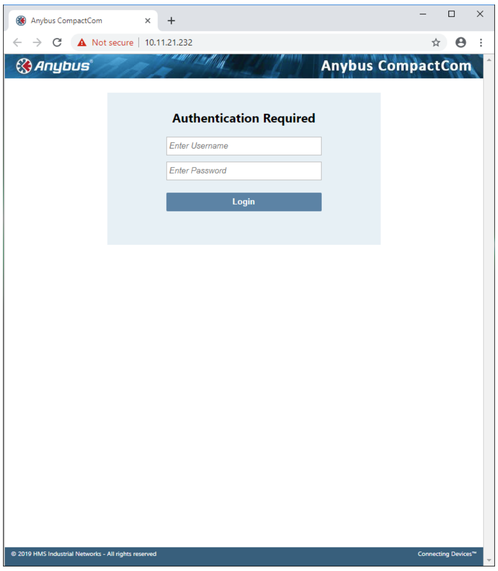
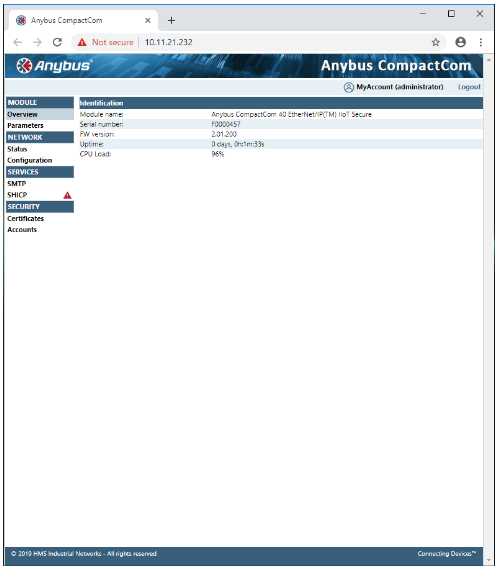


Fig. 2

3. When logged in using an administrator account, you can for example configure new user accounts of different types and install CA and device certificates.



Identification	
Module name:	Anybus CompactCom 40 EtherNet/IP(TM) IIoT Secure
Serial number:	F0000457
FW version:	2.01.200
Uptime:	0 days, 0h:1m:33s
CPU Load:	96%

Fig. 3

3.3 Install a Device Certificate

Follow the steps below to install a device certificate in the Anybus CompactCom.

You may have to install a CA certificate in your browser. See *Installing a CA Certificate in Windows, p. 241* for more information.

1. Create a device certificate, e.g. by using the tool from HMS Networks.
2. Select the Security tab in the column to the left on the start page. When delivered, the device has an initial device certificate installed. This certificate is not shown in the list, and can never be removed. Its sole purpose is to make it possible to access the internal web pages. The device will revert to this certificate if factory default settings are restored or if no certificate is configured for https.

   Add a new device certificate and proceed with the installation.

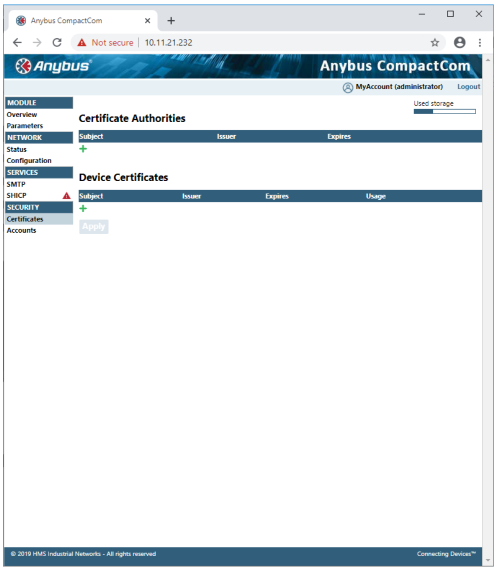
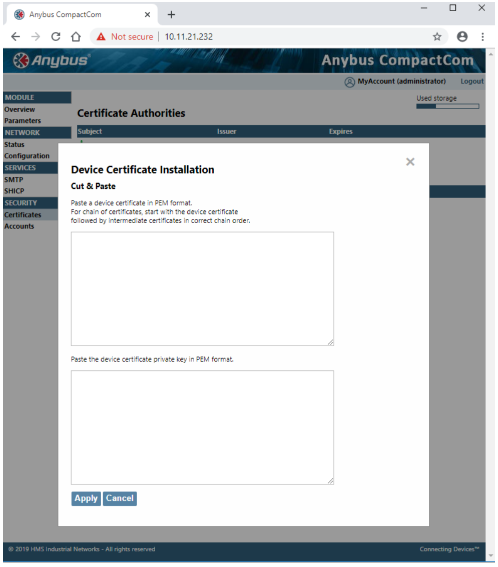


Fig. 4

3. Cut & paste the certificate text and the device certificate private key according to the instructions on the web page. Make sure that the certificate is issued by someone you trust. The certificate must be in PEM format.



Device Certificate Installation

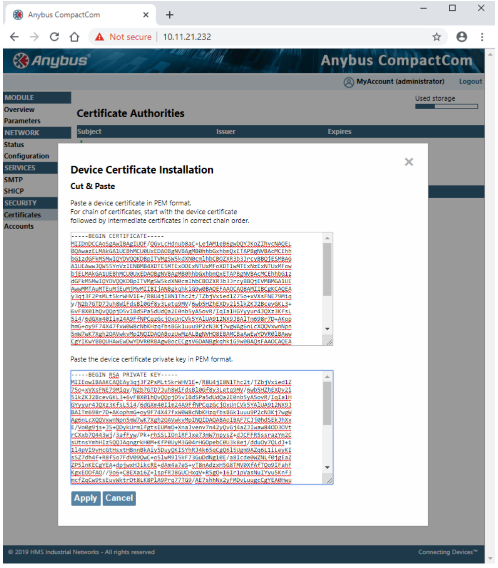
Cut & Paste

Paste a device certificate in PEM format.
For chain of certificates, start with the device certificate
followed by intermediate certificates in correct chain order.

Paste the device certificate private key in PEM format.

Fig. 5

If the certificate is to be used by HTTPS, it is important that the subject name “CN” parameter is set to the device address (IP number or DNS name).



Device Certificate Installation

Cut & Paste

Paste a device certificate in PEM format.
For chain of certificates, start with the device certificate
followed by intermediate certificates in correct chain order.

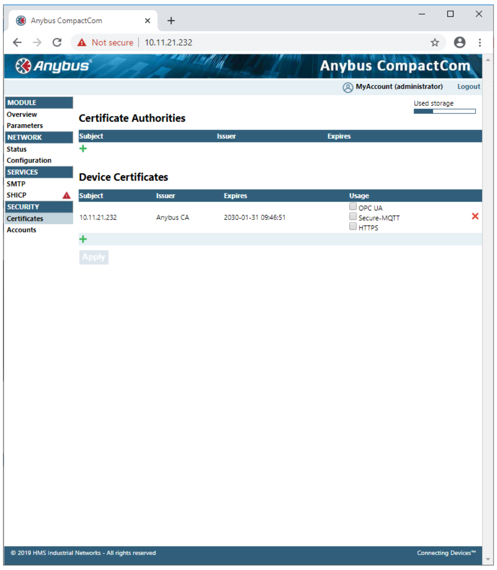
```
-----BEGIN CERTIFICATE-----
```

Paste the device certificate private key in PEM format.

```
-----BEGIN RSA PRIVATE KEY-----
```

Fig. 6

The device certificate is now listed on the security web page.



Certificate Authorities

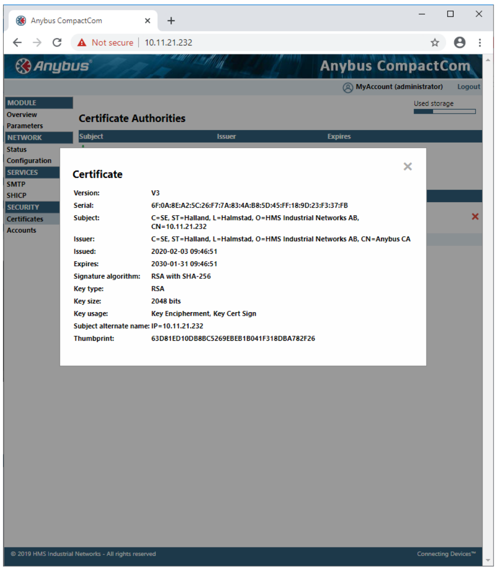
Subject	Issuer	Expires

Device Certificates

Subject	Issuer	Expires	Usage
10.11.21.232	Anybus CA	2030-01-31 09:46:51	□ OPC UA □ Secure-MQTT □ HTTPS

Fig. 7

Click on the device certificate to view the certificate information.



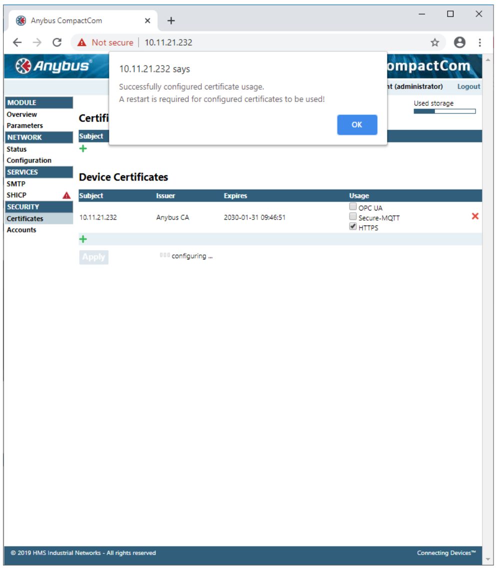
Certificate

Version:	V3
Serial:	6F:0A:8E:A2:5C:26:F7:7A:83:4A:B8:5D:45:FF:18:9D:23:F3:37:FB
Subject:	C=SE, ST=Halland, L=Halmstad, O=HMS Industrial Networks AB, CN=10.11.21.232
Issuer:	C=SE, ST=Halland, L=Halmstad, O=HMS Industrial Networks AB, CN=Anybus CA
Issued:	2020-02-03 09:46:51
Expires:	2030-01-31 09:46:51
Signature algorithm:	RSA with SHA-256
Key type:	RSA
Key size:	2048 bits
Key usage:	Key Encipherment, Key Cert Sign
Subject alternate name:	IP=10.11.21.232
Thumbprint:	63D81ED10DB8BC5269EBEB1B041F318DBA782F26

Fig. 8

4. The installed certificate can be used for HTTPS, MQTT and/or for OPC UA. Select the correct option for the newly installed certificate. For more information on requirements for OPC UA certificates please refer to OPC UA Specification 1.04: Part 6 – Mappings, section 6.2 Certificates.

5. Restart the module by switching the power off and on. The connection is still not trusted, unless a publicly signed certificate is used, or if you have already installed the CA certificate used to sign the device certificate in your browser. You may have to restart your browser.



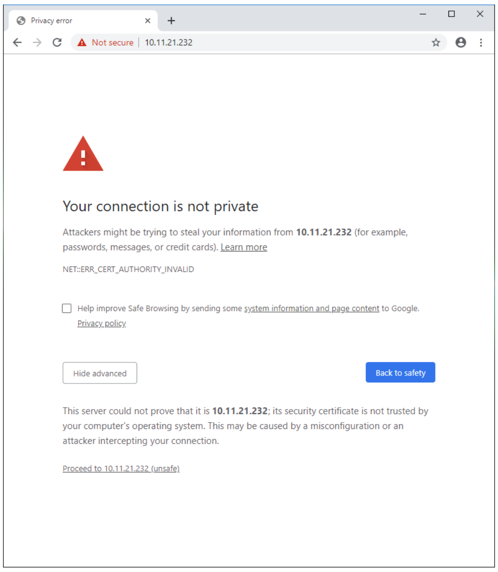
10.11.21.232 says

Successfully configured certificate usage.
A restart is required for configured certificates to be used!

Device Certificates

Subject	Issuer	Expires	Usage
10.11.21.232	Anybus CA	2030-01-31 09:46:51	□ OPC UA □ Secure-MQTT ☑ HTTPS

Fig. 9



Your connection is not private

Attackers might be trying to steal your information from **10.11.21.232** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

□ Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy

This server could not prove that it is **10.11.21.232**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

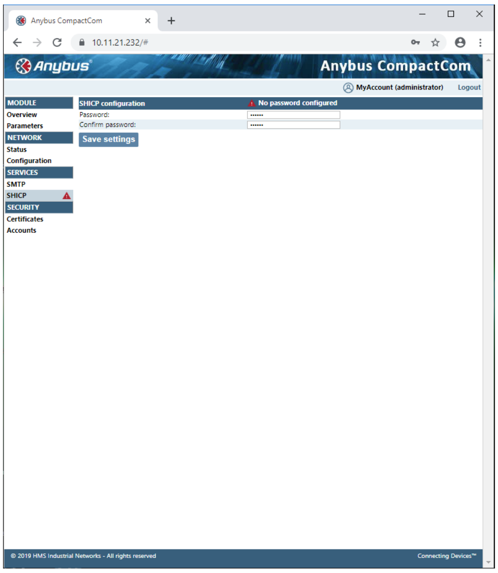
Proceed to 10.11.21.232 (unsafe)

Fig. 10

3.4 Protect the IP Configuration

The Anybus CompactCom 40 EtherNet/IP IIoT Secure supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP.

The IP configuration of the Anybus CompactCom 40 EtherNet/IP IIoT Secure can be protected by setting a password on the internal webpages. If a password is not set, a red triangle will appear.



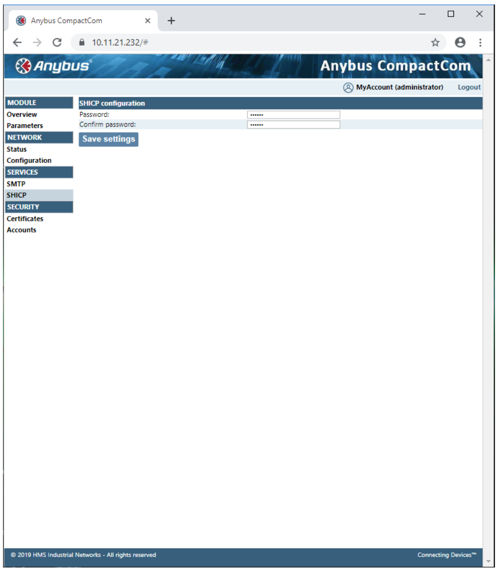
SHICP configuration ⚠ No password configured

Password: ••••••

Confirm password: ••••••

Fig. 11

> **!** It is strongly recommended to password protect this protocol. All users can see the settings, but a password will protect the possibility to set or change the configuration using the Anybus IPconfig utility.



SHICP configuration

Password: ••••••

Confirm password: ••••••

Fig. 12

3.5 Install a CA Certificate

A CA certificate is normally not needed for HTTPS and for the WebDAV file transfer protocol, but has to be installed for OPC UA. The certificate must be from a trusted source and it must be in PEM format. For OPC UA, this must be the CA certificate that issued the Device Certificate used by the OPC UA client, that the application shall communicate with.

1. Login as admin again.

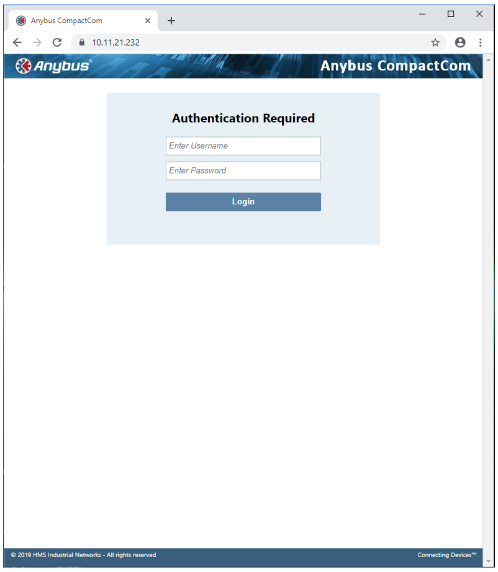
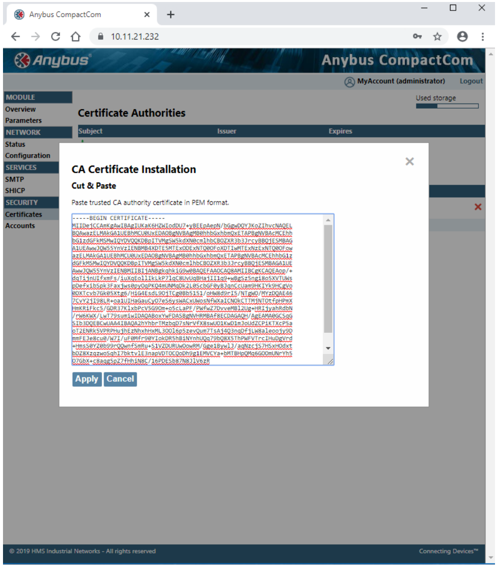


Fig. 13

2. Copy & paste the certificate as described on the web page. The certificate must be in PEM format.



CA Certificate Installation

Cut & Paste

Paste trusted CA authority certificate in PEM format.

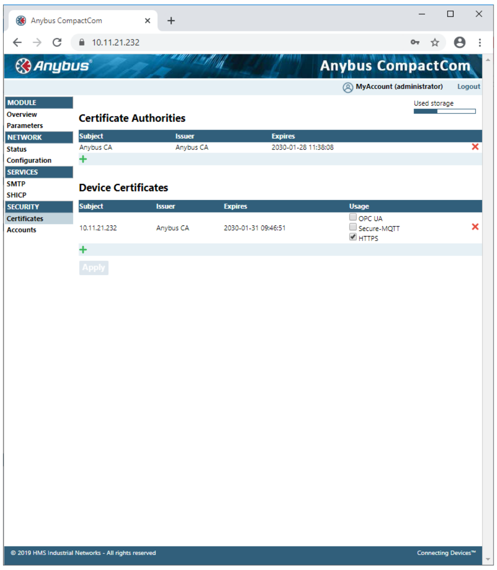
```
-----BEGIN CERTIFICATE-----
MIIDejCCAmKgAwIBAgIUKaK6HZWIodDU7+yBEEpAepN/bGgwDQYJKoZIhvcNAQEL
BQAwazELMAkGA1UEBhMCU0UxEDAOBgNVBAgMB0hhbGxhbmQxETAPBgNVBAcMCEhh
bG1zdGFkMSMwIQYDVQQKDBpITVMgSW5kdXN0cmlhbCBOZXR3b3JrcyBBQjESMBAG
A1UEAwwJQW55YnVzIENBMB4XDTE5MTExODExNTQ0OFoXDTIwMTExNzExNTQ0OFow
azELMAkGA1UEBhMCU0UxEDAOBgNVBAgMB0hhbGxhbmQxETAPBgNVBAcMCEhhbG1z
dGFkMSMwIQYDVQQKDBpITVMgSW5kdXN0cmlhbCBOZXR3b3JrcyBBQjESMBAGA1UE
AwwJQW55YnVzIENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAop/+
dqTijnUIfxmFs/iuXqEollIkLkP7lqC8UvUqBHajII1q9+wBgSz5ngi8o5XVTUWs
pDefxibSpk3Faxjws0pyOqPKQ4mUNMqDk2L0ScbGF0yBJqnCcUam9HKIYk9HCgVo
0DXTcvb7Gk05Xtg6/HiG4EsdL9OjTCg0Bb5151/oHW8d9rI5/NTgWO/MYzDQAE46
7CvY2jI98LR+oa1UIHaGauCyO7eS6ysWACxUWosNfWXaICNOkCTTMjNTOtfpHPmX
HmKRiFkc5/GDR37KlxbPcV5G9Om+o5cLaPF/PWfwZ7DvveMBl2Ug+HRIjyahRdbN
/rW6KWX/LwT79sumiwIDAQABoxYwFDASBgNVHRMBAf8ECDAGAQH/AgEAMA0GCSqG
SIb3DQEBCwUAA4IBAQA2hYhbrTMzbqD7sNrVfX8swUO1KwD1mJoUdZCPiKTXcP5a
oT2ENRk5VPRPHujhEzNhxhHxML3OOl6p5zevQum7TsAj4Q3nqDfjLW8aleoojy9D
mmFEJe8cu0/W7I/uF0Mfr90YIokDR5hBiNYnhUQq79bQ8X5ThPWFVTrcIHuDgVrd
+HmsS0YZ0b99rQQwnfSmRu+S1VZDURUwOowRM/Gge1BywlJ/aqNzcjS7HSxHOdxt
bDZ8XzqzwoSqhI7bktvlE3napVDTOCQoDh9g1EMVCYa+bMTBHpQMq6GOOmUNrYh5
O7GbX+c8aqg5pZ7fHhiN8C/16PDESb87N8JlV6zR
```

Apply Cancel

Fig. 14

3. Restart the Anybus CompactCom.

The installed certificates are listed on the internal web page (Security — Certificates).



Certificate Authorities

Subject	Issuer	Expires
Anybus CA	Anybus CA	2030-01-28 11:38:08

Device Certificates

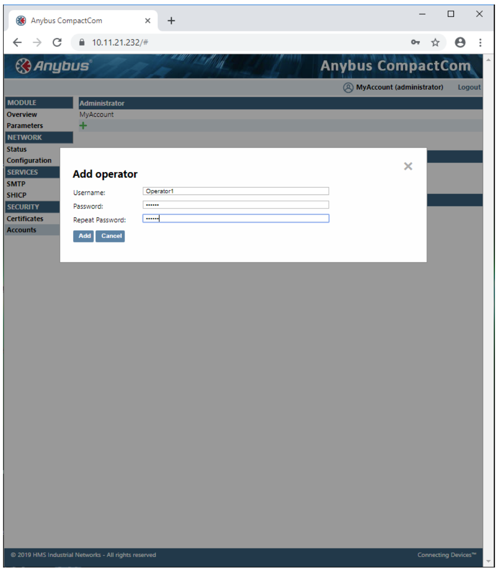
Subject	Issuer	Expires	Usage
10.11.21.232	Anybus CA	2030-01-31 09:46:51	□ OPC UA □ Secure-MQTT ☑ HTTPS

Fig. 15

3.6 Account Configuration

If you are logged in as administrator, you can add and configure user accounts.

1. Log in as administrator.
2. Select Accounts and then add an operator or user account.



Add operator

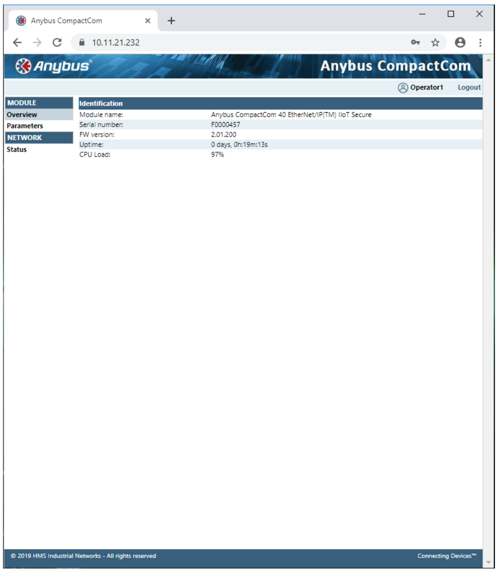
Username: Operator1

Password: ••••••

Repeat Password: ••••••

Fig. 16

Each user account is given a role, where each role is granted different access. The figure below shows the view that an operator will see. See *Authentication, Passwords, and User Roles, p. 26* for more information.



Identification	
Module name:	Anybus CompactCom 40 EtherNet/IP(TM) IIoT Secure
Serial number:	F0000457
FW version:	2.01.200
Uptime:	0 days, 0h:19m:13s
CPU Load:	97%

Fig. 17

4 Basic Operation

4.1 General Information

4.1.1 Software Requirements

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

- Certain functionality in the module requires that the command Get_Instance_Number_By_ Order (Application Data Object, FEh) is implemented in the host application.
- Up to 5 diagnostic instances (See *Diagnostic Object (02h), p. 149*) can be created by the host application during normal conditions. An additional 6th instance may be created in event of a major fault. This limit is set by the module, not by the network.
- EtherNet/IP in itself does not impose any specific timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period. The application that sends the request, also decides the timeout, e.g. EIPScan employs a timeout of 10 seconds.
- The use of advanced CIP-specific functionality may require in-depth knowledge in CIP networking internals and/or information from the official CIP and EtherNet/IP specifications. In such cases, the people responsible for the implementation of this product is expected either to obtain these specifications to gain sufficient knowledge or limit their implementation is such a way that this is not necessary.

See also...

- *Diagnostic Object (02h), p. 149* (Anybus Module Objects)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

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

4.1.2 Electronic Data Sheet (EDS)

On EtherNet/IP, the characteristics of a device is stored in an ASCII data file with the suffix EDS. This file is used by configuration tools etc. when setting up the network configuration. HMS Networks supplies a standard (generic) EDS file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the behavior of the product in ways which invalidate the generic EDS file. In such case, a custom EDS file needs to be created, which in turn invalidates the default identity information and require re-certification of the product.

Since the module implements the Parameter Object, it is possible for configuration tools such as RSNetWorx to automatically generate a suitable EDS-file. Note that this functionality requires that the command Get_Instance_Number_By_Order (Application Data Object, FEh) has been implemented in the host application.

See also..

- *Parameter Object (0Fh), p. 124* (CIP object)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

> **!** HMS Industrial Networks approves use of the standard EDS-file only under the condition that it matches the actual implementation and that the identity information remains unchanged.

4.2 Network Identity

By default, the module uses the following identity settings:

Vendor ID:	005Ah (HMS Industrial Networks)
Device Type:	002Bh (Generic Device)
Product Code:	005Eh (Anybus CompactCom 40 EtherNet/IP IIoT Secure)
Product Name:	“CompactCom 40 EtherNet/IP(TM)”

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

See also...

- *Identity Object (01h), p. 113* (CIP object)
- *EtherNet/IP Host Object (F8h), p. 222* (Host Application Object)

> **!** According to the CIP specification, the combination of Vendor ID and serial number must be unique. It is not permitted to use a custom serial number in combination with the HMS Vendor ID (005Ah), nor is it permitted to choose Vendor ID arbitrarily. Failure to comply to this requirement will induce interoperability problems and/or other unwanted side effects.
>
> To obtain a Vendor ID, contact the ODVA.

4.3 Authentication, Passwords, and User Roles

The secure protocols that are implemented in the Anybus CompactCom need to authenticate the users. Each user is assigned a role, that defines the user's access rights. The configuration of the users, their passwords and their roles, is administrated using the internal web interface. Role access rights are configured per protocol in a separate configuration file for each protocol. The table below shows the default role access rights. The configuration can be changed when developing an application.

Role	Description
administrator	• OPC UA access • account configuration • handling of certificates • firmware update possibility (via WebDAV) • access system configuration from internal web pages • get and set ADI data • access to module and network status and configuration information
operator	• OPC UA access • get and set ADI data • access to module and network status information The operator can view the module and network status information, but not set any system configuration. The operator is not granted any access to the security settings.
user	• get ADI data • access to module and network status information

4.4 Communication Settings

Network related communication settings are grouped in the Network Configuration Object (04h), and includes:

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 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 host application and the network through the built in web server, and through the TCP/IP Interface Object (CIP).

See also...

- *Secure Web Server (HTTPS), p. 37*
- *TCP/IP Interface Object (F5h), p. 138* (CIP object)
- *Ethernet Link Object (F6h), p. 141* (CIP object)
- *Network Configuration Object (04h), p. 151* (Anybus Module Object)
- *Secure HICP (Secure Host IP Configuration Protocol), p. 240*

4.4.1 Communication Settings in Stand Alone Shift Register Mode

If the Anybus CompactCom is used stand alone, there is no application from which to set the IP address. The IP address is instead set using the DIP1 switches (IP address byte 3) and the virtual attributes (Ethernet Host object (F9h), attribute #17), that are written to memory during setup (IP address byte 0 - 2). A flowchart is shown below.

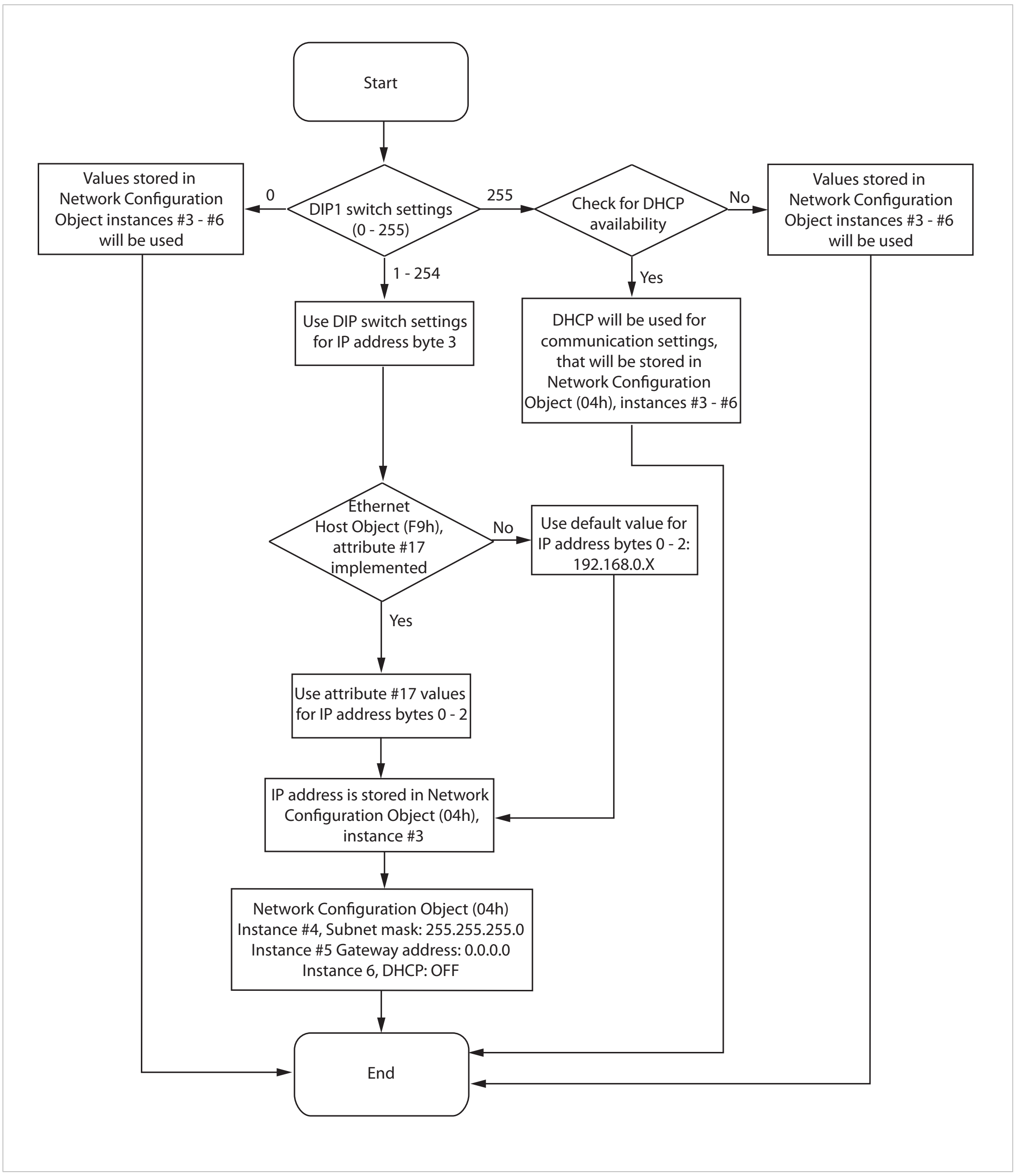

Start
DIP1 switch settings (0 - 255)
0
Values stored in Network Configuration Object instances #3 - #6 will be used
255
Check for DHCP availability
No
Values stored in Network Configuration Object instances #3 - #6 will be used
1 - 254
Use DIP switch settings for IP address byte 3
Yes
DHCP will be used for communication settings, that will be stored in Network Configuration Object (04h), instances #3 - #6
Ethernet Host Object (F9h), attribute #17 implemented
No
Use default value for IP address bytes 0 - 2: 192.168.0.X
Yes
Use attribute #17 values for IP address bytes 0 - 2
IP address is stored in Network Configuration Object (04h), instance #3
Network Configuration Object (04h) Instance #4, Subnet mask: 255.255.255.0 Instance #5 Gateway address: 0.0.0.0 Instance 6, DHCP: OFF
End

Fig. 18

See also ...

- *Ethernet Host Object (F9h), p. 231*
- Anybus CompactCom M40 Hardware Design Guide
- *Network Configuration Object (04h), p. 151*

4.5 Beacon Based DLR (Device Level Ring)

Device Level Ring (DLR) is a network technology for industrial applications that uses embedded switch functionality in automation end devices, such as programmable automation controllers and I/O modules, to enable Ethernet ring network topologies at the device level. DLR technology adds network resilience to optimize machine operation. Beacon based DLR networks consist of a ring supervisor and a number of ring nodes, and use “beacons” to detect breaks in the ring. When a DLR network detects a break in the ring, it provides ways to alternatively route the data to recover the network. Diagnostics built into DLR products can identify the point of failure, thus helping to speed maintenance and reduce repair time. The Anybus CompactCom 40 EtherNet/IP IIoT Secure implements the DLR protocol, and it is enabled by default. The device is able to process and act on beacon frames sent by ring supervisors, and supports beacon rates down to 100 µs. If needed, the DLR functionality can be disabled. This can be done by setting attribute #31 (Enable DLR) in the EtherNet/IP Host Object to False. See *EtherNet/IP Host Object (F8h), p. 222*.

4.6 Network Data Exchange

4.6.1 Application Data

Application Data Instances (ADIs) are represented through the ADI Object (CIP). Each instance within this objects corresponds directly to an instance in the Application Data Object on the host application side.

Accessible range of ADIs is 1 to 65535.

See also...

- *Parameter Object (0Fh), p. 124* (CIP object)
- *ADI Object (A2h), p. 134* (CIP object)

4.6.2 Process Data

Process Data is represented as dedicated instances in the Assembly Object (CIP).

See also...

- *Assembly Object (04h), p. 117* (CIP object)
- *Connection Manager (06h), p. 120* (CIP object)

4.6.3 Translation of Data Types

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

Anybus Data Type	CIP Data Type	Comments
BOOL	BOOL	Each ADI element of this type occupies one byte.
ENUM	USINT	
SINT8	SINT	
UINT8	USINT	
SINT16	INT	Each ADI element of this type occupies two bytes.
UINT16	UINT	
SINT32	DINT	Each ADI element of this type occupies four bytes.
UINT32	UDINT	
FLOAT	REAL	
CHAR	SHORT_STRING	SHORT_STRING consists of a single-byte length field (which in this case represents the number of ADI elements) followed by the actual character data (in this case the actual ADI elements). This means that a 10-character string occupies 11 bytes.

Anybus Data Type	CIP Data Type	Comments
SINT64	LINT	Each ADI element of this type occupies eight bytes.
UINT64	ULINT	
BITS8	BYTE	Each ADI element of this type occupies one byte.
BITS16	WORD	Each ADI element of this type occupies two bytes.
BITS32	DWORD	Each ADI element of this type occupies four bytes.
OCTET	USINT	
BITS1-7	BYTE	Bit fields of size 1 - 7
PAD0-8	BYTE	Bit fields of size 0 - 8 used for padding
PAD9-16	BYTE	Bit fields of size 9 - 16 used for padding
BOOL1	BOOL	

4.7 Web Interface

The web interface can be fully customized to suit a particular application. Dynamic content can be created by means of JSON. Data and web pages are stored in a FLASH-based file system, which can be accessed using a file transfer protocol.

See also...

- *File System, p. 30*
- *Secure Web Server (HTTPS), p. 37*
- *File Transfer Protocol (WebDAV), p. 76*
- *JSON, p. 46*

4.8 E-mail Client

The built-in e-mail client enables the host application to send e-mail messages stored in the file system, or defined directly within the SMTP Client Object (09h).

See also...

- *File System, p. 30*

4.9 Modular Device Functionality

Modular devices consist of a backplane with a certain number of slots. The first slot is occupied by the “coupler” which contains the Anybus CompactCom module. All other slots may be empty or occupied by modules.

When mapping ADIs to process data the application shall map the process data of each module in slot order.

A list of modules in a Modular Device is available to the EtherNet/IP network master by a request to the CIP Identity object.

See also ...

- “Modular Device Object (ECh)” (see Anybus CompactCom 40 Software Design Guide)
- *Identity Object (01h), p. 113* (CIP object)

4.10 File System

By default only the administrator role has access to the file system, and then only to the \firmware folder. To be able to customize this, the user has to enable admin mode in the Ethernet Host Object (F9h).

4.10.1 Overview

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

VFS	The virtual file system that e.g. holds the web pages of the module. The virtual file system is enabled by default in the Anybus File System Interface Object (0Ah).
Application	This directory provides access to the application file system through the Application File System Interface Object (EAh) (optional).
Firmware	Firmware updates are stored in this directory.

4.10.2 General Information

The built-in file system hosts 22 MByte of non volatile storage, which can be accessed by the HTTP and file transfer protocols, the email client, and the host application (through the Anybus File System Interface Object (0Ah)).

The maximum number of directories and files, that can be stored in the root directory, is 511 if only short filenames are used (8 bytes name + 3 bytes extension). The number of files that can be stored in other directories, than the root directory, is unlimited.

The file system uses the following conventions:

- \ (backslash) is used as a path separator
- Names may contain spaces, but must not begin or end with one.
- Valid characters in names are ASCII character numbers less than 127, excluding the following characters: \ / : * ? “ < > |
- Names cannot be longer than 48 characters
- A path cannot be longer than 126 characters (filename included)

See also ...

- *File Transfer Protocol (WebDAV), p. 76*
- *Secure Web Server (HTTPS), p. 37*
- *E-mail Client, p. 79*
- *Anybus File System Interface Object (0Ah), p. 189*
- *Application File System Interface Object (EAh), p. 212*

> ! *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

4.10.3 System Files

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

Example 1:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

5 EtherNet/IP Implementation Details

5.1 General Information

This chapter covers EtherNet/IP specific details in the Anybus implementation. Note that the use of such functionality may require in-depth knowledge in EtherNet/IP networking internals and/or information from the official EtherNet/IP and CIP specifications. In such cases, the people responsible for the implementation of this product are expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary. The EDS file must be changed to reflect all changes.

5.2 EtherNet/IP & CIP Implementation

By default, the module supports the generic CIP profile. Optionally, it is possible to re-route requests to unimplemented CIP objects to the host application, thus enabling support for other profiles etc.

To support a specific profile, perform the following steps:

1. Set up the identity settings in the EtherNet/IP Host Object according to profile requirements.
2. Implement the Assembly Mapping Object in the host application.
3. Set up the Assembly Instance Numbers according to profile requirements.
4. Enable routing of CIP messages to the host application in the EtherNet/IP Host Object.
5. Implement the required CIP objects in the host application.

See also...

- *Using the Assembly Mapping Object (EBh), p. 33*
- *EtherNet/IP Host Object (F8h), p. 222* (Host Application Object), details for the command Process_CIP_Object_Request.

5.3 Using the Assembly Mapping Object (EBh)

5.3.1 Introduction

This guide will describe how to map CIP instances to ADI data, using the assembly mapping object (EBh).

5.3.2 Adding Data - The Application Data Object

According to the Anybus object model, all data that is used in the application must be represented by application data instances (ADIs). ADIs are small portions of structured data, each representing only one of three possible different types: variable, array or structure.

See the Application Data Object (FEh) in the Software Design Guide for more information.

Below is an example with 30 ADIs. Instances 1 - 6 and 30 are implemented in the application, and 7 - 29 are not implemented.

Application Data Object (FEh) Instances

Instance #	Implemented	Order #
1	Yes	1
2	Yes	2
3	Yes	3
4	Yes	4
5	Yes	5
6	Yes	6
7...29	No	-
30	Yes	7

5.3.3 Grouping Data - The Assembly Mapping Object

The assembly mapping object makes it possible to create an arbitrary number of process data sets, called assembly mappings. Each assembly mapping instance represents a different logical set of process data, that can be chosen by the network and received over a single connection.

Every instance of the assembly mapping object, as seen below, contains an ADI map, referring to an arbitrary number of ADIs.

The instance numbers can be set freely.

Assembly Mapping Object (EBh) Instances

Instance #	Type	ADI Map
1	Read	1, 2
2	Read	2, 3
10	Write	3, 4, 30
11	Write	4, 5
30	Read	5, 6
51	Write	6, 30

There are two object instance attributes in the assembly mapping object, called Write PD Instance List and Read PD Instance List. These two attributes contain references to all read instances and all write instances, respectively. The example above will automatically generate the following content in these two attributes.

Name	Attribute	Values
Write PD Instance List	11	10, 11, 51
Read PD Instance List	12	1, 2, 30

The attributes Write PD Instance List and Read PD Instance List adopts the view of the network, e.g. an input will produce data on the network and an output will consume data on the network.

Write PD Instance List will contain all assembly mapping object instances with type “Read”. Read PD Instance List will contain all assembly mapping object instances with type “Write”.

5.3.4 Configuring CIP Assembly Numbers

The read and write instance list attributes in the assembly mapping object are bound to two corresponding attributes in the EtherNet/IP host object, according to the following table.

This routes application data to CIP assembly data, by linking CIP instance numbers to assembly mapping object instances.

The lists are matched index-wise, and must thus be of equal length.

Assembly Mapping Object Attribute	Value		Value	EtherNet/IP Host Object Instance Attribute
11 - Write PD Instance List	10	<—>	10	7 - Producing Instance Number
	11	<—>	22	
	51	<—>	100	
12 - Read PD Instance List	1	<—>	1	8 - Consuming Instance Number
	2	<—>	2	
	30	<—>	150	

For conformity with the CIP specification, both the Write_Assembly_Data and the Read_Assembly_Data services must be implemented.

5.3.5 Going Forward

During the initialization phase, in the NW_INIT state, all write assemblies (e.g. the instances of the assembly mapping object with type“write”) will be remapped to the write process data area. For this to happen, the device will issue the Remap_ADI_Write_Area command to the application data object in the host.

See the appendix about “Runtime Remapping of Process Data” in the Anybus CompactCom 40 Software Design Guide for more information.

When the network has been initialized, the device transitions from NW_INIT to the WAIT_PROCESS state. When the device receives a forward open request, the producing/consuming parameters in the request are verified and matched against the EtherNet/IP Host Object instance numbers (producing/consuming)

If the verification is successful, the read process data is remapped and the device transitions to the PROCESS_ACTIVE state. The I/O connection will then be established, and data can be exchanged over the network.

5.4 Socket Interface (Advanced Users Only)

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

See also..

- *Socket Interface Object (07h), p. 167* (Anybus Module Object)
- *Message Segmentation, p. 182*

5.5 Diagnostics

The severity value of all pending events are combined (using logical OR) and copied to the corresponding bits in the “Status” attribute of the Identity Object (CIP).

See also...

- *Identity Object (01h), p. 113* (CIP Object)
- *Diagnostic Object (02h), p. 149* (Anybus Module Object)

5.6 QuickConnect

The module supports the QuickConnect functionality. It is enabled in the EtherNet/IP Host Object. The module fulfills Class A with a startup time of less than 180 ms, with 16 bytes of I/O data mapped with parallel, SPI or shift register application interface.

See also ...

- *EtherNet/IP Host Object (F8h), p. 222* (Host Application Object)
- *TCP/IP Interface Object (F5h), p. 138* (CIP object)

5.7 CIP Safety

The Anybus CompactCom 40 EtherNet/IP IIoT Secure supports the CIP safety profile. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over EtherNet/IP using an add-on safety module, e.g. the IXXAT Safe T100. For an application to support CIP safety, the Functional Safety Object (E8h, host application object) has to be implemented. The Anybus CompactCom serial channel is used for the functional safety communication. When this channel is used for the host application, a second separate serial channel is implemented for the functional safety communication. See the Anybus CompactCom Hardware Design Guide for more information.

See ...

- *Functional Safety Module Object (11h), p. 194*
- *Functional Safety Object (E8h), p. 210*

5.7.1 Safety Module Firmware Upgrade

The firmware of the connected safety module can be upgraded through the Anybus CompactCom. The safety firmware (hiff file) has to be downloaded to the firmware directory in the Anybus CompactCom. At restart, the Anybus CompactCom detects and validates the firmware. Firmware upgrade in progress is indicated to the application by attribute #5 (instance #1) in the Functional Safety Object (E8h), which is set to TRUE during the firmware upgrade. The MS LED on the module will indicate by flashing red/green during firmware upgrade. The Anybus CompactCom will need more time to initialize , please do not restart the module during this time.

5.7.2 Reset Request from Network

When a reset request arrives from the network, a delay of 1 s is introduced before the Anybus CompactCom 40 EtherNet/IP IIoT Secure is reset, if CIP safety is enabled.

6 Secure Web Server (HTTPS)

6.1 General Information

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

HTTPS is always enabled to ensure security. When the Anybus CompactCom is delivered, the initial device identity certificate is enabled/activated for the HTTPS protocol to enable communication before the device is configured by the end user.

The web pages are stored in the file system, which can be accessed through the file transfer protocol (WebDAV) server. If necessary, the web server can be completely disabled in the Ethernet Host Object (F9h). HTTPS is enabled by default and can be disabled in the Ethernet Host Object (F9h), instance #1, attribute #3.

> ! To be able to modify configuration files and web pages, the user has to enable admin mode in the Ethernet Host Object (F9h), instance #1, attribute #7..

See also...

- *File Transfer Protocol (WebDAV), p. 76*
- *JSON, p. 46*
- *Certificates, p. 13*
- *Ethernet Host Object (F9h), p. 231*

6.2 Default Web Pages

The default pages are stored in \vfs. The first time the device is started, or after a factory reset request to the Anybus Object (01h), they allow for initial setup and configuration. For more information, see *Initial Setup and Account Configuration, p. 14*

Depending on the role of the user, the default web pages also may provide access to:

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

The default web pages are built of files stored in a virtual file system accessible through the vfs folder. These files are read only and cannot be deleted or overwritten. The web server will first look for a file in the web root folder. If not found it will look for the file in the vfs folder, making it appear as the files are located in the web root folder. By loading files in the web root folder with exactly the same names as the default files in the vfs folder, it is possible to customize the web pages, replacing such as pictures, logos and style sheets. To be able to replace the default web pages, the user has to enable admin mode in the Ethernet Host Object (F9h)

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

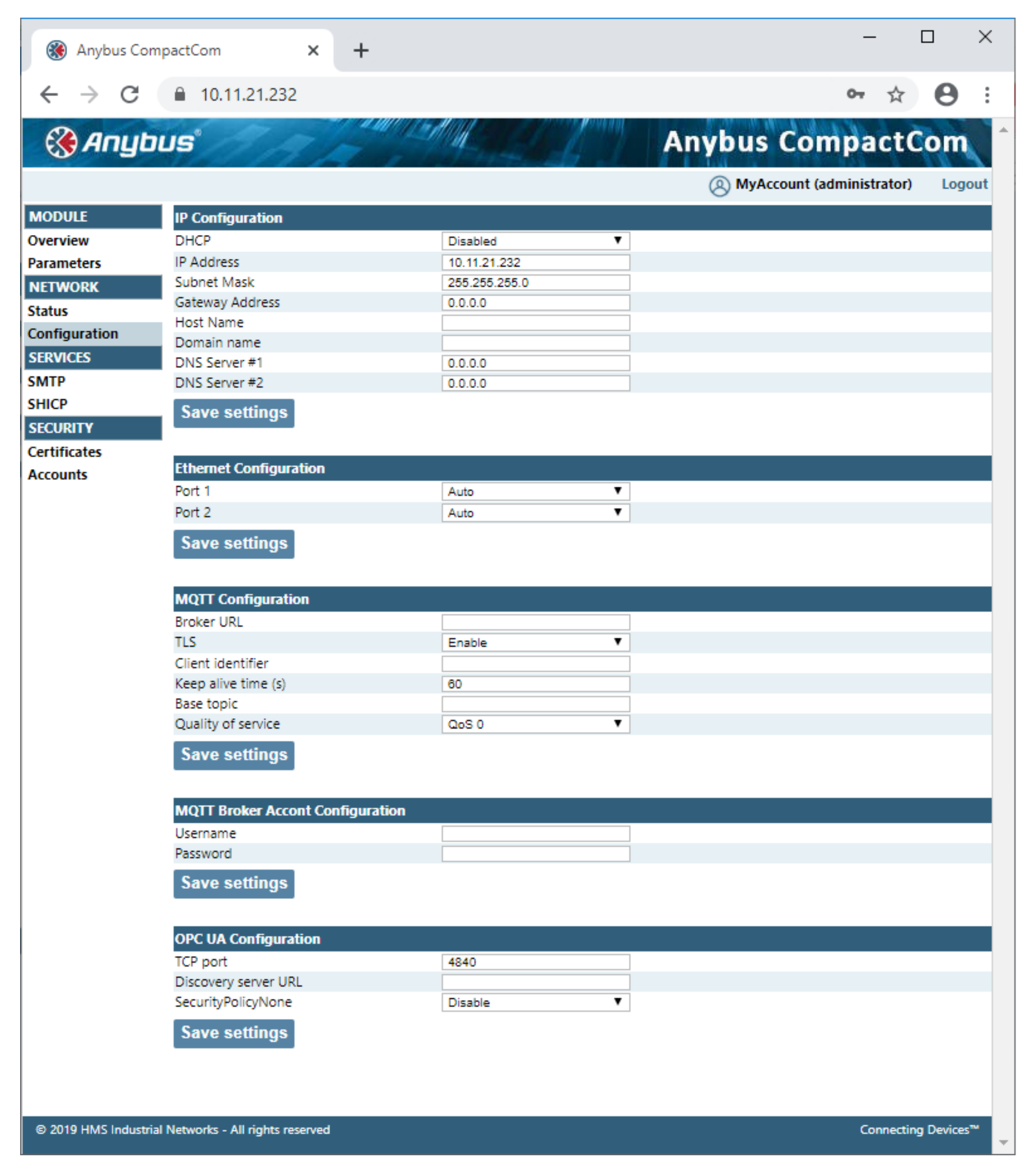
See also ...

- *File System, p. 30*

- *Anybus File System Interface Object (0Ah), p. 189*

6.2.1 Network Configuration

The network configuration page provides interfaces for changing TCP/IP settings in the Network Configuration Object.



IP Configuration	
DHCP	Disabled
IP Address	10.11.21.232
Subnet Mask	255.255.255.0
Gateway Address	0.0.0.0
Host Name	
Domain name	
DNS Server #1	0.0.0.0
DNS Server #2	0.0.0.0

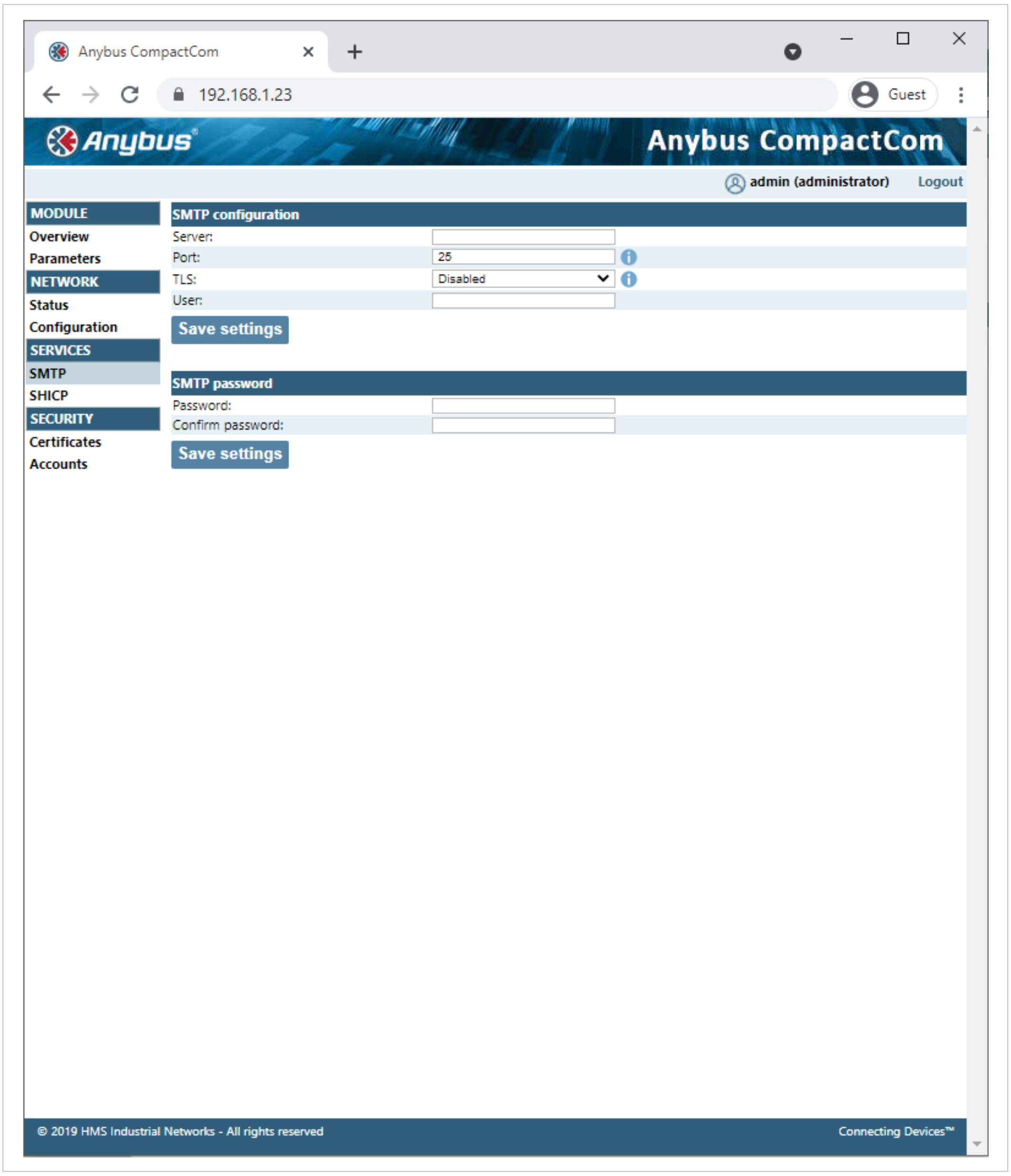
Ethernet Configuration	
Port 1	Auto
Port 2	Auto

MQTT Configuration	
Broker URL	
TLS	Enable
Client identifier	
Keep alive time (s)	60
Base topic	
Quality of service	QoS 0

MQTT Broker Accont Configuration	
Username	
Password	

OPC UA Configuration	
TCP port	4840
Discovery server URL	
SecurityPolicyNone	Disable

Fig. 19



SMTP configuration

Server:

Port: 25

TLS: Disabled

User:

SMTP password

Password:

Confirm password:

Fig. 20

The module needs to be reset for the TCP/IP and SMTP settings to take effect. The Ethernet Configuration settings will take effect immediately.

IP Configuration

The module needs a reset for any changes to take effect.

Name	Description
DHCP	Enable or disable DHCP Default value: enabled
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
Subnet mask	
Gateway	
Host name	IP address or name Max 64 characters
Domain name	IP address or name Max 48 characters
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
DNS 2	

Ethernet Configuration

Changes will take effect immediately.

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

SMTP Settings

The module needs a reset before any changes take effect

Name	Description
Server	IP address or name Max 64 characters
Port	Port number on SMTP server to connect to (1-65535) Default: SMTP = 25 SMTP TLS = 465
TLS	Enable or disable TLS
User	Max 64 characters
Password	Max 64 characters
Confirm password	

OPC UA Settings

These settings configure instances #40 - #42 of the Network Configuration Object.

Name	Description
TCP Port	OPC UA TCP Port Integer within the range 1 - 65535
Discovery Server URL	OPC UA Discovery Server URL 0 - 80 characters
SecurityPolicy None	Option to enable an Endpoint with SecurityPolicy None and UserIdentityToken with SecurityPolicy None

MQTT Client Settings

These settings configure instances #50 - #52 and #55 - #57 of the Network Configuration Object.

Name	Description
Broker URL	IP address or hostname 0 - 255 characters
TLS	Disable or Enable
Client Identifier	0 - 64 characters
Keep Alive (s)	Integer within the range 0 - 65535
Base topic	0 - 255 characters
Quality of service	Enumeration within the range 0 - 2

MQTT Broker Account Settings

These settings configure instances #53 - #54 of the Network Configuration Object.

Name	Description
Broker username	0 - 64 characters
Broker password	0 - 64 characters

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.

Ethernet/IP Statistics	Description
Established Class1 Connections	Current number of established class1 connections
Established Class3 Connections	Current number of established class3 connections
Connection Open Requests	Number of received connection open requests
Connection Open Format Rejects	Connection open requests rejected due to request format error
Connection Open Resource Rejects	Connection open requests rejected due to lack of resources
Connection Open Other Rejects	Connection open requests rejected due to other reasons
Connection Close Requests	Number of received connection open requests
Connection Close Format Rejects	Connection close requests rejected du to request format error
Connection Close Other Rejects	Connection close requests rejected due to other reasons
Connection Timeouts	Number of connection timeouts

MQTT State and Statistics	Description	
Broker address	Actual value of Network Configuration Object, Instance #50, Broker URL	
Connection status	State of the connection to the configured broker	
	Disconnected:	MQTT not started
	Connecting:	Connecting to the broker
	Connected:	Connected to the broker
	Rejected - <return description>	Connection rejected by the broker, description of the received return code
	Erroneous broker address	The broker address was not found on the netowrk or of an invalid format
	Failed	Connecting to the broker failed due to en internal error or a network error
Unexpected disconnections	Number of unexpected disconnections of the broker connection	
Connect errors	Number of failed connection attempts	

MQTT State and Statistics	Description
Successful publications	Number of successful publications
Publication errors too large	Number of publications too large to be transmitted to the network
Publication errors other	Number of publications that failed to be transmitted to the network

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.

Media Counters	Description
Alignment Errors	Frames received that are not an integral number of octets in length.
FCS Errors	Frames received that do not pass the FCS check.
Single Collisions	Successfully transmitted frames which experienced exactly one collision.
Multiple Collisions	Successfully transmitted frames which experienced more than one collision.
SQE Test Errors	Number of times SQE test error messages are generated. (Not provided with current PHY interface.)
Deferred Transmissions	Frames for which first transmission attempt is delayed because the medium is busy.
Late Collisions	Number of times a collision is detected later than 512 bit-times into the transmission of a packet.
Excessive Collisions	Frames for which a transmission fails due to excessive collisions.
MAC Receive Errors	Frames for which reception of an interface fails due to an internal MAC sublayer receive error.
MAC Transmit Errors	Frames for which transmission fails due to an internal MAC sublayer receive error.
Carrier Sense Errors	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame.
Frame Size Too Long	Frames received that exceed the maximum permitted frame size.
Frame Size Too Short	Frames received that are shorter than lowest permitted frame size.

6.3 Server Configuration

HTTPS is configured in the file http.cfg that is present in the vfs directory. The file can be overridden by creating http.cfg in the system root.

By default the three roles are configured to have HTTPS access according to this:

Role	Description/Default Configuration
administrator	• session timeout of 15 minutes (900 seconds) • full HTTPS access to web and API content
operator	• session time out of 15 minutes (900 seconds) • full HTTPS access to ADI API, meaning that an operator can get and set ADI data • access to module and network status information

Role	Description/Default Configuration
	The operator can view the module and network status information, but not set any system configuration. The operator is not granted any access to the security settings.
user	• no session timeout • access to module and network status information
$all	Reserved role. In order for non-authenticated users to access the login page, access is given to non-authenticated users to access images, stylesheets, some java script functionality, login and logoff pages as well as the “first run page”.

The default HTTPS configuration is stored in \vfs\http.cfg. It has the following content:

Content	Description

`[ServerName]`
`WebServerName`
	Configures the web server name included in the HTTP header of the responses from the module.

`[FileTypes]`
`FileType1:ContentType1`
`FileType2:ContentType2`
`...`
`FileTypeN:ContentTypeN`
	A list of file extensions and their reported content type.
	The default content types are listed in the table below.

`[IndexPage]`
`\web\index.html`
	Path to start page that will be returned if the url is empty.

`[LoginPage]`
`\web\login.html`
	Path to a web page that will be returned if login is required for access to a url.

`[FirstRunPage]`
`\web\firstrun.html`
	Path to a web page that will be returned if login is required for access to a url, but no accounts are configured.

`[SessionTimeout]`
`administrator:900`
`operator:900`
`user:0`
	Configures the session timeout per role (seconds).

`[Access]`
`administrator:\web`
`administrator:\api`
	Configures the access rights for each role. It is possible to give access to specific files or folders. Access to a folder gives access to all contents of the folder and subfolders.
	The default configuration of the device includes the three roles mentioned here. A custom configuration may change these roles.

`operator:\web\`
`operator:\api\adi\`
`operator:\api\module\status\`
`operator:\api\network\status\`	

`user:\web\`
`user:\api\adi\status\`
`user:\api\module\status\`
`user:\api\network\status\`	

`$all:\web\img\`
`$all:\web\css\`
`$all:\web\login.html`
`$all:\web\logoff.html`
`$all:\web\firstrun.html`
`$all:\web\js\csrf.js`
`$all:\web\js\jquery-1.9.1.js`
`$all:\api\security\config\addfirstaccount.json`
	$all is used for content that all shall have access to, including non-authenticated users.

See also...

- *File Transfer Protocol (WebDAV), p. 76*
- *JSON, p. 46*
- *Ethernet Host Object (F9h), p. 231*

6.3.1 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

6.4 Login

Login to the Anybus CompactCom is done by submitting a form with the following content to /login.cgi:

Username	Name of user to log in. Max 64 ASCII characters.
Password	Used to authenticate a user for a specific account. Max 64 ASCII characters
Target	Optional. If access is granted, the client is redirected to this location. Please note that if the user is not authorized to access “Target”, the user will still be logged in, but without access to “Target”.

The web server of the Anybus CompactCom will set the following cookies after a successful login:

Cookie	Description
wto	Contains creation time and user role. Signed by the server and cannot be modified by the client. Must accompany all requests. (Automatically added by all browsers).
csrf	Used to protect from CSRF attacks. The value of this cookie must be included as an HTML parameter (csrf) in all state-changing
role	The role of the user currently logged in. Can be used by the client web pages. (Information to client, not used by server) This cookie must be base64 encoded.
user	The name of the user currently logged in. Can be used by the client web pages. (Information to client, not used by server) This cookie must be base64 encoded.
stmo	The session timeout (in seconds) configured for the currently logged in role. Can be used by the client web pages. (Information to client, not used by server)
sret	Session retriggered. The value changes every time the session is used. Can be used by client web pages to trigger a timer used to know when a web session is going to time out. (Information to client, not used by server)

6.5 Logout

Logout is done by submitting a form with the following content to /logout.cgi:

Target	Optional. The client is redirected to this location.

/logout.cgi deletes all session cookies.

6.6 Cross Site Request Forgery (CSRF) Protection

The Anybus CompactCom web server uses a token based authentication system, where the authentication token is stored in a cookie (wto). Token based systems are prone to “Cross Site Request Forgery” (CSRF) attacks. The default web server includes protection against this type of attacks. If you are to develop custom web pages, see *Cross Site Request Forgery (CSRF) Protection, p. 47* (JSON) for more information.

7 JSON

7.1 General Information

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

A simple example of how to create a web page using JSON is added at the end of this chapter.

For easier security configuration all JSON functionality is accessed through the following URL: `api/[functionality]/[status/config]/[function].json`.

URL Part	Description
functionality	Categorizes the functionality that the JSON function accesses. The functionality may be split into multiple parts for sub-functionality.
status/config	Indicates if the function is only reading data or also can set data/configuration.
function	JSON function

7.1.1 Encoding

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

7.1.2 Access

> **!** Be careful not to provide JSON API access to unauthorized users.

Account configuration is done in a file called http.cfg in the root directory . The file is described in the “Web Server” section in this document.

7.1.3 Security

Account configuration, certificate installation and usage configuration can be performed using the JSON API. All security JSON functions are accessed through the api/security/ URL.

7.1.4 Error Response

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

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

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

7.2 Cross Site Request Forgery (CSRF) Protection

The Anybus CompactCom web server uses a token based authentication system, where the authentication token is stored in a cookie (wto). Token based systems are prone to “Cross Site Request Forgery” (CSRF) attacks.

In order to protect against CSRF attacks, the web server will always set a cookie named “csrf” which will be stored in the browser. Each request, submitting data which will have a state changing effect on the Anybus CompactCom, needs to read the value of the cookie and include it as a parameter named “csrf” in the request. The Anybus CompactCom module will only accept the request if the value in the csrf parameter matches the value in the csrf cookie. For a complete list of which API functions that are state changing see *Supported JSON functions, p. 48*.

This means that users creating web pages submitting state changing requests must handle this. This could for example be done by adding a hidden HTML input named “csrf” to every submitted HTML form. At submission of the HTML form, using Javascript, the value of the csrf input shall be populated with the data read from the csrf cookie.

7.3 Supported JSON functions

The table below lists all supported JSON functions in the Anybus CompactCom 40 EtherNet/IP IIoT Secure. The functions, that are considered state changing, require CSRF protection.

Function	State Changing
api/adi/status/info.json	No
api/adi/status/data.json	No
api/adi/status/metadata.json	No
api/adi/status/metadata2.json	No
api/adi/status/enum.json	No
api/adi/config/update.json	Yes
api/module/status/info.json	No
api/network/status/ethstatus.json	No
api/network/status/ipstatus.json	No
api/network/config/ethconf.json	Yes
api/network/config/ipconf.json	Yes
api/network/config/nwconf.json	Yes
api/network/status/ifcounters.json	No
api/network/status/mediacounters.json	No
api/network/status/nwstats.json	No
api/services/config/smtp.json	Yes
api/security/status/accounts.json	No
api/security/config/addfirstaccount.json	Yes
api/security/config/addaccount.json	Yes
api/security/config/deleteaccount.json	Yes
api/security/status/status.json	No
api/security/status/cacerts.json	No
api/security/status/devcerts.json	No
api/security/status/protocolinfo.json	No
api/security/config/installcacert.json	Yes
api/security/config/installdevcert.json	Yes
api/security/config/deletecacert.json	Yes
api/security/config/deletedevcert.json	Yes
api/security/config/cfgcertusage.json	Yes
api/security/config/shicpconf.json	Yes

7.4 JSON API

7.4.1 ADI

info.json

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

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

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

JSON response example:

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

data.json

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

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

Request data:

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

Response data:

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

JSON response example (using offset):

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

JSON response example (using inst):

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

metadata.json

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

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

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

The ADI metadata is static during runtime.

Request data:

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

Response data:

Name	Data Type	Description
instance	Number	-
name	String	May be NULL if no name is present.
numelements	Number	-
datatype	Number	-
min	String	Hex formatted string, see *Hex Format Explained, p. 73* for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see *Hex Format Explained, p. 73* for more information. May be NULL of no maximum value is present.
access	Number	Bit 0: Read access Bit 1: Write access

JSON response example (using offset):

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

JSON response example (using inst):

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

metadata2.json

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

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

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

The ADI metadata is static during runtime.

Request data:

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

Response data:

Name	Data Type	Description
instance	Number	-
numelements	Array of umbers	-
datatype	Array of Numbers	Array of datatypes. For Structures and Variables, each array element defines the data type of the corresponding element of the instance value. For Arrays, one array element defines the data type for all elements of the instance value.
descriptor		Array of descriptors. For Structures and Variables, each array element defines the descriptor of the corresponding element of the instance value. For Arrays, one array element defines the descriptor for all elements of the instance value.
name		May be NULL if no name is present.
min	String	Hex formatted string, see *Hex Format Explained, p. 73* for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see *Hex Format Explained, p. 73* for more information. May be NULL of no maximum value is present.
default	String	Hex formatted string, see *Hex Format Explained, p. 73* for more information. May be NULL of no default value is present.
numsubelements	Array of Numbers	For Structures and Variables each array element defines the number of subelements of the corresponding element of the instance value. May be NULL if not present.
elementname	Array of Strings	Array of names, one for each instance value element. May be NULL if not present.

JSON response example (using offset):

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

JSON response example (instance):

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

enum.json

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

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

The ADI enum strings are static during runtime.

Request data:

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

Response data:

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

JSON response example:

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

update.json

```
POST api/adi/config/update.json
```

Form data:

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

This function updates the value attribute of an ADI.

Request data:

Name	Data Type	Description
inst	Number	Instance number of the ADI
value	String	Value to set. If the value attribute is a number it shall be hes formatted, see *Hex Format Explained, p. 73* for more information.
elem	Number	Optional. If specified only a single element of the ADI value is set. Then <data> shall only contain the value of the specified <element>.
callback	String	Optional. A callback function for JSONP output.

Response data:

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

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

7.4.2 Module

info.json

```
GET api/module/status/info.json
```

Response data:

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

JSON response example:

```
{
   "modulename":  "ABCC M40",
   "serial":      "ABCDEF00",
   "fwver":       [ 1, 5, 0 ],
   "uptime":      [ 5, 123456 ],
   "cpuload":     55,
   "fwvertext":   "1.05.02",
   "vendorname":  "HMS Industrial Networks",
   "hwvertext":   "2",
   "networktype": 133,
}
```

7.4.3 Network

ethstatus.json

```
GET api/network/status/ethstatus.json.
```

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

Comm Object Definition:

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

JSON response example:

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

ipstatus.json & ipconf.json

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

```
GET api/network/status/ipstatus.json
```

or

```
GET api/network/config/ipconf.json
```

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

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

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

Example:

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

ethconf.json

```
GET api/network/config/ethconf.json
```

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

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

```
{
   "mac":        [00, 48, 17, 255, 2, 1],
   "comm1":      0,
   "comm2":      4
}
```

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

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

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

```
{
   "mac":        [00, 48, 17, 255, 2, 1],
   "comm1":      0,
   "comm2":      { error: 14 },
}
```

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

ifcounters.json

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

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

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

mediacounters.json

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

The argument <port> is either 1 or 2.

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

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

nwstats.json

```
GET api/network/status/nwstats.json
```

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

Example output:

```
[]
or
[ { "identifier": "eipstats", "title": "EtherNet/IP Statistics" } ]
```

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

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

“eipstats”

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

“eitstats”

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

“bacnetipstats”

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

“bacnetaplserverstats”

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

“bacnetaplclientstats”

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

“bacnetalarmstats”

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

"eplifcounters"

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

"ectstats"

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

"eoeifcounters"

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

"pnpof"

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

nwconf.json

```
GET api/network/config/nwconf.json
```

This is a product specific JSON function that can provide network specific configurations. The function lists available configuration options.

Response:

The response is a list with identifiers and titles of possible network specific configuration groups.

Example when product does not support any network specific configuration:

```
[]
```

Example when product supports OPC UA configurations:

```
[[ { "identifier": "opcua", "title": "OPC UA Configuration" } ]
```

Example when product supports OPC UA and FOO configurations:

```
[
  { "identifier": "opcua",   "title": "OPC UA Configuration" },
  { "identifier": "foo", "title": "FOO Configuration" },
]
```

To get current configuration for a specific configuration group use:

```
GET network/nwconf.json?get=[identifier]
```

The response is a list of current configurations of “identifier” containing:

name	Name of configuration parameter, used to identify the parameter on set requests.
value	Configuration value
min	Optional: Min value to accept.
max	Optional: Max value to accept
type	Optional: Indicate the HTML input type preferred to show the value
maxlength	Optional: Indicates the max length of value.enumstrings
enumstrings	Optional: For enum settings. A list of selectable value strings

Example to get an OPC UA configuration:

```
GET network/nwconf.json?get=opcua

[
   { "name" : "TCP port",             "value" : 4840 },
   { "name" : "Discovery server URL", "value" : "",    "type" : "text" }
]
```

Example to get an MQTT configuration:

```
GET network/nwconf.json?get=mqtt

[
   {"name":"Broker URL","value":"","type":"text","maxlength":64},,
   {"name":"Client identifier","value":"","type":"text","maxlength":23},
   {"name":"Keep alive time (s)","value":60,"type":"number","min":0,
          "max":65535},
   {"name":"Broker username","value":"","type":"text","maxlength":16},
   {"name":"Broker password","value":"","type":"password",
          "maxlength":32},
  {"name":"Base topic","value":"","type":"text","maxlength":128},
  {"name":"Quality of service","value":0,"enumstrings":
          ["QoS 0","QoS 1","QoS 2"]}
]
```

To set current configuration for a specific configuration group use:

```
GET network/nwconf.json?set=[identifier]&param1=value1&param2=value2…
```

The response contains:

result	The result code is a standard ABCC (ABP) error code (0=Success).
message	Optional; Indicates a response message to the user.

Example to set OPC UA TCP Port:

```
GET network/nwconf.json?set=opcua&TCP port=4841

{ "result" : 0 }
```

7.4.4 Services

api/services/config/smtp.json

```
GET api/services/config/smtp.json
```

ⓘ *Password is not returned when retrieving the settings.*

Name	Data Type	Note
server	String	Server URL, 64 characters in the format [<protocol>://]<server address>[:<port>]. See *SMTP Server URL format, p. 156* for URL format options.
user	String	-

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

Set:

Form data:

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

7.4.5 Security

This section describes the JSON API for account configuration, certificate installation and usage configuration.

status.json

```
GET api/security/status/status.json
```

This object is used to get general security status information,

Response data:

Name	Data Type	Note
storage:total	Integer	Number of bytes totally available for security content storage
storage:free	Integer	Number of bytes currently available for security contents storage

JSON response example:

```
{"storage":
   {
      "total":241664",
      "free":151552
    },
    "device_certificates":
       "slots":16,
       "free":15
    }
}
```

accounts.json

```
GET api/security/status/accounts.json
```

This object is used to receive an array of all configured accounts.

Response data:

Name	Data Type	Note
username	String	Account username
role	String	Account role

JSON response example:

```
{
      "<username1>":{"role" : "account1 role"   },
      "<username2>":{"role" : "account2 role"   },
       ,,,
      "<usernameN>":{"role" : "accountX role"   },
}
```

addfirstaccount.json & addaccount.json

```
GET api/security/config/addfirstaccount.json?
                Username=<username>&Password=<password>&Role=<role>
GET api/security/config/addaccount.json?
                Username=<username>&Password=<password>&Role=<role>
```

The function addfirstcount.json is used from a “first run page” to create the first user account. It will only be accepted if no previous accounts exist.

The function addaccount.json is used to add a user account.

Request data:

Name	Data Type	Note
Username	String	Account username (see Account Configuration)
Password	String	Account password (see Account Configuration)
role	String	Account role (see Account Configuration)

JSON success response example:

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

JSON failure response example:

Name	Data Type	Note
result	Integer	The result of the operation (0 = success)
errordesc	String	A readable description of the fault. Included in case of error.

```
{
   "result" : x,
   "errordesc" : "Failure description"
}
```

deleteaccount.json

```
GET api/security/config/deleteaccount.json?Username=<username>
```

This function deletes a user account.

Request data:

Name	Data Type	Note
Username	String	User name of account to delete (see Account Configuration)

JSON success response example:

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

JSON failure response example:

Name	Data Type	Note
result	Integer	The result of the operation (0 = success)
errordesc	String	A readable description of the fault. Included in case of error.

```
{
   "result" : x,
   "errordesc" : "Failure description"
}
```

protocolinfo.json

```
GET api/security/status/protocolinfo.json
```

This function is used to get information about number of certificates that can be configured for each supported security protocol.

JSON response example:

```
{
       "https" : {  "max_certs"  :  2 },
        …
      "protocol_x" : {  "max_certs"  :  1 }
}
```

7.4.6 cacerts.json & devcerts.json

```
GET api/security/status/cacerts.json
GET api/security/status/devcerts.json
```

The function cacerts.json gets an array of descriptions of installed CA certificates.

The function devcerts.json gets an array of descriptions of installed device certificates.

Response data:

For properties not existing in the certificate null will be given as value. See “nscerttype” below.

For each certificate in the list, the following attributes are given:

Attribute Name	Data Type	Description
certificate:version	Integer	Certificate version
certificate:serial	String	Certificate serial number
certificate:subject	String (comma separated)	Certificate subject name
certificate:issuer	String (comma separated)	Certificate issuer name
certificate:expires	DateString (see description below)	Date when certificate expires
certificate:issued	DateString (see description below)	Date when certificate was created
certificate:sigalg	String	Algorithm certificate is signed with
certificate:keytype	String	Type of the key used by this certificate
certificate:keysize	Integer	Size of the key (number of bits)
certificate:basicconst	String	Certificate basic constraints
certificate:subjaltname	String (comma separated)	Subject alternate namesCould be DNS names and/or IP Numbers
certificate:nscerttype	String	Netscape certificate type description
certificate:keyusage	String	Key usage description
certificate:thumbprint	String	The SHA1 sum of the certificateThis is a 40 byte hexadecimal formatted string that can be used to identify the certificate. This thumbprint will internally be used as the filename of the certificate.
usage:<protocol>	Bool (true, false)	For each, by the ABCC 40 security module supported secure protocols, a protocol name attribute is included with boolean value indicating if the certificate is configured for usage with the protocol Note that this attribute is only present for devcerts.json

DateString format: YYYY-MM-DD hh:mm:ss

YYYY	four-digit year
MM	two-digit month (01=January, etc.)
DD	two-digit day of month (01 through 31)
hh	two-digit of hour (00 through 23)
mm	two-digit of minute (00 through 59)
ss	two-digit of second (00 through 59)

JSON response example (cacerts.json):

The hex string at the beginning of the example is the SHA1 thumbprint of the certificate.

```
    {
       "24C7E186CA125AB5C49CA6945E3D37D85B84FACF" :
       {
          "certificate" :
          {
              "version" :     3,
              "serial" :      "C0:78:27:6E:A6:25:46:23",
              "subject" :     "C=SE, ST=Halland, L=Halmstad, O=HMS,
                   OU=Dev, CN=HMS-CA",
              "issuer" :      "C=SE, ST=Halland, L=Halmstad, O=HMS,
                   OU=Dev, CN=HMS-CA",
              "issued" :      "2017-01-19 14:43:54",
              "expires" :     "2027-01-17 14:43:54",
              "sigalg" :      "RSA with SHA-256",
              "keytype" :     "RSA",
              "keysize" :     2048,
              "basicconst":   "Subject Type=CA, Path Length Constraint=0",
              "subjaltname":  "IP=10.10.12.88, DNS=abccmodule.hms.se",
              "nscerttype" :  null,
              "keyusage" :    "Key Cert Sign",
              "thumbprint"    "24C7E186CA125AB5C49CA6945E3D37D85B84FACF",
           }
       } ,
       "F3A5EF014702937F37AC540898F36235E7A435B3" : {
             "certificate" : { Certificate description… }
        },
       …
       "A3C72403A85EA577DEB4661772E2D1D4B99904D2" : {
              "certificate" : { Last certificate… }
    ]
```

JSON response example (devcerts.json):

The hex string at the beginning of the example is the SHA1 thumbprint of the certificate.

```
    {
       "F3A5EF014702937F37AC540898F36235E7A435B3" :
       {
          "certificate" :
          [
          {
              "version" :     3,
              "serial" :      "02",
              "subject" :     "C=SE, ST=Halland, L=Halmstad,
                    O=HMS Industrial Networks AB, OU=BU Anybus,
                    CN=10.11.20.55",
              "issuer" :      "C=SE, ST=Halland, L=Halmstad,
                    O=HMS Industrial Networks AB, OU=BU Anybus,
                    CN=10.11.20.55",
              "issued" :      "2019-05-29 06:41:21",
              "expires" :     "2020-05-28 06:41:21",
              "sigalg" :      "ECDSA with SHA256",
              "keytype" :     "EC",
              "keysize" :     256,
              "basicconst":  null,
              "subjaltname":  "IP=10.11.20.55,",
              "nscerttype" :  null,
              "keyusage" :   null,
              "thumbprint"    "F3A5EF014702937F37AC540898F36235E7A435B3",
           }
        ],
       "usage" : { "https" : true }
        }
```

```
}
```

7.4.7 installcacert.json & installdevcert.json

Install a CA certificate:

```
POST api/security/config/installcacert.json
```

Request data:

Name	Data Type	Note
CaCert	String	CA certificate in PEM format

Install a device certificate:

```
POST api/security/config/installdevcert.json - formdata:
   DevCert=<Device certificate in PEM format>&DevKey=
                     <Device certificate private key in PEM format>
```

Request data:

Name	Data Type	Note
DevCert	String	Device certificate in PEM format
DevKey	String	Device certificate private key in PEM format

Response data:

Name	Data Type	Note
result	Integer	The result of the operation (0 = success)
errordesc	String	A readable description of the fault in case of error.

JSON success response example:

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

JSON error response example:

```
{
   "result" : 21,
   "errordesc":"Failed to parse certificate"
}
```

7.4.8 deletecacert.json & deletedevcert.json

Delete a CA certificate:

```
GET api/security/config/deletecacert.json?thumbprint=
     <SHA1 thumbprint of certificate to delete>
```

Delete a device certificate:

```
GET api/security/config/deletedevcert.json?thumbprint=
     <SHA1 thumbprint of certificate to delete>
```

Response data:

Name	Data Type	Note
result	Integer	The result of the operation (0 = success)

JSON success response example:

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

JSON error response example:

```
{
   "result" : 21,
   "errordesc":"Failed to parse certificate"
}
```

7.4.9 cfgcertusage.json

```
GET api/security/config/cfgcertusage.json?<DevThumbprint>=<ProtocolName>
```

This function configures the usage of certificates for specific protocols.

Attribute Name	Data Type	Description
DevThumbprint	40 octet HEX string	The SHA1 thumbprint of the certificate to configure usage for.
Protocolname	Shall be one of the protocol names provided by security/certusage.json.	Name of the protocol to configure usage for.

Response data:

Attribute Name	Data Type	Description
result	Integer	The result of the operation (0 = success)

JSON response example:

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

JSON error response example:

```
{
   "result":12
   "errordesc":" Unsupported protocol"
  }
```

7.4.10 Hex Format Explained

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

Examples:

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

```
0500
```

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

```
414243
```

7.5 Example

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

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

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

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

8 File Transfer Protocol (WebDAV)

WebDAV is an extension to the HTTPS protocol, giving access to the file system of the Anybus CompactCom. It replaces FTP that was the standard protocol when downloading files in earlier versions of Anybus CompactCom 40 EtherNet/IP IIoT Secure. Using a separate port number for WebDAV makes it possible to block WebDAV operations in routers and firewalls, but still letting web traffic through. WebDAV also offers the possibility to add the Anybus CompactCom as a network drive in Microsoft Windows.

The following port number is used for WebDAV communication:

- TCP port 4443

WebDAV is enabled by default and can be disabled in the Ethernet Host Object (F9h), instance #1, attribute #25. If WebDAV is turned off, it is not possible to update the module firmware using File Download or Firmware Manager.

See also...

- *Ethernet Host Object (F9h), p. 231*

8.1 WebDAV Configuration

Accounts can be added and removed using the default web pages. Role access capacities are configured in the file webdav.cfg. By default only the administrator role has access to the file system, and is configured to have access to \firmware. To change the configuration the Anybus CompactCom must be set in admin mode.

By default there are three roles, administrator, operator and user. The number of roles and their capabilities can be defined differently.

The following configuration options can be set in \webdav.cfg:

[HomeDir]	Configures locations from system root where each role has its home directory. Once logged in to WebDAV this directory will be presented as the user’s root directory. Format: `role:path`
[SessionTimeout]	Session timeout in seconds. If not configured or set to 0, no timeout will be used. (Optional) Format: `role:timeout`

It is recommended not to configure access to system root, as this would also give access to the system configuration.

The default WebDAV configuration is stored in /vfs/webdav.cfg and has the following content:

```
[HomeDir]
administrator:\firmware

[SessionTimeout]
administrator:900
```

8.2 WebDAV

If a trusted certificate is configured for HTTPS, the file system of the Anybus CompactCom 40 EtherNet/IP IIoT Secure can be mapped as a network drive on your PC. See *File Transfer Protocol (WebDAV), p. 76* for more information.

1. Select Map network drive on your PC.

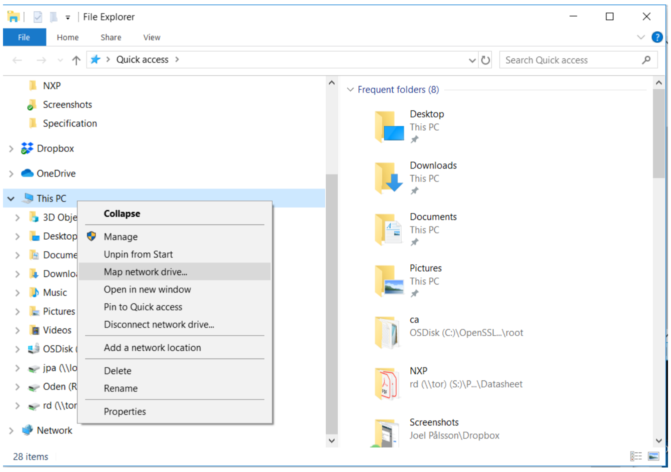
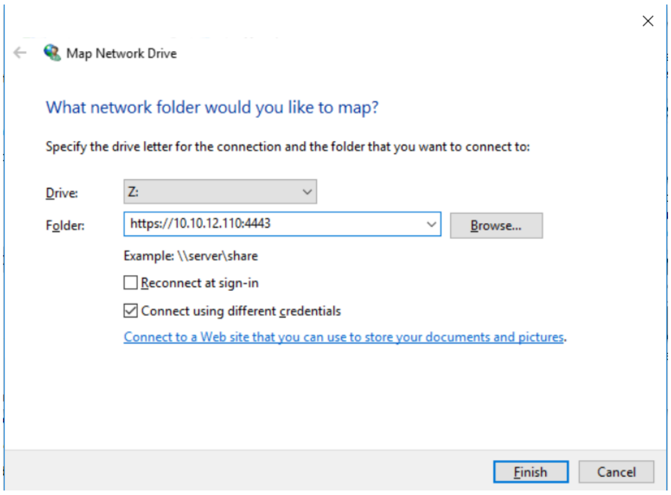


Fig. 21

2. Enter the module IP address (or DNS name) followed by 4443 which is the TCP port number used for WebDAV.



What network folder would you like to map?

Specify the drive letter for the connection and the folder that you want to connect to:

Drive: Z:

Folder: https://10.10.12.110:4443

Example: \\server\share

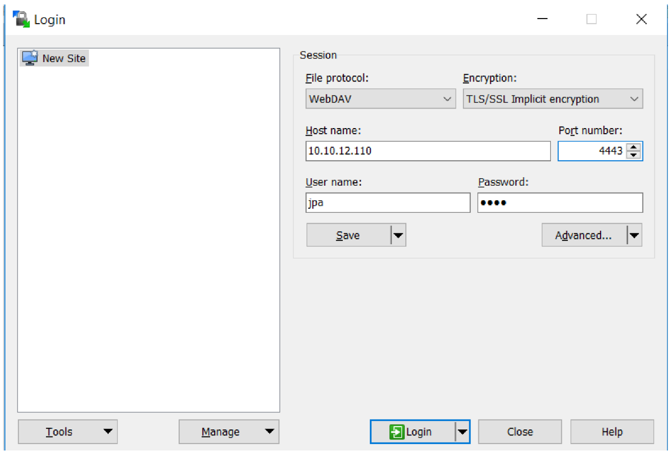
☐ Reconnect at sign-in

☑ Connect using different credentials

Connect to a Web site that you can use to store your documents and pictures.

Fig. 22

3. The WebDAV client, that is embedded in Windows, demands a certificate that is trusted by Windows. The file system of a device that is not correctly configured can be accessed by e.g. WinSCP.



Login

Session

File protocol: WebDAV

Encryption: TLS/SSL Implicit encryption

Host name: 10.10.12.110

Port number: 4443

User name: jpa

Password: ••••

Fig. 23

WinSCP is shown in the figure, but other third party programs, supporting WebDAV, are available.

9 E-mail Client

9.1 General Information

The built-in e-mail client allows the application to send e-mail messages through an SMTP server. Messages can either be specified directly in the SMTP Client Object (04h), or retrieved from the file system.

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

Supported protocols are SMTP and SMTPS (SMTP over TLS). To use SMTPS, the root CA certificate of the SMTP server must be installed in Certificate Authorities under Security > Certificates.

9.2 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 URL
- A valid username
- A valid password

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

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

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

10 OPC UA

10.1 General

The OPC Unified Architecture standard makes it possible to exchange information among devices from multiple vendors. It is platform independent and connects the industry to IT. You can sit at your local PC or handheld device and exchange information with any device that is modelled on OPC UA.

The Anybus CompactCom implements an OPC UA server which models the Anybus CompactCom as a device in its address space using the OPC UA Device model. The modelled device is of type CompactCom40DeviceType which is a subtype of the OPC UA DeviceType. It is possible for the user to change certain parameters e.g. the name of the device and its device type, to make the application manufacturer specific. See *CompactCom 40 Device Type Information Model, p. 82*.

It is also possible to model and download an application defined information model based on e.g. an existing Companion specification. See *Application Defined Information Model, p. 93*.

The functionality is disabled by default. It has to be enabled during startup of the Anybus CompactCom. This is done by modifying attribute #1 (OPC UA Model), in the OPC UA Object (E3h), instance #1. Set the attribute to 1 for the CompactCom40DeviceType and 2 for the downloaded application defined information model.

See also...

- opcfoundation.org
- *OPC UA Object (E3h), p. 207*
- “Application Object (FFh)” (see Anybus CompactCom 40 Software Design Guide)

10.2 Configuration

10.2.1 Parameters

If OPC UA is enabled, the Anybus CompactCom will set up a default configuration for the parameters needed. It is possible to change this configuration, either by setting the values of the instances in the Network Configuration Object, or by using the internal web page.

The table below lists the parameters that are configurable via the internal web page and the Network Configuration Object.

Parameter	Instance No. (in the Network Configuration Object)	Default
OPC UA TCP Port	40	4840
OPC UA Discovery Server URL	41	Empty string
OPC UA SecurityPolicyNone	42	Disabled

10.2.2 Access Configuration

The access control configuration is performed in a configuration file, stored in the file system of the CompactCom. The file is named “opcua.cfg” and has the following default content:

```
[Access]
administrator
operator
user:r,b
```

This configuration can be changed by creating an opcua.cfg file in the file system root. Set the Anybus CompactCom in admin mode in the Ethernet Host Object (F9) to expose the file system root on WebDAV.

Each line of the Access section configures an access rule with the following format:

```
<role>:<access rights>:<namespace index>
```

Fields that have specified default behavior can be omitted.

Field	**Description**
Role	Name of a role. Must be present in the user database.
Access rights	Comma separated list with access rights. Possible access rights are: r – Gives read access of the value attribute of variable nodes w – Gives write access of the value attribute of variable nodes b – Gives browse access of nodes No specified access rights implies full access
Namespace index	Namespace index of the namespace to apply the access rights on. No specified namespace implies all namespaces.

Some examples on how the access rules can be formatted:

Example	
operator	The role operator gets full access to all namespaces
operator:b	The role operator gets browse access to all namespaces
operator:b:0	The role operator gets browse access to namespace 0
operator::0	The role operator gets full access to namespace 0
operator:r,b operator:w:4	The role operator gets read and browse access to all namespaces and write access to namespace 4
operator:r,w,b operator:r:4	The role operator gets read, write and browse access to all namespaces (the second access rule does not affect the access in this case)

10.3 CompactCom 40 Device Type Information Model

The CompactCom40DeviceType is a subtype of the DeviceType of the OPC Foundation's Device Integration (OPC UA DI) model. The CompactCom40 instance of this subtype is organized by the DeviceSet node, which is a well-known node defined by the OPC UA device integration model. The CompactCom40DeviceType inherits several mandatory properties from the device type in the device integration model. These properties present some asset information about the device. The properties that can be changed from the application are: SerialNumber, Manufacturer, Model, SoftwareRevision and HardwareRevision. They correspond to certain attributes in the OPC UA Object (E3h) and the Application Object (FFh) of the Anybus CompactCom 40. The ADIs are represented as a set of parameters and are modelled as components to the ParameterSet, an object that is a component of the CompactCom40 device type.

The picture below shows how the device instance CompactCom40 of type CompactCom40DeviceType is structured. The names of the instance and the subtype can be changed to reflect the application. The variables on the left are used to identify the device. They are inherited from the OPC UA DeviceType and are mandatory. The variables of the object ParameterSet are examples of how ADIs are modelled in the OPC UA address space.

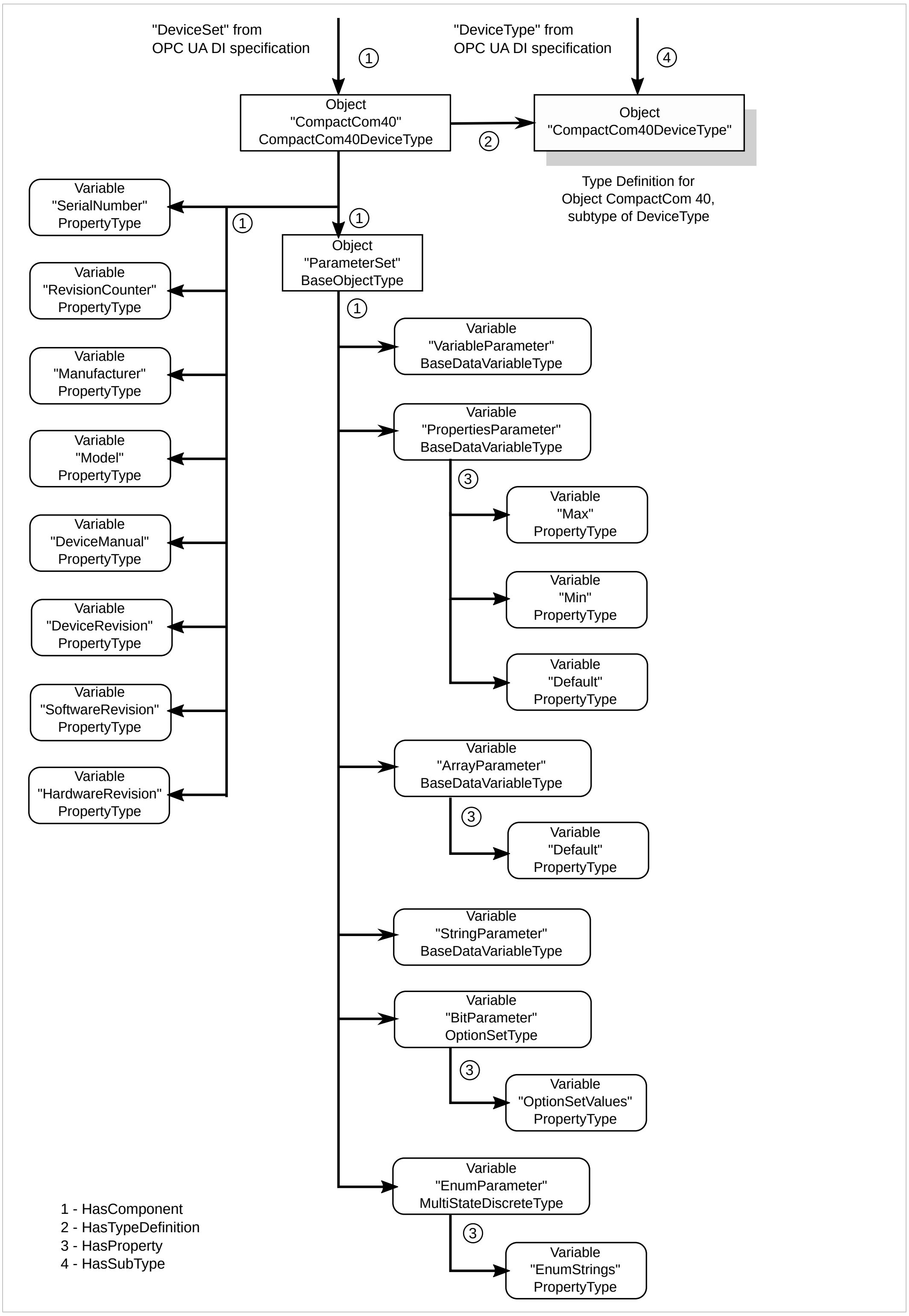

"DeviceSet" from
OPC UA DI specification
1
"DeviceType" from
OPC UA DI specification
4
Object
"CompactCom40"
CompactCom40DeviceType
2
Object
"CompactCom40DeviceType"
Type Definition for
Object CompactCom 40,
subtype of DeviceType
1
Variable
"SerialNumber"
PropertyType
Variable
"RevisionCounter"
PropertyType
Variable
"Manufacturer"
PropertyType
Variable
"Model"
PropertyType
Variable
"DeviceManual"
PropertyType
Variable
"DeviceRevision"
PropertyType
Variable
"SoftwareRevision"
PropertyType
Variable
"HardwareRevision"
PropertyType
1
Object
"ParameterSet"
BaseObjectType
1
Variable
"VariableParameter"
BaseDataVariableType
Variable
"PropertiesParameter"
BaseDataVariableType
3
Variable
"Max"
PropertyType
Variable
"Min"
PropertyType
Variable
"Default"
PropertyType
Variable
"ArrayParameter"
BaseDataVariableType
3
Variable
"Default"
PropertyType
Variable
"StringParameter"
BaseDataVariableType
Variable
"BitParameter"
OptionSetType
3
Variable
"OptionSetValues"
PropertyType
Variable
"EnumParameter"
MultiStateDiscreteType
3
Variable
"EnumStrings"
PropertyType
1 - HasComponent
2 - HasTypeDefinition
3 - HasProperty
4 - HasSubType

Fig. 24

10.3.1 CompactCom 40 Device Type Namespaces

The namespace is part of the node identity used to address a specific node. A namespace is defined by a naming authority (e.g. an organization, a vendor or the local server) and is responsible for managing the identifiers of all nodes defined in the namespace.

The namespaces supported by an OPC UA server are presented in the NamespaceArray of the Server object. When addressing a node in the address space the position (array index) in the NamespaceArray is used to point out the namespace that the node belongs to.

Index 0 of the NamespaceArray is reserved for the OPC UA namespace and index 1 is reserved for the local server. Further indexes can contain any namespace supported by the server. As the Anybus CompactCom implements the OPC UA Device Integration model, the namespace of this model is present in the NamespaceArray. The Anybus CompactCom also has its own DeviceType defined and is part of a namespace managed by HMS Networks. The product has two namespaces, one for its parameters and one for its device representation.

The table below shows the namespaces supported.

Namespace index	Namespace URI	Description
0	http://opcfoundation.org/UA/	OPC UA Foundation
1	urn:<hostname/serialnumber>:anybus:compactcom40	Local server namespace. Globally-unique logical name for a server within the scope of the network in which it is installed. The hostname is used as a component of the URI if configured. If the serial number of the application is available, this can be used, otherwise the serial number of the Anybus CompactCom is used. This namespace string can be replaced by the OPC UA host object (E3h) for branding purposes.
2	http://opcfoundation.org/UA/DI/	OPC UA Device Integration (DI)
3	http://hms-networks.com/UA/Anybus/CompactCom40/	Vendor namespace. The namespace that defines the device type of the CompactCom module that is a sub-type of the Device Integration (DI) device type. Default is the CompactCom 40 device model defined by HMS Networks. The OPC UA host object (E3h) can configure the namespace URI for branding purposes.
4	<ProductUri>/parameters	The namespace that implements all variable nodes that represent Application Data Instances in the information model. It is expected that all instances of a product that shares the ProductUri has the same parameter set. The ProductUri portion of the namespace string can be replaced by the OPC UA host object (E3h) for branding purposes.
5	<ProductUri>/modules	The namespace that implements the module representation in the information model. It is expected that all instances of a product that shares the ProductUri has the same device representation. The ProductUri portion of the namespace string can be replaced by the OPC UA host object (E3h) for branding purposes.
6 and up	(reserved)	

Vendor Namespace

The vendor namespace contains all types defined by the vendor device model. By default, the vendor namespace belongs to HMS Networks and defines the CompactCom 40 device model. It is possible to change the URI from the host application. At the moment there is one node in the Anybus CompactCom 40 namespace:

Node ID type	Node ID	Browse name	Description
Numeric	0x00010000	CompactCom40Type	CompactCom40 type definition based on OPC UA DI DeviceType

Parameters Namespace

The Parameters namespace contains all nodes that models Application Data Instances. The translation of Application Data instance (ADI) numbers and the instance attributes to nodes are specified by the table below.

Node ID	Description
0x01000000-0x01FFFFFF	Node IDs of nodes that models Application Data Instances. The node IDs have the following encoding: 0x01MMMMNN • MMMM = ADI number • NN = ID of a node that models certain information of an ADI NN: Description 0x00: Value variable nodes of the ADIs 0x01: Max value variable nodes of the ADIs 0x02: Min value variable nodes of the ADIs 0x03: Default value variable nodes of the ADIs 0x04: OptionSetValues property of OptionSetType (only exists forf ADIs with BITS or BITx data types) 0x05: EnumStrings property of MultiStateDiscreteType (Only exists for ADIs with ENUM data type)

Modules Namespace

The Modules namespace contains all nodes that models the device representation. The nodes that model the device are specified by the table below.

Node ID	Description
0x01000100	Device instance
0x01000101	SerialNumber property of the device instance
0x01000102	RevisionCounter property of the device instance
0x01000103	Manufacturer property of the device instance
0x01000104	Model property of the device instance
0x01000105	DeviceManual property of the device instance
0x01000106	DeviceRevision property of the device instance
0x01000107	SoftwareRevision of the device instance
0x01000108	HardwareRevision of the device instance
0x01000109	Instance of the DeviceType_ParameterSet of the OPC UA DI namespace. A component of the device instance

10.3.2 Identification Parameters

A number of identification parameters are presented on OPC UA by the Server Object and by the Device Instance of the OPC UA Device Integration model. Some parameters are only used in either the Server Object or the Device Object while some are used in both objects. This chapter specifies the sources of each parameter and what parameters in the Server Object or the Device Object that are equal.

Parameter	Used by/in	DataType	Source/Default
Product URI	Server object/Server Status/BuildInfo	String	Can be configured in the OPC UA Host Object (E3h). Default value = “http://hms-networks.com/UA/Anybus/CompactCom40/[networktype]/[softwareversion]” [networktype] = Abbreviation of network [softwareversion] = Shall equal SoftwareVersion of this table
	ApplicationDescription		
	Parameters and modules namespace URIs		
Manufacturer name	Server object/Server Status/BuildInfo	String	Equals the manufacturer name of the Anybus CompactCom. The name is provided by the Anybus CompactCom following the priority specified for the Application Object (FFh), instance attribute #8 (Vendor Name). The user (the host application) can configure the manufacturer name in the host object of the industrial network (if applicable) or in the application host object.
Manufacturer	DeviceType	LocalizedText	
Product name	Server object/Server Status/BuildInfo	String	Equals the product name of the Anybus CompactCom. The name is provided by the Anybus CompactCom following the priority specified for the Application Object (FFh), instance attribute #9 (Product Name).
Application name	ApplicationDescription	LocalizedText	
Software version	Server object/Server Status/BuildInfo	String	Equals the software version of the Anybus CompactCom as a string. The software version is provided by the Anybus CompactCom following the priority specified for Application Object (FFh), instance attribute #10 (Firmware version). In case the software version isn’t already in string format it is converted by the Anybus CompactCom to string format X.YY where X is major version and YY is minor version.
Software revision	Property of the device instance		
Build number	Server object/Server Status/BuildInfo	String	Equals the build number of the Anybus CompactCom. The build number is provided by the Anybus CompactCom according to the following: 1. If a network host object is implemented, the build number specified here is used. If the network host object is implemented, but no build number is available for the firmware version, the build number will default to 1. 2. If network host object isn’t implemented, but the Application Object (FFh), attribute #10 (Firmware version) is, the build number in this attribute will be used. 3. Anybus CompactCom default value will be used.

Parameter	Used by/in	DataType	Source/Default
Application URI	ApplicationDescription	String	Application URI of the server. Can be configured in the OPC UA Host Object (E3h). Default value = “urn:<hostname/serialnumber>: anybus:compactcom40”. Use host name as a part of URI if available, otherwise use the serial number. The Application URI is part of the ApplicationDescription, the EndpointDescription and also used as Local server namespace URI.
	Server object/ServerArray		
Localnamespace URI	NamespaceArray		
Application type	ApplicationDescription	Enum	Set to 0 = UA_APPLICATIONTYPE_SERVER.
Serial number	Property of the device instance	String	Value from Anybus CompactCom. The serial number is provided by the Anybus CompactCom following the priority specified for Application Object (FFh), instance attribute #3 (Serial number)
Hardware revision	Property of the device instance	String	Equals the hardware revision of the Anybus CompactCom as a string. The hardware revision is provided by the Anybus CompactCom following the priority specified for Application Object (FFh), instance attribute #11 (Hardware version).
Vendor namespace URI	Server object/ NamespaceArray	String	Vendor namespace URI. This namespace collects type definitions specific for the product. Retrieved from OPCUA Host object. Default value = “http://hms-networks.com/UA/ Anybus/CompactCom40/”.
DeviceType name	The DeviceType instantiated by the Anybus CompactCom module	LocalizedText	The name of the DeviceType. Can be configured in OPC UA Host Object (E3h). Default value = “CompactCom40DeviceType”.
Device instance name	Device instance	LocalizedText	The name of the instance of the DeviceType above that represents the device in the local namespace. Retrieved from OPC UA Host object. Default value = “CompactCom40”.

The ApplicationDescription mentioned in the column “Used by/in” in the table above, is used in the responses to discovery services and in the response to the CreateSessionRequest. See OPC UA Part 4 for more information.

10.3.3 Application Data Exchange

An Anybus CompactCom ADI is mapped to a variable node in the OPC UA address space. In the current implementation only the first 256 defined instances (ADIs) of the Application Data Object are accessible via OPC UA. All data access is explicit, and no process data access is available. Struct ADIs are not supported.

Translation of Data Types

An OPC UA variable node has a data type that describes the actual value of the variable and a variable type reference that points out the variable type. Variable types provide type definitions for variables. A variable type can for instance define components and properties that are either mandatory or optional to implement by the variable. The Anybus data types are translated to OPC UA data types and variable types according to the table below.

Anybus data type	OPC UA data type	OPC UA variable type	Description
BOOL	Boolean	BaseDataVariableType	
SINT8	SByte	BaseDataVariableType	
SINT16	Int16	BaseDataVariableType	
SINT32	Int32	BaseDataVariableType	
UINT8	Byte	BaseDataVariableType	
UINT16	UInt16	BaseDataVariableType	
UINT32	UInt32	BaseDataVariableType	
CHAR	String	BaseDataVariableType	Array of CHAR will be translated to a single string
ENUM	Byte	MultiStateDiscreteType	The enum strings of an Enum ADI will be presented in the property node EnumStrings of the variable type MultiStateDiscreteType
BITS8	BitFieldMaskDataType	OptionSetType	This variable type has a mandatory property called OptionSetValue, which value attribute holds a text string array that describes the value, bit by bit. This property does not exist in the ADI implementation in Anybus, but is generated by the Anybus OPC UA implementation [“Bit0”, “Bit1”, ... “Bitn”]
BITS16	BitFieldMaskDataType	OptionSetType	This variable type has a mandatory property called OptionSetValue, which value attribute holds a text string array that describes the value, bit by bit. This property does not exist in the ADI implementation in Anybus, but is generated by the Anybus OPC UA implementation [“Bit0”, “Bit1”, ... “Bitn”]
BITS32	BitFieldMaskDataType	OptionSetType	This variable type has a mandatory property called OptionSetValue, which value attribute holds a text string array that describes the value, bit by bit. This property does not exist in the ADI implementation in Anybus, but is generated by the Anybus OPC UA implementation [“Bit0”, “Bit1”, ... “Bitn”]
OCTET	ByteString	BaseDataVariableType	Struct element of type OCTET with subelements is translated to a single ByteString. Struct with only one element of type OCTET is translated to a ByteString variable.
SINT64	Int64	BaseDataVariableType	
UINT64	UInt64	BaseDataVariableType	

Anybus data type	OPC UA data type	OPC UA variable type	Description
FLOAT	Float	BaseDataVariableType	
DOUBLE	Double		
PADx	N/A	N/A	Not supported in OPC UA. No OPC UA node is created for an ADI of data type PADx.
BOOL1	Boolean	BaseDataVariableType	
BITx	BitFieldMaskDataType	OptionSetType	This variable type has a mandatory property called OptionSetValue, which value attribute holds a text string array that describes the value, bit by bit. This property does not exist in the ADI implementation in Anybus, but is generated by the Anybus OPC UA implementation [“Bit0”, “Bit1”, ... “Bitn”]

ADI Variable Node

An ADI is represented as an OPC UA variable node, with the attributes as in the table below.

The max, min and default value attributes of an ADI, if its data type is translated to BaseDataVariableType, are translated to variable nodes in the OPC UA address space. These nodes are referenced from the variable node that holds the actual value of the ADI.

Attribute name	Data type	Value	Description
NodeId	NodeId	See table in *Parameters Namespace, p. 85*	Instance number is translated to a NodeId in the Parameters namespace.
NodeClass	NodeClass	VARIABLE (2)	
BrowseName	QualifiedName	Namespace: Local server namespace String: “Param”+<ADI instance number>	The BrowseName attribute is used to create a path to a certain node. It is possible to translate a browse name path to the node ID of the node. E.g. “Param1” or “Param105”
DisplayName	LocalizedText	Equals the value from Application Data Object, Instance attribute #1 (Name)	
Description	LocalizedText	Null	
WriteMask	UInt32	0	It is not possible to write any attribute
UserWriteMask	UInt32	0	It is not possible to write any attribute
Value	Defined by DataType attribute	Equals the value from Application Data Object, Instance attribute #5 (Value)	
DataType	NodeId	Node ID of the data type	For ADI classes variable and array, this is the node ID of the data type, according to the translation in *Translation of Data Types, p. 88*
ValueRank	Int32	1 or -1	The ValueRank is set to 1 for arrays (one dimension) and -1 for variables and structures (scalar)
ArrayDimensions	Uint32	Null or a single entry array	The ArrayDimensions is null for variables and structures. If the ADI is an array the ArrayDimensions is an array with one entry. The value of this single entry is equal to the ADI attribute 3, Number of elements.
AccessLevel	Byte	Bit 0: Get access bit of the ADI descriptor Bit 1: Write access bit of the ADI descriptor	

Attribute name	Data type	Value	Description
		All other bits set to 0.	
UserAccessLevel	Byte	Bit 0 & 1: Access rights from the configuration file of the current user All other bits set to 0.	
MinimumSamplingInterval	Duration	0	The server monitors the item continuously
Historizing	Boolean	False	Collecting data for the history of the variable is not supported

A variable node representing an ADI has the references in the table below.

References	Number	Description
HasProperty	0..3	Only valid for ADI data types translated to BaseDataVariableType. Possible properties of an ADI variable node: • Max value • Min value • Default value The properties are present if the ADI implements the corresponding attribute. They are presented through referencing another variable node, see *Max, Min and Default Value Variable Node, p. 90*.
HasProperty	1	Only valid for ADI data types translated to OptionSetType. The following property is referenced: • OptionSetValues (see description in table in *Translation of Data Types, p. 88*)
HasProperty	1	Only valid for ADI data types translated to MultiStateDiscreteType. The following property is referenced: • EnumStrings (see description in table in *Translation of Data Types, p. 88*)
HasTypeDefinition	1	For an ADI of class variable or array the reference points to the the variable type specified per data type in *Translation of Data Types, p. 88*.

Max, Min and Default Value Variable Node

The max, min and default value attributes of an ADI, if its data type is translated to BaseDataVariableType, are translated to variable nodes in the OPC UA address space. These nodes are referenced from the variable node that holds the actual value of the ADI.

The variable node representing the max, min or default value of an ADI has the attributes in the table below.

Attribute name	Data type	Value	Description
NodeId	NodeId	See table in *Parameters Namespace, p. 85*	
NodeClass	NodeClass	VARIABLE (2)	
BrowseName	QualifiedName	Namespace: Local server namespace String: “Max”, “Min” or “Default”	The browse name are used to create paths to certain nodes. It is possible to translate a browse name path to the node’s node ID. Select string depending on which value the node represent.
DisplayName	LocalizedText	“Max”, “Min” or “Default”	
Description	LocalizedText	“Max”, “Min” or “Default”	
WriteMask	UInt32	0	It is not possible to write any attribute
UserWriteMask	UInt32	0	It is not possible to write any attribute
Value	Defined by DataType attribute	Use the value from Application Data Object, attribute 6,7, or 8.	ADI attribute 6, 7 or 8 (Max value, Min value or Default value)

Attribute name	Data type	Value	Description
DataType	NodeId	Equals the data type of the variable node representing the ADI an that has the HasComponent reference to this node.	
ValueRank	Int32	-1	The ValueRank is set to -1 for all ADI classes (arrays only have 1 max, min or default value).
ArrayDimensions	UINT32[]	Null	The ArrayDimensions is null for all ADI classes (arrays only have 1 max, min or default value)
AccessLevel	Byte	0x01	It is possible to read the current value.
UserAccessLevel	Byte	Bit 0: access rights from the configuration file of the current user All other bits set to 0	
MinimumSamplingInterval	Duration	0	The server monitors the item continuously
Historizing	Boolean	False	Collecting data for the history of the variable is not supported

A variable node representing the max, min or default value of an ADI uses the references in the table below.

References	Number	Description
HasTypeDefinition	1	This reference points to the PropertyType variable type.

Language Support

The name attribute of an ADI can be multilingual.

OPC UA has a LocalizedText data type that can present a text together with a language code. It is a structured type containing a locale id and a string. The OPC UA simple data type LocaleId is used to present a particular language. It uses two letter ISO 639 codes.

The active language is set by the host application in the Anybus Object (01h, attribute #9). The setting is transferred to OPC UA in the data type LocaleId.

Anybus language	OPC UA LocaleId
English (0x00)	“en”
Deutsch (0x01)	“de”
Español (0x02)	“es”
Italiano (0x03)	“it”
Français (0x04)	“fr”

Data Mapping Example

This section gives an example of how a number of ADIs of different types are mapped into the OPC UA information model, see also figure in section *CompactCom 40 Device Type Information Model, p. 82*.

ADI	Name	Descriptor	Num Elements	Data type	Value	Max	Min	Default
1	VariableParameter	Get, Set	1	UINT16	100	Not implemented	Not implemented	Not implemented
2	VariableProperties	Get	1	UINT16	90	100	10	15
3	ArrayParameter	Get	10	UINT8	[0, 0, 253, 1, 73, 42, 3, 143, 10, 0]	Not implemented	Not implemented	10
4	StringParameter	Get	11	CHAR	[“S”, “t”, “r”, “i”, “n”, “g”, “V”, “a”, “l”, “u”, “e”]	Not implemented	Not implemented	Not implemented
100	BitParameter	Get	1	BIT3	0x05	N/A	N/A	N/A
101	EnumParameter	Get	1	ENUM	3	N/A	N/A	N/A

OPC UA Variable nodes						
NodeId	0x01000100	0x01000200	0x01000300	0x01000400	0x01006400	0x01006500
NodeClass	Variable	Variable	Variable	Variable	Variable	Variable
BrowseName	Param1	Param2	Param3	Param4	Param100	Param101
DisplayName	VariableParameter	VariableProperties	ArrayParameter	StringParameter	BitParameter	EnumParameter
Description	Null	Null	Null	Null	Null	Null
WriteMask	0	0	0	0	0	0
UserWriteMask	0	0	0	0	0	0
DataType	UInt16	UInt16	UInt8	String	BitFieldMaskDataType	Byte
ValueRank	-1	-1	1	-1	-1	-1
ArrayDimensions	Null	Null	[10]	Null	Null	Null
Value	100	10	[0, 0, 253, 1, 73, 42, 3, 143, 10, 0]	“StringValue”	0x0700000005000000	3
AccessLevel	0x03	0x01	0x01	0x01	0x01	0x01
MinimumSamplingInterval	0	0	0	0	0	0
Historizing	False	False	False	False	False	False
HasTypeDefinition reference	BaseDataVariableType	BaseDataVariableType	BaseDataVariableType	BaseDataVariableType	OptionSetType	MultiStateDiscreteType
HasProperty references	None	To max, min, default variable nodes: * 0x01000201 (NodeId max value) * 0x01000202 (NodeId min value) * 0x01000203 (NodeId default value)	To default variable node: * 0x01000203 (NodeId default value)	None	To OptionSetValues node: * 0x01006404	To EnumStrings node: * 0x01006505

10.4 Application Defined Information Model

This model enables the Anybus CompactCom to present an application defined model created in an OPC UA modeler tool. The application defined information model can be completely custom or based on existing companion specifications that define profiles of devices according to OPC UA standards. Variable nodes of the application defined information model can be mapped to ADIs of the host application.

The information model is defined by a Nodeset2 XML file that can be generated by an OPC UA Modeler Tool. The finished Nodeset2 file is converted by a tool from HMS, the *Anybus OPC UA NodeSet Encoder*, to a binary file (binarynodeset.hiff, and this filename must not be changed), that is downloaded to the root of the file system of the Anybus CompactCom 40 IIoT Secure.

Variable nodes can either be tied to ADIs in the host application or be statically modeled in the information model. For nodes tied to ADIs, a separate namespace must be created. The *Anybus OPC UA NodeSet Encoder* will along with the binary file generate an ADI list as C source code, customized for the information model. This ADI list can be directly used in the CompactCom Host Application Example Code.

See also...

- *Using OPC UA Application Defined Information Models with Anybus CompactCom IIoT Secure* application note

10.4.1 Application Defined Namespaces

The application defined namespaces contain the information model designed for the specific application. The namespaces can be device specific or be defined by Companion specifications, organizations and vendors.

Namespace Index	Namespace
0	OPC Foundation namespace
1	Local server namespace
2–N	Application defined namespaces
M (M ⊂ 2-N)	Parameters namespace, see *Parameters Namespace, p. 85*. If mapping an OPC UA variable to an ADI, this namespace must be included in the information model. The namespace URI of this namespace shall be set to urn:compactcom40:parameters.

NodeClasses

The NodeClasses table describes the supported node classes and any limitations when using them in an application defined information model namespace. No multilingual support (except variable nodes mapped to ADIs) is available.

NodeClass	**Considerations**
OBJECT	The EventNotifier attribute must be set to 0 (no event support)
VARIABLE	Variables tied to ADIs must be modeled in the Parameters namespace, see *Parameters Namespace, p. 85*. Other variables can be modeled in any namespace. Variables tied to ADIs can only use data types (no structured data types) from the OPC Foundation namespace (namespace 0).
METHOD	A call to a method will result in an error response.
OBJECT_TYPE	
VARIABLE_TYPE	The value attribute of this node class defines the default value of any variable node of this type. The host application must be aware of this default value and use it for any ADI that maps to a variable node of this type.
REFERENCE_ TYPE	
DATA_TYPE	The following data types are **not** supported: • Built-in types and all simple types derived of them: – Guid – XmlElement – ExpandedNodeId – StatusCode – ExtensionObject – DataValue – Variant – DiagnosticInfo – Decimal • Structured data types with optional fields • Data types derived from Union data type
VIEW	The EventNotifier attribute must be set to 0 (no event support). Use of view nodes in the Browse service of the view service set is not supported.

10.4.2 Identification Parameters

A number of identification parameters are presented on OPC UA by the Server Object and in service responses. This section specifies the source of each parameter.

Parameter	Used by/in	DataType	Source/Default
Product URI	Server object/Server Status/BuildInfo	String	Can be configured in the OPC UA Host Object (E3h). Default value = “http://hms-networks.com/UA/Anybus/CompactCom40/[networktype]/[softwareversion]” [networktype] = Abbreviation of network [softwareversion] = Shall equal SoftwareVersion of this table
	ApplicationDescription		
	Parameters namespace URI		
Manufacturer name	Server object/Server Status/BuildInfo	String	Equals the manufacturer name of the Anybus CompactCom. The name is provided by the Anybus CompactCom following the priority specified for the Application Object (FFh), instance attribute #8 (Vendor Name). The user (the host application) can configure the manufacturer name in the host object of the industrial network (if applicable) or in the application host object.
Product name	Server object/Server Status/BuildInfo	String	Equals the product name of the Anybus CompactCom. The name is provided by the Anybus CompactCom following the priority specified for the Application Object (FFh), instance attribute #9 (Product Name).
Application name	ApplicationDescription	LocalizedText	
Software version	Server object/Server Status/BuildInfo	String	Equals the software version of the Anybus CompactCom as a string. The software version is provided by the Anybus CompactCom following the priority specified for Application Object (FFh), instance attribute #10 (Firmware version). In case the software version isn’t already in string format it is converted by the Anybus CompactCom to string format X.YY where X is major version and YY is minor version.
Build number	Server object/Server Status/BuildInfo	String	Equals the build number of the Anybus CompactCom. The build number is provided by the Anybus CompactCom according to the following: 1. If a network host object is implemented, the build number specified here is used. If the network host object is implemented, but no build number is available for the firmware version, the build number will default to 1. 2. If network host object isn’t implemented, but the Application Object (FFh), attribute #10 (Firmware version) is, the build number in this attribute will be used. 3. Anybus CompactCom default value will be used.

Parameter	Used by/in	DataType	Source/Default
Application URI	ApplicationDescription	String	Application URI of the server. Can be configured in the OPC UA Host Object (E3h). Default value = “urn:<hostname/serialnumber>: anybus:compactcom40”. Use host name as a part of URI if available, otherwise use the serial number. The Application URI is part of the ApplicationDescription, the EndpointDescription and also used as Local server namespace URI.
	Server object/ServerArray		
Localnamespace URI	NamespaceArray		
Application type	ApplicationDescription	Enum	Set to 0 = UA_APPLICATIONTYPE_SERVER.

The ApplicationDescription mentioned in the column “Used by/in” in the table above, is used in the responses to discovery services and in the response to the CreateSessionRequest. See OPC UA Part 4 for more information.

10.4.3 Application Data

ADI Variable Node

An application defined information model must specify variable nodes tied to ADIs completely. However, some attributes will be manipulated by the CompactCom module to contain values that matches the ADI setup of the host application and configured access rights. The table below specifies this behavior per attribute.

Node Attribute	Description
NodeId	Instance number is translated to a NodeId in the Parameters namespace. This attribute is static and will not be manipulated by the CompactCom.
NodeClass	This attribute is static and will not be manipulated by the CompactCom.
BrowseName	This attribute is static and will not be manipulated by the CompactCom.
DisplayName	This attribute will be manipulated by the CompactCom.
Description	This attribute is static and will not be manipulated by the CompactCom.
WriteMask	Only supported for the value attribute by the Anybus CompactCom. Other node attributes should be configured according to the application. This attribute is static and will not be manipulated by the CompactCom.
UserWriteMask	Only supported for the value attribute by the Anybus CompactCom. Other node attributes should be configured according to the application. This attribute is static and will not be manipulated by the CompactCom.
Value	This attribute will be manipulated by the CompactCom.
DataType	Can be set to any OPC UA type that origins from an OPC UA built-in type that translates to the data type of the ADI that the variable points to. Only types of OPC Foundation namespace (namespace 0) is supported. This attribute is static and will not be manipulated by the CompactCom.
ValueRank	The ValueRank is set to 1 for arrays (one dimension) and -1 for variables and structures (scalar) This attribute will be manipulated by the CompactCom.
ArrayDimensions	The ArrayDimensions is null for variables and structures. If the ADI is an array the ArrayDimensions is an array with one entry. The value of this single entry is equal to the ADI attribute 3, Number of elements. This attribute will be manipulated by the CompactCom.
AccessLevel	This attribute will be manipulated by the CompactCom.
UserAccessLevel	This attribute will be manipulated by the CompactCom.
MinimumSamplingInterval	Not supported by the Anybus CompactCom.
Historizing	The attributes should be configured according to the application. This attribute is static and will not be manipulated by the CompactCom.

Translation of Supported Data Types

Anybus data type	OPC UA data type	OPC UA variable type	Description
BOOL/BOOL1	Boolean	BaseDataVariableType	
SINT8	SByte	BaseDataVariableType	
SINT16	Int16	BaseDataVariableType	
SINT32	Int32	BaseDataVariableType	
UINT8	Byte	BaseDataVariableType	
UINT16	UInt16	BaseDataVariableType	
UINT32	UInt32	BaseDataVariableType	
CHAR	String	BaseDataVariableType	Array of CHAR will be translated to a single string
SINT64	Int64	BaseDataVariableType	
UINT64	UInt64	BaseDataVariableType	
FLOAT	Float	BaseDataVariableType	
DOUBLE	Double	BaseDataVariableType	
	DateTime	BaseDataVariableType	A DateTime value shall be encoded as a 64-bit signed integer which represents the number of 100 nanosecond intervals since January 1, 1601 (UTC).
Struct of: UINT32 UINT16 UINT16 OCTET[8]	Guid	BaseDataVariableType	A 16-byte value that can be used as a globally unique identifier.
OCTET	ByteString	BaseDataVariableType	
UINT32	StatusCode	BaseDataVariableType	
CHAR	LocalizedText	BaseDataVariableType	

Anybus Data Types Without Matching OPC UA Built-in Type

Some Anybus data types do not have an equivalent OPC UA built-in type. These Anybus data types have either no translation defined or translates to a simple type that match the data type well.

Anybus data type	Translation description
BITx	Translated to BitFieldMaskDataType which is a subtype of Uint64
BITS8/BITS16/BITS32	
Enum	Translated to Byte.

10.5 Time

An OPC UA server needs a mechanism for knowing the current UTC time and facilities to convert to and from local time. The Anybus CompactCom will fetch the time from the network by cyclically sending a request to a OPC UA Discovery Server. The server will be polled every 60 seconds with a timeout of 5 seconds. To configure the address of the Discovery Server, set Network Configuration Object (04h), instance #41, attribute #5 with the URL of the Discovery Server.

! If the application is to pass the conformance test for OPC UA, a Discovery Server must be configured in the Network Configuration Object (04h), instance #41.

10.6 Server Endpoints

An OPC UA server offers one or several endpoints which clients can connect to. An endpoint specifies a SecurityPolicy and a message security mode to be used when connecting. It also presents identity information and what UserIdentityTokens that are available to use for authentication. If required by the SecurityPolicy it also provides the server certificate. This section specifies what SecurityPolicies the Anybus CompactCom shall support, what UserIdentityToken it will offer for clients to use for authentication and the endpoints it will provide.

10.6.1 SecurityPolicies

OPC UA specifies a set of security policies. A SecurityPolicy specifies a set of algorithms for encryption and signing of data to be used when communicating. A device can support one or multiple policies and the client and the server negotiate which policy to use when connecting.

The Anybus CompactCom supports the SecurityPolicies listed in the table below.

SecurityPolicy	Security Policy URI
SecurityPolicy – None	http://opcfoundation.org/UA/SecurityPolicy#None
SecurityPolicy [A] – Aes128-Sha256-RsaOaep	http://opcfoundation.org/UA/SecurityPolicy#Aes128_Sha256_RsaOaep
SecurityPolicy [B] – Basic256Sha256	http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256

10.6.2 UserIdentityTokens

OPC UA specifies a set of UserIdentityTokens to provide different methods of authentication. The CompactCom module only supports the UserNameIdentityToken. Access will be granted to user accounts assigned to roles that are configured to have OPC UA access, see *Configuration, p. 80*.

The Anybus CompactCom supports the UserIdentityTokens listed in the table below.

PolicyId	UserTokenType	SecurityPolicy
Username_Null	UserName	SecurityPolicy specified to NULL will make the UserIdentityToken to inherit the SecurityPolicy of the endpoint.
Username_None	UserName	SecurityPolicy None
Username_Basic256Sha256	UserName	SecurityPolicy [B] – Basic256Sha256
Username_Aes128_Sha256_RsaOaep	UserName	SecurityPolicy [A] – Aes128-Sha256-RsaOaep

Which UserIdentityTokens that are enabled on an endpoint depends on the security mechanisms that the endpoint uses.

10.6.3 Endpoints

The Anybus CompactCom implements the Endpoints listed in the table below.

Endpoint #	SecurityPolicy	Message security mode	Security level	UserIdentityTokens
1	SecurityPolicyNone	None	0	Username_Basic256Sha256, Username_Aes128_Sha256_RsaOaep (Username_None)
2	SecurityPolicy [B] – Basic256Sha256	Sign	2	Username_Null (Username_None)
2	SecurityPolicy [B] – Basic256Sha256	SignAndEncrypt	4	Username_None

Endpoint #	SecurityPolicy	Message security mode	Security level	UserIdentityTokens
3	SecurityPolicy [A] – Aes128-Sha256-RsaOaep	Sign	1	Username_Null (Username_None)
4	SecurityPolicy [A] – Aes128-Sha256-RsaOaep	SignAndEncrypt	3	Username_None

Endpoints with SecurityPolicyNone and the UserIdentityToken “Username_None” on all endpoints are disabled by default and can only be enabled by an administrator from the internal web pages or from the Network Configuration object (04h). By enabling the UserIdentityToken “Username_None” it is possible to connect to the CompactCom module without configuring any certificates and private keys.

Endpoints with SecurityPolicy other than None are disabled if no device certificate is installed for OPC UA. How to install device and CA certificates are described in *Initial Setup and Account Configuration, p. 14*.

Endpoints implementing SecurityPolicy other than None and message security mode Sign offer the UserIdentityToken “Username_Null” for session authentication. UserIdentityTokens without any SecurityPolicy specified used on a secure channel with a SecurityPolicy other than None, inherits the SecurityPolicy from the secure channel.

Endpoints implementing SecurityPolicy other than None and message security mode SignAndEncrypt offer the UserIdentityToken “Username_None” for session authentication. The secure channel encrypts all data, hence there is no need to also encrypt the authentication credentials.

! If the application is to pass the conformance test for OPC UA, the UserIdentityToken “Username_None” must be disabled (does not apply to endpoints with message security mode SignAndEncrypt) or some external security mechanism must be applied to encrypt the authentication credentials.

10.7 Error Code Translation

The Anybus CompactCom error codes are translated to OPC UA status codes as described in the table below.

#	Anybus error code	#	OPC UA status code	Comment
0x02	Invalid message format	0x80020000	BadInternalError	
0x03	Unsupported object	N/A	N/A	Not possible to define a generic translation. Depends on the context of the request.
0x04	Unsupported instance	0x80340000	BadNodeIdUnknown	Not possible to define a generic translation. Depends on the context of the request.
0x05	Unsupported command	0x803D0000	BadNotSupported	
0x06	Invalid CmdExt[0]	N/A	N/A	Not possible to define a generic translation. Depends on what CmdExt[0] contains.
0x07	Invalid CmdExt[1]	N/A	N/A	Not possible to define a generic translation. Depends on what CmdExt[1] contains.
0x08	Attribute not set-able	0x803B0000	BadNotWritable	
0x09	Attribute not get-able	0x803A0000	BadNotReadable	
0x0A	Too much data	0x80740000	BadTypeMismatch	
0x0B	Not enough data	0x80740000	BadTypeMismatch	
0x0C	Out of range	0x803C0000	BadOutOfRange	
0x0D	Invalid state	0x80AF0000	BadInvalidState	
0x0E	Out of resources	0x80030000	BadOutOfMemory	
0x0F	Segmentation failure	0x80020000	BadInternalError	
0x10	Segmentation buffer overflow	0x80020000	BadInternalError	
0x11	Value too high	0x803C0000	BadOutOfRange	
0x12	Value to low	0x803C0000	BadOutOfRange	
0x13	Attribute controlled	0x801F0000	BadUserAccessDenied	
0x14	Message channel too small	0x80020000	BadInternalError	
0x15	General error	0x80010000	BadUnexpectedError	
0x16	Protected access	0x801F0000	BadUserAccessDenied	
0xFF	Object specific	N/A	N/A	Not possible to define a generic translation. Depends on what Anybus object that is accessed.

10.7.1 Error Code Translation when Accessing the Application Data Object

When accessing nodes translated to ADIs, the error codes are translated to OPC UA status codes according to the table below.

#	Anybus error code	#	OPC UA status code	Comment
0x03	Unsupported object	0x80340000	BadNodeIdUnknown	If the Application Data object is not implemented, no nodes representing ADIs will be implemented.
0x04	Unsupported instance	0x80340000	BadNodeIdUnknown	If the instance doesn’t exist, no node exist.
0x06	Invalid CmdExt[0]	0x80340000	BadNodeIdUnknown	Every attribute of an Application Data Instance is represented by a node on OPC UA
0x07	Invalid CmdExt[1]	0x80340000	BadNodeIdUnknown	Every attribute of an Application Data Instance is represented by a node on OPC UA
0xFF	Object specific	0x80020000	BadInternalError	The translation of ADIs to OPC UA variable nodes does not use any object specific services and therefore doesn’t expect any object specific error codes.

10.8 Stack Configuration

This section specifies the configuration of the OPC UA stack, implemented in Anybus CompactCom. The configuration defines the capabilities of the stack implementation.

10.8.1 Connection Configuration

The connection configuration is set according to the table below:

Configuration Parameter	Value	Description/Comment
TCP send buffer size	8196 bytes	Minimum requirement from OPC UA
TCP receive buffer size	8196 bytes	Minimum requirement from OPC UA
Max message size	25576 bytes	-
Number of TCP connections	4	One TCP connection per secure channel plus one to provide a better error message to clients when out of sessions
Number of secure channels	3	One secure channel supported per session plus one more to provide a better error message to clients when out of sessions
TCP connection lifetime	2 min	If not attached to a secure channel the TCP connection can be overtaken after 2 minutes of inactivity
Security token lifetime	5 min	Minimum lifetime required
Number of sessions	2	Minimum requirement of the Micro Embedded Device Server profile
Max session timeout	5 min	-

10.8.2 Data Subscription Configuration

The data subscription configuration is set according to the table below:

Configuration Parameter	Value	Description/Comment
maxSubscriptions	10	-
publishingInterval Min	1000 ms	Configurable via the OPC UA Host Object (E3h). See *OPC UA Object (E3h), p. 207*
publishingInterval Max	86400000 ms	24h
lifeTimeCount Min	3	-
lifeTimeCount Max	15000	-
keepAliveCount Min	1	-
keepAliveCount Max	100	-
maxNotificationsPerPublish	10	
maxRetransmissionQueueSize	20	The sequence number can be used to detect a missing response
maxMonitoredItems	100 (Default 8)	Configurable via the OPC UA Host Object (E3h). See *OPC UA Object (E3h), p. 207*
samplingInterval Min	1000 ms	Configurable via the OPC UA Host Object (E3h). See *OPC UA Object (E3h), p. 207*
samplingInterval Max	86400000 ms	24h
queueSize Min	1	-
queueSize Max	1	-
maxPublishReqPerSession	10	-

10.8.3 Resource Configuration

The resource configuration is set according to the table below:

Configuration Parameter	Value	Description/Comment
MinSupportedSampleRate	0	Part of the ServerCapabilitiesType.
MaxBrowseContinuationPoints	5	Part of the ServerCapabilitiesType.
MaxNodesPerRead	100	Part of the OperationLimitsType.
MaxNodesPerBrowse	20	Part of OperationLimitsType.
MaxNodesPerRegisterNodes	1	Part of OperationLimitsType.
MaxNodesPerTranslateBrowsePathsToNodeIds	20	Part of OperationLimitsType.
MaxMonitoredItemsPerCall	100	100 MonitoredItems per subscription are supported.
MaxReferencesPerBrowseResponse	30	

11 MQTT

MQTT is a publish-subscribe messaging protocol that runs on top of TCP/IP. It was first developed to transmit data from field devices on remote locations over unreliable satellite links with limited bandwidth. This initial use case has shaped the protocol to offer a limited number of features and only adds a small overhead to the data to be transmitted. This resource constrained protocol has proved useful when pushing data (e.g. diagnostics) from devices to IT systems. All devices that produce or consume data are clients. The clients connect to a common broker device to either publish data, subscribe for data or both. The MQTT message flow is shown in the figure below.

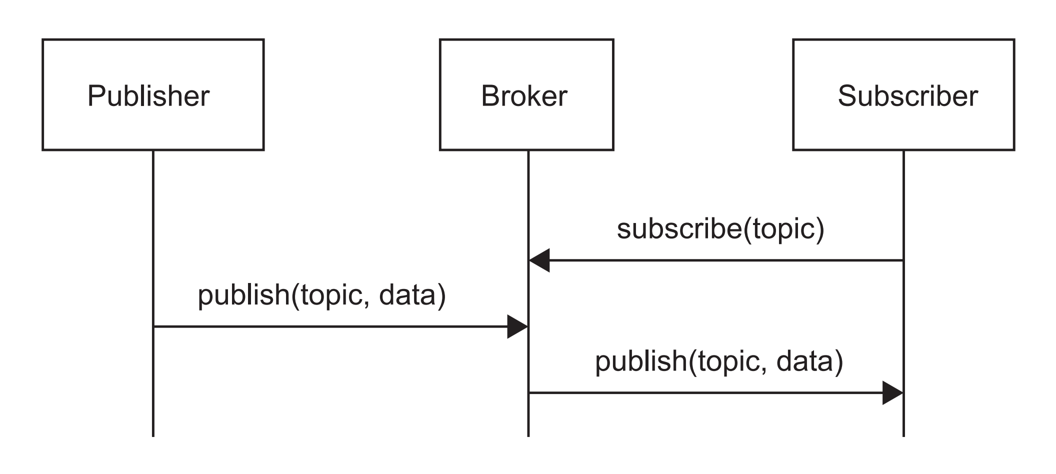


Fig. 25

Data is tagged with a topic string. The topic is used as a unique identifier to route the publications to its subscribers when they are published by a certain device to the broker.

See also....

- mqtt.org
- Network Configuration Object (04h) (Anybus Module Object)
- MQTT Host Object (E2h) (Host Application Object)
- Application Object (FFh), command Get_Data_Notification (13h) in *Anybus CompactCom 40 Software Design Guide*

11.1 MQTT Configuration

The MQTT functionality of the Anybus CompactCom is disabled by default. Implement the attribute #1 (MQTT mode) in the MQTT host object (E2h), instance #1, to enable it.

The MQTT host object provides the possibility to configure the Last will message and to provide publication options per publication.

The Network Configuration object (04h) offers a set of instances to configure the connection to the MQTT broker and also a base topic used for publications. These instances, which also are available via the internal web page, are summarized in the table below.

Parameter	**Instance No. (in Network Configuration Object)**	**Default**
Broker URL	50	Empty string
Client Identifier	51	Empty string
Keep Alive (s)	52	60
Broker Username	53	Empty string
Broker Password	54	Empty string
Base Topic	55	Empty string
Quality of service	56	0
Secure MQTT	57	Disable

11.2 Connection Setup

The Anybus CompactCom connects automatically to an MQTT broker if the following requirements are fulfilled:

- MQTT is enabled by the MQTT host object (E2h)
- A valid broker URL is configured in the Network Configuration instance #50
- The Anybus CompactCom is assigned a valid IP address on the network

If a connection attempt fails or an existing connection is disconnected, a new connection attempt will be performed in 10 seconds except if any of the following connection errors occur:

- Erroneous broker address
- Connection rejected by broker
- Bad server certificate

If Secure MQTT is enabled in the Network Configuration Object, instance #57, the server identity and server certificate of the MQTT broker will be verified against the CA certificates installed in the Anybus CompactCom CA store. If the MQTT broker requests mutual authentication during the TLS handshake the device certificate enabled for MQTT will be sent to the broker. If there is no device certificate enabled for MQTT, the TLS handshake will fail. How to install device and CA certificates is described in *Initial Setup and Account Configuration, p. 14*.

11.3 Publications

To publish data on MQTT, the generic Get_Data_Notification command of the Application host object (FFh) shall be used. The command makes it possible to publish either a single ADI value, the values of ADIs that belong to an Assembly Mapping instance, or vendor specific data that is published transparently from the host application. The Get_Publish_Configuration command of the MQTT host object (E2h) gives the possibility to customize some options of the MQTT publish packet per publication. Once the Anybus CompactCom has successfully set up a connection to the configured broker it will be possible to publish data. The publish sequence is described below.

1. The Anybus CompactCom sends a Get_Data_Notification request to the Application host object once a connection to a broker is active. If the host application has data to publish it can respond to it immediately. If there is currently no data available, the host application can choose to either keep the request to be able to respond immediately when data is available or respond with error code 17h (no data available). In the later case, the Anybus CompactCom will enter a poll mode and periodically poll the host application by sending the request again.
2. When the application has data to publish, e.g. if data is changed or if someone pushes a button, it responds to the Get_Data_Notification request from the Anybus CompactCom that has been kept from Step 1, or waits for the next request. The response includes the dataset to publish to the broker. The dataset is either a single ADI value, the values of ADIs that belong to an Assembly Mapping instance, or vendor specific data that is published transparently from the host application. Optionally, a timestamp can be included.
3. The Anybus CompactCom then sends a Get_Publish_Configuration request to the MQTT host object (E2h) to retrieve any defined publication options for this publication.
4. The application responds with its publication configuration or with an error code, if default options are wanted.
5. Depending on dataset, the Anybus CompactCom requests more information from the application.
6. The Anybus CompactCom encodes the dataset using JSON, if the dataset is either a single ADI value or the values of ADIs that belong to an Assembly Mapping instance. Vendor specific data is not encoded and is published transparently.
7. The Anybus CompactCom builds the MQTT message and publishes it to the MQTT broker. If the retain bit is set, the message will be saved in the MQTT broker for future subscribers to collect.

The figure below shows an example of a publishing sequence with the numbers from the sequence above included. The order of 3 and 5 may be different depending on the Anybus CompactCom implementation.

When the Anybus CompactCom has published a message to the broker, it repeats the sequence and sends a new Get_Data_Notification request. This sequence is repeated as long as the Anybus CompactCom is connected to the broker.

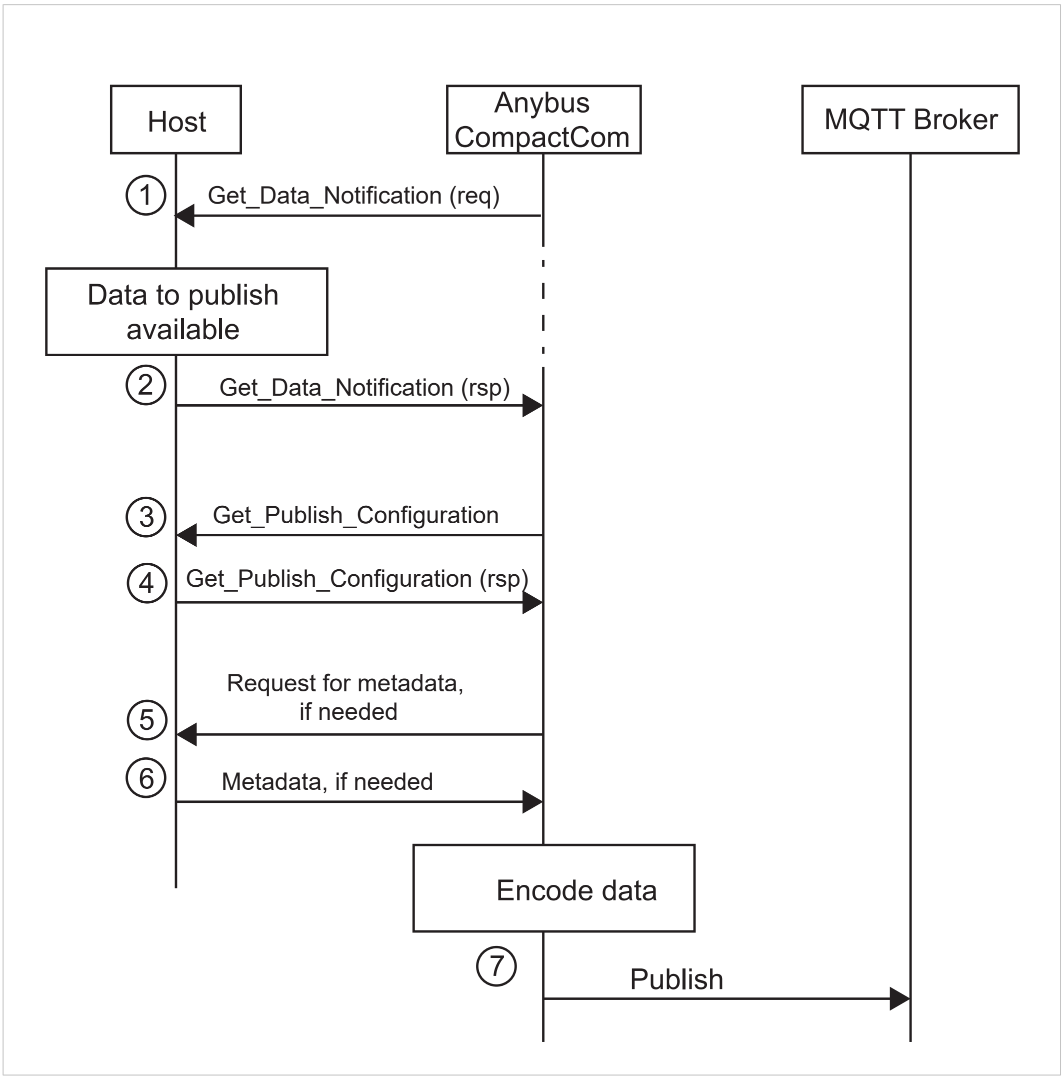

Host
Anybus CompactCom
MQTT Broker
1
Get_Data_Notification (req)
Data to publish available
2
Get_Data_Notification (rsp)
3
Get_Publish_Configuration
4
Get_Publish_Configuration (rsp)
Request for metadata, if needed
5
6
Metadata, if needed
Encode data
7
Publish

Fig. 26

11.3.1 Topic

All data is tagged with a topic string. The topic is a unique identifier used to route publications to its subscribers when they are published by a certain device to the broker.

The topic is either set by the host application, by responding to the Get_Publish_Configuration command that is sent to the MQTT Host Object (E2h), or generated by the Anybus CompactCom with help of metadata of the ADIs that are being published.

The generated topic string has the following format:

```
<base topic>/<dataset type>/<dataset identifier>
```

Each entry of the generated topic format can consist of several levels separated by "/" characters. The base topic level is retrieved from the Network Configuration Object (04h), instance #55. The other topic levels are specific to each dataset, according to the table below.

Dataset	**Dataset Type**	**Dataset Identifier**
Single ADI	"Parameter"	ADI name as given by attribute #1 of the ADI instance in the Application Data Object (FEh). If this attribute is not implemented, the default value will be used. Max 32 characters Default value: "Param<ADI number>"
Assembly mapping instance	"Assembly"	Assembly mapping instance name as given by attribute #13 of the assembly mapping instance in the Assembly Mapping Object (EBh). If this attribute is not implemented, the default value will be used. Max 32 characters Default value: "Asm<assembly mapping instance number>"
Transparent network payload	"Transparent"	The dataset identifier in decimal format without leading zeroes. Vendor specific.

11.3.2 Dataset Encoding

The Anybus CompactCom encodes the different datasets using JSON, see examples below.

Vendor specific datasets are sent transparently, and not encoded.

Data Type Translation

Anybus data types are translated to JSON data types according to the table below.

Anybus Data Type	JSON Data Type
BOOL	Boolean
SINT8/16/32/64	Number
UINT8/16/32/64	Number
FLOAT	Number
CHAR	String
OCTET	String A hexadecimal string representation of the octet value. E.g: 0xFF is translated to “FF”.
ENUM	Object , see below
BOOL1	Boolean
BITSx / BITx	JSON Array of Number where each bit is encoded to a Number of either 0 or 1. The array starts with the most significant bit.
PADx	Data type is ignored and will not be published.

The ENUM data type is encoded as a JSON object:

```
{
   "EnumValue" : <ADI value encoded as a Number>,
   "EnumStr" : <Enum string of the ADI value encoded as a String>
}
```

ADI variables of data type PADx will be ignored and not published.

ADI variables with more than one subelement, are encoded if the data type of the ADI is CHAR or OCTET. These are encoded and published as a single string. ADI variables with subelements of other types will be ignored and not published.

An ADI with an array of the data type CHAR will be published as a single string, while other arrays will be published as arrays.

Anybus Data Type	JSON Encoding with Example Values
UINT16	10
Array of UINT16	[0, 1, 2, 3, 4]
BOOL	True
OCTET string	“EDDAEBFE”
Array of OCTET	[“ED”, “DA”, “EB”, “FE”]

ENUM	{
	“EnumValue” : 2,
	“EnumStr” : “Two”,
	}

BIT2	[1,0]
Array of BIT2	[[1,0], [1,1], [0,0]]

JSON Encoding of Single ADI Datasets

Single ADI Datasets are encoded as follows:

```
{
  "<ADI name>" : {
                      "Value" : <ADI value>
                   },
  "Timestamp" : <Timestamp of the dataset>
}
```

Example of JSON encoding of a single ADI:

```
{
  "Single ADI" : {
                      "Value" : true
                   },
  "Timestamp" : 1526643062
}
```

Example of JSON encoding of a variable ADI:

```
{
  "Variable ADI" : {
                      "Value" : "String value"
                   },
  "Timestamp" : 1526643062
}
```

Example of JSON encoding of an array ADI with 3 elements:

```
{
  "Array ADI" : {
                   "Value" : [ 1, 2, 3 ]
                },
  "Timestamp" : 1526643062
}
```

JSON Encoding of Assembly Mapping Datasets

Assembly Mapping Datasets are encode as follows:

```
{
  "<Assembly mapping instance name>" : {
                          "<ADI #1 name>" : { "Value" : <ADI value> },
                          ...
                          <ADI #N name>" : { "Value" : <ADI value> },
                       },
  "Timestamp" : <Timestamp of the dataset>
}
```

Example of JSON encoding of an Assembly Mapping instance:

```
{
  "Example Assembly" : {
                          "Variable ADI": { "Value" : "String value" },
                          "Array ADI" : { "Value" : [ 1, 2, 3 ] },
                       },
  "Timestamp" : 1526643062
}
```

11.4 Stack Configuration

This section specifies the configuration of the MQTT stack, implemented in Anybus CompactCom. The configuration defines the capabilities of the stack implementation.

Parameter	
MQTT version	3.1.1
Clean session support	Yes
Retain bit support	Yes
QoS level support	Yes
Maximum payload	32768 bytes

12 CIP Objects

12.1 General Information

This chapter specifies the CIP-object implementation in the module. These objects can be accessed from the network, but not directly by the host application.

Mandatory objects

- *Identity Object (01h), p. 113*
- *Message Router (02h), p. 116*
- *Assembly Object (04h), p. 117*
- *Connection Manager (06h), p. 120*
- *QoS Object (48h), p. 129*
- *TCP/IP Interface Object (F5h), p. 138*
- *Ethernet Link Object (F6h), p. 141*

CIP Energy Objects:

- *Base Energy Object (4Eh), p. 130*
- *Power Management Object (53h), p. 132*

Optional Objects:

- *Port Object (F4h), p. 136* (Optional)
- *Parameter Object (0Fh), p. 124*
- *DLR Object (47h), p. 128*

Vendor Specific Objects:

- *ADI Object (A2h), p. 134*

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see *EtherNet/IP Host Object (F8h), p. 222* and commend details for Process_CIP_ Object_Request.

Unconnected CIP routing is supported, which means that a message can be sent to a device without first setting up a connection.

12.2 Translation of Status Codes

If an error occurs when an object is requested from the application, an error code is returned. These Anybus CompactCom error codes are translated to CIP status codes according to the table below.

Anybus CompactCom 40 Error Code		**CIP Status Code**	
Value	**Error**	**Value**	**Status**
00h	Reserved	1Eh	Embedded service error
01h	Reserved	1Eh	Embedded service error
02h	Invalid message format	1Eh	Embedded service error
03h	Unsupported object	05h	Path destination unknown
04h	Unsupported instance	05h	Path destination unknown
05h	Unsupported Command	08h	Service not supported
06h	Invalid CmdExt(0)	14h	Depending on Anybus CompactCom Service returning this reply, e.g. attribute not supported
07h	Invalid CmdExt(1)	-	Depending on Anybus CompactCom Service returning this reply
08h	Attribute not settable	0Eh	Attribute not settable
09h	Attribute not gettable	2Ch	Attribute not gettable
0Ah	Too Much Data	15h	Too much data
0Bh	Not Enough Data	13h	Not enough data
0Ch	Out of range	09h	Invalid attribute value
0Dh	Invalid state	0Ch	Object state conflict
0Eh	Out of resources	02h	Resource unavailable
0Fh	Segmentation failure	1Eh	Embedded service error
10h	Segmentation buffer overflow	23h	Buffer overflow
11h	Value too high	09h	Invalid attribute value
12h	Value too low	09h	Invalid attribute value
13h	Attribute controlled	0Fh	A permission/privilege check failed
14h	Message channel too small	11h	Reply data too large
FFh	Object Specific Error	1Fh	Vendor specific error. No additional error codes will be sent on EtherNet/IP
Other	-	1Eh	Embedded service error

For further information about the Anybus CompactCom error codes, please consult the Anybus CompactCom 40 Software Design Guide.

12.3 Identity Object (01h)

Category

Extended

Object Description

The Identity Object provides identification of and general information about the module.

The object supports multiple instances. Instance 1, which is the only mandatory instance, describes the whole product. It is used by applications to determine what nodes are on the network and to match an EDS file with a product on the network. The other (optional) instances describe different parts of the product, e.g. the software.

If modular device functionality is enabled, a list of the modules in the slots can be retrieved and made available to the network master by sending a get request to class attribute 100.

Instance attributes 1 - 7 can be customized by implementing the EtherNet/IP Host Object.

Additional identity instances can be registered by implementing the CIP Identity Host Object (host application object).

See also ....

- *EtherNet/IP Host Object (F8h), p. 222*
- *CIP Identity Host Object (EDh), p. 213*

Supported Services

Class:	Get_Attribute_Single Get_Attribute_All
Instance:	Get_Attribute_Single Set_Attribute_Single Get_Attribute_All Get_Attribute_List Set_Attribute_List Reset

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max instance	Get	UINT	Maximum instance number
3	Number of instances	Get	UINT	Number of instances
100	Module ID List	Get	Array of UINT32	If modular device functionality is enabled, a request to this attribute will generate a Get_List request to the Modular Device Object in the host application.

Instance Attributes

Attributes #1–4 and #6–7 can be customized by implementing the EtherNet/IP Host Object, see *EtherNet/IP Host Object (F8h), p. 222*

#	Name	Access	Type	Value/Description
1	Vendor ID	Get	UINT	005Ah (HMS Industrial Networks)
2	Device Type	Get	UINT	002Bh (Generic Device)
3	Product Code	Get	UINT	0037h (Anybus CompactCom 40 EtherNet/IP)
4	Revision	Get	Struct of: USINT USINT	Major and minor firmware revision
5	Status	Get	WORD	See Device Status table below
6	Serial Number	Get	UDINT	Unique serial number (assigned by HMS)
7	Product Name	Get	SHORT_STRING	“Anybus CompactCom 40 EtherNet/IP (TM)”
11	Active language	Set	Struct of: USINT USINT USINT	Requests sent to this instance are forwarded to the Application Object. If the request is accepted, the module will update the language accordingly.
30	Supported Language List 2	Get	Struct of: UINT Array of: USINT USINT USINT	List of languages supported by the host application. The list is read from the Application Object and translated to CIP standard. By default the only supported language is English. The application has to implement the corresponding attributes in the application object to enable more languages.

Device Status

bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured This bit shows if the product has other settings than "out-of-box". The value is set to true if the configured attribute in the Application Object is set and/or the module’s NV storage is changed from default.
3	(reserved)
4... 7	Extended Device Status: Value: Meaning: 0000b Unknown 0010b Faulted I/O Connection 0011b No I/O connection established 0100b Non volatile configuration bad 0101b Major fault 0110b Connection in Run mode 0111b Connection in Idle mode (other) (reserved)
8	Set for minor recoverable faults. See *Diagnostic Object (02h), p. 149*
9	Set for minor unrecoverable faults. See *Diagnostic Object (02h), p. 149*
10	Set for major recoverable faults. See *Diagnostic Object (02h), p. 149*
11	Set for major unrecoverable faults. See *Diagnostic Object (02h), p. 149*
12... 15	(reserved)

Service Details: Reset

ℹ *This service is not supported if safety is enabled in the Functional Safety Object (E8h).*

The module forwards reset requests from the network to the host application. For more information about network reset handling, consult the general Anybus CompactCom 40 Software Design Guide.

There are two types of network reset requests on EtherNet/IP:

Type 0: Power Cycling Reset This service emulates a power cycling of the module, and corresponds to Anybus reset type 0 (Power cycling). For further information, consult the general Anybus CompactCom 40 Software Design Guide.

Type 1: Out of box reset This service sets a “out of box” configuration and performs a reset, and corresponds to Anybus reset type 2 (Power cycling + factory default). For further information, consult the general Anybus CompactCom 40 Software Design Guide.

12.4 Message Router (02h)

Category

Extended

Object Description

The Message Router Object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical module.

In the Anybus CompactCom module it is used internally to direct object requests.

Supported Services

Class:	-
Instance:	-

Class Attributes

-

Instance Attributes

-

12.5 Assembly Object (04h)

Category

Extended

Object Description

The Assembly object uses static assemblies and holds the Process Data sent/received by the host application. It allows data to and from each object to be sent or received over a single connection. The default assembly instance IDs used are in the vendor specific range.

It is possible for the application to create and support up to six consuming and six producing instances if the Assembly Mapping Object is implemented.

The terms “input” and “output” are defined from the network’s point of view. An input will produce data on the network and an output will consume data from the network.

See also ....

- *EtherNet/IP Host Object (F8h), p. 222*
- Assembly Mapping Object (see Anybus CompactCom 40 Software Design Guide)

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max instance	Get	UINT	Maximum instance number

Instance 03h Attributes (Heartbeat, Input-Only)

This instance is used as heartbeat for Input-Only connections. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

#	Name	Access	Type	Value/Description
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

Instance 04h Attributes (Heartbeat, Listen-Only)

This instance is used as heartbeat for listen-only connections. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

#	Name	Access	Type	Value/Description
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

Instance 05h Attributes (Configuration Data)

Configuration Data that is sent through the service Forward_Open will be written to this instance.

#	Name	Access	Type	Value/Description
3	Data	Set	N/A	- (Configuration data written to the application when the forward open command has the configuration data included)- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

See command details for Set_Configuration_Data nad Get_Contfiguration_Data in the *EtherNet/IP Host Object (F8h), p. 222*.

Instance 06h Attributes (Heartbeat, Input-Only Extended)

This instance is used as heartbeat for input-only extended connections, and does not carry any attributes. The state of connections made to this instance does not affect the state of the Anybus CompactCom module, i.e. if the connection times out, the module does not switch to the Error state. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value/Description
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

Instance 07h Attributes (Heartbeat, Listen-Only Extended)

This instance is used as heartbeat for listen-only extended connections, and does not carry any attributes. The state of connections made to this instance does not affect the state of the Anybus CompactCom 40 module, i.e. if the connection times out, the module does not switch to the Error state. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value/Description
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

Instance 64h Attributes (Producing Instance)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value/Description
3	Produced Data	Get	Array of BYTE	This data corresponds to the Write Process Data.
4	Size	Get	UINT	Number of bytes in attribute 3

See also...

Network Data Exchange, p. 28

EtherNet/IP Host Object (F8h), p. 222(Instance attribute #7)

Instance 96h Attributes (Consuming Instance)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value/Description
3	Produced Data	Set	Array of BYTE	This data corresponds to the Read Process Data.
4	Size	Get	UINT	Number of bytes in attribute 3

See also...

Network Data Exchange, p. 28

EtherNet/IP Host Object (F8h), p. 222 (Instance attribute #8)

12.6 Connection Manager (06h)

Category

Extended

Object Description

This object is used for connection and connectionless communications, including establishing connections across multiple subnets.

Supported Services

Class:	-
Instance:	Get Attribute All Get Attribute Single Set Attribute Single Large_Forward_Open Forward_Open Forward_Close Unconnected Send (when unconnected routing is enabled)

Class Attributes

(No supported class attributes)

Instance Attributes

#	Name	Access	Type	Value/Description
1	Open Requests	Set	UINT	Number of Forward Open service requests received.
2	Open Format Rejects	Set	UINT	Number of Forward Open service requests which were rejected due to bad format.
3	Open Resource Rejects	Set	UINT	Number of Forward Open service requests which were rejected due to lack of resources.
4	Open Other Rejects	Set	UINT	Number of Forward Open service requests which were rejected for reasons other than bad format or lack of resources.
5	Close Requests	Set	UINT	Number of Forward Close service requests received.
6	Close Format Rejects	Set	UINT	Number of Forward Close service requests which were rejected due to bad format.
7	Close Other Rejects	Set	UINT	Number of Forward Close service requests which were rejected for reasons other than bad format.
8	Connection Timeouts	Set	UINT	Total number of connection timeouts that have occurred in connections controlled by this Connection Manager.

Class 0 Connection Details

General

Class 0 connections are only supported for safety connections. The Anybus CompactCom device will act as a transparent bridge for safety connections, meaning that open and close requests for safety connections and safety I/O data will be forwarded to the safety module. Class 0 connections use UDP transport.

Total number of supported class 0 connections:	2
Max input connection size:	241 bytes (Including the Mode Byte, Actual, Complement and Time stamp sections.)
Max output connection size:	239 bytes (Including the Mode Byte, Actual, Complement and Time stamp sections.)
Supported RPI (Requested Packet Interval):	1... 20000 ms

Class 1 Connection Details

General

Class 1 connections are used to transfer I/O data, and can be established to instances in the Assembly Object. Each Class 1 connection will establish two data transports; one consuming and one producing. The heartbeat instances can be used for connections that shall only access inputs. Class 1 connections use UDP transport. Null forward open is supported.

Total number of supported class 1 connections:	4
Max input connection size:	1448 bytes with Large_Forward_Open, 509 bytes with Forward_Open
Max output connection size:	1448 bytes with Large_Forward_Open, 505 bytes with Forward_Open
Supported RPI (Requested Packet Interval):	1... 3200ms
T→O Connection type:	Point-to-point, Multicast, Null
O→-T Connection type:	Point-to-point, Null
Supported trigger types:	Cyclic, CoS (Change of State)
Supported priorities:	Low, High, Scheduled, Urgent

T	Target, in this case the module
O	Origin, in this case the master

Connection Types

- **Exclusive-Owner connection**

  This type of connection controls the outputs of the Anybus module and does not depend on other connections.

Max. no. of Exclusive-Owner connections:	1
Connection point O →T:	Assembly Object, instance 96h (Default)
Connection point T →O:	Assembly Object, instance 64h (Default)

- **Input-Only connection**

  This type of connection is used to read data from the Anybus module without controlling the outputs. It does not depend on other connections.

Max. no. of Input-Only connections:	Up to 4 (Shared with Exclusive-Owner and Listen-Only connections)
Connection point O →T:	Assembly Object, instance 03h (Default)
Connection point T →O:	Assembly Object, instance 64h (Default)

  Please not that if an Exclusive-Owner connection has been opened towards the module and times out, the Input-Only connection times out as well. If the Exclusive-Owner connection is properly closed, the Input-Only connection remains unaffected.

- **Input-Only Extended connection**

  This connections functionality is the same as the standard Input-Only connection. However when this connection times out it does not affect the state of the application.

Connection point O →T:	Assembly Object, instance 06h (Default)
Connection point T →O:	Assembly Object, instance 64h (Default)

- **Listen-Only connection**

  This type of connection requires another connection in order to exist. If that connection (Exclusive-Owner or Input-Only) is closed, the Listen-Only connection will be closed as well.

Max. no. of Input-Only connections:	Up to 4 (Shared with Exclusive-Owner and Input-Only connections)
Connection point O →T:	Assembly Object, instance 04h (Default)
Connection point T →O:	Assembly Object, instance 64h (Default)

- **Listen-Only Extended connection**

  This connections functionality is the same as the standard Listen-Only connection. However when this connection times out it does not affect the state of the application.

Connection point O →T:	Assembly Object, instance 07h (Default)
Connection point T →O:	Assembly Object, instance 64h (Default)

- **Redundant-Owner connection**

  This connection type is not supported by the module.

Class 3 Connection Details

General

Class 3 connections are used to establish connections towards the message router. Thereafter, the connection is used for explicit messaging. Class 3 connections use TCP transport.

No. of simultaneous Class 3 connections:	6
Supported RPI (Requested Packet Interval):	100... 10000 ms
T→O Connection type:	Point-to-point
O→-T Connection type:	Point-to-point
Supported trigger type:	Application
Supported connection size:	1526 bytes

12.7 Parameter Object (0Fh)

Category

Extended

Object Description

The Parameter Object provides an interface to the configuration data of the module. It can provide all the information necessary to define and describe each of the module configuration parameters, as well as a full description of each parameter, including minimum and maximum values and a text string describing the parameter. Configuration tools, such as RSNetworx, can extract information about the Application Data Instances (ADIs) and present them with their actual name and range to the user.

Since this process may be somewhat time consuming, especially when using the serial host interface, it is possible to disable support for this functionality in the EtherNet/IP Host Object.

Each parameter is represented by one instance. Instance numbers start at 1, and are incremented by one, with no gaps in the list. Due to limitations imposed by the CIP standard, ADIs containing multiple elements (i.e. arrays etc.) cannot be represented through this object. In such cases, default values will be returned.

See also ....

- *ADI Object (A2h), p. 134* (CIP Object)
- *EtherNet/IP Host Object (F8h), p. 222* (Host Application Object)

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Set_Attribute_Single Get_Attributes_All Get_Enum_String

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)
2	Max instance	Get	UINT	Maximum created instance number = class attribute 3 in the Application Data Object (see Anybus CompactCom 40 Software Design Guide)
8	Parameter Class Descriptor	Get	WORD	Default: 0000 0000 0000 1011b Bit: Contents: 0 Supports parameter instances 1 Supports full attributes 2 Must do non-volatile storage save command 3 Parameters are stored in non-volatile storage
9	Configuration Assembly Instance	Get	UINT	0000h (Application does not support configuration data) 0005h (If the application supports configuration data, unless the configuration instance number has been changed using attribute 15 in the EtherNet/IP Host Object.)

Instance Attributes

#	Name	Access	Type	Value/Description
1	Parameter Value	Get/Set	Specified in attributes 4, 5 & 6.	Actual value of parameter This attribute is read-only if bit 4 of Attribute #4 is true
2	Link Path Size	Get	USINT	0007h (Size of link path in bytes)
3	Link Path	Get	Packed EPATH	20 A2 25 nn nn 30 05h (Path to the object from where this parameter's value is retrieved, in this case the ADI Object)
4	Descriptor	Get	WORD	Bit: Contents: 0 Supports Settable Path (N/A) 1 Supports Enumerated Strings 2 Supports Scaling (N/A) 3 Supports Scaling Links (N/A) 4 Read only Parameter 5 Monitor Parameter (N/A) 6 Supports Extended Precision Scaling (N/A)
5	Data Type	Get	USINT	Data type code
6	Data Size	Get	USINT	Number of bytes in parameter value
7	Parameter Name String	Get	SHORT_STRING	Name of the parameter, truncated to 16 chars
8	Units String	Get	SHORT_STRING	“” (default string)
9	Help String	Get	SHORT_STRING	
10	Minimum Value	Get	(Data type)	Minimum value of parameter The Data Type is defined in attribute 5.
11	Maximum Value	Get	(Data type)	Maximum value of parameter The Data Type is defined in attribute 5.
12	Default Value	Get	(Data type)	Default value of parameter The Data Type is defined in attribute 5.
13	Scaling Multiplier	Get	UINT	0001h
14	Scaling Divisor	Get	UINT	
15	Scaling Base	Get	UINT	
16	Scaling Offset	Get	INT	0000h
17	Multiplier Link	Get	UINT	
18	Divisor Link	Get	UINT	
19	Base Link	Get	UINT	
20	Offset Link	Get	UINT	
21	Decimal Precision	Get	USINT	00h

Default Values

#	Name	Value	Comments
1	Parameter Value	0	-
2	Link Path Size	0	Size of link path in bytes.
3	Link Path	-	NULL Path
4	Descriptor	0010h	Read only Parameter
5	Data type	C6h	USINT
6	Data size	1	-
7	Parameter Name String	“Reserved”	-
8	Units String	“”	-
9	Help String	“”	-
10	Minimum value	N/A	0
11	Maximum value	N/A	0
12	Default value	N/A	0
13	Scaling Multiplier	N/A	1
14	Scaling Divisor	N/A	1
15	Scaling Base	N/A	1
16	Scaling Offset	N/A	0
17	Multiplier Link	N/A	0
18	Divisor Link	N/A	0
19	Base Link	N/A	0
20	Offset Link	N/A	0
21	Decimal Precision	N/A	0

12.8 DLR Object (47h)

Category

Extended

Object Description

The Device Level Ring (DLR) Object provides the status information interface for the DLR protocol. This protocol enables the use of an Ethernet ring topology, and the DLR Object provides the CIP application-level interface to the protocol.

This object is not available if DLR is disabled in the EtherNet/IP Host Object, see *Ethernet Host Object (F9h), p. 231*

.

Supported Services

Class:	Get_Attribute_Single Get_Attributes_All
Instance:	Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0003h (Object revision)

Instance Attributes

Attributes #1–4 and #6–7 an be customized by implementing the EtherNet/IP Host Object, see *EtherNet/IP Host Object (F8h), p. 222*

#	Name	Access	Type	Value/Description
1	Network Topology	Get	USINT	Bit: Contents: 0 “Linear” 1 “Ring”
2	Network Status	Get	USINT	Bit: Contents: 0 “Normal” (N/A) 1 “Ring Fault” 2 “Unexpected Loop Detected” 3 “Partial Network Fault” 4 “Rapid Fault/Restore Cycle”
10	Active Supervisor Address	Get	Struct of: UDINT Array of: 6 USINTs	This attribute holds the IP address (IPv4) and/or the Ethernet Mac address of the active ring supervisor.
12	Capability Flags	Get	DWORD	82h (Beacon-based ring node, Flush_Table frame capable)

12.9 QoS Object (48h)

Category

Extended

Object Description

Quality of Service (QoS) is a general term that is applied to mechanisms used to treat traffic streams with different relative priorities or other delivery characteristics. Standard QoS mechanisms include IEEE 802.1D/Q (Ethernet frame priority) and Differentiated Services (DiffServ) in the TCP/IP protocol suite.

The QoS Object provides a means to configure certain QoS related behaviors in EtherNet/IP devices.

The object is required for devices that support sending EtherNet/IP messages with nonzero DiffServ code points (DSCP), or sending EtherNet/IP messages in 802.1Q tagged frames.

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

Instance Attributes

Attributes #1–4 and #6–7 an be customized by implementing the EtherNet/IP Host Object, see *EtherNet/IP Host Object (F8h), p. 222*

#	Name	Access	Type	Value/Description
1	802.1Q Tag Enable	Set	USINT	Enables or disables sending 802.1Q frames. Value: Contents: 0 Disabled (Default) 1 Enabled
4	DSCP Urgent	Set	USINT	CIP transport class 1 messages with priority Urgent Default: 55
5	DSCP Scheduled	Set	USINT	CIP transport class 1 messages with priority Scheduled Default: 47
6	DSCP High	Set	USINT	CIP transport class 1 messages with priority High Default: 43
7	DSCP Low	Set	USINT	CIP transport class 1 messages with priority Low Default: 31
8	DSCP Explicit	Set	USINT	CIP UCMM and CIP class 3 Default: 27

12.10 Base Energy Object (4Eh)

Category

Extended

Object Description

The Base Energy Object acts as an “Energy Supervisor” for CIP Energy implementations. It is responsible for providing a time base for energy values, provides energy mode services, and can provide aggregation services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is energy type independent and allows energy type specific data and functionality to be integrated into an energy system in a standard way. The Anybus CompactCom 40 EtherNet/IP IIoT Secure supports one instance of the Base Energy Object. For instance, an electric power monitor may count metering pulse output transitions of a separate metering device. The count of such transitions, represented by a Base Energy Object instance, would reflect the energy consumption measured by the separate metering device.

An instance of the Base Energy Object may exist as a stand-alone instance, or it may exist in conjunction with an Electrical and/or Non-Electrical Energy Object instance (These objects are not implemented in the Anybus CompactCom 40 EtherNet/IP IIoT Secure). If an instance of any of these objects is implemented in a device, it must be associated with a Base Energy Object instance in the device.

For this object to be able to access the network, the Energy Reporting Object (E7h) must be implemented in the host application, see *Energy Reporting Object (E7h), p. 209*.

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

Instance Attributes

Attributes #1–4 and #6–7 an be customized by implementing the EtherNet/IP Host Object, see *EtherNet/IP Host Object (F8h), p. 222*

#	Name	Access	Type	Value/Description
1	Energy/ Resource Type	Get	UINT	Type of energy managed by this instance Always 0 (Generic)
2	Base Energy Object Capabilities	Get	UINT	Always 0 (Energy measured)
3	Energy Accuracy	Get	UINT	Specifies the accuracy of power and energy metering results, either in 0.01 percent of reading (default) or 0.01 of other units specified in attribute #4. If 0, unknown.
4	Energy Accuracy Basis	Get	UINT	Always 0 (Percent of reading)
7	Consumed Energy Odometer	Get	ODOMETER	The value of the consumed energy.
8	Generated Energy Odometer	Get	ODOMETER	The value of the generated energy.
12	Energy Type Specific Object Path	Get	Struct of: UINT (Path size) padded EPATH (Path)	NULL path

- Depending on whether the instance reports consumed or generated energy, either attribute #7 or attribute #8 is required.
- The struct data type ODOMETER makes it possible to represent very large values, for more information please consult the CIP specification Volume 1 (CIP Common).

12.11 Power Management Object (53h)

Category

Extended

Object Description

The Power Management Object provides standardized attributes and services to support the control of devices into and out of paused or sleep states. The Energy Control Object (F0h) has to be implemented for this object to gain access to the network.

See also ..

- Energy Control Object (F0h) (Anybus CompactCom 40 Software Design Guide)

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Power_Management Set_Pass_Code Clear_Pass_Code

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

Instance Attributes

#	Name	Access	Type	Value/Description
1	Power Management Command	Get	DWORD	Collection of bit fields comprising the most recent power management request.
2	Power Management Status	Get	DWORD	Collection of bit fields providing Power Management status information.
3	Client Path	Get	Struct of:	Specifies the EPATH from this instance (server) to its current owner (client).
			UINT (Path Size)	Size of path (in words
			Padded EPATH (Path)	
4	Number of Power Management Modes	Get	UINT	Number of Power Management Mode array entries in attribute 5.
5	Power Management Nodes	Get	Array of:	Array of low power modes
			Struct of:	Modes (Array of mode structures)
			USINT	Minimum Pause Units (Specifies the unit of Minimum Pause Time)
			UINT	Minimum Pause Time
			USINT	Resume Units (Specifies the unit of Resume Time)
			UINT	Resume Time (Required time to transition from the paused stated to the owned state.
			REAL	Power Level (Power in kW for this mode)
			BOOL	Availability (Specifies whether this mode can be entered given the current device state)
6	Sleeping State Support	Get	BOOL	0 (Sleeping state not supported)

12.12 ADI Object (A2h)

Category

Extended

Object Description

This object maps instances in the Application Data Object to EtherNet/IP. All requests to this object will be translated into explicit object requests towards the Application Data Object in the host application; the response is then translated back to CIP-format and sent to the originator of the request.

The ADI Object can be disabled using attribute 30 in the EtherNet/IP Host Object (F8h). This attribute can also be used to change the ADI Object number.

See also ..

- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- *Parameter Object (0Fh), p. 124* (CIP Object)
- *EtherNet/IP Host Object (F8h), p. 222*

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max Instance	Get	UINT	Equals attribute #4 in the Application Data Object
3	Number of instances	Get	UINT	Equals attribute #3 in the Application Data Object

For information about the Application Data Object, please consult the Anybus CompactCom 40 Software Design Guide.

Instance Attributes

Each instance corresponds to an instance within the Application Data Object (for more information, please consult the general Anybus CompactCom 40 Software Design Guide).

#	Name	Access	Type	Value/Description
1	Name	Get	SHORT_STRING	Parameter name (Including length)
2	ABCC Data type	Get	Array of USINT	Data type of instance value
3	No. of Elements	Get	USINT	Number of elements of the specified data type
4	Descriptor	Get	Array of USINT	Bit field describing the access rights for this instance Bit: Meaning: 0 1 = Get Access 1 1 = Set Access 2 (reserved, set to 0) 3 1 = Write process data mapping possible 4 1 = Read process data mapping possible 5 1 = NVS parameter 6 1 = Data notification enabled
5	Value	Get/Set	Determined by attributes #2, #3 and #9	Instance value
6	Max Value	Get		The maximum permitted parameter value.
7	Min Value	Get		The minimum permitted parameter value.
8	Default Value	Get		The default parameter value.
9	Number of subelements	Get	Array of UINT	Number of subelements in the ADI. Default value is 1 unless implemented in the application. The size of the array depends on attribute #3.

Attributes #5–8 are converted to/from CIP standard by the module

12.13 Port Object (F4h)

Category

Extended

Object Description

The Port Object describes the CIP ports present on the device. Each routable CIP port is described in a separate instance. Non-routable ports may be described. Devices with a single CIP port are not required to support this object.

The object exists only if enabled in the EtherNet/IP Host Object (Instance Attribute #17).

See also ..

- *EtherNet/IP Host Object (F8h), p. 222* (Anybus Module Object)
- *CIP Port Configuration Object (0Dh), p. 192* (Host Application Object)

Supported Services

Class:	Get_Attributes_All Get_Attribute_Single
Instance:	Get_Attributes_All Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max Instance	Get	UINT	Max. instance number
3	Number of Instances	Get	UINT	Number of ports currently created.
8	Entry Port	Get	UINT	Returns the instance of the Port Object that describes the port through which this request entered the device.
9	Port Instance Info	Get	Array of:	Array of structures containing instance attributes 1 and 2 from each instance. The array is indexed by instance number, up to the maximum number of instances. The value at index 1 (offset 0) and any non-instantiated instances will be zero.
			Struct of: UINT (Type) UINT (Number)	Enumerates the type of port (see instance attribute #1) CIP port number associated with this port (see instance attribute #2)

Instance Attributes (Instance #1)

#	Name	Access	Type	Value/Description
1	Port Type	Get	UINT	0h (default) 4h (if the application registers a port)
2	Port Number	Get	UINT	2h
3	Link Object	Get	Struct of: UINT Padded EPATH	- 2h (Path Length) 20 F5 24 01h (Link Path)
4	Port Name	Get	SHORT_STRING	“EtherNet/IP”
5	Port Type Name	Get	SHORT_STRING	“”
6	Port Description	Get	SHORT_STRING	“”
7	Node Address	Get	Padded EPATH	-
10	Port Routing Capabilities	Get	UDINT	1h (Routing of incoming Unconnected Messaging supported)

See also...

CIP Port Configuration Object (0Dh), p. 192

Instance Attributes (Instances #2... #8)

#	Name	Access	Type	Value/Description
1	Port Type	Get	UINT	Enumerates the type of port
2	Port Number	Get	UINT	CIP port number associated with this port
3	Link Object	Get	Struct of: UINT Padded EPATH	- Path length (number of 16-bit words) Logical path segments which identify the object for this port. The path must consist of one logical class segment and one logical instance segment. The maximum size is 12 bytes.
4	Port Name	Get	SHORT_STRING	Name of port, e.g. “Port A”. Max. 64 characters.
5	Port Type Name	Get	SHORT_STRING	“”
6	Port Description	Get	SHORT_STRING	“”
7	Node Address	Get	Padded EPATH	Node number of this device on port. The range within this data type is restricted to a Port Segment.
8	Port Node Range	Get	Struct of: UINT (Min.) UINT (Max.)	- Min. node number on port Max. node number on port
10	Port Routing Capabilities	Get	UDINT	1h (Routing of incoming Unconnected Messaging supported)

See also...

CIP Port Configuration Object (0Dh), p. 192 , “Instance Attributes.”.

12.14 TCP/IP Interface Object (F5h)

Category

Extended

Object Description

This object provides the mechanism to configure the TCP/IP network interface of the module. It groups the TCP/IP-related settings in one instance for each TCP/IP capable communications interface.

See also ..

- *Communication Settings, p. 26*
- *Network Configuration Object (04h), p. 151* (Anybus Module Object)

Supported Services

Class:	Get_Attribute_All Get_Attribute_Single
Instance:	Get_Attribute_All Get_Attribute_Single Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0004h (Object revision)
2	Max instance	Get	UINT	1 (Maximum instance number)
3	Number of instances	Get	UINT	1 (Number of instances)
6	Maximum ID Number Class Attributes	Get	UINT	7 (The attribute number of the last implemented class attribute)
7	Maximum ID Number Instance Attributes	Get	UINT	13 (The attribute number of the last implemented instance attribute)

Instance Attributes

#	Name	Access	Type	Value	Comments
1	Status	Get	DWORD	-	Bit: Meaning: (reserved, set to 0) 0–3 When set to h, attribute #5 contains valid configuration from DHCP or non-volatile storage. When set to 2h, attribute #5 contains valid configuration from hardware settings. Remaining values are reserved for future use. 4 Multicast pending if set to 1. 5 Interface configuration pending if set to 1. A new configuration will be loaded at the next reset. 6 AcdStatus. Set to 1 if an address conflict is detected. Address conflict detection is enabled/ disabled in attribute #10. 7 AcdFault 8–31 (reserved, set to 0)
2	Configuration Capability	Get	DWORD	-	Bit: Meaning: 0-1: Always 0. For more information, consult the CIP specifications. 2: If set to 1, the device is capable of acting as a DHCP client. The bit is set to 0 if attribute #24 (Enable DHCP Client) is disabled in the *Ethernet Host Object (F9h), p. 231* 3: Always 0. For more information, consult the CIP specifications. 4: The 'Configuration Settable'-bit reflects the value of instance attribute #9 in the "EtherNet/IP Host Object (F8h)" on page 161. 5: The module is hardware configurable when this bit is set to 1. The bit will be set if any of the address attributes is set in the Network Configuration Object (04h) during setup or if attribute #6 (Hardware configurable address) in the Application Object (FFh) is set. 6: Always 0. For more information, consult the CIP specifications. 7: If set to 1, the device is capable of detecting address conflicts. The bit is set to 0 if address conflict detection is disabled in the Ethernet Host Object, see page *231* 8 - 31: (reserved, set to 0)
3	Configuration Control	Get/Set	DWORD	-	Value: Meaning 0: Configuration from non-volatile memory 2: Configuration from DHCP
4	Physical Link Object	Get	Struct of: UINT (Path size) Padded EPATH	- 0002h 20 F6 24 03h	- - Path to Ethernet Link Object, Instance #3
5	Interface Configuration	Get/Set	Struct of: UDINT (IP) UDINT (Mask) UDINT (GW) UDINT (DNS1) UDINT (DNS2) STRING (Domain)		- IP address Subnet mask Default gateway Primary DNS Secondary DNS Default domain
6	Host Name	Get/Set	STRING	-	Host name of Anybus module
8	TTL Value	Get/Set	USINT	1	TTL value for EtherNet/IP multicast packets
9	Mcast Config	Set	Struct of:		IP multicast configuration.

#	Name	Access	Type	Value	Comments
	Alloc Control		USINT	0	Value: Meaning: 0: Use default allocation algorithm to generate multicast addresses 1: Allocate multicast addresses according to the values in the “Num Mcast”- and “Mcast Start Addr”-fields.
	(reserved)		USINT	0	Set to zero. Do not change.
	Num Mcast		UINT	-1	Number of multicast addresses to allocate for EtherNet/IP
	Mcast Start Addr		UDINT	-	Starting multicast address from which to begin allocation
10	SelectAcd	Set	Bool	1	Value: Meaning: 0: Disable ACD 1: Enable ACD (Default). If ACD (address conflict detection) is enabled, bit 6 in attribute #1 will be set if an ACD conflict is detected. The Network Status LED will also indicate a detected conflict, see *Front View, p. 244* .
11	LastConflictDetected	Set	Struct of:		ACD Diagnostic parameters Related to the last conflict detected.
	AcdActiviity		USINT	-	State of ACD activity when last conflict detected.
	RemoteMAC		ARRAY of 6 USINT	-	MAC address of remote node form the ARP PDU in which a conflict was detected.
	ArpPdu		ARRAY of 28 USINT	-	Copy of the raw ARP PDU in which a conflict was detected.
12	EIP QuickConnect	Set	Bool	0	Value: Meaning: 0: Disable EIP QuickConnect (Default) 1: Enable EIP QuickConnect If EIP QuickConnect is enabled, the QuickConnect feature will direct EtherNet/IP target devices to quickly power up and join an EtherNet/IP network.
13	Encapsulation inactivity timeout	Set	UINT	0 - 3600	Number of seconds of inactivity before a TCP connection is closed. 0: Disabled

- Support for configuring network settings (attributes #3 and #5) from the network can be disabled by implementing attribute #9 in the EtherNet/IP Host Object, see *EtherNet/IP Host Object (F8h), p. 222*
- Attributes #10 and #11 will not be available if ACD is disabled using attribute #11 in the Ethernet Host Object (F9h).
- Attribute #12:
  - If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification.
  - This attribute exists if attribute #26 in the EtherNet/IP Host Object is implemented, see *EtherNet/IP Host Object (F8h), p. 222*.

12.15 Ethernet Link Object (F6h)

Category

Extended

Object Description

This object maintains link specific counters and status information for an IEEE 802.3 communications interface. Exactly one instance for each communications interface on the module is supported. Instances for internally accessible interfaces can also be supported.

See also ..

- *Communication Settings, p. 26*
- *Network Configuration Object (04h), p. 151* (Anybus Module Object)

Supported Services

Class:	Get_Attributes_All Get_Attribute_Single
Instance:	Get_Attributes_All Get_Attribute_Single Set_Attribute_Single Get_And_Clear

Class Attributes

By default, three instances (port 1, port 2 and the internal port) are implemented, meaning that two ports are activated.

If port 2 is inactivated in the Port 2 State attribute of the Ethernet Host Object (F9h), only one instance (port 1) should be implemented.

#	Name	Access	Type	Value
1	Revision	Get	UINT	0004h (Object revision)
2	Max Instance	Get	UINT	1 or 3 (Maximum instance number)
3	Number of Instances	Get	UINT	1 or 3 (Number of instances)
6	Maximum ID Number Class Attributes	Get	UINT	7 (The attribute number of the last implemented class attribute.)
7	Maximum ID Number Instance Attributes	Get	UINT	11 (The attribute number of the last implemented instance attribute.)

Instance Attributes

#	Name	Access	Type	Value	Comments
1	Interface Speed	Get	UDINT	10 or 100	Actual Ethernet interface speed.
2	Interface Flags	Get	DWORD	-	See table “Interface Flags” below.
3	Physical Address	Get	Array of 6 USINTs	(MAC ID)	Physical network address, i.e. assigned MAC address.
4	Interface Counters	Get	Struct of:		
	In Octets		UDINT	N/A	Octets received on the interface
	In Ucast Packets		UDINT	N/A	Unicast packets received on the interface
	In NUcast Packets		UDINT	N/A	Nonunicast packets received on the interface
	In Discards		UDINT	N/A	Inbound packets with unknown protocol
	In Errors		UDINT	N/A	Inbound packets that contain errors (does not include In discards)
	In Unknown Protos		UDINT	N/A	Inbound packets with unknown protocol
	Out Octets		UDINT	N/A	Octets sent on the interface
	Out Ucast Packets		UDINT	N/A	Unicast packets sent on the interface
	Out NUcast Packets		UDINT	N/A	Nonunicast packets sent on the interface
	Out Discards		UDINT	N/A	Outbound packets with unknown protocol
	Out Errors		UDINT	N/A	Outbound packets that contain errors (does not include Out discards)
5	Media Counters	Get	Struct of:		Media specific counters
	Alignment Errors		UDINT	N/A	Frames received that are not an integral number of octets in length
	FCS Errors		UDINT	N/A	Frames received that do not pass the FCS check
	Single Collisions		UDINT	N/A	Successfully transmitted frames that have experienced exactly one collision
	Multiple Collisions		UDINT	N/A	Successfully transmitted frames that have experienced more than one collision
	SQE Test Errors		UDINT	0	The number of times the SQE test error message is generated(Counter not provided with current PHY interface)
	Deferred Transmissions		UDINT	N/A	Frames for which the first transmission attempt is delayed because the medium is busy
	Late Collisions		UDINT	N/A	The number of times a collision is detected later than 512 bit-times into the transmission of a packet
	Excessive Collisions		UDINT	N/A	Frames for which a transmission fails due to excessive collisions
	MAC Transmit Errors		UDINT	N/A	Frames for which a transmission fails due to an internal MAC sublayer receive error
	Carrier Sense Errors		UDINT	N/A	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
	Frame Too Long		UDINT	N/A	Frames received that exceed the maximum permitted frame size
	MAC Receive Errors		UDINT	N/A	Frames for which reception on an interface fails due to an internal MAC sublayer receive error

#	Name	Access	Type	Value	Comments
6	Interface Control	Get/Set	Struct of:		
	Control Bits		WORD	-	Interface control bits
	Forced Interface Speed		UINT	-	Speed at which the interface shall be forced to operate. Returns 'Object state Conflict' if auto-negotiation is enabled
7	Interface Type	Get	USINT	-	See table "Interface State" below.
8	Interface State	Get	USINT	-	See table "Interface Type" below.
9	Admin State	Get/Set	USINT	-	See table "Admin State" below.
10	Interface Label	Get	SHORT_STRING	—	See table "Interface Label" below.
11	Interface Capability	Get	Struct of:	-	Indication of the capabilities of the interface
	Capability Bits		DWORD	-	Interface capabilities, other than speed/duplex See table "Interface Capability" below.
	Speed/Duplex Options		Struct of:	-	Indicates speed/duplex pairs supported in the Interface Control Attribute
			USINT	-	Speed/duplex array count
			Array of Struct of:	-	Speed/duplex array
			UINT	-	Interface speed
			USINT	-	Interface Duplex Mode 0 = half duplex 1 = full duplex 2 - 255 = Reserved

- Support for attribute #6 can be disabled by implementing attribute #9 in the EtherNet/IP Host Object (F8h). see *EtherNet/IP Host Object (F8h), p. 222*
- Support for attribute #9 can be disabled by implementing the port state attributes (#12 or #13) in the Ethernet Host object (F9h) see *Ethernet Host Object (F9h), p. 231*

Interface Flags

Bit	Name	Description
0	Link status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. Value: Meaning: 0 Inactive link 1 Active link
1	Half/full duplex	Indicates the duplex mode currently in use. Value: Meaning: 0 Half duplex 1 Full duplex
2 - 4	Negotiation Status	Indicates the status of link auto-negotiation. Value: Meaning: 0 Auto-negotiation in progress. 1 Auto-negotiation and speed detection failed (using default values) (Recommended default values are 10 Mbps, half duplex) 2 Auto negotiation failed but detected speed (using default duplex value) 3 Successfully negotiated speed and duplex. 4 Auto-negotiation not attempted. Forced speed and duplex.
5	Manual Setting requires Reset	Value: Meaning: 0 Interface can activate changes to link parameters during runtime 1 Reset is required in order for changes to have effect
6	Local Hardware Fault	Value: Meaning: 0 No local hardware fault detected 1 Local hardware fault detected
7-31	(reserved)	Set to 0.

Interface State

This attribute indicates the current operational state of the interface.

Value	Description
0	Unknown interface state.
1	The interface is enabled and is ready to send and receive data.
2	The interface is disabled.
3	The interface is testing.

Admin State

This attribute controls the administrative setting of the interface state.

Value	Description
0	(reserved)
1	Enable the interface.
2	Disable the interface.
3-255	(reserved)

Interface Label

This attribute is configurable via the EtherNet/IP Host Object, see page *222*

Instance	Value
1	Port 1
2	Port 2
3	Internal

Interface Type

Instance	Value	Description
1	2	Twisted-pair
2	2	Twisted-pair
3	1	Internal interface

Interface Capability

Bit	Name	Description	Implementation
0	Manual setting requires reset	Indicates whether or not the device requires a reset to apply changes made to the Interface Control attribute (#6). 0 Indicates that the device automatically applies changes made to the Interface Control attribute (#6) and, therefore, does not require a reset in order for changes to take effect. This bit shall have this value when the Interface Control attribute (#6) is not implemented. 1 1 = Indicates that the device does not automatically apply changes made to the Interface Control attribute (#6) and, therefore, will require a reset in order for changes to take effect. Note: this bit shall also be replicated in the Interface Flags attribute (#2), in order to retain backwards compatibility with previous object revisions.	Return 0
1	Auto-negotiate	0 Indicates that the interface does not support link auto-negotiation 1 Indicates that the interface supports link auto-negotiation	0 for internal interface, 1 for external interfaces
2	Auto-MDIX	0 Indicates that the interface does not support auto MDIX operation 1 Indicates that the interface supports auto MDIX operation	0 for internal interface, 1 for external interfaces
3	Manual speed/ duplex	0 Indicates that the interface does not support manual setting of speed/duplex. The Interface Control attribute (#6) shall not be supported. 1 Indicates that the interface supports manual setting of speed/duplex via the Interface Control attribute (#6)	0 for internal interface, 1 for external interfaces
4 - 31	Reserved	Shall be set to 0	Return 0

13 Anybus Module Objects

13.1 General Information

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

Standard Objects:

- *Anybus Object (01h), p. 147*
- *Diagnostic Object (02h), p. 149*
- *Network Object (03h), p. 150*
- *Network Configuration Object (04h), p. 151*

Network Specific Objects:

- *Socket Interface Object (07h), p. 167*
- *SMTP Client Object (09h), p. 184*
- *Anybus File System Interface Object (0Ah), p. 189*
- *Network Ethernet Object (0Ch), p. 190*
- *CIP Port Configuration Object (0Dh), p. 192*
- *Functional Safety Module Object (11h), p. 194*
- *Time Object (13h), p. 201*

13.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

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

Extended

#	Name	Access	Type	Value
17	Virtual attributes	Get/Set	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
18	Black list/White list	Get/Set		
19	Network time	Get	UINT64	0 (Not supported)

Command Details: Reset

Details

Command Code	05h
Valid for:	Object Instance

Description

This command is sent from the host application to the Anybus object (instance 0). A power-on reset is a request to make the module ready for a power-on reset from the host application. A power-on + factory default reset is a request a to return the module to the application specific out-of-box state and then make the module ready for a power-on reset from the host application.

A power-on reset shuts down the network and then sets the module in the EXCEPTION state waiting for the host to perform the power-on reset. Note that this command does not clear or reset any functionality stored in non-volatile memory. No command data shall be supplied together with this reset type of reset command.

A power-on + factory default request shuts down the network, resets the functionality specified by the bit field in the command data, reports the result in the response data and then sets the module in EXCEPTION state waiting for the host application to perform the power-on reset.

If a power-on + factory default reset is successful the response bit field indicates what was reset successfully.

- Command details:

Field	Contents	
CMDExt[0]	(reserved)	
CMDExt[1]	00h:	Power-on reset (actual power-on or simulated)
	01h:	(reserved)
	02h:	Power-on + Factory default reset
Data[0-3]	Bitmask specifying what to reset to factory default state (UINT32)	
	Bit(s):	Description:
	0:	Network configuration parameters
	1:	Anybus file system
	2:	All existing configuration like user created accounts and stored certificates
	3-23	(reserved)
	24-31	Reserved or used for network specific functionality, see the respective network guides

- Response details:

Field	Contents
Data[0-3]	Bitmask specifying what the Anybus CompactCom was supported to reset. See command data for bit specification.

13.3 Diagnostic Object (02h)

Category

Basic

Object Description

This object provides a standardized way of handling host application events & diagnostics, and is thoroughly described in the general *Anybus CompactCom 40 Software Design Guide*.

Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1 (Of the maximum number of instances there should always be one instance reserved for an event of severity level ‘Major, unrecoverable’, to force the module into the ‘EXCEPTION’-state.)
12	Supported functionality	Get	BITS32	Bit 0: “0” (Latching events are not supported) Bit 1 - 31: reserved (shall be “0”)

Instance Attributes (Instance #1)

Extended

#	Name	Access	Data Type	Value
1	Severity	Get	UINT8	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
2	Event Code	Get	UINT8	
3	-	-	-	Not implemented in product
4	Slot	Get	UINT16	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
5	ADI	Get	UINT16	
6	Element	Get	UINT8	
7	Bit	Get	UINT8	

Attributes #2 and #4–7 can not be represented on the network and are ignored by the module.

In this implementation, the severity level of all instances are combined (using logical OR) and represented on the network through the CIP Identity Object.

13.4 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String Map_ADI_Write_Area Map_ADI_Read_Area Map_ADI_Write_Ext_Area Map_ADI_Read_Ext_Area

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	00ABh (EtherNet/IP Beacon Based 2–port + IIoT)
2	Network type string	Get	Array of CHAR	“Ethernet/IP(TM)”
3	Data format	Get	ENUM	00h (LSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area. (Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area. (Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state. Value: Meaning: 00h No information available 01h Invalid assembly instance mapping 02h Missing MAC address (Only valid for Anybus IP)

13.5 Network Configuration Object (04h)

Category

Extended

Object Description

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

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

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

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

See also...

- *Communication Settings, p. 26*
- *TCP/IP Interface Object (F5h), p. 138* (CIP-object)
- *Ethernet Link Object (F6h), p. 141*
- *Ethernet Host Object (F9h), p. 231*

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
3	Number of instances	Get	UINT16	25	Supported number of instances
4	Highest instance number	Get	UINT16	56	Highest instance number

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

Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“IP address” (Multilingual, see page *166*)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Subnet mask” (Multilingual, see page *166*)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #5, Gateway Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Gateway” (Multilingual, see page *166*)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #6, DHCP Enable)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DHCP" (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. (Multilingual, see page *166*) Value — String — Meaning 00h — "Disable" — DHCP disabled 01h — "Enable" — DHCP enabled (default)
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset. Value — String — Meaning 00h — "Disable" — DHCP disabled 01h — "Enable" — DHCP enabled

Instance Attributes (Instance #7 Ethernet Communication Settings 1)

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Comm 1" (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	Value — String — Meaning (Multilingual, see page *166*) 00h — "Auto" — Auto negotiation (default) 01h — "10 HDX" — 10Mbit, half duplex 02h — "10 FX" — 10Mbit, full duplex 03h — "100HDX" — 100Mbit, half duplex 04h — "100FX" — 100Mbit, full duplex
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset. Value — String — Meaning (Multilingual, see page *166*) 00h — "Auto" — Auto negotiation 01h — "10 HDX" — 10Mbit, half duplex 02h — "10 FX" — 10Mbit, full duplex 03h — "100HDX" — 100Mbit, half duplex 04h — "100FX" — 100Mbit, full duplex

Instance Attributes (Instance #8 Ethernet Communication Settings 2)

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Comm 2” (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	Value String Meaning (Multilingual, see page *166*) 00h “Auto” Auto negotiation (default) 01h “10 HDX” 10Mbit, half duplex 02h “10 FX” 10Mbit, full duplex 03h “100HDX” 100Mbit, half duplex 04h “100FX” 100Mbit, full duplex
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset. Value String Meaning (Multilingual, see page *166*) 00h “Auto” Auto negotiation 01h “10 HDX” 10Mbit, half duplex 02h “10 FX” 10Mbit, full duplex 03h “100HDX” 100Mbit, half duplex 04h “100FX” 100Mbit, full duplex

Instance Attributes (Instance #9, DNS1)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“DNS1” (Multilingual, see page *166*)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #10, DNS2)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS2" (Multilingual, see page *166*)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #11, Host name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Host name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 64 characters

Instance Attributes (Instance #12, Domain name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h (48 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Domain name, 48 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Domain name, 48 characters

Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server URL.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP server" (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP server URL, 64 characters in the format [<protocol>://]<server address>[:<port>]. See *SMTP Server URL format, p. 156* for URL format options. If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset.
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP server URL, 64 characters.

Changes are valid after reset.

SMTP Server URL format

The URL of the SMTP server has the following components:

Protocol	Optional. If not specified, non-encrypted SMTP will be used. Specifies the protocol to use when communicating with the server. The available protocols are: • smtp — Standard, non-encrypted SMTP. • smtps — Secure SMTP. A TLS connection will be used when communicating with the server.
Server address	Mandatory. Sets the IP address or host name of the SMTP server. If host name is used, a valid DNS server must also be configured.
Port	Optional. If not specified, the SMTP default port 25 will be used. Specifies the port to connect to for the SMTP server.

These examples are in valid SMTP server URL format:

smtp.server.com	Connects to smtp.server.com using SMTP on port 25.
smtp://smtp.server.com:1025	Connects to smtp.server.com using SMTP on port 1025.
smtps://smtp.server.com:465	Connects to smtp.server.com using SMTPS (SMTP over TLS) on port 465.

Instance Attributes (Instance #14, SMTP User)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP user" (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account username, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account username, 64 characters

Instance Attributes (Instance #15, SMTP Password)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“SMTP Pswd” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account password, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account password, 64 characters

Instance Attributes (Instance #16, MDI 1 Settings)

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

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

Instance Attributes (Instance #17, MDI 2 Settings)

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

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

Instance Attributes (Instances #18 and #19)

These instances are reserved for future attributes.

Instance Attributes (Instance #20, QuickConnect)

This instance enables or disables the QuickConnect functionality from the application. Changes are valid after reset or power cycle. The value of the QuickConnect attribute (#12) in the TCP/IP Interface object (F5h), will change immediately.

This instance has no effect unless QuickConnect is enabled in the EtherNet/IP host object. If QuickConnect is disabled in the EtherNet/IP host object the application is advised to hide this instance to the end-user.

See also...

- *TCP/IP Interface Object (F5h), p. 138*
- *EtherNet/IP Host Object (F8h), p. 222*

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“QuickConnect”
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	bit 0: 1 = Read access bit 1: 1 = Write access bit 2: 1 = Shared access
5	Value	Get/Set	ENUM	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Value: Meaning: 00h Disable (default) 01h Enable
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset. Value: Meaning: 00h Disable 01h Enable

Instance Attributes (Instance #40, OPC UA TCP Port)

This instance holds the TCP port address for OPC UA communication.

If this value is changed by the host application during runtime, a reset is required in order for changes to have effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“OPC Port” (Multilingual, see page *166*)
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	bit 0: 1 = Read access bit 1: 1 = Write access bit 2: 1 = Shared access
5	Value	Get/Set	UINT16	Actual OPC UA TCP port Range: 1 - 65535 (Default: 4840) If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset.
6	Configured value	Get	UINT16	The configured value that will be used after restart

Instance Attributes (Instance #41, OPC UA Discovery Server)

This instance holds the URL of the OPC UA Discovery server used by the Anybus CompactCom.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“OPC DS URL” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	80 (Max Discovery server URL length is 80 characters.)
4	Descriptor	Get	UINT8	bit 0: 1 = Read access bit 1: 1 = Write access bit 2: 1 = Shared access
5	Value	Get/Set	CHAR[80]	Actual OPC UA Discovery Server Format: “opc.tcp://<hostname/ip-address>[:<port>]”. The port is optional to have in the URL, if left out the default value is used: 4840. Value set to this attribute will be used on the next connection attempt towards the discovery server. If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a restart.
6	Configured value	Get	CHAR[80]	The configured value that will be used after restart

Instance Attributes (Instance #42, OPC UA SecurityPolicyNone)

This instance provides the possibility to enable an endpoint with SecurityPolicyNone as a first step. In a second step, it is also possible to enable the UserIdentityTokenPolicy Username_None on all endpoints. In the first step all data will be transmitted in clear text, except protected login credentials. This mode is suitable when debugging the network. In the second step it is possible to connect to the OPC UA server, completely without certificates. However, the login credentials are transmitted in clear text in this mode and it should only be used during development or fault investigation of a device, on a network that cannot be monitored by external parties.

For more information, see *Endpoints, p. 98*.

> ! If the application is to pass the conformance test for OPC UA, this setting must be protected by administrator rights and must be disabled by default. Value 2 (Endpoint + UserIdentityToken) is not conformant.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“OPC Unsecure” (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h
3	Number of elements	Get	UINT8	One data element
4	Descriptor	Get	UINT8	bit 0: 1 = Read access bit 1: 1 = Write access bit 2: 1 = Shared access
5	Value	Get/Set	ENUM	Actual OPC UA “Enable unsecure endpoint” setting 0 = “Disable” (Unsecure endpoint disabled) 1 = “Endpoint” (Unsecure endpoint enabled) 2 = “EndP+UserId” (Unsecure endpoint with Username_None UserTokePolicy enabled) (Multilingual, see page *166*) Value set to this attribute will be used after next restart. If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a restart.
6	Configured value	Get	ENUM	Default value: 0 Configured OPC UA “Enable unsecure endpoint” setting

Instance Attributes (Instance #50, MQTT Broker URL)

This instance holds the MQTT Broker URL.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT Broker” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	255 (Max broker URL length is 255 characters)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	CHAR[64]	Actual MQTT broker URL Format: “<hostname/ip-address>[:<port>]”. The port is optional to have in the URL, if left out the default value is used: 1883 (TLS disabled) or 8883 (TLS enabled). Value set to this attribute will be used in the next connection attempt.
6	Configured value	Get	CHAR[64]	Configured MQTT broker URL

Instance Attributes (Instance #51, MQTT Client Identifier)

This instance holds the MQTT Client Identifier.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT ClientID” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	64 (Max Client Identifier length is 64 characters)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	CHAR[23]	Actual Client Identifier The ID must be unique at least within each network, and must only contain numbers and/or letters [0-9, a-z, A-Z]. Value set to this attribute will be used in the next connection attempt.
6	Configured value	Get	CHAR[23]	Configured MQTT client identifier.

Instance Attributes (Instance #52, MQTT Keep Alive)

This instance holds the MQTT Keep Alive value. This value defines the max allowed time between two messages, for the broker to keep the connection to a client alive.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT KA Time” (Multilingual, see page *166*)
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	1
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Actual keep alive value (s). Default: 60. Value set to this attribute will be used in the next connection attempt.
6	Configured value	Get	UINT16	Configured keep alive value.

Instance Attributes (Instance #53, MQTT Username)

This instance holds the MQTT username used when connecting to the broker.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT Username” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	64 (Max username length is 64 characters)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Actual MQTT username Value set to this attribute will be used in the next connection attempt. If the username is of zero length no username will be included in the connection packet of MQTT.
6	Configured value	Get	Array of CHAR	Configured MQTT username.

Instance Attributes (Instance #54, MQTT Password)

This instance holds the MQTT password.

If the MQTT username is not set or of zero length, the MQTT password will not be used.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT Password” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	64 (Max password length is 64 characters)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Actual MQTT password Value set to this attribute will be used in the next connection attempt. If the MQTT username is of zero length the MQTT password will not be included in the connection packet of MQTT.
6	Configured value	Get	Array of CHAR	Configured MQTT password.

Instance Attributes (Instance #55, MQTT Base Topic)

This instance configures the base topic level of datasets, that the host application have not specified a custom topic for.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT Topic” (Multilingual, see page *166*)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	255 (Max base topic length is 255 characters)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Actual MQTT base topic level The value set to this attribute will be used in the next PUBLISH packet.
6	Configured value	Get	Array of CHAR	Configured MQTT base topic level This attribute always has the same value as attribute #5.

Instance Attributes (Instance #56, MQTT QoS)

This instance configures the MQTT QoS level.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT QoS” (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	1
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	Actual MQTT QoS level 0: “QoS 0” (default) 1: “QoS 1” 2: “QoS 2” The value set to this attribute will be used in the next PUBLISH packet.
6	Configured value	Get	ENUM	Configured MQTT QoS level This attribute always has the same value as attribute #5.

Instance Attributes (Instance #57, MQTT TLS)

This instance enables or disables TLS for the MQTT protocol.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MQTT TLS” (Multilingual, see page *166*)
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	1
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	Actual setting for TLS 0: “Disable” (MQTT over TCP) (default) 1: “Enable” (MQTT over TLS) Value set to this attribute will be used after next restart.
6	Configured value	Get	ENUM	Configured setting for TLS Default Value: 0

Multilingual Strings

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

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
7	Comm 1	Komm 1	Comu 1	Connessione 1	Comm 1
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
8	Comm 2	Komm 2	Comu 2	Connessione 2	Comm 2
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
9	DNS1	DNS 1	DNS Primaria	DNS1	DNS1
10	DNS2	DNS 2	DNS Secundia.	DNS2	DNS2
11	Host name	Host name	Nombre Host	Nome Host	Nom hôte
12	Domain name	Domain name	Nobre Domain	Nome Dominio	Dom Domaine
13	SMTP Server	SMTP Server	Servidor SMTP	Server SMTP	SMTP serveur
14	SMTP User	SMTP User	Usuario SMTP	Utente SMTP	SMTP utilisa.
15	SMTP Pswd	SMTP PSWD	Clave SMTP	Password SMTP	SMTP mt passe
40	OPC Port	OPC Port	OPC Puerto	OPC Porta	OPC Port
41	OPC DS URL	OPC DS URL	OPC DS URL	OPC DS URL	OPC DS URL
42	OPC Unsecure	OPC Unsicher	OPC NoSegura	OPC NonSicur	OPC NonSécur
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
	Endpoint	Endpoint	Endpoint	Endpoint	Endpoint
	EndP+UserId	EndP+UserId	EndP+UserId	EndP+UserId	EndP+UserId
50	MQTT Broker	MQTT Broker	Broker MQTT	Broker MQTT	Serveur MQTT
51	MQTT ClientID	MQTT ClientID	ClientID MQTT	ClientID MQTT	Clientid MQTT
52	MQTT KA Time	MQTT KA Time	Tpo. KA MQTT	Tempo KA MQTT	Tps KA MQTT
53	MQTT Username	MQTT Benutzer	Usuario MQTT	Utente MQTT	Utilisa. MQTT
54	MQTT Password	MQTT Password	Clave MQTT	Password MQTT	Mt passe MQTT
55	MQTT Topic	MQTT Topic	Mt passe MQTT	Topic MQTT	Sujet MQTT
56	MQTT QoS	MQTT QoS	QoS MQTT	QoS MQTT	QDS MQTT
57	MQTT TLS	MQTT TLS	TLS MQTT	TLS MQTT	TLS MQTT
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé

13.6 Socket Interface Object (07h)

Category

Extended

Object Description

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

Note that some of the commands used when accessing this object may require segmentation. A message will be segmented if the amount of data sent or received is larger than the message channel can handle. For more information, see *Message Segmentation, p. 182*.

ⓘ *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 below) Delete (See below) DNS_Lookup (See below)
Instance:	Get_Attribute Set_Attribute Bind (See below) Shutdown (See below) Listen (See below) Accept (See below) Connect (See below) Receive (See below) Receive_From (See below) Send (See below) Send_To (See below) P_Add_membership (See below) IP_Drop_membership (See below)

Object Attributes (Instance #0)

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

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

Extended

#	Name	Access	Data Type	Description
1	Socket Type	Get	UINT8	Value: Socket Type 00h SOCK_STREAM, NONBLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NONBLOCKING (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 and received 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 Ah 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). If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75s. The connection is terminated after 8 failed probe attempts. 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) Must belong to group in order to get the loop backed message Value Meaning 1 Enabled (default) 0 Disabled
12	(reserved)			
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)

Command Details: Create

Category

Extended

Details

Command Code	03h
Valid for:	Object Instance

Description

This command creates a socket.

This command is only allowed in WAIT_PROCESS, IDLE and PROCESS_ACTIVE states.

- Command Details

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

Extended

Details

Command Code	04h
Valid for:	Object Instance

Description

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

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

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

- Response Details

  (no data)

Command Details: Bind

Category

Extended

Details

Command Code	10h
Valid for:	Instance

Description

This command binds a socket to a local port.

- Command Details

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

- Response Details

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

Command Details: Shutdown

Category

Extended

Details

Command Code	11h
Valid for:	Instance

Description

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

- Command Details

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

- Response Details

  (no data)

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

Application initiates shutdown:

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

Host initiates shutdown:

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

Command Details: Listen

Category

Extended

Details

Command Code	12h
Valid for:	Instance

Description

This command puts a TCP socket in listening state.

- Command Details

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

- Response Details

  (no data)

Command Details: Accept

Category

Extended

Details

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.

NONBLOCKING 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.

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

- Command Details

  (no data)

- Response Details

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

Command Details: Connect

Category

Extended

Details

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

- Response Details

  (no data)

Command Details: Receive

Category

Extended

Details

Command Code	15h
Valid for:	Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (for more information, see *Message Segmentation, p. 182*).

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	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 182*
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- Response Details

  The data in the response may be segmented (For more information, see *Message Segmentation, p. 182*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 182*
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Extended

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 (For more information, see *Message Segmentation, p. 182*).

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	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 182*
Data[0]	Receive data size (low byte)	Only used in the first segment
Data[1]	Receive data size (high byte)	

- Response Details

The data in the response may be segmented (For more information, see *Message Segmentation, p. 182*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 182*
Data[0]	Host IP address byte 4	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Received data	

Command Details: Send

Category

Extended

Details

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 (For more information, see *Message Segmentation, p. 182*).

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

- Command Details

  To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see *Message Segmentation, p. 182*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see *Message Segmentation, p. 182*)
Data[0...n]	Data to send	-

- Response Details

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

Command Details: Send_To

Category

Extended

Details

Command Code	18h
Valid for:	Instance

Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (For more information, see appendix For more information, see *Message Segmentation, p. 182*).

- Command Details

  To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see *Message Segmentation, p. 182*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see *Message Segmentation, p. 182*
Data[0]	Host IP address byte 4	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Data to send	

- Response Details

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

Command Details: IP_Add_Membership

Category

Extended

Details

Command Code	19h
Valid for:	Instance

Description

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

- Command Details

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

- Response Details

  (no data)

Command Details: IP_Drop_Membership

Category

Extended

Details

Command Code	1Ah
Valid for:	Instance

Description

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

- Command Details

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

- Response Details

  (no data)

Command Details: DNS_Lookup

Category

Extended

Details

Command Code	1Bh
Valid for:	Object

Description

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

- Command Details

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

- Response Details (Success)

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

Socket Interface Error Codes (Object Specific)

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

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

Message Segmentation

General

Category: Extended

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

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

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

The module supports 1 (one) segmented message per instance

Command Segmentation

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

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

- Send
- Send To

When issuing a segmented command, the following rules apply:

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

Segmentation Control Bits (Command)

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

Segmentation Control Bits (Response)

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

Response Segmentation

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

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

- Receive
- Receive From

When receiving a segmented response, the following rules apply:

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

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

13.7 SMTP Client Object (09h)

Category

Extended

Object Description

This object groups functions related to the SMTP client.

Supported Commands

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

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Instances are created dynamically by the application.

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

Command Details: Create

Category

Extended

Details

Command Code	03h
Valid for:	Object

Description

This command creates an e-mail instance.

- Command Details

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

- Response Details

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

Command Details: Delete

Category

Extended

Details

Command Code	04h
Valid for:	Object

Description

This command deletes an e-mail instance.

- Command Details

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

- Response Details

  (no data)

Command Details: Send E-mail From File

Category

Extended

Details

Command Code	11h
Valid for:	Object

Description

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

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- Command Details

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

- Response Details

  (no data)

Command Details: Send E-mail

Category

Extended

Details

Command Code	10h
Valid for:	Instance

Description

This command sends the specified e-mail instance.

- Command Details

  (no data)

- Response Details

  (no data)

Object Specific Error Codes

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

13.8 Anybus File System Interface Object (0Ah)

Category

Extended

Object Description

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

This provides the host application with access to the built-in file system of the module, e.g. when application specific web pages are to be installed.

Instances are created and deleted dynamically during runtime.

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

13.9 Network Ethernet Object (0Ch)

Category

Extended

Object Description

This object provides Ethernet-specific information to the application.

The object has three instances, each corresponding to a port:

Instance #	Port
1	Internal port
2	Port 1
3	Port 2

Each instance provides statistic counters for the port. This information can e.g be presented on internal web pages, if present, using the JSON script language.

Instance attribute #1 is reserved and used for backwards compatibility with earlier applications.

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	3
4	Highest instance no.	Get	UINT16	3

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Reserved, used for backwards compatibility. (Device MAC address.) (See also *Ethernet Host Object (F9h), p. 231*)
2	(Reserved)			
3	(Reserved)			
4	MAC Address	Get	Array of UINT8	Device MAC address
5	Interface Counters	Get	Array of UINT32	Array containing MIB-II interface counters (rfc1213) See table below for array indices.
6	(Reserved)			

Instance Attributes (Instances #2 - #3)

#	Name	Access	Data Type	Description
1 - 4	(Reserved)			
5	Interface Counters	Get	Array of UINT32	Array containing MIB-II interface counters (rfc1213) See table below for array indices.
6	Media Counters	Get	Array of UINT32	Array containing Ethernet-Like MIB counters for the port. See table below for array indices.

Interface Counters

Array indices of Interface Counters attribute (#5)

Index	Name	Description
0	In octets	Octets received on the interface
1	In Unicast Packets	Unicast packets received on the interface
2	In Non-Unicast Packets	Non-unicast packets (multicast/broadcast) packets received on the interface
3	In Discards	Inbound packets received on the interface but discarded
4	In Errors	Inbound packets that contain errors (does not include In Discards)
5	In Unknown Protos	Inbound packets with unknown protocol
6	Out Octets	Octets transmitted on the interface
7	Out Unicast packets	Unicast packets transmitted on the interface
8	Out Non-Unicast Packets	Non-unicast (multicast/broadcast) packets transmitted on the interface
9	Out Discards	Outbound packets discarded
10	Out Errors	Outbound packets that contain errors

Media Counters

Array indices of Media Counters attribute (#6)

Index	Name	Description
0	AlignmentErrors;	Frames received that are not an integral number of octets in length
1	FCSErrors;	Frames received that do not pass the FCS check
2	SingleCollisions;	Successfully transmitted frames which experienced exactly one collision
3	MultipleCollisions;	Successfully transmitted frames which experienced more than one collision
4	SQETestErrors;	Number of times SQE test error is generated
5	DeferredTransmissions;	Frames for which first transmission attempt is delayed because the medium is busy
6	LateCollisions;	Number of times collision is detected later than 512 bit-times into the transmission of a packet
7	ExcessiveCollisions;	Frames for which transmission fails due to excessive collisions
8	lMACTransmitErrors;	Frames for which transmission fails due to an internal MAC sublayer transmit error
9	lCarrieSenseErrors;	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
10	lFrameTooLong;	Frames received that exceed the maximum permitted frame size
11	lMACRecieveErrors;	Frames for which reception on an interface fails due to an internal MAC sublayer receive error

13.10 CIP Port Configuration Object (0Dh)

Category

Extended

Object Description

This object is used to populate and enumerate the CIP Port Object (see *Port Object (F4h), p. 136*) on the network side. Basically, this is a matter of creating and updating instances and attributes which shall represent a CIP Port within the host application. This process is necessary in case support for Unconnected CIP Routing has been enabled (see *EtherNet/IP Host Object (F8h), p. 222*, Instance Attribute #17).

Each instance within this object corresponds to an instance in the CIP Port Object. The object supports up to 8 instances, where instance #1 is dedicated to the local TCP port, enabling the host application to implement up to 7 additional ports. Instance #1 will automatically be populated with default values, however it is possible for the host application to customize instance attributes #2 and #4.

Apart from attribute #7, it is possible to write to the instance attributes only during setup. The host application is responsible for keeping instance attribute #7 updated for all ports located within the host application.

> **!** Note that the module does not take over the host application responsibility for error control; the module will not verify that the data set by the host application is correct.

Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“CIP Port Configuration”
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 (Instance #1)

#	Name	Access	Data Type	Description
1	Port Type	Set	UINT16	Enumerates the port (See CIP specification, available from www.odva.org)
2	Port Number	Set	UINT16	CIP port number associated with this port
3	Link Path	Set	Array of UINT8	Logical path segments which identify the object for this port.
4	Port Name	Set	Array of CHAR	String (max. no. of characters is 64) which names the port.
5	-	-	-	(reserved)
6	-	-	-	(reserved)
7	Node Address	Set	Array of UINT8	Node number of this device on port. The data type restricts the range to a Port Segment. The encoded port number must match the value specified in attribute #2. A device which does not have a node number on the port can specify a zero length node address within the Port Segment (i.e. 10h 00h). In case the node address changes during runtime, the host application is responsible for updating this attribute as well.
8	Port Node Range	Set	Struct of: UINT16 (Min) UINT16 (Max)	Minimum and maximum node number on port. Support for this attribute is conditional; the attribute shall be supported provided that the node number can be reported within the range of the data type (e.g. DeviceNet). If not (as is the case with networks such as EtherNet/IP which uses a 4 byte IP address), the attribute shall not be supported.

13.11 Functional Safety Module Object (11h)

Category

Extended

Object Description

This object contains information provided by the Safety Module connected to the Anybus CompactCom module. Please consult the manual for the Safety Module used, for values of the attributes below.

Supported Commands

Object:	Get_Attribute Error_Confirmation Set_IO_Config_String Get_Safety_Output_PDU Get_Safety_Input_PDU
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Functional Safety Module”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	State	Get	UINT8	Current state of the Safety Module Please consult the manual for the Safety Module used.
2	Vendor ID	Get	UINT16	Identifies vendor of the Safety Module. E.g. 0001h (HMS Industrial Networks) Please consult the manual for the Safety Module used.
3	IO Channel ID	Get	UINT16	Describes the IO Channels that the Safety Module is equipped with. Please consult the manual for the Safety Module used.
4	Firmware version	Get	Struct of UINT8 (Major) UINT8 (Minor) UINT8 (Build)	Safety Module firmware version. Format: version “2.18.3” would be represented as: first byte = 0x02, second byte = 0x12, third byte = 0x03.
5	Serial number	Get	UINT32	32 bit number, assigned to the Safety Module at production. Please consult the manual for the Safety Module used.
6	Output data	Get	Array of UINT8	Current value of the Safety Module output data, i.e. data FROM the network Note: This data is unsafe, since it is provided by the Anybus CompactCom module.
7	Input data	Get	Array of UINT8	Current value of the Safety Module input data, i.e. data sent TO the network. **Note**: This data is unsafe, since it is provided by the Anybus CompactCom module.
8	Error counters	Get	Struct of UINT16 (ABCC DR) UINT16 (ABCC SE) UINT16 (SM DR) UINT16 (SM SE)	Error counters (each counter stops counting at FFFFh) ABCC DR: Responses (unexpected) from the Safety Module, discarded by the Anybus CompactCom module. ABCC SE: Serial reception errors detected by the Anybus CompactCom module. SM DR: Responses (unexpected) from the Anybus CompactCom module, discarded by the Safety Module. SM SE: Serial reception errors detected by the Safety Module.
9	Event log	Get	Array of UINT8	Latest Safety Module event information (if any) is logged to this attribute. Any older event information is erased when a new event is logged. For evaluation by HMS support.
10	Exception information	Get	UINT8	If the Exception Code in the Anybus object is set to “Safety communication error” (09h), additional exception information is presented here, see table below.
11	Bootloader version	Get	Struct of UINT8 Major UINT8 Minor	Safety Module bootloader version. Format: version “2.12” would be represented as: first byte = 0x02, second byte = 0x0C
12	Vendor block safe uc1	Get	Array of UINT8	The Safety Module may supply additional vendor-specific data to the Anybus CompactCom. If such data is available it is presented in this attribute.
13	Vendor block safe uc2	Get	Array of UINT8	The Safety Module may supply additional vendor-specific data to the Anybus CompactCom. If such data is available it is presented in this attribute.

Exception Information

If Exception Code 09h is set in the Anybus object, there is an error regarding the functional safety module in the application. Exception information is presented in instance attribute #10 according to this table:

Value	Exception Information
00h	No information
01h	Baud rate not supported
02h	No start message
03h	Unexpected message length
04h	Unexpected command in response
05h	Unexpected error code
06h	Safety application not found
07h	Invalid safety application CRC
08h	No flash access
09h	Answer from wrong safety processor during boot loader communication
0Ah	Boot loader timeout
0Bh	Network specific parameter error
0Ch	Invalid IO configuration string
0Dh	Response differed between the safety microprocessors (e.g. different module types)
0Eh	Incompatible module (e.g. supported network)
0Fh	Max number of retransmissions performed (e.g. due to CRC errors)
10h	Firmware file error
11h	The cycle time value in attribute #4 in the Functional Safety Host Object can not be used with the current baud rate
12h	Invalid SPDU input size in start-up telegram
13h	Invalid SPDU output size in start-up telegram
14h	Badly formatted input SPDU
15h	Anybus to safety module initialization failure

Command Details: Error_Confirmation

Category

Extended

Details

Command Code	10h
Valid for:	Object

Description

When the Safety Module has entered the Safe State, for any reason, it must receive an error confirmation before it can leave the Safe State. With this command it is possible to reset all safety channels of the safety which, for any reason, are in the Safe State at the same time. The application issues this command to the Anybus CompactCom module, when an error has been cleared by for example an operator. The Anybus CompactCom forwards the command to the Safety Module.

The channel Safe State can also be confirmed by the safety PLC or by the safety module.

With this command

- Command Details

  (no data)

- Response Details

  (no data)

Command Details: Set_IO_Config_String

Category

Extended

Details

Command Code	11h
Valid for:	Object

Description

This command is sent from the host application when there is a need to change the default configuration of the safety inputs and outputs. This string is used by networks where there are no other means (e.g. PLC or some other tool) to provide the configuration to the safety module. Consult the specification of the safety module for more information. The byte string passed is generated by HMS and need to be passed unmodified using this command.

Information about this string is located in the specification of the safety module to which the string shall be sent.

- Command Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Data (byte string) The data consists of an IO configuration string, where the data format depends on the safety network.

- Response Details

  (no data)

Command Details: Get_Safety_Output_PDU

Category

Extended

Details

Command Code	12h
Valid for:	Object

Description

This command can be issued by the application to get the complete safety output PDU sent by the PLC. The Anybus CompactCom 40 EtherNet/IP IIoT Secure will respond with the complete safety PDU, that the application then has to interpret.

- Command Details

  (no data)

- Response Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Safety PDU from PLC

Command Details: Get_Safety_Input_PDU

Category

Extended

Details

Command Code	13h
Valid for:	Object

Description

This command can be issued by the application to get the complete safety input PDU sent by the safety module. The Anybus CompactCom 40 EtherNet/IP IIoT Secure will respond with the complete safety PDU, that the application then has to interpret.

- Command Details

  (no data)

- Response Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Safety PDU from safety module

Object Specific Error Codes

Error Code	Description	Comments
01h	The safety module rejected a message.	Error code sent by safety module is found in MsgData[2] and MsgData[3].
02h	Message response from the safety module has incorrect format (for example, wrong length).	-

13.12 Time Object (13h)

Category

Extended

Object Description

In some networks there are multiple possible time sources. This object is used to present all known time sources using a common format. The quality of the different time sources may vary, which the host application has to consider when using the time value.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
1	Name	Get	STRING	“Time Object”	Object name
2	Revision	Get	UINT8	1	Revision of object
3	Number of instances	Get	UINT16	N/A	Supported number of instances
4	Highest instance number	Get	UINT16	N/A	Highest implemented instance
11	Protocols	Get	Array of: Struct of: UINT16 Instance ENUM Protocol UINT8 Reserved	N/A	Array of available time protocols. Instance: Corresponding instance number. Protocol: Enumeration of time protocols. See *Time Protocols, p. 202*. Reserved: Should not be used.

Instance Attributes (Instance #n)

Instance 1 is dedicated to CIP Sync. Instance 2 is dedicated to OPC UA Discovery server timestamp.

> **!** To make the instances available they have to be activated:
>
> CIP Sync is activated with attribute 32 in EtherNet/IP Host Object. See *EtherNet/IP Host Object (F8h), p. 222*.
>
> OPC UA is activated with attribute 1 in OPC UA Object. See *OPC UA Object (E3h), p. 207*.

#	Name	Access	Data Type	Description
1	Protocol	Get	ENUM	Enumeration identifier of the time protocol. See *Time Protocols, p. 202* for supported protocols.
2	Current time	Get	UINT64	Current time in protocol specific format. If the time is not valid the value will be set to 0. See *Time Protocols, p. 202* for protocol formats.

Time Protocols

Enum value	Priority	Protocol	Format	Epoch
0	0	CIP Sync (IEEE 1588 PTP)	64 bit nanoseconds	23:59:51. 51.999918, December 31, 1969
8	2	OPC UA Discovery server timestamp	64 bit signed number of 100 nanosecond intervals	January 1, 1601

14 Host Application Objects

14.1 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 EtherNet/IP implementation.

Standard Objects:

- *MQTT Host Object (E2h), p. 204*
- *OPC UA Object (E3h), p. 207*
- *Energy Control Object (F0h), p. 216*
- Assembly Mapping Object (EBh) - (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (ECh) - (see Anybus CompactCom 40 Software Design Guide)
- *Sync Object (EEh), p. 215*
- *Energy Reporting Object (E7h), p. 209*
- Application Data Object (FEh) - (see Anybus CompactCom 40 Software Design Guide)
- Application Object (FFh) - (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- *Functional Safety Object (E8h), p. 210*
- *Application File System Interface Object (EAh), p. 212*
- *CIP Identity Host Object (EDh), p. 213*
- *EtherNet/IP Host Object (F8h), p. 222*
- *Ethernet Host Object (F9h), p. 231*

14.2 MQTT Host Object (E2h)

Category

Extended

Object Description

This object implements MQTT functionality for the host application.

See also ...

- *MQTT, p. 103*

Supported Commands

Object:	Get_Attribute Get_Publish_Configuration
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MQTT mode	Get	UINT8	Defines the MQTT mode 0: Disabled (default) 1: Enabled with JSON data encoding
2	Last will message configuration	Get	Struct	Struct that configures the MQTT last will message. For a detailed description see table below. Default: No last will message.

Attribute #2, Last Will Message Configuration

If a struct member of attribute #2 contains invalid data, e.g. out of range values or invalid string lengths, the last will message configuration is discarded.

Struct Element	Data Type	Description
1	UINT8	Specifies the QoS level of the last will message when it is published Value: Meaning: 0: QoS 0 1: QoS 1 2: QoS 2
2	BOOL	Specifies if the last will message is to be retained when it is published FALSE: Retain bit cleared TRUE: Retain bit set
3	UINT16	Length of the last will message topic string Valid range: 0 - 128
4	Array of CHAR	Topic string for the last will message The length of the array must match the topic length given by struct element 3 Max length: 128 characters
5	UINT16	Length of the last will message data Valid range: 0 - 256
6	Array of OCTET	Message data of the last will message The length of the array must match the message length given by struct element 5 Max lengt: 256 octets.

Command Details: Get_Publish_Configuration

Category

Extended

Details

Command Code:	10h
Valid for:	Object

Description

This command is issued at least once for every dataset, following the Get_Data_Notification response, if the following conditions are fulfilled:

- The MQTT bit is set in the network channels field of the dataset in the Get_Data_Notification response.
- The dataset is supported by MQTT
- MQTT is enabled in instance #1, attribute #1
- The Anybus CompactCom is connected to an MQTT broker on the network

For details on how MQTT is used, see *MQTT, p. 103*.

- Command Details

Field	**Contents**	**Comments**
CmdExt[0]	Dataset type (UINT8)	See the description of the Get_Data_Notificaton command of the Application Object (FFh) in the Software Design Guide.
CmdExt[1]	(reserved)	
MsgData[0...1]	Dataset identifier (UINT16)	

- Response Details

Field	**Contents**		**Comments**
CmdExt[0]	0		Reserved, set to 0
CmdExt[1]	0		Reserved, set to 0
MsgData[0]	Value: TRUE: FALSE:	Meaning: Publish the dataset with the retain bit set Publish the dataset with the retain bit cleared (Default)	If the retain bit is set, the topic will be kept in the broker for additional recipients.
MsgData[1... n]	Array of char, Max length: 128 char		MQTT topic, published by the dataset on the network. Omit this field to not customize the topic.

14.3 OPC UA Object (E3h)

Category

Extended

Object Description

This object implements OPC UA functionality for the host application.

See also ...

- *OPC UA, p. 80*

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	OPC UA Model	Get	UINT8	Defines the model of OPC UA functionality. 0: Disabled (default) 1: CompactCom 40 model 2: Application defined model
2	Application/ Localnamespace URI	Get	Array of CHAR	Application URI, also used as Local server namespace URI of the server. Default value: “urn:<hostname/serialnumber>:anybus: compactcom40”. If the host name is available, it is used as a part of the URI. If no host name is available, use the serial number instead. Max length: 128.
3	Vendor namespace URI	Get	Array of CHAR	Vendor namespace URI. This namespace collects type definitions specific for the product. Default value: “http://hms-networks.com/UA/Anybus/ CompactCom40”. Max length: 128.
4	DeviceType Name	Get	Array of CHAR	The name of the DeviceType. Default value = “CompactCom40DeviceType”. Max length: 64.
5	Device instance name	Get	Array of CHAR	The name of the instance of the DeviceType above that represents the device in the local namespace. Default value = “CompactCom40”. Max length: 64.
6	Product URI	Get	Array of CHAR	URI that identifies the software. Part of the BuildInfo structure in the Server object. Default value = “http://hms-networks.com/UA/Anybus/ CompactCom40/[networktype]/[softwareversion]” [networktype] = Abbreviation of network [softwareversion] = String representation of software version. Max length: 128.
7	Limits	Get	Struct of: UINT16 UINT32	Configuration of server limits. See the table below for structure description. If one of the provided values are out of range none of the values will be used and the discard counter will be incremented. *Warning: The limits are configurable as these options are highly dependent on the response time of the host application. Setting too low/high values may result in a scenario where the Anybus CompactCom is unable to publish all monitored items within the expected interval. Please verify configuration while using other interfaces required by the module such as process data and web interface.*

Server limits structure description

Name	Data type	Default	Min	Max
Max number of monitored items	UINT16	8	8	100
Minimum sampling/publishing interval in milliseconds	UINT32	1000	1000	1000*3600*24

14.4 Energy Reporting Object (E7h)

Category

Extended

Object Description

Using this object, the host application has a standardized way of reporting its energy consumed or produced. The reporting capabilities of this object are limited. On networks providing more elaborate reporting functionality, the reporting functionality will have to be implemented in a transparent manner by 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	“Energy Reporting”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description
1	Energy Reading	Get	Struct of: UINT32 UINT32	Amount of energy (Wh) consumed or produced by the application. Stored in nonvolatile memory. The first UINT32 represents the lower part of the Energy Reading, the second UINT32 represents the higher part of the Energy Reading
2	Direction	Get	BOOL	Indicates if the host is consuming or producing energy. Value: Meaning: 0: Producing 1: Consuming
3	Accuracy	Get	UINT16	Accuracy in 0.01% of reading 0: Unknown
4	Current Power Consumption	Get	UINT16	The current power consumption in 0.01% of the Nominal Power consumption
5	Nominal Current Consumption	Get	UINT32	The nominal power consumption in mW

14.5 Functional Safety Object (E8h)

Category

Extended

Object Description

! Do not implement this object if a safety module is not used.

This object specifies the safety settings of the application. It is mandatory if Functional Safety is to be supported and a Safety Module is connected to the Anybus CompactCom module.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Default Value	Comment
1	Safety enabled	Get	BOOL	-	When TRUE, enables communication with the Safety Module. **Note**: If functional safety is not supported, this attribute must be set to FALSE.
2	Baud Rate	Get	UINT32	1020 kbit/s	This attribute sets the baud rate of the communication in bits/s between the Anybus CompactCom and the Safety Module. Valid values: • 625 kbit/s • 1000 kbit/s • 1020 kbit/s (default) Any other value set to this attribute, will cause the module to enter the EXCEPTION state. The attribute is optional. If not implemented, the default value will be used. **Note**: The host application shall never implement this attribute when using the IXXAT Safe T100.
3	(reserved)				

#	Name	Access	Data Type	Default Value	Comment
4	Cycle Time	Get	UINT8	-	Communication cycle time between the Anybus and the Safety module in milliseconds. **Note**: The host application shall never implement this attribute when using the IXXAT Safe T100. Valid values: • 2 ms • 4 ms • 8 ms • 16 ms If another value is set in this attribute the Anybus will enter Exception state. Optional attribute; If not implemented the minimum cycle time for the chosen baud rate will be used: • 2 ms for 1020 kbit/s • 2 ms for 1000 kbit/s • 4 ms for 625 kbit/s The Anybus CompactCom validates the cycle time according to the minimum values above. If e.g. baud rate is 625 kbit/s and the cycle time is set to 2 ms the Anybus CompactCom will enter the EXCEPTION state.
5	FW upgrade in progress	Set	BOOL	False	Indicates if the Anybus CompactCom is upgrading the connected Safety module firmware. This means that the Anybus CompactCom will stay in the NW_ INIT state longer than normal.

14.6 Application File System Interface Object (EAh)

Category

Extended

Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations. This allows the user to download software through the file transfer protocol server to the application. The application decides the available memory space.

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

14.7 CIP Identity Host Object (EDh)

Category

Extended

Object Description

This object allows for applications to support additional CIP identity instances. It is used to provide additional product identity information, e.g. concerning the software installed.

The first instance in the CIP identity object will not change its behavior. When implementing instances in the CIP identity host object, they will be mapped to the CIP identity object starting at instance 2. Instance no. 1 in the CIP identity host object will be mapped to instance no. 2 in the CIP identity object and so on.

See also ...

- *Identity Object (01h), p. 113* (CIP object)

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Get_Attribute_All

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“CIP Identity”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	Depends on application
4	Highest instance no.	Get	UINT16	Depends on application

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	Vendor ID	Get	UINT16	These values replace the values for the CIP identity object instance #2 and upwards. See also... *Identity Object (01h), p. 113* (CIP-object)
2	Device Type	Get	UINT16	
3	Product Code	Get	UINT16	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	
5	Status	Get	UNIT16	
6	Serial Number	Get	UINT32	
7	Product Name	Get	Array of CHAR	

Command Details: Get_Attribute_All

Category

Extended

Details

Command Code:	10h
Valid for:	Object

Description

This service must be implemented by the application for all instances that exist in the CIP identity host object. If identity data is requested from the network the Anybus module will issue this command to the application. The application will then respond with a message containing a struct of all attributes in the requested instance.

- Command Details

  (no data)

- Response Details

Field	Contents	Comments
MsgData[0, 1]	Vendor ID	ABCC CIP identity data
MsgData[2, 3]	Device type	
MsgData[4, 5]	Product code	
MsgData[6]	Major revision	
MsgData[7]	Minor revision	
MsgData[8,9]	Status	
MsgData[10 .. .13]	Serial number	
MsgData[14 .... n]	Product name	

14.8 Sync Object (EEh)

Category

Extended

Object Description

The Anybus CompactCom 40 EtherNet/IP IIoT Secure does not support CIP Sync. This object is only used to store the cycle time for the last established IO connection that consumes data.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

The attributes are represented on EtherNet/IP as follows:

#	Name	Access	Data Type	Description
1	Cycle time	Get/Set	UINT32	The RPI for the last established IO connection that consumes data (O→T RPI)
2–8	(not implemented)			

14.9 Energy Control Object (F0h)

Category

Extended

Object Description

This object implements energy control functionality, i.e. energy specific settings, in the host application. The implementation of this object is optional. All instance attributes shall be seen as required and must be implemented in the application. If the Anybus module detects that an attribute is missing during run time an appropriate network error is sent and the Discard Responses counter is increased in the Anybus Object instance attribute Error Counter.

Each enabled instance in the object corresponds to an Energy saving mode. The number of available modes is device specific, and must be defined by the application. The higher the instance number, the more energy is saved. The instance with the highest number always corresponds to the “Power off” mode, i.e. the state where the device is essentially shut down. Instance 1 of the object represents “Ready to operate”, i.e. the mode where the device is fully functional and does not save energy at all. Consequently a meaningful implementation always contains at least two instances, one for energy saving and one for operating. If this object is implemented for PROFINET, at least three instances are needed: “Ready to operate”, “Energy saving mode 1”, and “Power off”.

Highest number of instances is 8. Please note that these modes are always present – they are not dynamically created or deleted. It is not allowed to leave holes in the list of instances.

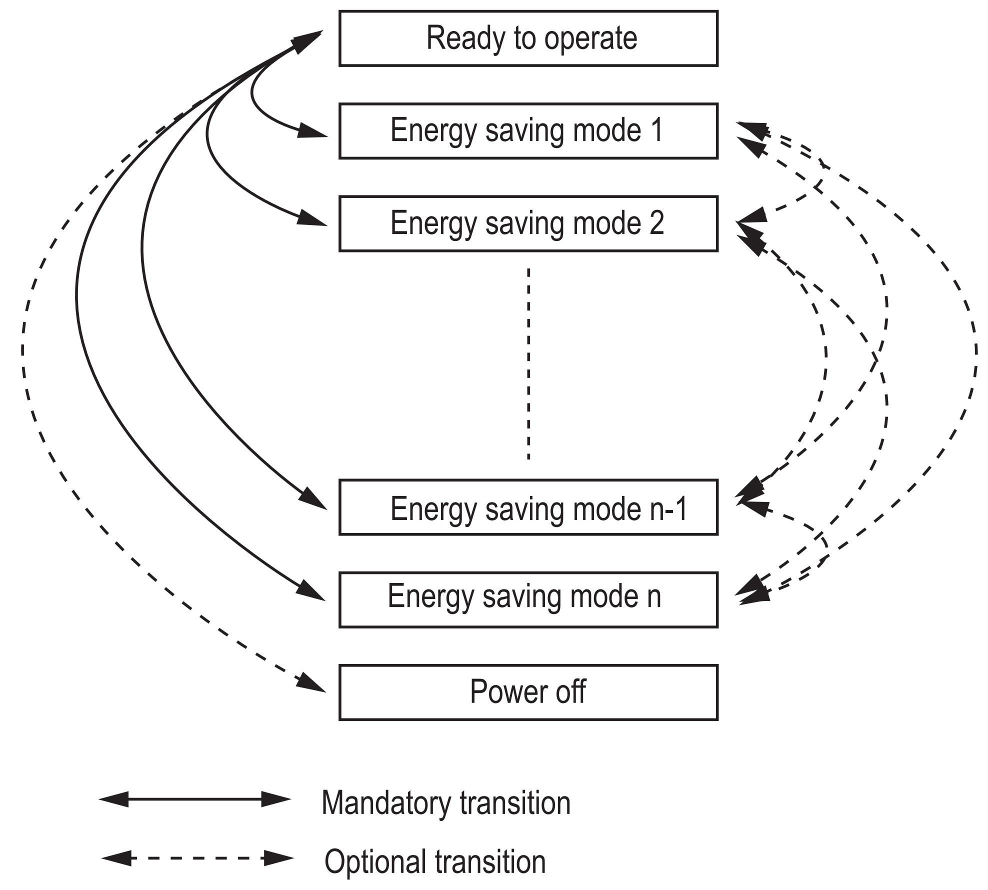


Fig. 27

Supported Commands

Object:	Get_Attribute StartPause EndPause Preview_Pause_Time (not PROFINET)
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Energy Control”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	Highest created instance number. Maximum value is 8.
11	Current Energy Saving Mode	Get	UINT16	Instance number of the currently used Energy saving mode. During a mode transition the new Energy saving mode shall be presented. “Ready to operate” will equal instance #1, and “Power off” mode will equal Highest instance number.
12	Remaining time to destination	Get	UINT32	When changing mode this parameter will reflect the actual time (in milliseconds) remaining until the shift is completed. If a dynamic value cannot be generated the static value for the transition from the source to destination mode shall be used. If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.
13	Energy consumption to destination	Get	FLOAT	When changing mode this parameter will reflect the actual energy (in kWh) which will be consumed until the shift is completed. If a dynamic value cannot be generated the static value for the transition from the source to destination mode shall be used. If the value is undefined the value 0.0 shall be used.
14	Transition to “Power off” mode supported	Get	BOOL	Indicates whether transition to “Power off” mode is supported or not. 0: Not supported 1: Supported

Instance Attributes (Instance #1 - #8)

#	Name	Access	Data Type	Description	
1	ModeAttributes	Get	BITS16	Bit 0:	Meaning:
				0:	Only static time and energy values are available (Value of bit 0 attribute is not implemented)
				1:	Dynamic time and energy values are available
				Bit 1-15:	Reserved
2	TimeMinPause	Get	UINT32	Minimum pause time in milliseconds. (t_{pause}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
3	TimeToPause	Get	UINT32	Maximum time to go to this Energy saving mode.(ms, t_{off}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
4	TimeToOperate	Get	UINT32	Maximum time needed to go to the “Ready to operate” mode. (ms, t_{on}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
5	TimeMinLengthOfStay	Get	UINT32	The minimum time that the device must stay in this mode. In milliseconds.(ms, t_{off_min}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
6	TimeMaxLengthOfStay	Get	UINT32	Maximum time that is allowed to stay in this mode. In milliseconds. If no maximum value is available or if not implemented, the maximum value FFFFFFFFh shall be used.	
7	ModePowerConsumption	Get	FLOAT	Amount of power consumed in this mode. (kW) If the value is undefined the value 0.0 shall be used.	
8	EnergyConsumptionToPause	Get	FLOAT	Amount of energy required to go to this mode. (kWh) If the value is undefined the value 0.0 shall be used.	
9	EnergyConsumptionToOperate	Get	FLOAT	Amount of energy required to go to the “Ready to operate” mode from this mode. (kWh) If the value is undefined the value 0.0 shall be used.	
10	Availability	Get	BOOL	Indicates if this energy saving mode is available given the current device state. Not used for PROFINET.	
				False	Not available
				True	Available (Value if attribute not implemented)
11	Power Consumption	Get	UINT32	Indicates the power consumption of the device when in this state. Not used for PROFINET.	

Command Details: Start_Pause

Details

Command Code	10h
Valid for:	Object

Description

This command is sent to the host application when the system wants to initialize a pause of the system. The length of the pause is specified in milliseconds. The response of the message contains the destination mode (i. e. the instance number of the selected energy saving mode).

- Command Details

Field	Contents	Comments
Data[0]	Pause time (low word, low byte)	Pause time (ms)
Data[1]	Pause time (low word, high byte)	
Data[2]	Pause time (high word, low byte)	
Data[3]	Pause time (high word, high byte)	

- Response Details

Field	Contents	Comments
Data[0]	Instance number (low byte)	Instance number of selected Energy mode
Data[1]	Instance number (low byte)	

If the application is unable to select a state, given the requested pause time, it shall return one of the error codes in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device and the requested pause time it is currently not possible to enter any energy saving mode
0x12	Value too low	The requested pause time is too short

Command Details: End_Pause

Details

Command Code	11h
Valid for:	Object

Description

This command is sent to the host application when the system wants to return the system from a pause mode back to “Ready to operate” mode. In the response message the number of milliseconds to actualize the switch is returned.

- Command Details

  (none)

- Response Details

Field	Contents	Comments
Data[0]	Time To Operate (low word, low byte)	Time needed to switch to “Ready to operate”
Data[1]	Time To Operate (low word, high byte)	
Data[2]	Time To Operate (high word, low byte)	
Data[3]	Time To Operate (high word, high byte)	

If the application is unable to end the pause it shall return the error code in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device, it is currently not possible to end the pause

Command Details: Preview_Pause_Time

Details

Command Code	12h
Valid for:	Object

Description

Not used for PROFINET devices.

This command is sent to the host application when the system wants to preview the application’s choice of Energy saving mode. The length of the pause is specified in milliseconds. The response shall contain the destination mode the application would have chosen if the StartPause service was sent (that is, the instance number of the selected energy saving mode). No transition to an Energy saving mode occurs.

- Command Details

Field	Contents	Comments
Data[0]	Pause time (low word, low byte)	Pause time (ms)
Data[1]	Pause time (low word, high byte)	
Data[2]	Pause time (high word, low byte)	
Data[3]	Pause time (high word, high byte)	

- Response Details

Field	Contents	Comments
Data[0]	Instance number (low byte)	Instance number of selected Energy mode
Data[1]	Instance number (low byte)	

If the application is unable to select a state, given the requested pause time, it shall return one of the error codes in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device and the requested pause time it is currently not possible to enter any energy saving mode
0x12	Value too low	The requested pause time is too short

14.10 EtherNet/IP Host Object (F8h)

Category

Basic, Extended

Object Description

This object implements EtherNet/IP specific features in the host application. Note that this object must not be confused with the Ethernet Host Object, see *Ethernet Host Object (F9h), p. 231*.

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

Note that some of the commands used when accessing this object may require segmentation. For more information, see *Message Segmentation, p. 182*.

If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification..

See also ...

- *Identity Object (01h), p. 113* (CIP object)
- *Assembly Object (04h), p. 117* (CIP object)
- *Port Object (F4h), p. 136* (CIP object)
- *CIP Port Configuration Object (0Dh), p. 192*
- Anybus CompactCom 40 Software Design Guide, “Error Codes”

Supported Commands

Object:	Get_Attribute Process_CIP_Object_Request Set_Configuration_Data Process_CIP_Routing_Request Get_Configuration_Data
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“EtherNet/IP”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Data Type	Default Value	Comment
1	Vendor ID	Get	UINT16	005Ah	These values are set in the Identity Object (CIP) at startup. See also... • *Network Identity, p. 25* • *Identity Object (01h), p. 113* Please note that changing any of these attributes requires a new Vendor ID.
2	Device Type	Get	UINT16	002Bh	
3	Product Code	Get	UINT16	0037h	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	(software revision)	
5	Serial Number	Get	UINT32	(set at production)	
6	Product Name	Get	Array of CHAR	“Anybus CompactCom 40 EtherNet/IP(TM)”	

Extended

#	Name	Access	Data Type	Default Value	Comment
7	Producing Instance No.	Get	Array of UINT16	-	The values in this array are the EtherNet/IP Assembly instance numbers that matches the host application Assembly Mapping Object instances that are listed in attribute #11 (Write PD Instance List). If the Assembly Mapping Object is not implemented, one element in this array is allowed, to set the producing instance number. The maximum number of entries in the array is 6. See “Multiple Assembly Instances” below for an example.
8	Consuming Instance No.	Get	Array of UINT16	-	The values in this array are the EtherNet/IP Assembly instance numbers that matches the host application Assembly Mapping Object instances that are listed in attribute #12 (Read PD Instance List). If the Assembly Mapping Object is not implemented, one element in this array is allowed, to set the consuming instance number. The maximum number of entries in the array is 6. See “Multiple Assembly Instances” below for an example.
9	Enable communication settings from net	Get	BOOL	True	Value Meaning True Can be set from network False Cannot be set from network See also ... • *TCP/IP Interface Object (F5h), p. 138* (CIP-object) • *Ethernet Link Object (F6h), p. 141* CIP-object) • *Network Configuration Object (04h), p. 151*(Anybus Module Object)
11	Enable CIP forwarding	Get	BOOL	False	Value Meaning True Requests to unknown CIP objects and unknown assembly object instances are routed to the application. False Requests to unknown CIP objects and unknown assembly object instances are not routed to the application. See also.command details for Process _CIP_Object_Request below
12	Enable Parameter Object	Get	BOOL	True	Value Meaning True Enable CIP Parameter Object False Disable CIP Parameter Object
13	Input-Only heartbeat instance number	Get	UINT16	0003h	See “Instance 03h Attributes (Heartbeat, Input-Only)” in *Assembly Object (04h), p. 117* (CIP-object).
14	Listen-Only heartbeat instance number	Get	UINT16	0004h	See “Instance 04h Attributes (Heartbeat, Listen-Only)” in *Assembly Object (04h), p. 117* (CIP-object).
15	Assembly object Configuration instance number	Get	UINT16	0005h	See “Instance 05h Attributes (Configuration Data)” in *Assembly Object (04h), p. 117* (CIP-object).
16	Disable Strict IO Match	Get	BOOL	False	If true, the module will accept Class1 connection requests that have sizes that’s less than or equal to the configured IO sizes.

#	Name	Access	Data Type	Default Value	Comment
17	Enable unconnected routing	Get	BOOL	False	If true, the module enables unconnected CIP routing. This also triggers an initial upload of the contents of the CIP Port Mapping object.
18	Input-Only extended heartbeat instance number	Get	UINT16	0006h	See “Instance 06h Attributes (Heartbeat, Input-Only Extended)” in *Assembly Object (04h), p. 117* (CIP-object).
19	Listen-Only extended heartbeat instance number	Get	UINT16	0007h	See “Instance 06h Attributes (Heartbeat, Listen-Only Extended)” in *Assembly Object (04h), p. 117* (CIP-object).
20	Interface label port 1	Get	Array of CHAR	Port 1	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #1
21	Interface label port 2	Get	Array of CHAR	Port 2	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #2
22	Interface label internal port	Get	Array of CHAR	Internal	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #3
23 - 25	(reserved)				
26	Enable EtherNet/IP QuickConnect	Get	BOOL	False	Value – Meaning True – EtherNet/IP QuickConnect functionality enabled. False – False EtherNet/IP QuickConnect functionality disabled. If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification.
27 - 28	(reserved)				
29	Ignore Sequence Count Check	Get	BOOL	False	Setting this attribute to “true” makes the module ignore the Sequence Count Check for consumed Class 1 data. This means that all data, not just changed/new data, received from the Originator, will be copied to the application. Copying all data and not just changed data is a violation of the CIP specification. It will also affect the performance of the module. Use precaution when setting this flag to“true”. HMS Networks will do NO performance measurements and states NO guarantees about how performance will be affected when copying all data.
30	ABCC ADI Object Number	Get	UINT16	00A2h	This attribute either changes the object number of theADI Object (CIP object) or disables the ADI Object (CIP object). Valid object numbers are within the vendor specific ranges (0064h - 00C7h and 0300h - 04FFh). Any other value will disable the ADI object.
31	Enable DLR	Get	BOOL	True	Value – Meaning True – DLR functionality enabled False – DLR functionality disabled
32	Enable CIP Sync	Get	BOOL	False	Value – Meaning True – CIP Sync functionality enabled False – CIP Sync functionality disabled

Multiple Assembly Instances

The Assembly Mapping Object has two arrays on class level (Write PD Instance List and Read PD Instance List) listing instances defined by the application. The arrays of attributes 7 and 8 in the EtherNet/IP host object (Producing Instance Number and Consuming Instance number) are bound to the instance lists in the Assembly Mapping Object. The arrays list the corresponding CIP instance numbers representing each assembly instance defined by the application.

For more information, see

- *Using the Assembly Mapping Object (EBh), p. 33*
- Anybus CompactCom 40 Software Design Guide, “Assembly Mapping Object (EBh)”

Command Details: Process_CIP_Object_Request

Category

Extended

Details

Command Code:	10h
Valid for:	Object

Description

By setting the 'Enable CIP Request Forwarding'-attribute (#11), all requests to unimplemented CIP-objects and unknown assembly object instances, will be forwarded to the host application through this command. The application then has to evaluate the request and return a proper response. The module supports one CIP-request; additional requests will be rejected by the module.

Note that since the telegram length on the host interface is limited, the request data size must not exceed 1524 bytes. If it does, the module will send a 'resource unavailable' response to the originator of the request and the message will not be forwarded to the host application.

1524 bytes are available for class 3 connections using Large Forward Open. If UCMM is used, the total size (Message Router request/response) is limited to 504 bytes.

If the legacy message channel is used, message data is limited to 255 bytes.

This command is similar - but not identical - to the 'Process_CIP_Request'-command in the Anybus CompactCom 40 DeviceNet.

- Command Details

Field	Contents	Comments
CmdExt[0]	CIP Service Code	CIP service code from original CIP request
CmdExt[1]	Request Path Size	Number of 16-bit words in the Request Path field
MsgData[0... m]	Request Path	CIP EPATH (Class, Instance, Attr. etc.)
MsgData[m... n]	Request Data	Service-specific data

- Response Details

Field	Contents	Comments
CmdExt[0]	CIP Service Code	(Reply bit set)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	General Status	CIP General Status Code
MsgData[1]	Size of Additional Status	Number of 16-bit words in Additional Status array
MsgData[2... m]	Additional Status	Additional Status, if applicable
MsgData[m... n]	Response data	Actual response data, if applicable

! When using this functionality, make sure to implement the common CIP Class Attribute (attribute #1, Revision) for all objects in the host application firmware. Failure to observe this will prevent the module from successfully passing conformance tests.

Command Details: Set_Configuration_Data

Category

Extended

Details

Command Code:	11h
Valid for:	Object

Description

If the data segment in the CIP “Forward_Open” service contains Configuration Data, this will be forwarded to the host application through this command. If implemented, the host application should evaluate the request and return a proper response. Segmentation is used, see “Message Segmentation” on page 189 for more information. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

This command must be implemented in order to support Configuration Data. If not implemented, “Forward_ Open” requests will be rejected.

- Command Details

Field	Contents	Comments
CmdExt[0]	-	(reserved, ignore)
CmdExt[1]	Segmentation Control bits	See *Message Segmentation, p. 182*
MsgData[0 - 1]	Producing connection point	Producing connection point, requested by the originator.
MsgData[2 - 3]	Consuming connection point	Consuming connecition point, requested by the originator.
MsgData[4... n]	Data	Actual configuration data

When the Set_Configuration_Data command is sent to the application as a result of a CIP Forward_open service containing Configuration Data, the producing connection point will be indicated by MsgData[0-1], and the consuming connection point by MsgData[2-3]. However, the Set_Configuration command may also come as a result of a CIP Set_Attribute_Single service to the CIP Assembly Object or a non matching NULL Forward Open service request. For both cases, MsgData[0-1] and MsgData[2-3] will contain 0 (zero).

- Response Details (Success)

Field	Contents	Comments
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)

- Response Details (Error)

Field	Contents	Comments
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	Error code	Anybus error code
MsgData[1]	Extended error code	If the Anybus error code is set to FFh, the extended error code shall be translated as shown in the table below.
MsgData[2... 3]	Index	If the Extended error code is set to 02h (invalid configuration), this parameter points to the attribute that failed.

Extended Error Code

If the Error code equals FFh (Object specific error), the extended code will be translated as below:

Code	Contents	CIP no.	CIP status code	Additional Information
01h	Ownership conflict	01h	Connection failure	The configuration data was supplied in a forward open request.
		10h	Device State conflict	The configuration data was supplied in a set request to the Assembly object.
02h	Invalid configuration	09h	Bad attribute data	CIP extended error code: Use value from MsgData[2 - 3]. The extended error code shall only be used if the request originated from a Forward Open request, not for explicit set requests.

- *Connection Manager (06h), p. 120* (CIP object)
- Message segmentation

Command Details: Process_CIP_Routing_Request

Category

Extended

Details

Command Code:	12h
Valid for:	Object

Description

The module will strip the first path within the “Unconnected_Send” service and evaluate whether or not it’s possible to continue with the routing (e.g. check that the requested port exists within the port object). If the stripped path was the last path the contents delivered to the application will be the CIP request sent to the destination node, otherwise it will be an “Unconnected_Send” service with updated route path information.

The module supports one pending request. Additional requests will be rejected by the module.

Please note that since the telegram length on the host interface is limited, the data must not exceed 1524 bytes in length. If it does, the module will reject the originator of the request (“Resource unavailable”), and this command will not be issued towards the host application.

If the legacy message channel is used, message data is limited to 255 bytes.

- Command Details

Field	Contents	Comments
CmdExt[0]	-	(reserved, ignore)
CmdExt[1]	-	(reserved, ignore)
MsgData[0... n]	Destination Path	Destination path encoded as an EPATH.
MsgData[n+1]	Time_tick	Valid after timeout parameters have been updated
MsgData[n+2]	Time-out_ticks	Valid after timeout parameters have been updated
MsgData[n+3... m]	CIP message	CIP message to route

- Response Details

Field	Contents	Comments
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	CIP Service	Actual CIP service code, response bit set
MsgData[1]	00h	(reserved, set to zero)
MsgData[2]	General Status	Actual CIP General status code
MsgData[3]	Size of Additional Status	No. of 16-bit words in Additional Status Array
MsgData[4... n]	Additional Status Array	Additional status, if applicable
MsgData[n+1... m]	Response Data	Actual response data

See also..

- *Port Object (F4h), p. 136* (CIP object)
- *CIP Port Configuration Object (0Dh), p. 192*

Command Details: Get_Configuration_Data

Category

Extended

Details

Command Code:	13h
Valid for:	Object

Description

If the configuration data is requested from the network, the Anybus will issue this command to the application. The application shall send the stored configuration data in the response message.

Segmentation is used since the telegram length on the host interface is limited. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

This command must be implemented in order to support Configuration Data. If not implemented, the request will be rejected by the Anybus module.

- Command Details

Field	Contents	Comments
CmdExt[0]	00h	-
CmdExt[1]	00h	-
MsgData[0... n]	-	No extended message data

- Response Details (Success)

Field	Contents	Comments
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	Segmentation Control bits	See *Message Segmentation, p. 182*
MsgData[0 - n]	Status	Configuration data from the application

- Response Details (Error)

Field	Contents	Comments
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	Segmentation Control bits	See *Message Segmentation, p. 182*
MsgData[0]	Status	Anybus protocol error code

14.11 Ethernet Host Object (F9h)

Object Description

This object implements Ethernet features in the host application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

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

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

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	**Note**: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
10	Port 2 MAC address	Get	Array of UINT8	-	**Note**: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
11	Enable ACD	Get	BOOL	True (Enabled)	Enable/Disable ACD protocol. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.
12	Port 1 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 1. • This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 1 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.
13	Port 2 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 2. • This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 2 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website. 02h: Inactive. The attribute is set to this value for a device that only has one physical port. All two-port functionality is disabled. No references can be made to this port. **Note:** This functionality is available for PROFINET, Ethernet/IP and Modbus-TCP devices.
14	(reserved)				
15	Enable reset from HICP	Get	BOOL	0 = False	Enables the option to reset the module from HICP.
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	Whenever the configuration is assigned or changed, the Anybus CompactCom module will update this attribute.

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

Network Status

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

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

Bit	Contents	Description	Comment
4	Link port 2	Current link status for port 2 1 = Link sensed 0 = No link	Not used for EtherCAT
5... 15	(reserved)	(mask off and ignore)	

DHCP Option 61 (Client Identifier)

Only valid for EtherNet/IP devices

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

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

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

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

Example:

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

Example 2:

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

This page intentionally left blank

A Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

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

B Compatibility to Standard Anybus CompactCom 40

The Anybus CompactCom 40 EtherNet/IP IIoT Secure is not compatible to standard Anybus CompactCom. The differences are summarized in the table below.

Issue	Description
User account configuration	“web_acces.cfg” and “ftp.cfg” files are not used for account configuration. User accounts are given roles and are configured using the internal web pages. They can also be configured using the JSON API. For more information see *Initial Setup and Account Configuration, p. 14*.
HTTP(S) configuration	Changes are made to the “http.cfg” configuration file. • Web root cannot be configured. All URLs is from system root, but accounts have different access rights. • Access rights are controlled per role and are configured in this file. • The index page must be configured. For more information see *Secure Web Server (HTTPS), p. 37*.
File transfer protocol	File transfer protocol FTP changed to WebDAV
SSI	SSI removed.
JSON functions URL	The URLs to JSON functions is changed, to make it possible to configure access rights for each JSON API. For more information see *JSON, p. 46*.
JSON function protection	For standard Anybus CompactCom 40 all JSON functions are protected by the root “web_acces.cfg” file. For Anybus CompactCom Security modules, JSON functions are instead protected independently in the same way as protecting file system resources. For more information see *Secure Web Server (HTTPS), p. 37*.
OPC UA	File format changed for opcua.cfg.

C Implementation Details

C.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of EtherNet/IP, this means that the SUP-bit is set when one or more CIP (Class 1 or Class 3) connections has been opened towards the module.

C.2 Anybus State Machine

The table below describes how the Anybus Statemachine relates to the EtherNet/IP network

Anybus State	Implementation	Comment
WAIT_PROCESS	The module stays in this state until a Class 1 connection has been opened.	-
ERROR	Class 1 connections errors Duplicate IP address detected	-
PROCESS_ACTIVE	Error free Class 1 connection active (RUN-bit set in the 32-bit Run/Idle header of an Exclusive-Owner connection)	Only valid for consuming connections.
IDLE	Class 1 connection idle.	
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

C.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.

D Secure HICP (Secure Host IP Configuration Protocol)

D.1 General

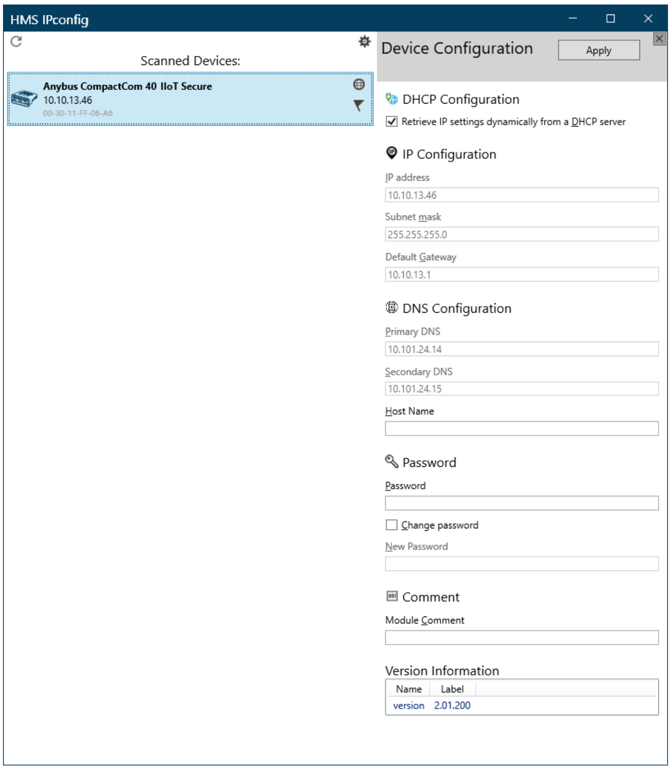
The Anybus CompactCom 40 EtherNet/IP IIoT Secure supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/ disable DHCP. Anybus IPconfig can be downloaded free of charge from the support pages at HMS Networks, www.anybus.com/support. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

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

D.2 Operation

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

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



HMS IPconfig

Scanned Devices:

Anybus CompactCom 40 IIoT Secure
10.10.13.46
00-30-11-FF-06-A6

Device Configuration

DHCP Configuration

☑ Retrieve IP settings dynamically from a DHCP server

IP Configuration

IP address: 10.10.13.46

Subnet mask: 255.255.255.0

Default Gateway: 10.10.13.1

DNS Configuration

Primary DNS: 10.101.24.14

Secondary DNS: 10.101.24.15

Host Name:

Password

Password:

☐ Change password

New Password:

Comment

Module Comment:

Version Information

Name	Label
version	2.01.200

Fig. 28

Validate the new settings by clicking **Set**, or click **Cancel** to cancel all changes.

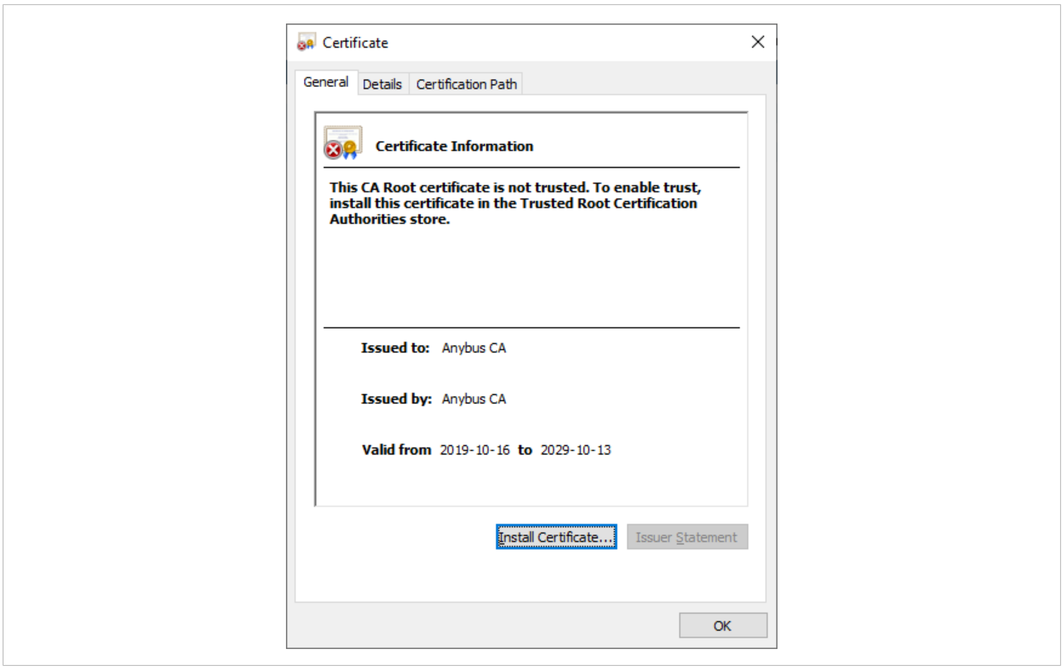
For Anybus CompactCom 40 EtherNet/IP IIoT Secure, you can not set a password to protect the configuration from unauthorized access in this view. The password is set from the internal web pages.

! It is strongly recommended to password protect the IP configuration.

E Installing a CA Certificate in Windows

This section describes how to install an Anybus CA Certificate in the trusted certificate store in Windows. Chrome and IE will use the CA certificate. Other browsers may use other stores for their trusted certificates.

1. Open the certificate.
2. Click on “Install Certificate....”



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Anybus CA

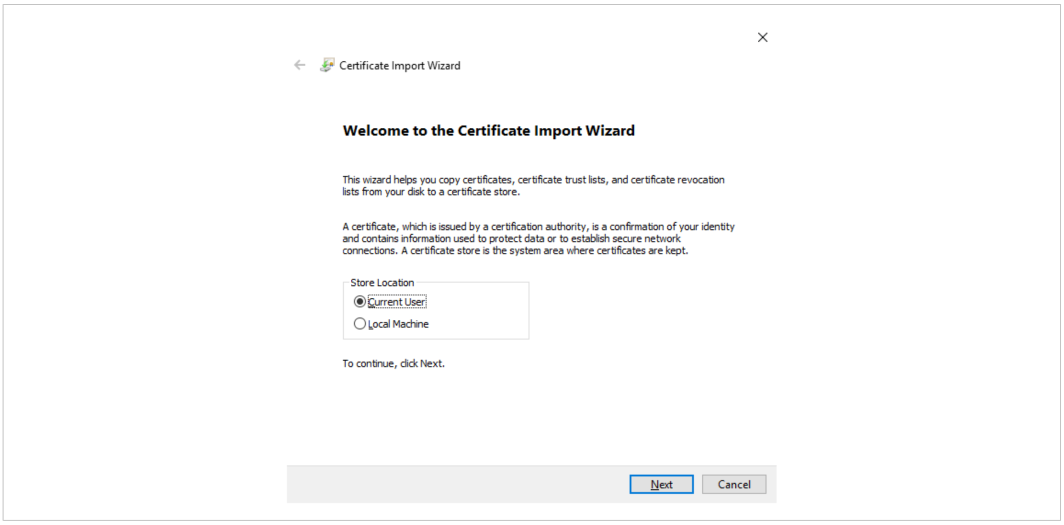
Issued by: Anybus CA

Valid from 2019-10-16 to 2029-10-13

Fig. 29

3. The Certificate Import Wizard is opened. Select Store Location to Current User.

   Click Next to continue the installation.



Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

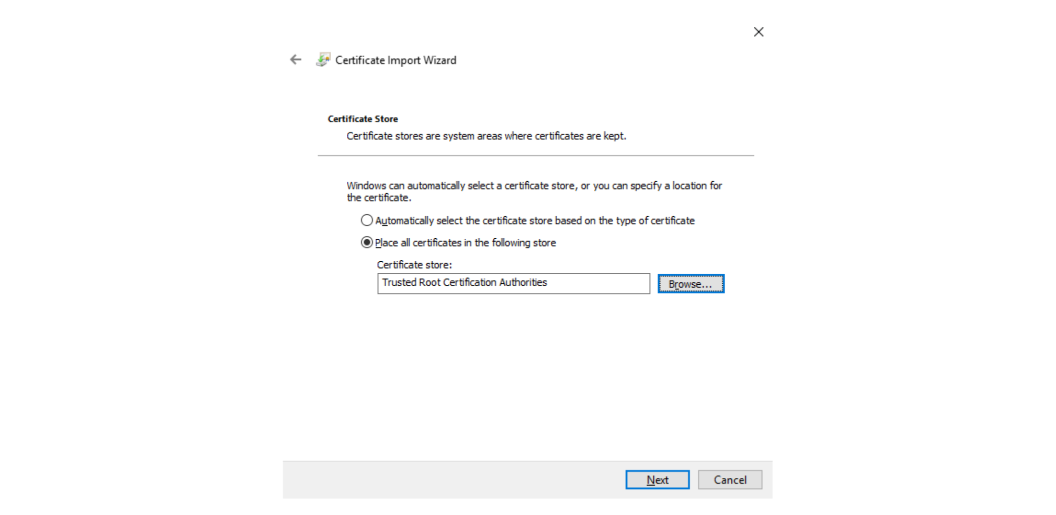
A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

Store Location
● Current User
○ Local Machine

To continue, click Next.

Fig. 30

4. Continue the installation according to the instructions in the wizard. When asked to specify a certificate store, select the trusted store as shown in the figure.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

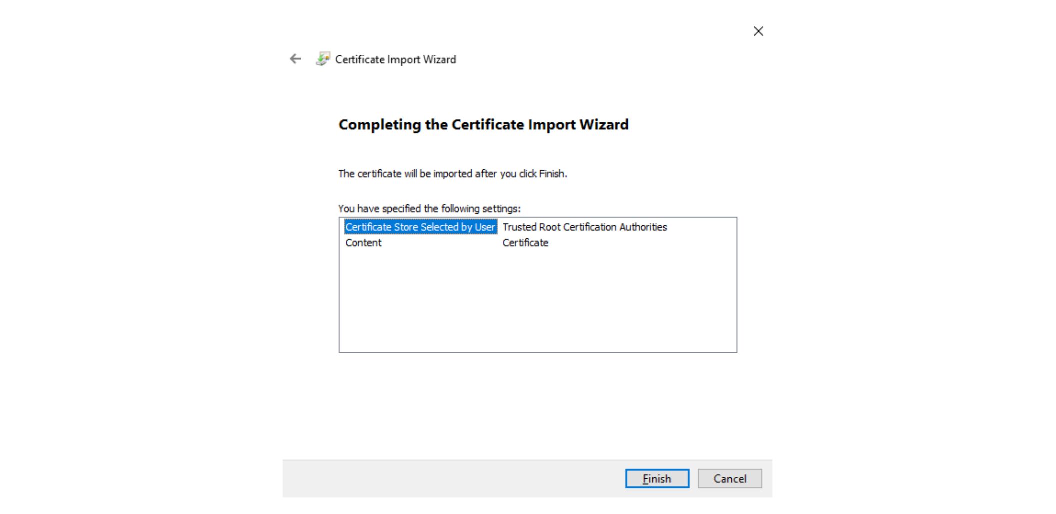
○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store: Trusted Root Certification Authorities

Fig. 31

5. Check that the settings are correct before you complete the wizard.



Certificate Import Wizard

Completing the Certificate Import Wizard

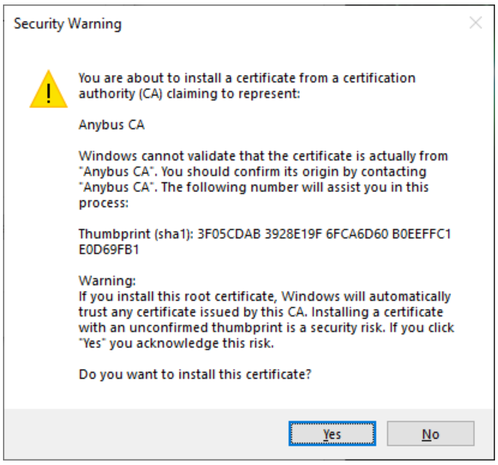
The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certification Authorities
Content	Certificate

Fig. 32

6. The wizard will finally ask you to confirm that you want to install this certifcate.



Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

Anybus CA

Windows cannot validate that the certificate is actually from "Anybus CA". You should confirm its origin by contacting "Anybus CA". The following number will assist you in this process:

Thumbprint (sha1): 3F05CDAB 3928E19F 6FCA6D60 B0EEFFC1 E0D69FB1

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

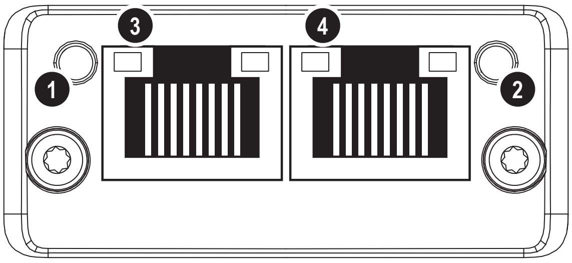
Do you want to install this certificate?

Fig. 33

F Technical Specification

F.1 Front View

F.1.1 Front View (Ethernet Connectors)

#	Item	Connector	
1	Network Status LED	Ethernet, RJ45	
2	Module Status LED		
3	Link/Activity LED (port 1)		
4	Link/Activity LED (port 2)		

Test sequences are performed on the Network and Module Status LEDs during startup.

F.1.2 Network Status LED

LED State	Description
Off	No power or no IP address
Green	Online, one or more connections established (CIP Class 1 or 3)
Green, flashing	Online, no connections established
Red	Duplicate IP address, FATAL error
Red, flashing	One or more connections timed out (CIP Class 1 or 3)

F.1.3 Module Status LED

LED State	Description
Off	No power
Green	Controlled by a Scanner in Run state
Green, flashing	Not configured, or Scanner in Idle state
Red	Major fault (EXCEPTION-state, FATAL error etc.)
Red, flashing	Recoverable fault(s). Module is configured, but stored parameters differ from currently used parameters
Alternate red and green, flashing	Firmware upgrade in progress

F.1.4 LINK/Activity LED 3/4

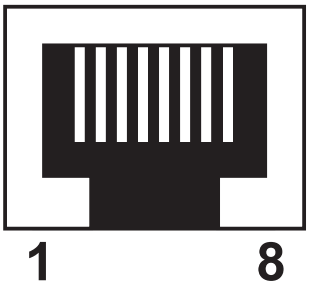
LED State	Description
Off	No link, no activity
Green	Link (100 Mbit/s) established
Green, flickering	Activity (100 Mbit/s)
Yellow	Link (10 Mbit/s) established
Yellow, flickering	Activity (10 Mbit/s)

F.1.5 Ethernet Interface

Ethernet Interface (RJ45 connectors)

The Ethernet interface 10/100Mbit, full or half duplex operation.

Pin no	Description
4,5,7,8	Connected to chassis ground over serial RC circuit
6	RD-
3	RD+
2	TD-
1	TD+
Housing	Cable Shield



F.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to functional earth via the FE pad/FE mechanism described in the *Anybus CompactCom 40 Hardware Design Guide*. Proper EMC behavior is not guaranteed unless these FE requirements are fulfilled.

F.3 Power Supply

F.3.1 Supply Voltage

The Anybus CompactCom 40 EtherNet/IP IIoT Secure requires a regulated 3.3 V power source as specified in the general *Anybus CompactCom 40 Hardware Design Guide*.

F.3.2 Power Consumption

The Anybus CompactCom 40 EtherNet/IP IIoT Secure is designed to fulfil the requirements of a Class C module. The current hardware design consumes up to 510 mA.

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.

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

F.4 Environmental Specification

Consult the *Anybus CompactCom 40 Hardware Design Guide* for further information.

F.5 EMC Compliance

Consult the *Anybus CompactCom 40 Hardware Design Guide* for further information.

G Conformance Test Guide

G.1 General

When using the default settings of all parameters, the Anybus CompactCom 40 EtherNet/IP IIoT Secure is precertified for network compliance. This precertification is done to ensure that the end product *can* be certified.

To be allowed to use EtherNet/IP in a product the vendor is required to be a licensed EtherNet/IP vendor, with a vendor ID of its own. Please contact www.odva.org to obtain a vendor ID.

Changes in the parameters in the example EDS file, supplied by HMS Networks, will require a certification. This chapter provides a guide for successfully conformance testing your product, containing the Anybus CompactCom 40 EtherNet/IP IIoT Secure, to comply with the demands for network certification set by the ODVA.

The actions described in this appendix have to be accounted for in the certification process, e.g. the identity of the product needs to be changed to match your company and device.

> ! This appendix provides guidelines and examples of what is needed for conformance testing and certification. Depending on the functionality of your application, there may be additional steps to take.

All screenshots within this document are taken from the ODVA Conformance Test Software Tool for EtherNet/IP CT14, © ODVA Inc. This software is available for order through the ODVA website. It is required to perform pre-testing with this software prior to submitting the product for conformance testing.

Also, a Statement of Conformance file (STC file), describing the EtherNet/IP application, has to be prepared prior to submitting the product for conformance testing.

G.2 Suggested Test Tools

G.2.1 Wireshark

This free, open source tool is the de facto standard for network capture and analysis. It is heavily used by ODVA TSPs, HMS Networks, and the greater EtherNet/IP user base. Wireshark (www.wireshark.org) captures Ethernet traffic using your computers network interface card, and displays the contents in an intuitive fashion that allows for detailed analysis of the packets. Developers from HMS Networks have contributed to the EtherNet/IP dissectors (the analysis engine), and it is possible for users to create their own dissectors for their application data. The use of Wireshark is well documented, but there are a few good tips for EtherNet/IP testing that will help users get to the crucial information.

- Use viewing filters “CIP” to see only EtherNet/IP traffic.
- It is possible to filter by the HMS MAC ID. This will only show Ethernet messages with HMS devices as the source or destination “eth.addr[0:3] == 00:30:11”.
- There are many other useful filters available on the Wireshark webpage.

G.2.2 NMAP

NMAP is a free, open source tool for network discovery and security testing. NMAP will discover which TCP and UDP ports are open or responding. It will also determine which layer 3 services are supported by your device. ODVA has strict guidelines for open ports, and mandatory layer 3 services. For the NMAP procedure used by TSPs please see the Sample Test Report that comes with Conformance Test Software from ODVA.

G.2.3 ODVA Conformance Test Software

This automated test software is designed to query, provoke, and detect software flaws in your device. ODVA sells yearly subscriptions of this software to vendors so that they can prepare for conformance testing. This software is also the best way to modify or create the Statement of Conformance (STC) file. Pressing CTRL+D will bring up a GUI for the Data section of the STC file.

Getting Started

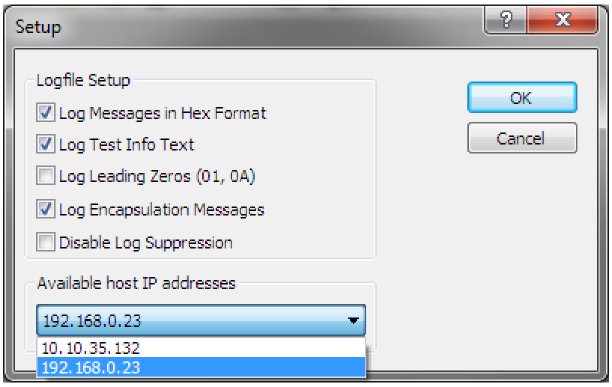
After completing the install, a webpage is brought up in the default browser. This page gives an overview of the test software and lists the relevant documentation with a brief summary. The setup for testing is covered in the Conformance Test Software User Manual.

Chapter	Contents
1	System requirements and installation
2	How to select a device and how to modify the Statement of Conformance file
3	How to set up the network to prepare for testing
4	How to run the test software

The User Manual - Critical Points

Users are strongly encouraged to read through the Conformance Test Software User Manual to fully understand the testing software. The following points are meant to recapture the critical sections of this document.

- The Network Interface that will be used for testing needs to be selected from the available network interface cards in the Setup menu.



Setup

Logfile Setup

☑ Log Messages in Hex Format

☑ Log Test Info Text

□ Log Leading Zeros (01, 0A)

☑ Log Encapsulation Messages

□ Disable Log Suppression

Available host IP addresses

192.168.0.23

10.10.35.132

192.168.0.23

Fig. 34

- Most devices will comply with the default timeout settings, but some require more relaxed standards for responses. This can be set in the Set Message Wait Timers menu.
- The latest version of the CT Test software requires users to allocate a second IP address for their network interface card.
- Enabling the Encapsulation Logging feature of the CT test will allow users to efficiently work with Wireshark captures and Conformance Test logs.

G.2.4 EZ-EDS

EZ-EDS is a free utility made available by ODVA. This tool is very helpful for editing and testing Electronic Data Sheets. Electronic Data Sheets are ASCII formatted files that describe data organization, configuration, and performance capabilities. They are commonly called EDS files, and have the extension .eds. EDS files can be built and modified using a text editor, but EZ-EDS provides a graphical user interface that brings attention to major fields. EZ-EDS also tests EDS files for correct formatting. Much of the possible content of EDS files is optional, and ODVA tests stress correct formatting and not content.

G.2.5 Anybus EDS Generator

The Anybus EDS Generator tool automatically generates an EDS file by scanning a device using the Anybus CompactCom 40 EtherNet/IP. This tool is easy to use and will provide a correctly configured EDS file that matches your product. It is still required to validate the EDS file via EZ-EDS.

The tool is available from the HMS Networks web site.

G.2.6 Sample Test Reports

The subscription to the conformance test software includes the EtherNet/IP Sample Test Report document. This document outlines the manual procedure that testers will perform in addition to running the automated test software.

Manual Test Procedure

Some features of EtherNet/IP cannot be properly verified by automated test software or the development of a fully automated test may be impractical. For these features, a manual test procedure, as well as passing criteria, is listed in the Sample Test Report. This is the exact procedure and criteria used by the Test Service Providers.

The majority of functionality that needs to be tested manually is provided by the systems of the Anybus CompactCom, and has no interaction with the host application. Therefore, developers using the Anybus CompactCom may omit this lengthy procedure, but they must check the following:

- HMS recommends everyone to complete the Physical layer and EDS test sections of the sample test report. This ensures that produce labeling of LED’s is correct and that the EDS file is verified prior to submitting the product to the TSP for conformance test
- If DLR is enabled in the product, it is required to be able to configure the speed and the duplex of all Ethernet ports in some way. The host application may elect to disable the standard means of configuring the speed and duplex in the Anybus CompactCom by:
  - Disabling set access to the Ethernet Link object by setting instance attribute #9 (Enable Communication Settings from NET) in the Host EtherNet/IP object to False.
  - Disabling the web server. On the standard web pages of the Anybus CompactCom it is possible to configure speed and duplex of the ethernet ports. For applications using transparent ethernet functionality the web server is always disabled.

  If none of the these ways of configuring the speed and duplex is possible, the host application must provide some other way to configure them. For example the application can have a keypad interface which can be used for configuration.
- If the host application includes hardware switches (for example DIP switches or rotary switches) for configuring the IP address or has disabled either ACD or DLR, HMS Networks recommends to perform the manual test cases in TCP/IP Interface Object Tests (section 4), Ethernet Link Object Tests (section 5), and Address Conflict Detection (ACD) Tests (section 10) in the sample test report.

G.3 Statement of Conformance (STC)

> ! This document is not a comprehensive guide. Following the steps below will not absolutely guarantee that a device will complete conformance testing.

The goal of this section is to explain the relation the Anybus Objects to the Conformance Test and the Statement of Conformance (STC). The objects listed below exist in the host application, the Anybus CompactCom, and not in the EtherNet/IP interface. The objects are described in the Anybus CompactCom 40 Software Design Guide and in the Anybus CompactCom 40 EtherNet/IP Network Guide.

It is recommended to read the CIP Protocol Test Specification and the EtherNet/IP Test Specification prior to testing. In these documents the expected response and/or the acceptable behavior are stated, which is useful to be able to avoid a lot of initial errors. Modifications can be made to the Statement of Conformance and to the host application at an early stage, reducing time and effort.

G.3.1 Implementation of Host Objects

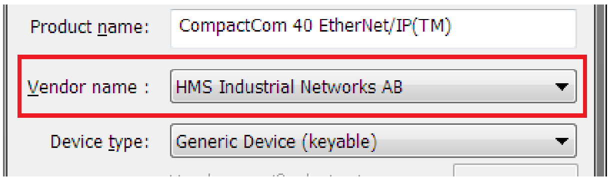
The implementations of the host objects may have to be adapted, to make sure that the end product will pass a conformance test. Using the CT Software, follow the instructions below. Only the host objects relevant to EtherNet/IP will be discussed.

EtherNet/IP Host Object (F8h)

The implementation of the EtherNet/IP Host Object (F8h) has impact on the following objects: Identity Object (01h, CIP object), Assembly Object (04h, CIP object), Port Object (F4h, CIP object), and CIP Port Configuration Object (host object, 0Dh). It also has impact on how the STC is configured. The instance attributes, listed below, need to be considered.

EtherNet/IP Host Object (F8h) - Attribute #1 - Vendor ID

The Vendor ID must match the Vendor name in the CT software and the STC.



Product name: CompactCom 40 EtherNet/IP(TM)

Vendor name : HMS Industrial Networks AB

Device type: Generic Device (keyable)

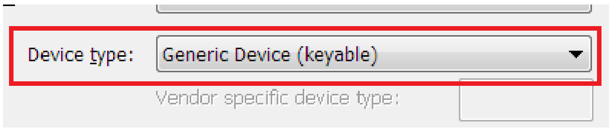
Fig. 35

First time EtherNet/IP vendors may not find their name available from the drop down menu, as it’s not certain that the test software has been updated. It is possible to pre-test with any Vendor ID in the list, to reduce the number of errors reported due to Vendor ID mismatch, as long as the Vendor ID is changed in both the device and in the STC before actual conformance testing.

Alternatively, vendors can add the vendor information to the VID.dat file.

EtherNet/IP Host Object (F8h) - Attribute #2 - Device Type

The Device Type must match the Device Type in the drop down list:

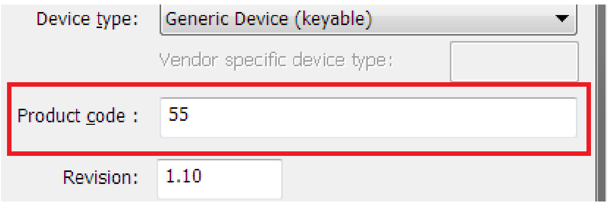


Device type: Generic Device (keyable)

Fig. 36

EtherNet/IP Host Object (F8h) - Attribute #3 - Product Code

The Product Code must match the Product Code in the drop down list:



Device type: Generic Device (keyable)

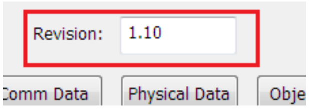
Product code : 55

Revision: 1.10

Fig. 37

EtherNet/IP Host Object (F8h) - Attribute #4 - Revision

The Revision must match the revision field <major>. <minor>.



Revision: 1.10

Fig. 38

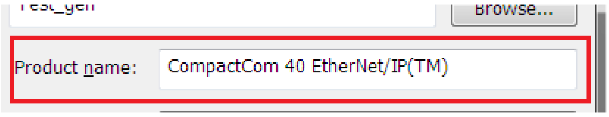
EtherNet/IP Host Object (F8h) - Attribute #5 - Serial Number

The current version of CT test does not check serial number.

ⓘ *According to the CIP specification, the combination of Vendor ID and serial number must be unique. It is not permitted to use a custom serial number in combination with the HMS Vendor ID (005Ah).*

EtherNet/IP Host Object (F8h) - Attribute #6 - Product Name

The Product Name must match the Product Name field.

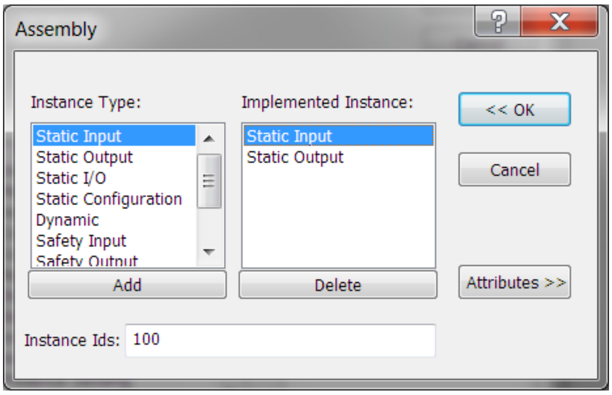


Product name: CompactCom 40 EtherNet/IP(TM)

Fig. 39

EtherNet/IP Host Object (F8h) - Attribute #7 - Producing Instance No.

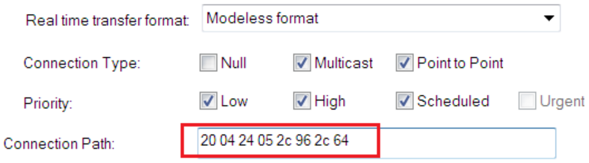
The Producing Instance(s) will impact the assembly object, and will need to be listed. For most applications the producing instance(s) will be Static Inputs.



Instance Ids: 100

Fig. 40

Producing Instances will also impact the Connections of the Connection Manager object. Each of the connections must have the connection path modified to point to the correct instance(s). The example below lists 0x64 as the producing instance. See *Volume 1: Common Industrial Protocol Specification appendix C* for the encoding of the Connection Path.



Real time transfer format: Modeless format

Connection Type: □ Null ☑ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

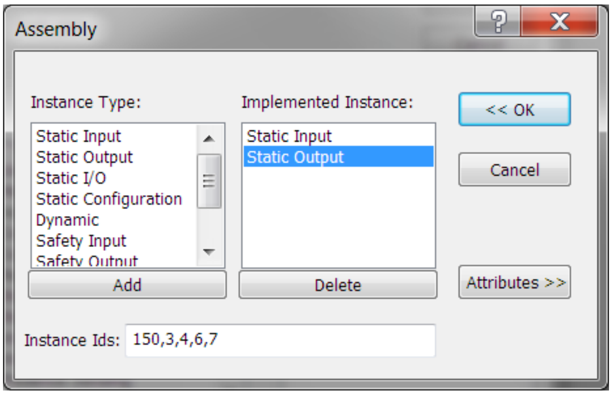
Connection Path: 20 04 24 05 2c 96 2c 64

Fig. 41

This attribute is an array if the host application implements the Assembly Mapping Host object, see details of this object below.

EtherNet/IP Host Object (F8h) - Attribute #8 - Consuming Instance No.

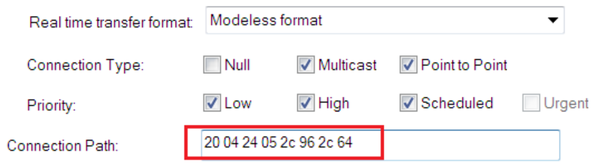
The response field will impact the assembly object. For most applications the producing instance (s) will be Static Outputs.



Instance Ids: 150,3,4,6,7

Fig. 42

Producing Instances will also impact the Connections of the Connection Manager object. Each of the connections must have the connection path modified to point to the correct instance(s). The example below lists 0x96 as the consuming instance. See *Volume 1: Common Industrial Protocol Specification appendix C* for the encoding of the Connection Path.



Real time transfer format: Modeless format

Connection Type: ☐ Null ☑ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled ☐ Urgent

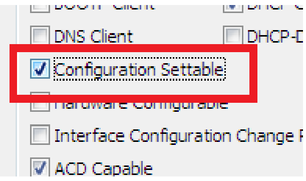
Connection Path: 20 04 24 05 2c 96 2c 64

Fig. 43

This attribute is an array if the host application implements the Assembly Mapping Host object, see details of this object below.

EtherNet/IP Host Object (F8h) - Attribute #9 - Enable Communication Settings from Net

This attribute sets the ability for other devices on the network to adjust the communication settings using access to the CIP TCP/IP object and the CIP Ethernet Link Object. Check the box in the Physical Data section if this method is supported.



☐ DNS Client ☐ DHCP-D

☑ Configuration Settable

☐ Hardware Configurable

☐ Interface Configuration Change F

☑ ACD Capable

Fig. 44

The STC file must be configured to support the ability to set the affected attributes in the TCP/IP Interface object and in the Ethernet Link object according to the table below.

Enable communication settings from net	CIP TCP/IP Object	CIP Ethernet Link Object
True	Set service enabled on attributes 3, 5, 6, 8, 9, 10, 11 and 12.	Attribute 6 enabled. Set service enabled on object level.
False	Set service enabled on attributes 3, 8, 9, 10, 11 and 12. Set service disabled on attributes 5 and 6.	Attribute 6 disabled. Set service disabled on object level (does not apply if the Admin state attribute is implemented). Attribute 3 shall be settable in all modes except when hardware switches are used.

Please note that using the DLR functionality requires that Ethernet ports are able to have forced settings e.g. 10 Mbps Half-duplex. For DLR devices with Communication Setting from the network disabled, the application must provide the ability to force Ethernet settings. For example, the web server can provide the user the ability to force Ethernet settings.

EtherNet/IP Host Object (F8h) - Attribute #11 - Enable CIP Forwarding

Enabling CIP Forwarding allows the host application to respond to all requests to both CIP objects and instances of the Assembly Object not implemented by the Anybus CompactCom. The Conformance Test software will check to see if those requests are handled properly by the application. All Objects and Assembly Object Instances listed in the section about CIP objects in the corresponding EtherNet/IP network appendices, are handled by the Anybus CompactCom. This means that all requests to CIP objects and instances not listed in the CIP objects section need to be handled by responses to the Process_CIP_Request command if CIP Forwarding is enabled.

Enabling CIP Forwarding can be necessary when users support device profiles defined by Volume 1 of the CIP Network Libraries. Additionally, vendors may define and support objects and

Assembly Object Instances that are not specified in the CIP network specification as long as those objects and Assembly Object Instances are in the vendor specific range.

CIP defined device profiles define which object(s) instance(s) attributes(s), and Instances of the Assembly Object need be supported by a device. Additionally, mandatory services and behaviors are defined. Chapter 6 of Volume 1 CIP Network Libraries details device profiles. The default profile supported by the Anybus CompactCom is Generic Device, Keyable. This device profile, and some other profiles do not require any additional objects or assembly instances to be supported making it not necessary to enable CIP forwarding.

Responding properly can mean different things for different requests at different times. The following list gives advice how to reduce the complexity.

- Decide which Object/Instance/attributes combinations will be implemented. Consult specifications to ensure that mandatory/optional/vendor specific combinations are correct.
- Decide which services are supported for the implemented combinations. Consult specification to ensure that mandatory/optional/vendor specific services are implemented properly.
- Verify that proper application behavior is provided for the correct interaction of the implemented services and paths.
- Provide the correct error response for all paths not supported by application.

  CIP status code 0x05 (path destination unknown) will be reported, when the application returns any of the following Anybus CompactCom error codes: Unsupported Object (3), Unsupported Instance (4). Consult the CIP network libraries Vol1 appendix B for status codes, and the section on CIP Objects in the Anybus CompactCom EtherNet/IP Network Guide for a translation of CIP error codes to Anybus CompactCom error codes.
- Provide the correct error response for all unsupported commands.

  CIP status code 0x08 service not supported will be reported when the application returns the Anybus CompactCom error code Unsupported Command (5). Consult the CIP network libraries Vol1 appendix B for status codes, and the CIP objects chapter in the Anybus CompactCom EtherNet/IP Network Guide for a translation of CIP error codes to Anybus CompactCom error codes.
- Provide error checking for all commands that modify variables, and respond with the correct CIP defined error code. Consult the CIP network libraries Vol1 appendix B for status codes, and the CIP Objects chapter in the Anybus CompactCom EtherNet/IP Network Guide for a translation of CIP error codes to Anybus CompactCom error codes.

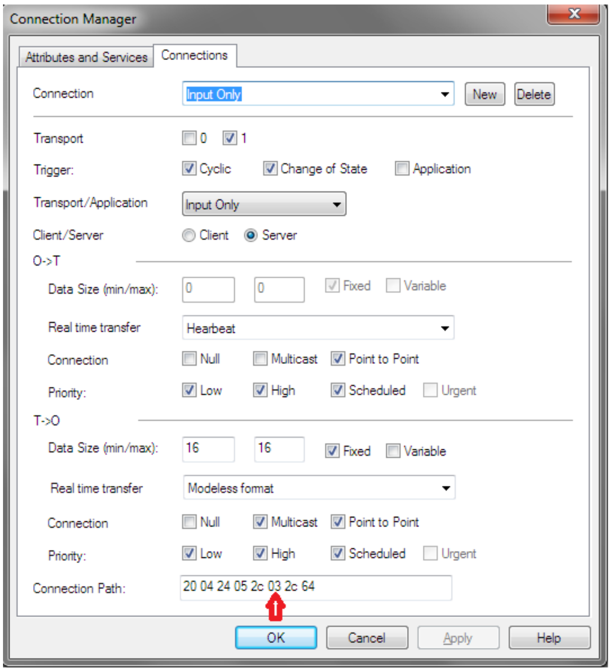
Please consult the profile requirements, services, and behaviors as well as the object definitions specified in the CIP Network Libraries.

EtherNet/IP Host Object (F8h) - Attribute #12 - Enable Parameter Object

The purpose of the CIP Parameter Object is to provide a uniform interface for device configuration. EtherNet/IP requires one instance of the parameter object per configurable parameter. A request to the CIP parameter object is converted into a request to the Host Application Data object. It is possible to disable access to the Parameter Object by responding FALSE to this request. The required and optional instance attributes are listed in table 5A-14.7 of The CIP Network Libraries Volume 1. If the object is disabled the parameter object must be removed from the list of supported objects in the STC file.

EtherNet/IP Host Object (F8h) - Attribute #13 - Input-Only Heartbeat Instance Number

By default this instance number is 3. Changing this value from the default will require users to modify the Input only connection listed in the Connection Manager portion of the .stc file. The following figure shows that for the input only connection 03 shows up as the 0->T connection point for the connection path. For an explanation of the configuration path please see The CIP Networks Library Volume 1 Appendix C.



Connection Manager

Attributes and Services | Connections

Connection: Input Only

Transport: ☐ 0 ☑ 1

Trigger: ☑ Cyclic ☑ Change of State ☐ Application

Transport/Application: Input Only

Client/Server: ○ Client ● Server

O->T

Data Size (min/max): 0 0 ☑ Fixed ☐ Variable

Real time transfer: Hearbeat

Connection: ☐ Null ☐ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled ☐ Urgent

T->O

Data Size (min/max): 16 16 ☑ Fixed ☐ Variable

Real time transfer: Modeless format

Connection: ☐ Null ☑ Multicast ☑ Point to Point

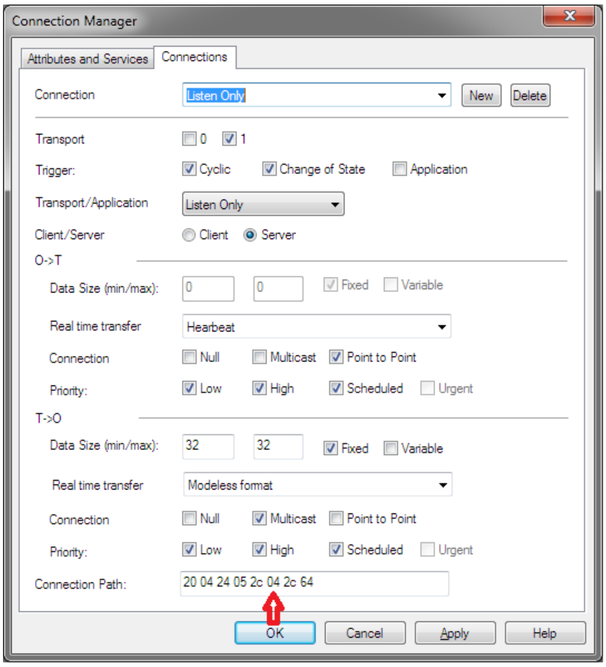
Priority: ☑ Low ☑ High ☑ Scheduled ☐ Urgent

Connection Path: 20 04 24 05 2c 03 2c 64

Fig. 45

EtherNet/IP Host Object (F8h) - Attribute #14 - Listen-Only Heartbeat Instance Number

This attribute will set the Assembly Instance for the Heartbeat connection point (Originator to Target). This instance should be listed as a connection point in the connection manager object for the input only connection. The default instance number is 4.



Connection: Listen Only

Transport: □ 0 ☑ 1

Trigger: ☑ Cyclic ☑ Change of State □ Application

Transport/Application: Listen Only

Client/Server: ○ Client ● Server

O->T

Data Size (min/max): 0 0 ☑ Fixed □ Variable

Real time transfer: Hearbeat

Connection: □ Null □ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

T->O

Data Size (min/max): 32 32 ☑ Fixed □ Variable

Real time transfer: Modeless format

Connection: □ Null ☑ Multicast □ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

Connection Path: 20 04 24 05 2c 04 2c 64

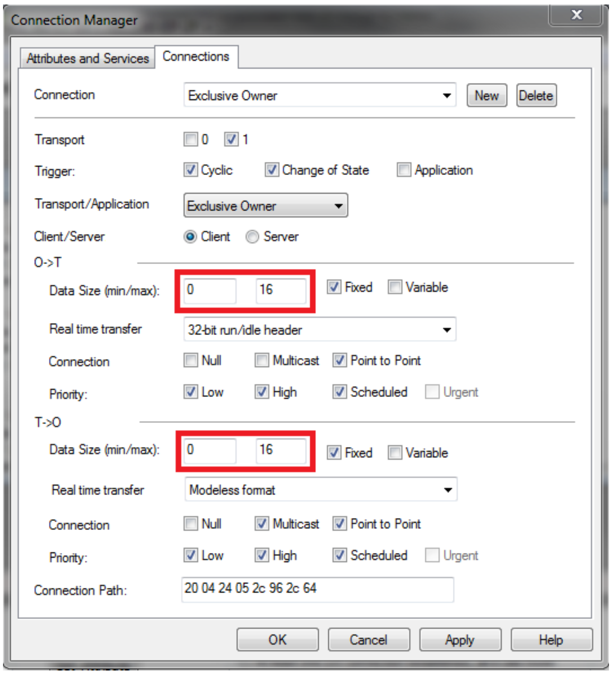
Fig. 46

EtherNet/IP Host Object (F8h) - Attribute 15 - Assembly Object Configuration Instance Number

Device configuration parameters can be grouped together in an assembly instance. By default this Instance is 5. Support for the Configuration instance is provided by the functions Get_ Configuration_Instance and Set_Configuration_Instance of the EtherNet/IP host object. If this instance is used to pass configuration data, this assembly should be listed in the Assembly object as a Static Configuration, and should be listed in the connection paths in the Connection Manager.

EtherNet/IP Host Object (F8h) - Attribute 16 - Disable Strict IO Match

Device configuration parameters can be grouped together in an assembly instance. By default this Instance is 5. Support for the Configuration instance is provided by the functions Get_Configuration_Instance and Set_Configuration_Instance of the EtherNet/IP host object. If this instance is used to pass configuration data, this assembly should be listed in the Assembly object as a Static Configuration, and should be listed in the connection paths in the Connection Manager.



Connection: Exclusive Owner

Transport: □ 0 ☑ 1

Trigger: ☑ Cyclic ☑ Change of State □ Application

Transport/Application: Exclusive Owner

Client/Server: ● Client ○ Server

O->T

Data Size (min/max): 0 16 ☑ Fixed □ Variable

Real time transfer: 32-bit run/idle header

Connection: □ Null □ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

T->O

Data Size (min/max): 0 16 ☑ Fixed □ Variable

Real time transfer: Modeless format

Connection: □ Null ☑ Multicast ☑ Point to Point

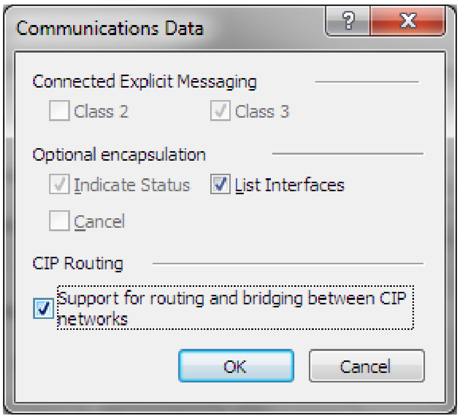
Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

Connection Path: 20 04 24 05 2c 96 2c 64

Fig. 47

EtherNet/IP Host Object (F8h) - Attribute 17 - Enable Unconnected Routing

Enabling this attribute will allow unconnected routing, and allow access to the CIP Port Object (F4h). It is possible for originators to use CIP routing to link to other subnets or backplanes through the Anybus CompactCom. For EtherNet/IP, multiple Port Object Instances can share the single or dual Physical ports. For each CIP routable port, one instance of the CIP Port Object should exists. Enabling this attribute also requires that applications support for Hosts CIP Port Configuration Object (ODh). The Statement of Conformance file can be configured to reflect the ability for port forwarding, by selecting the check box in the Communication data section, and the port object can be added to the list of implemented objects.



Communications Data

Connected Explicit Messaging

□ Class 2 ☑ Class 3

Optional encapsulation

☑ Indicate Status ☑ List Interfaces

□ Cancel

CIP Routing

☑ Support for routing and bridging between CIP networks

Fig. 48

EtherNet/IP Host Object (F8h) - Attribute 18 - Input-Only Extended Heartbeat Instance Number

The extended version of the Input-only Heartbeat connection is functionally the same with one exception. The state of the connection does not affect the state of the module, i.e., a timeout of this connection will not force the module into the Exception state. This Instance can be used for a connection point and that connection should be annotated in the Connection Manager. The instance number should appear in the connection path.

EtherNet/IP Host Object (F8h) - Attribute 19 - Listen-Only Extended Heartbeat Instance Number

The extended version of the Listen-only Heartbeat connection is functionally the same with one exception. The state of the connection does not affect the state of the module, i.e., a timeout of this connection will not force the module into the Exception state. This Instance can be used for a connection point and that connection should be annotated in the Connection Manager. The instance number should appear in the connection path.

EtherNet/IP Host Object (F8h) - Attribute 20 - Interface label port 1

This label is not checked by the CT test software, however if changed please ensure that the EDS file is updated with the equivalent string.

EtherNet/IP Host Object (F8h) - Attribute 21 - Interface label port 2

This label is not checked by the CT test software, however if changed please ensure that the EDS file is updated with the equivalent string.

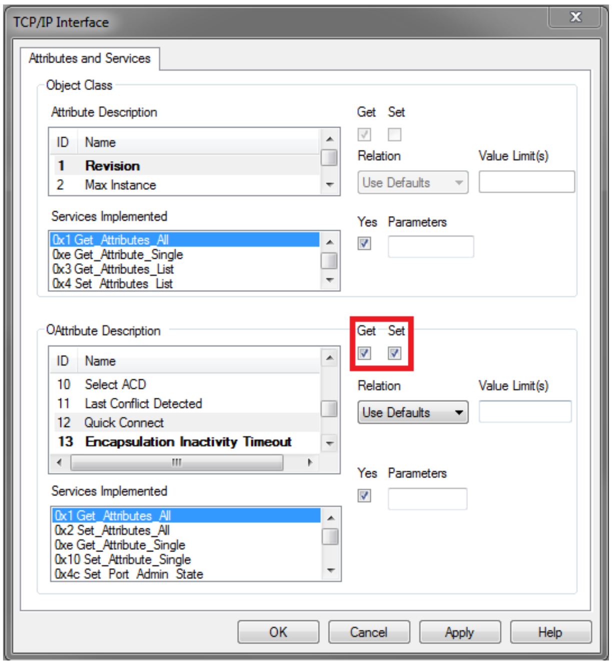
EtherNet/IP Host Object (F8h) - Attribute 22 - Interface label internal port

This label is not checked by the CT test software, however if changed please ensure that the EDS file is updated with the equivalent string.

EtherNet/IP Host Object (F8h) - Attribute 26 - Enable EtherNet/IP QuickConnect

Enabling QuickConnect will make the duration from power-on to available on the network as short as possible. QuickConnect will require a change to the default EDS file, and also require that, for two port modules, ports 1 and 2 labeled externally on the device.

If QuickConnect is enabled, attribute #12 of TCP/IP Interface object needs to be set to Set and Get access in the STC file. If QuickConnect is disabled Get and Set access must be unchecked.



TCP/IP Interface

Attributes and Services

Object Class

Attribute Description

ID	Name
1	**Revision**
2	Max Instance

Get ☑ Set □

Relation: Use Defaults

Value Limit(s):

Services Implemented

- 0x1 Get_Attributes_All
- 0xe Get_Attribute_Single
- 0x3 Get_Attributes_List
- 0x4 Set_Attributes_List

Yes ☑ Parameters:

OAttribute Description

ID	Name
10	Select ACD
11	Last Conflict Detected
12	Quick Connect
13	**Encapsulation Inactivity Timeout**

Get ☑ Set ☑

Relation: Use Defaults

Value Limit(s):

Services Implemented

- 0x1 Get_Attributes_All
- 0x2 Set_Attributes_All
- 0xe Get_Attribute_Single
- 0x10 Set_Attribute_Single
- 0x4c Set_Port_Admin_State

Yes ☑ Parameters:

Fig. 49

EtherNet/IP Host Object (F8h) - Attribute 29 - Ignore Sequence Count Check

Enabling this functionality violates the CIP Network Libraries specifications.

EtherNet/IP Host Object (F8h) - Attribute 30 - ABCC ADI Object Number

The default behavior of the Anybus CompactCom EtherNet/IP family of modules provide access to Instances of the Application Data Objects through the CIP ADI Object (A2h). It is possible to change this Object Class number or disable access altogether. It is important to note that A2h is in the vendor specific range where Vendors are free to implement their own objects. Choosing an object class number outside the vendor specific range should only be done when the device provides the functionality specified by the object, and adheres to the organization of attributes and services set in the CIP Networks Library. The vendor specific range is 64h – C7h and 300h – 4FFh.

EtherNet/IP Host Object (F8h) - Attribute 31 - Enable DLR

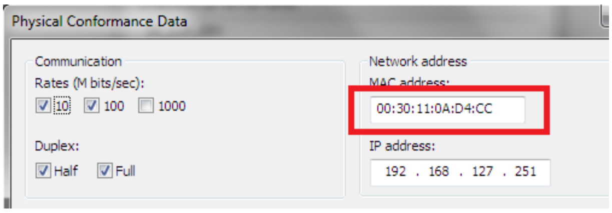
The default behavior of the Anybus CompactCom EtherNet/IP is to have DLR enabled, if for some reason the DLR is disabled the DLR object must be removed from the list of supported objects in the STC file.

Ethernet Host Object (F9h)

Many of the attributes for this object are outside ODVA’s specification and have no bearing on the conformance test, and will not be listed in this document.

Ethernet Host Object (F9h) - Attribute #1 - MAC Address (Also Attribute #8 and #9)

The MAC address should be listed in the Statement of Conformance. This attribute can be accessed in the Physical Data section.



Physical Conformance Data

Communication

Rates (M bits/sec): ☑ 10 ☑ 100 □ 1000

Duplex: ☑ Half ☑ Full

Network address

MAC address: 00:30:11:0A:D4:CC

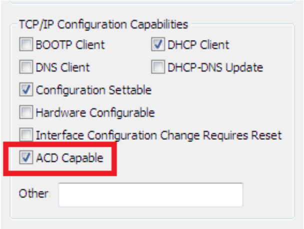
IP address: 192 . 168 . 127 . 251

Fig. 50

Ethernet Host Object (F9h) - Attribute #11 - Enable ACD (Automatic Collision Detection)

The MAC address should be listed in the Statement of Conformance. This attribute can be accessed in the Physical Data section.

Auto Collision Detection is a feature of EtherNet/IP that will detect and mitigate the errors due to multiple devices having the same IP address. This attribute can be accessed in the Physical Data section. Also, there is a section in ODVA’ Conformance Test Details form that indicates if the device is ACD capable.



TCP/IP Configuration Capabilities

□ BOOTP Client ☑ DHCP Client

□ DNS Client □ DHCP-DNS Update

☑ Configuration Settable

□ Hardware Configurable

□ Interface Configuration Change Requires Reset

☑ ACD Capable

Other

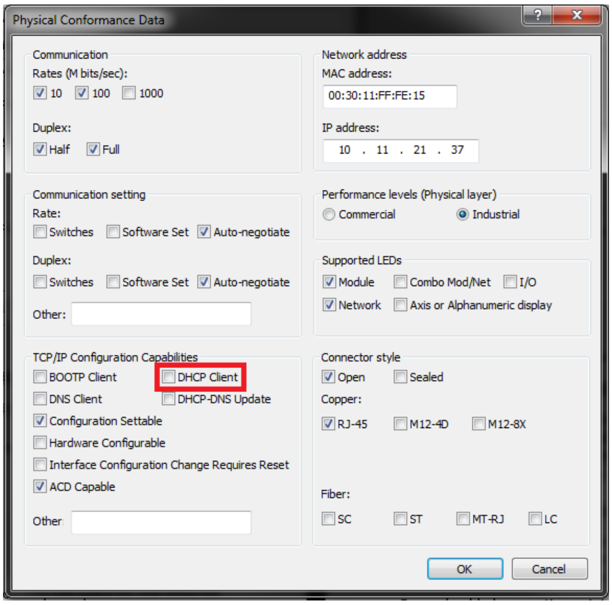
Fig. 51

Ethernet Host Object (F9h) - Attribute 13 - Port 2 State

For Anybus CompactCom B40 and C40 applications port 2 may not be mounted if the application only has space for one ethernet port or for some other reason only need one ethernet port. If this attribute is set to inactive, the DLR object will automatically be disabled, hence the object must be removed from the list of supported objects in the STC file.

Ethernet Host Object (F9h) - Attribute #24 - Enable DHCP Client

If the host application for any reason does not want to support DHCP, this attribute shall be set to False. The DHCP client support in Physical data section of the stc file must be unchecked.



Physical Conformance Data

Communication
Rates (M bits/sec): ☑ 10 ☑ 100 □ 1000
Duplex: ☑ Half ☑ Full

Network address
MAC address: 00:30:11:FF:FE:15
IP address: 10 . 11 . 21 . 37

Communication setting
Rate: □ Switches □ Software Set ☑ Auto-negotiate
Duplex: □ Switches □ Software Set ☑ Auto-negotiate
Other:

Performance levels (Physical layer)
○ Commercial ● Industrial

Supported LEDs
☑ Module □ Combo Mod/Net □ I/O
☑ Network □ Axis or Alphanumeric display

TCP/IP Configuration Capabilities
□ BOOTP Client □ DHCP Client
□ DNS Client □ DHCP-DNS Update
☑ Configuration Settable
□ Hardware Configurable
□ Interface Configuration Change Requires Reset
☑ ACD Capable
Other

Connector style
☑ Open □ Sealed
Copper: ☑ RJ-45 □ M12-4D □ M12-8X
Fiber: □ SC □ ST □ MT-RJ □ LC

Fig. 52

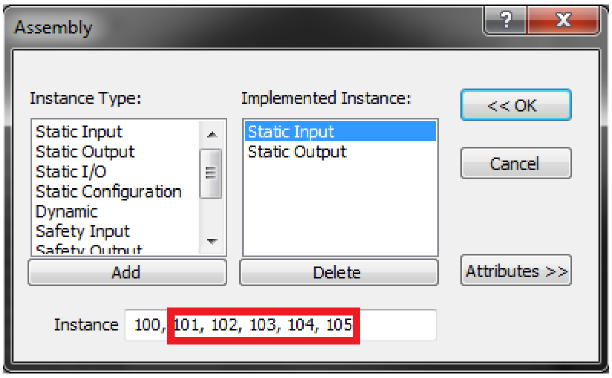
CIP Identity Host Object (EDh)

This object allows devices to support additional instances of the Identity Object (CIP object, 01h) beyond the 1st instance which is supported by default. The support for additional instances of the Identity Object must be reflected in the Statement of Conformance by changing Identity Class attributes Max Instance, and Number of Instances to the proper values.

Assembly Mapping Object (EBh)

The assembly mapping host object can be used to create up to 6 producing and 6 consuming CIP assembly instances. These additional assemblies will also create connection points in the connection manager which will be equivalent to the standard exclusive owner connection in the Anybus CompactCom40 EtherNet/IP.

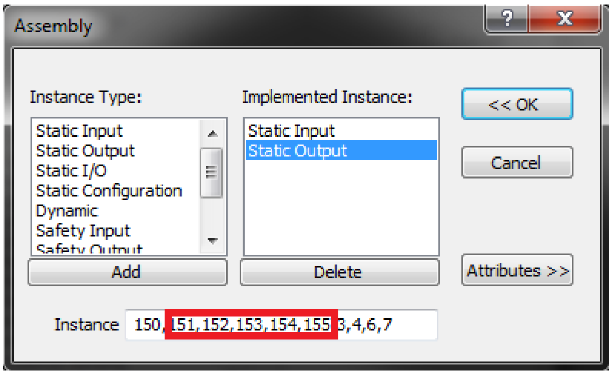
For each added connection point the Assembly object and Connection manager object in the STC file must be updated to describe these new assembly instances.



Instance 100, 101, 102, 103, 104, 105

Fig. 53

In this example five additional producing assemblies has been added.

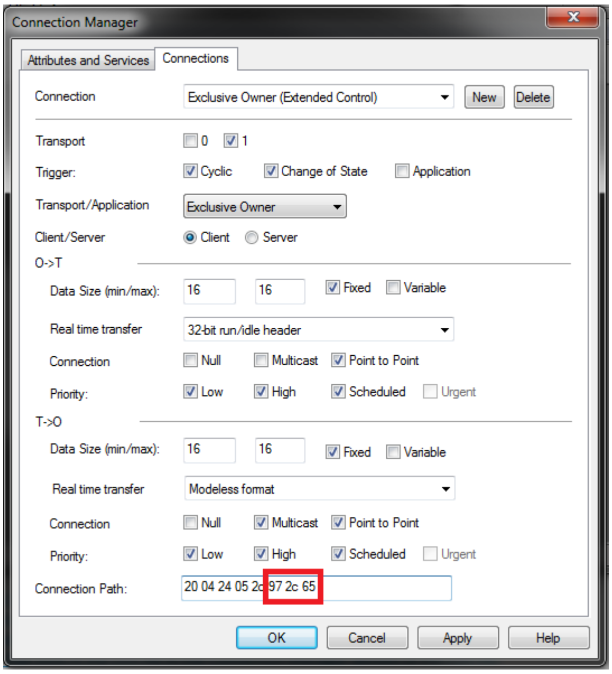


Instance 150, 151,152,153,154,155 3,4,6,7

Fig. 54

In this example five additional consuming assemblies has been added.

It is not required to list all possible combinations of connections between the producing and consuming connection points in the Connection Manager section of the STC file. It is up to the vendor do decide which connection combinations that should be available for the customer in the EDS file, it is however required to list all connections that exist in the EDS file in the Connection Manager section of the EDS file.



Connection: Exclusive Owner (Extended Control)

Transport: □ 0 ☑ 1

Trigger: ☑ Cyclic ☑ Change of State □ Application

Transport/Application: Exclusive Owner

Client/Server: ● Client ○ Server

O->T

Data Size (min/max): 16 16 ☑ Fixed □ Variable

Real time transfer: 32-bit run/idle header

Connection: □ Null □ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

T->O

Data Size (min/max): 16 16 ☑ Fixed □ Variable

Real time transfer: Modeless format

Connection: □ Null ☑ Multicast ☑ Point to Point

Priority: ☑ Low ☑ High ☑ Scheduled □ Urgent

Connection Path: 20 04 24 05 2c 97 2c 65

Fig. 55

This is an example of an additional exclusive owner connection connecting to connection points 101/151.

Please note that it is required to implement the Write_Assembly_Data and Read_Assembly_Data services of the Assembly mapping host object in the application to pass Conformance testing.

G.3.2 Implementation of Anybus Module Objects

Only the Anybus module objects relevant to EtherNet/IP will be discussed.

Network Object (03h)

Network Object (03h) - Attribute #5 - Write Process Data Size

The Write Process Data Size represents the current amount of data mapped for the T-> O connection. In most cases, but not always, this value will correspond to the T->O connection size in the connection manager. Please note that it is possible to support multiple assemblies for connection points by supporting process data remapping.

Network Object (03h) - Attribute #6 - Read Process Data Size

The Read Process Data Size represents the current amount of data mapped for the O->T connection. In most cases, but not always, this value will correspond with the O->T connection size in the connection manager. Please note that it is possible to support multiple assemblies for connection points by supporting process data remapping.

Network Object (03h) - CIP Port Configuration Object (0Dh)

CIP routing can be enabled by instance attribute #17 of the Host EtherNet/IP object. Each instance of this object corresponds to an instance of the CIP Port object (F4h).

Please note that a CIP port does not necessarily correspond to a Physical port. The two network connectors on the two-port Anybus CompactCom correspond to a single CIP routable port. Devices with a single CIP port are not required to support the Port Object, but the Communication Adapter device profile does require support for this object.

H Licensing Information

```
lwIP is licensed under the BSD licence:

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

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

1.   Redistributions of source code must retain the above copyright
notice, this list of conditions  and the following disclaimer.
2.   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.
3.  The name of the author may not be used to endorse or promote
products derived from this software without specific prior written
 permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE AUTHOR 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.

----------------------------------------------------------------------

Print formatting routines

Copyright (C) 2002 Michael Ringgaard. All rights reserved.

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

1.    Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2.    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.
3. Neither the name of the project 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 COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS " AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER 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) 2002 Florian Schulze.
All rights reserved.

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

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the authors nor the names of the contributors may
be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS 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.

ftpd.c - This file is part of the FTP daemon for lwIP

----------------------------------------------------------------------

FatFs - FAT file system module  R0.09b                   (C)ChaN, 2013

FatFs module is a generic FAT file system module for small embedded
systems. This is a free software that opened for education, research and
commercial developments under license policy of following terms.

Copyright (C) 2013, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY. No
restriction on use. You can use, modify and redistribute it for personal,
non-profit or commercial products UNDER YOUR RESPONSIBILITY.
Redistributions of source code must retain the above copyright notice.

----------------------------------------------------------------------

Copyright (c) 2016 The MINIX 3 Project.
All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIEDWARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR 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.

Author: David van Moolenbroek <david@minix3.org>

MD5 routines

Copyright (C) 1999, 2000, 2002 Aladdin Enterprises. All rights reserved.

This software is provided "as-is", without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch

ghost@aladdin.com

Copyright 2013 jQuery Foundation and other contributors

http://jquery.com/

Permission is hereby granted, free of charge, to any person obtaining a copy of this "software" and associated documentation files (the Software), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,FITNESS FOR A PARTICULAR PURPOSE ANDNONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

----------------------------------------------------------------------

rsvp.js

Copyright (c) 2013 Yehuda Katz, Tom Dale, and contributors

Permission is hereby granted, free of charge, to any person obtaining a
copy of this "software" and associated documentation files (the
Software), to deal in the Software without restriction, including without
limitation the rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and to permit persons
to whom the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,FITNESS FOR A PARTICULAR PURPOSE ANDNONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

----------------------------------------------------------------------

libb (big.js)
The MIT Expat Licence.
Copyright (c) 2012 Michael Mclaughlin

Permission is hereby granted, free of charge, to any person obtaining a
copy of this "software" and associated documentation files (the
Software), to deal in the Software without restriction, including without
limitation the rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell copies of the Software, and to permit persons
to whom the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE ANDNONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

----------------------------------------------------------------------

The "inih" library is distributed under the New BSD license:
Copyright (c) 2009, Ben Hoyt
All rights reserved.
```

```
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 Ben Hoyt 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 BEN HOYT "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL BEN HOYT 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.

**********************************************************************

open62541 is licensed under the Mozilla Public License v2.0

This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this file,
You can obtain one at http://mozilla.org/MPL/2.0/.

To obtain customized changes please contact foss@anybus.com.

**********************************************************************

musl as a whole is licensed under the following standard MIT license:

----------------------------------------------------------------------

Copyright ©  2005-2014 Rich Felker, et al.

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject
to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE ANDNONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

----------------------------------------------------------------------
```

****

PCG Random Number Generation for C.

Copyright 2014 Melissa O'Neill

Licensed under the Apache License, Version 2.0 ("the License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

For additional information about the PCG random number generation scheme, including its license and other licensing options, visit

http://www.pcg-random.org
****

queue.h

Copyright (c) 1991, 1993

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

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

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

2. 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.

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

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OFMERCHANTABILITY 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.

@(#)queue.h
8.5 (Berkeley) 8/20/94

Format - lightweight string formatting library.

```
Copyright (C) 2010-2013, Neil Johnson
All rights reserved.

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 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 COPYRIGHT HOLDERS AND CONTRIBUTORS
 "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED.IN NO EVENT SHALL THE
COPYRIGHT OWNER 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 THEORYOF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THISSOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.
```

This page intentionally left blank

Box 4126
300 04 Halmstad, Sweden

info@hms.se