ENGLISH

Anybus® CompactCom™ 40 - PROFINET IRT IIoT Secure

NETWORK GUIDE

SCM-1202-067
Version 2.8
Publication date 2022-02-14

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.

Contact Information
Postal address:
Box 4126
300 04 Halmstad, Sweden

E-Mail: info@hms.se

Table of Contents

This page is 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 PROFINET IRT IIoT Secure. The document describes the features that are specific to Anybus CompactCom 40 PROFINET IRT 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 PROFINET specific functionality is to be used, in-depth knowledge of PROFINET networking internals and/or information from the official PROFINET specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the PROFINET 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
PROFINET IO specification, rev. 2.3	Profibus International	
PROFIenergy Technical Specification, rev. 1.2	Profibus International	
GSDML Technical Specification for PROFINET IO, rev 2.31	Profibus International	
OPC UA Specification 1.04	OPC Foundation	

1.3. Document History

Version	Date	Description
1.6	2017-11-28	First public release
1.7	2017-12-15	Updated Copyright Appendix
1.8	2018-06-18	SYNC functionality description updated Minor corrections
1.9	2018-09-19	Added MQTT functionality
2.2	2018-10-23	Minor updates
2.3	2019-06-10	Rebranding Minor updates
2.4	2019-07-02	Removed reference to Transparent Ethernet (which is not supported by this product) in section 2.1
2.5	2020-05-11	S2 redundancy 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)
2.6	2021-05-24	Added Time Object (13h) Updated OPC UA and MQTT sections Updated SMTP sections
2.7	2021-10-29	Minor updates Added OPC UA application defined information models
2.8	2022-02-14	Added Command Details: Method_Call Minor updates

1.4. Document Conventions

Lists

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

User Interaction Elements

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

Program Code and Scripts

```
Program code and script examples
```

Cross-References and Links

Cross-reference within this document: Document Conventions (page 2)

External link (URL): www.anybus.com

Safety Symbols

DANGER
Instructions that must be followed to avoid an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING
Instructions that must be followed to avoid a potential hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION
Instruction that must be followed to avoid a potential hazardous situation that, if not avoided, could result in minor or moderate injury.

IMPORTANT
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.

Information Symbols

NOTE
Additional information which may facilitate installation and/or operation.

TIP
Helpful advice and suggestions.

1.5. Terminology

- 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. Trademarks

Anybus® is a registered trademark of HMS Networks.

All other trademarks are the property of their respective holders.

2. About the Anybus CompactCom 40 PROFINET IRT IIoT Secure

2.1. General

The Anybus CompactCom 40 PROFINET IRT IIoT Secure communication module provides instant PROFINET Real Time connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

This product conforms to all aspects of the host interface for Active modules defined in the Anybus CompactCom Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to take 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
- Ethernet connectors
- Supports OPC UA technology
- Supports MQTT functionality
- Up to 128 submodules in total
- Up to 32767 ADIs
- Max. read process data: 1308 bytes
- Max. write process data: 1308 bytes
- Max. process data (read + write, in bytes): 2616 bytes
- Generic and PROFINET specific diagnostic support
- Complies with PROFINET IO Conformance class C
- Supports up to 1440 bytes I/O data in each direction, status bytes included
- Supports 250 µs cycle time
- SNMP agent
- Secure Web server w. customizable content
- Secure file transfer server (WebDAV)
- E-mail client
- JSON functionality
- Device identity customization
- Modular Device Functionality
- PROFIenergy profile supported
- GSD file template provided by HMS
- Supports Media Redundancy Protocol (MRP)
- Supports S2 system redundancy
- Supports PROFIsafe

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 PROFINET IRT IIoT Secure supports the protocols OPC UA and MQTT.

See also ...

- OPC UA (page 89)
- MQTT (page 112)

2.4. Security

Anybus CompactCom 40 PROFINET IRT 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 PROFINET IRT 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.

NOTE
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 PROFINET IRT 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 PROFINET IRT IIoT Secure.

IMPORTANT
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 PROFINET IRT 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.

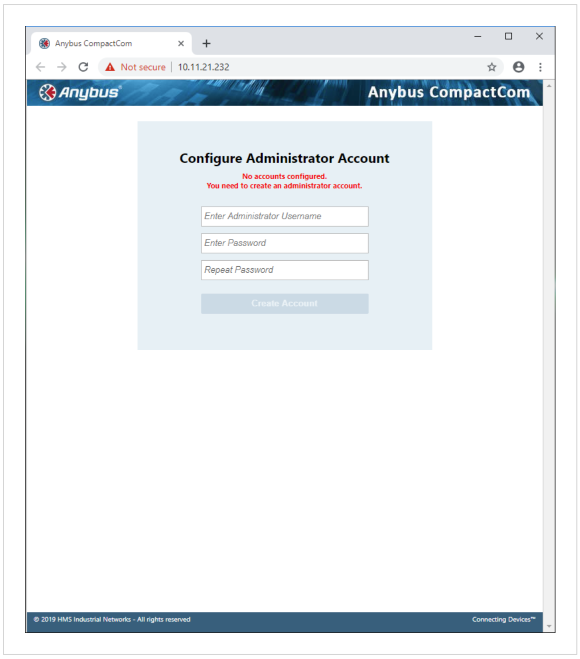
IMPORTANT
It is recommended to perform this initial configuration offline, with the device connected directly to your computer. This to ensure that anyone who is not authorized, will gain access to the device and configure the first account.

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) (page 266). 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 PROFINET IRT 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.


Anybus CompactCom
Not secure | 10.11.21.232
Anybus
Anybus CompactCom

Configure Administrator Account

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

Enter Administrator Username

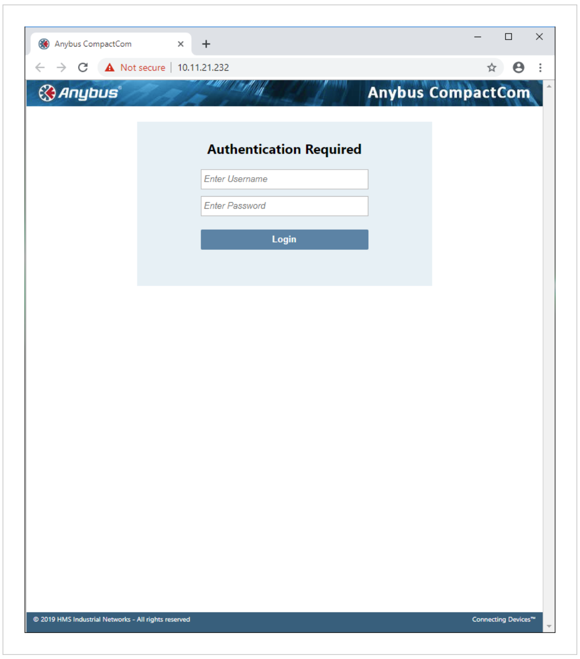
Enter Password

Repeat Password

Create Account
© 2019 HMS Industrial Networks - All rights reserved
Connecting Devices™

Figure 1.

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


Anybus CompactCom
Not secure | 10.11.21.232
Anybus
Anybus CompactCom

Authentication Required

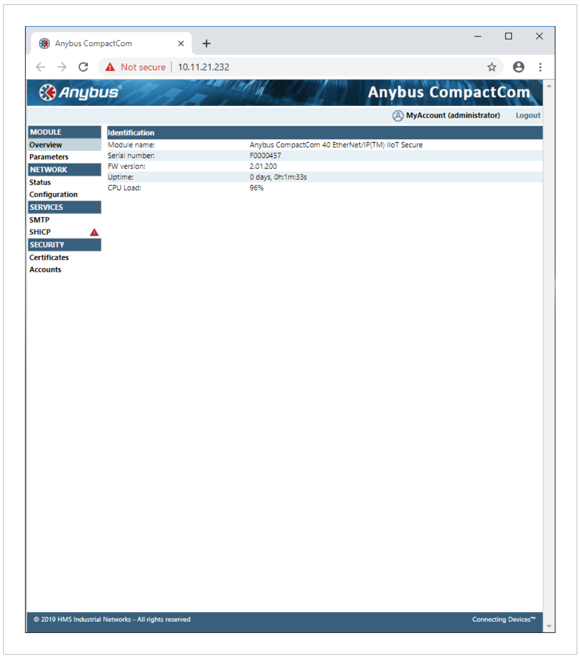
Enter Username

Enter Password

Login
© 2019 HMS Industrial Networks - All rights reserved
Connecting Devices™

Figure 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%

© 2019 HMS Industrial Networks - All rights reserved

Figure 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 (page 268) 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.

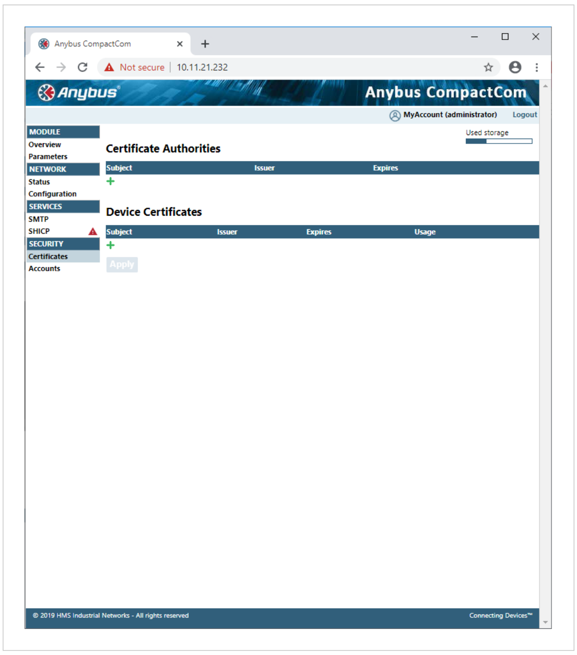
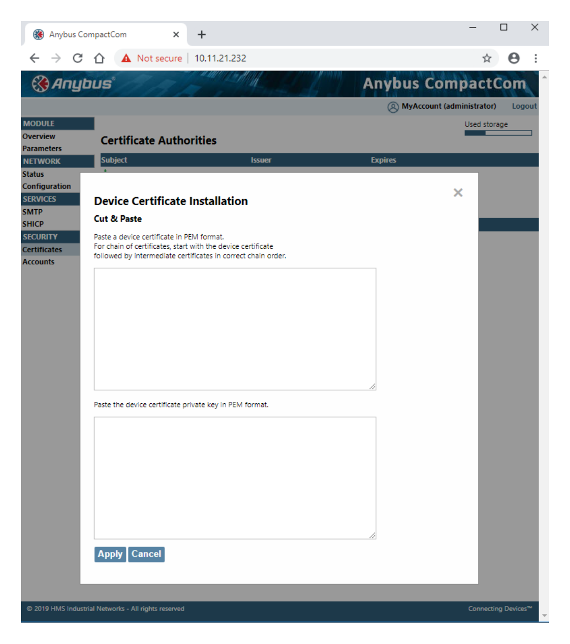


Figure 4.

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

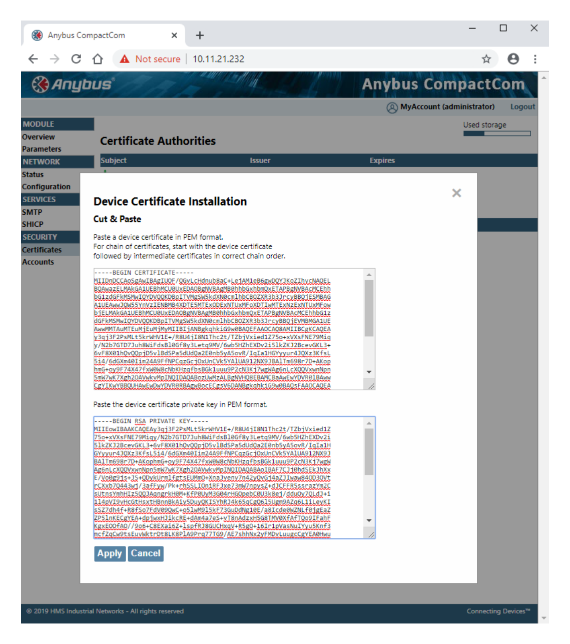
Paste a device certificate in PEM format.
For chain of certificates, start with the device certificate
followed by intermediate certificates in correct chain order.

Paste the device certificate private key in PEM format.

Apply Cancel

Figure 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-----
MIIDnDCCAoSgAwIBAgIUOF/QGvLcHdnub8aC+LejAM1eB6gwDQYJKoZIhvcNAQEL
BQAwazELMAkGA1UEBhMCU0UxEDAOBgNVBAgMB0hhbGxhbmQxETAPBgNVBAcMCEhh
bG1zdGFkMSMwIQYDVQQKDBpITVMgSW5kdXN0cmlhbCBOZXR3b3JrcyBBQjESMBAG
A1UEAwwJQW55YnVzIENBMB4XDTE5MTExODExNTUxMFoXDTIwMTExNzExNTUxMFow
bjELMAkGA1UEBhMCU0UxEDAOBgNVBAgMB0hhbGxhbmQxETAPBgNVBAcMCEhhbG1z
dGFkMSMwIQYDVQQKDBpITVMgSW5kdXN0cmlhbCBOZXR3b3JrcyBBQjEVMBMGA1UE
AwwMMTAuMTEuMjEuMjMyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
y3qj3F2PsMLt5krWHV1E+/R8U4jI8N1Thc2t/TZbjVxied1Z75o+xVXsFNE79Miq
y/N2b7GTD7Juh8WiFdsBl0Gf8y3Letq9MV/6wb5HZhEXDv2i5lkZKJ2BcevGKL3+
6vF8X01hQvQQpjD5vlBdSPa5dUdQa2E0nb5yA5ovR/IqIa1HGYyyur4JQXz3KfsL
5i4/6dGXm40Iim24A9FfNPCqzGcjOxUnCVk5YAlUA912NX9JBAlTm698r7D+AKop
hmG+oy9F74X47fxW0W8cNbKHzqfbsBGk1uuu9P2cN3Kj7wgWAg6nLcXQQVxwnNpn
5mW7wK7Xgh2OAVwkvMpINQIDAQABozUwMzALBgNVHQ8EBAMCBaAwEwYDVR0lBAww
CgYIKwYBBQUHAwEwDwYDVR0RBAgwBocECgsV6DANBgkqhkiG9w0BAQsFAAOCAQEA
```

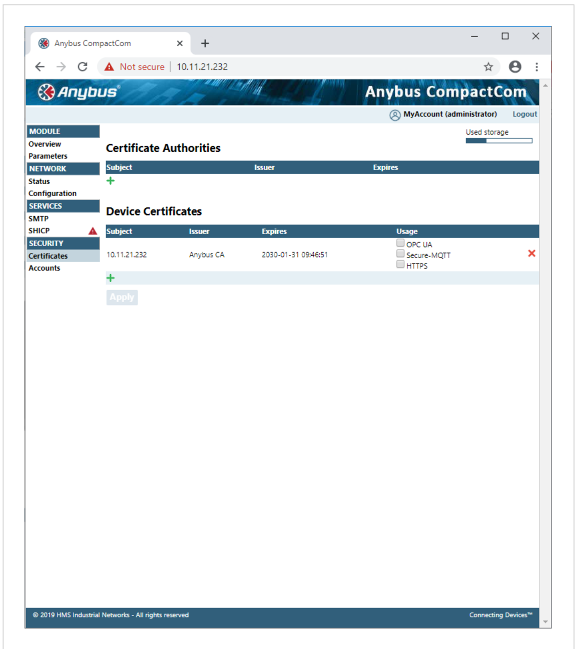
Paste the device certificate private key in PEM format.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAy3qj3F2PsMLt5krWHV1E+/R8U4jI8N1Thc2t/TZbjVxied1Z
75o+xVXsFNE79Miqy/N2b7GTD7Juh8WiFdsBl0Gf8y3Letq9MV/6wb5HZhEXDv2i
5lkZKJ2BcevGKL3+6vF8X01hQvQQpjD5vlBdSPa5dUdQa2E0nb5yA5ovR/IqIa1H
GYyyur4JQXz3KfsL5i4/6dGXm40Iim24A9FfNPCqzGcjOxUnCVk5YAlUA912NX9J
BAlTm698r7D+AKophmG+oy9F74X47fxW0W8cNbKHzqfbsBGk1uuu9P2cN3Kj7wgW
Ag6nLcXQQVxwnNpn5mW7wK7Xgh2OAVwkvMpINQIDAQABAoIBAF7CJj0hdSEkJhXx
E/Vo0g9js+JS+QDykUrmlfgtsEUMmO+XnaJvenv7n42yQvGj4aZJIwaw84OD3OVt
rCXxb7Q443wj/3affyw/Pk+rhSSLIOniRFJxe73mW7npysZ+dJCFFR5ssrazYm2C
sUtnsYmhHIz5QQJAqngrkH0M+KfP0UyM3G04rHGOpebC0U3k8ej/dduOy7QLdJ+i
1l4pVI9vHcGtHsxtHBnnBkAiy5DuyQKISYhRJ4k65qCgQ6l5Ugm9AZq6L1iLeyKI
sSZ7dh4f+R8fSo7FdV09QwC+o5lwM9l5kF73GuDdNg10E/a8Icde0WZNLf0jgEaZ
ZP5lnKECgYEA+dpjwxHJikcRE+dAm4a7eS+vT8nAdzxH5G8TMV0XfAfTQo9IFahF
KgxEOOfAO//9o6+C8EXai6Z+lspfRJ8GUCHxqV+R5gO+16Ir1pVasNuIYyu5Knf3
mcfZqCw9tsEuvWktrDt8LK8PlA9Prq77TG9/AE7shhNx2yFMDvLuugcCgYEA0Hwu
```

Apply Cancel

Figure 6.

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



Certificate Authorities

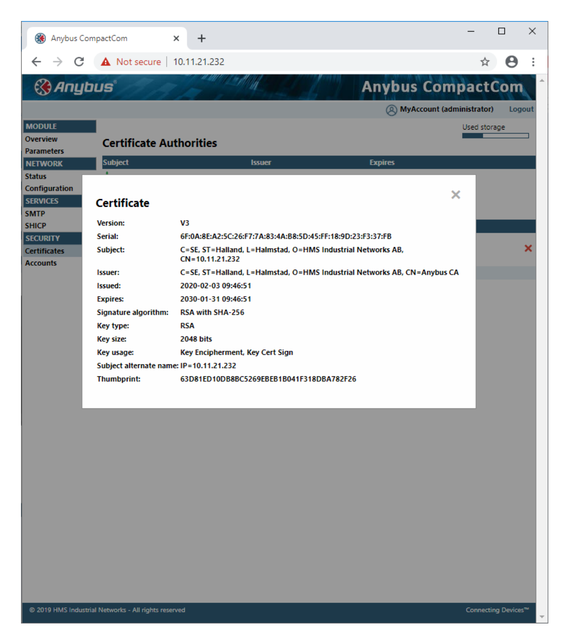
Subject	Issuer	Expires

Device Certificates

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

Figure 7.

Click on the device certificate to view the certificate information.



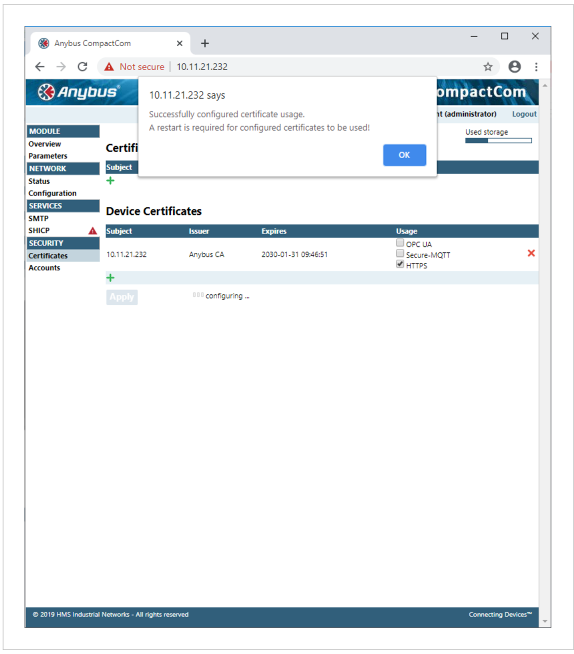
Certificate

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

Figure 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

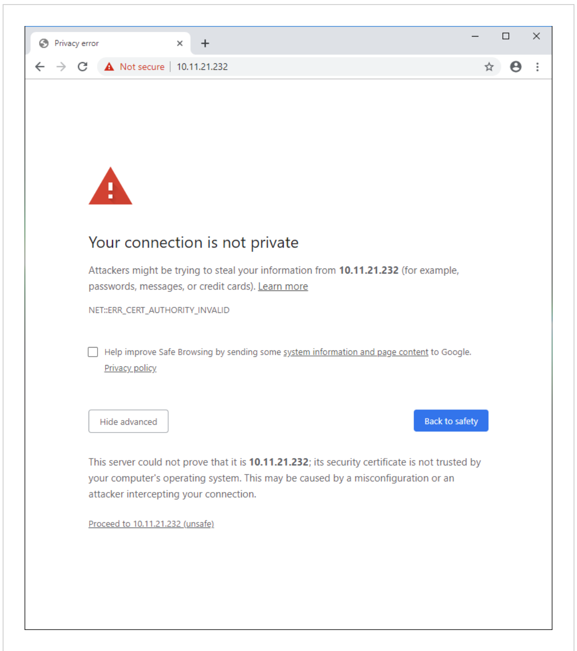
Successfully configured certificate usage.
A restart is required for configured certificates to be used!

Device Certificates

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

configuring ...

Figure 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.

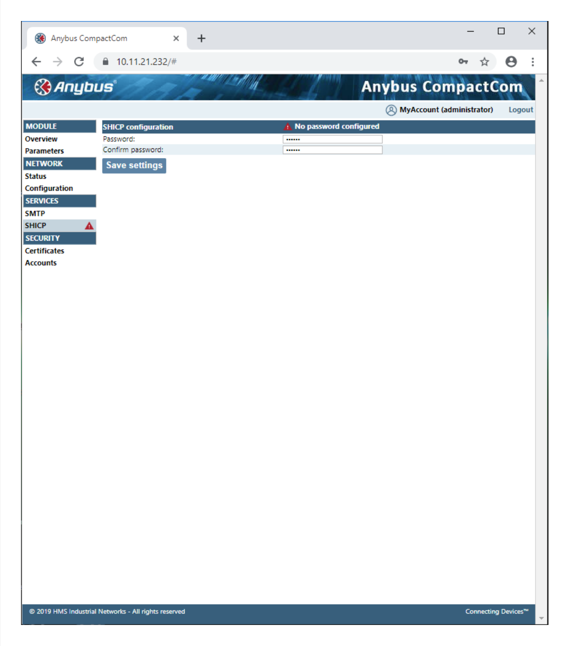
Proceed to 10.11.21.232 (unsafe)

Figure 10.

3.4. Protect the IP Configuration

The Anybus CompactCom 40 PROFINET IRT 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 PROFINET IRT 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

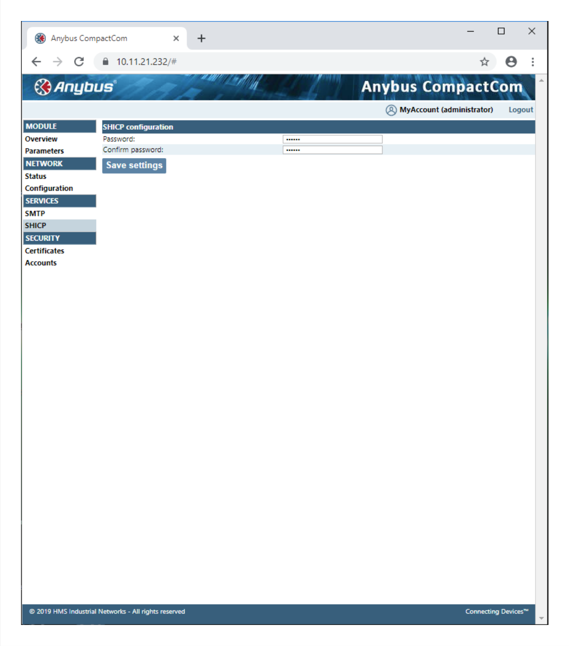
Password: ••••••

Confirm password: ••••••

Figure 11.

IMPORTANT

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: ••••••

Figure 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.

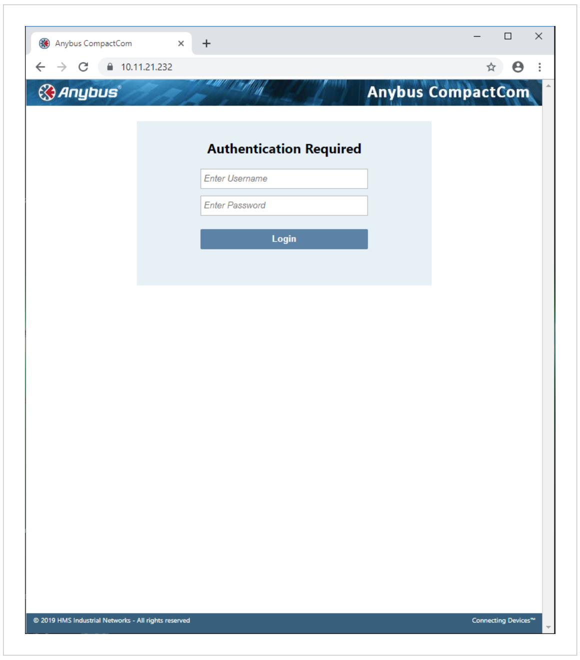
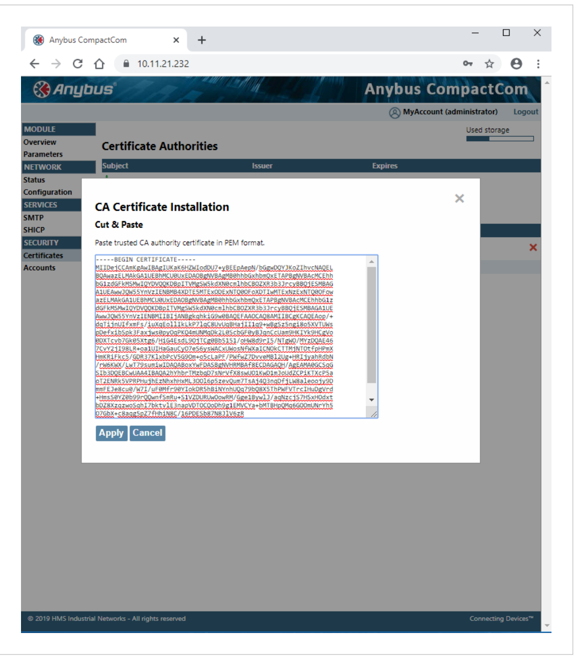


Figure 13.

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



CA Certificate Installation

Cut & Paste

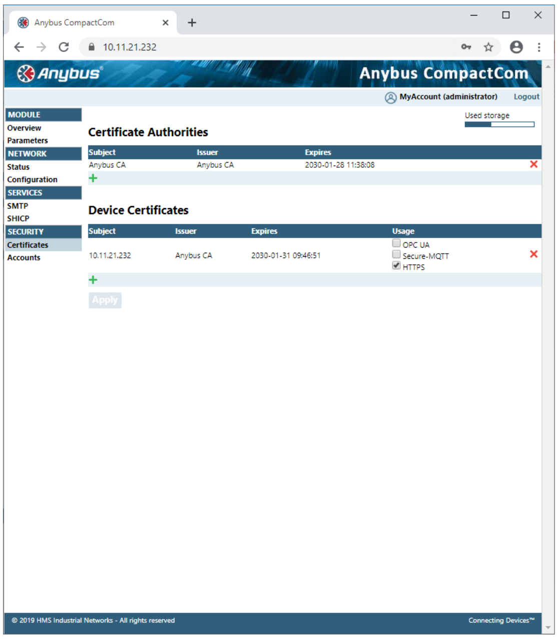
Paste trusted CA authority certificate in PEM format.

```
-----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
```

Figure 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

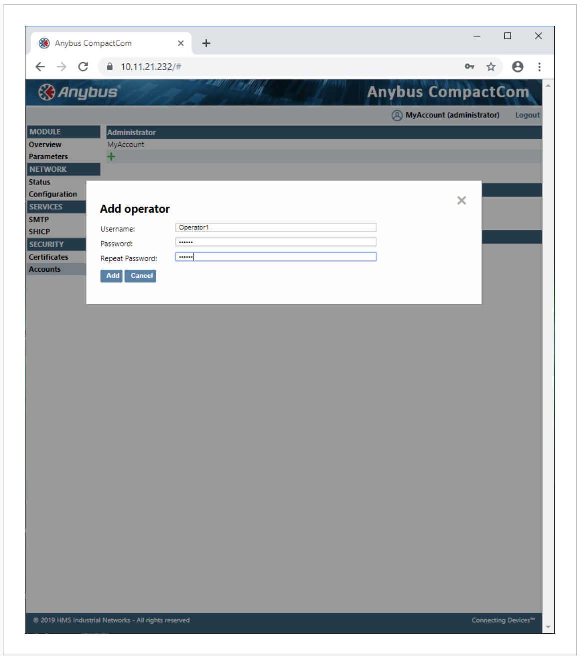
Subject	Issuer	Expires	Usage	
10.11.21.232	Anybus CA	2030-01-31 09:46:51	□ OPC UA □ Secure-MQTT ☑ HTTPS	✕

Figure 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

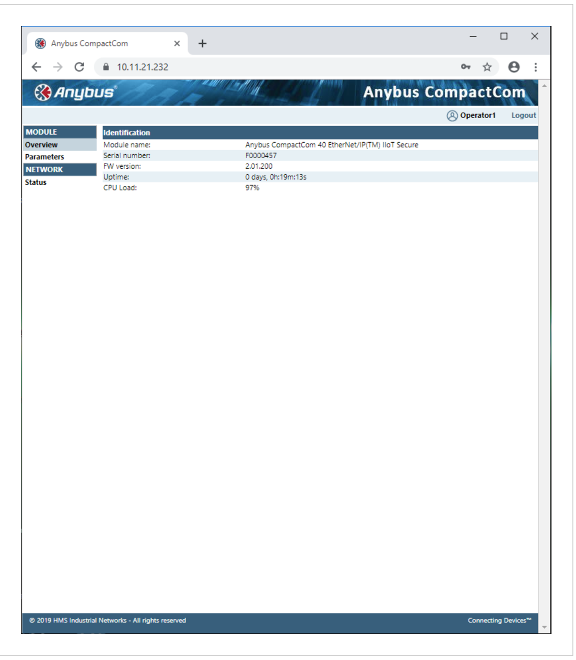
Username: Operator1

Password: ••••••

Repeat Password: ••••••

Figure 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 (page 21) 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%

Figure 17.

4. Basic Operation

4.1. General Information

4.1.1. Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 40 PROFINET IRT IIoT Secure, however due to the nature of the PROFINET networking system certain things must be taken into account:

- Up to 32767 ADIs can be represented on PROFINET.
- ADI names, types and similar attributes cannot be accessed via PROFINET. They are however represented on the network through the built in web server.
- Up to 5 diagnostic instances can be created by the host application. An additional 6th instance may be created in event of a major fault.
- For conformance reasons, the host application must implement support for network reset types 00h (Power-on) and 02h (Power-on + Factory Default) in the Application Object (FFh).
- PROFINET in itself does not impose any particular 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 (exactly what this means in practice depends on the implementation and the actual installation).
- The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). In case of a configuration mismatch, see Configuration Mismatch (page 31) for more information.

See also ...

- Application Data Instances (ADIs) (page 23)
- Diagnostic Object, Anybus Module Objects (page 120)
- Anybus CompactCom 40 Software Design Guide, Application Data Object (FEh)

4.1.2. General Station Description (GSD)

On PROFINET, the characteristics of a device is stored in an XML data file. This file, referred to as the “GSD” file, is used by PROFINET engineering tools when setting up the network configuration. HMS Networks provides an example GSD file, which must be adapted by the user to suit the application.

NOTE
A GSD checker tool and a GSD upgrade tool is available to PI members on the PI website.

Setting Identity and Function Information

The GSD file must be adapted to your implementation. First thing is the device identity.

In the GSD file there is a section called “DeviceIdentity”. It looks like this.

```
<DeviceIdentity VendorID="0x010C" DeviceID="0x0010">
      <InfoText TextId="T_ID_DEV_DESCRIPTION"/>
     <VendorName Value="HMS Industrial Networks"/>
</DeviceIdentity>
```

The identity in the example represents HMS values.

- Replace VendorID value 0x010C with the value which corresponds to your vendor name.
  If you do not have a Vendor ID you can obtain this by contacting PI.
  The VendorID must match what is configured for the PROFINET IO Object (0xF6) attribute #2 (Vendor ID).
- Replace the DeviceID value 0x0010 with the value you have selected for this device.
  The DeviceID must match what is configured for the PROFINET IO Object (0xF6) attribute #1 (Device ID). Please note that if you change the Device ID you MUST also change the Vendor ID, as the Device ID is unique for the Vendor ID.
- Replace the VendorName value “HMS Industrial Networks” with your vendor name.
  Please note that the keyword VendorName is found not only here, but also at other places in the GSD file. Use “Search”to find all instances and replace them with your vendor name.

Specify the function of the device.

```
<DeviceFunction>
      <Family MainFamily="General" ProductFamily="Anybus CompactCom 40
PIR"/>
</DeviceFunction>
```

The example GSD specifies a kind of “General” device as the usage of it is unclear.

- Replace the MainFamily with the class that best describes the device. The following are the allowed values:
  – General
  – Drives
  – Switching Devices
  – I/O
  – Valves
  – Controllers
  – HMI
  – Encoders
  – NC/RC
  – Gateway
  – PLCs
  – Ident Systems
  – PA Profiles
  – Network Components
  – Sensors
- Replace the ProductFamily value “Anybus CompactCom 40 PIR” with a string which describes your device.

In addition to the above, there are a few more places where identity related information is present in the GSD file.

```
DNS_CompatibleName="CompactCom-40-PIR"
```

For the Device access point (DeviceAccessPointItem) there is a keyword which is called DNS_CompatibleName that may be used by engineering tools to build the Station Name. Locate this by using the search function.

- Replace DNS_CompatibleName value with the default name of the device.

The order number of the device is set with the keyword “OrderNumber”.

```
<OrderNumber Value="CompactCom 40 PIR"/>
```

In many cases the value of the OrderNumber equals the Station Type string, but it does not necessarily need to be that way.

- Replace the OrderNumber value “CompactCom 40 PIR” with the order number used for the device.
  The OrderNumber must match what is configured for the PROFINET IO Object (0xF6) attribute #8 (IM Order ID).

It is recommended to change the name of the DAP (keyword Name in the “ModuleInfo” section of the DAP) to something more descriptive. The DAP name is often visible in the PROFINET configuration tool.

- Replace the name “DAP” of the T_ID_DAP text ID with e.g. the name of the product type.

How to Enable Initial Record Data

During the establishment of an IO connection between the IO device and the IO controller it is possible for the IO controller to send initial record data. This initial record data is sent using the PROFINET IO service record write. This service can be used at any time and will write data to a defined ADI. The initial record data is defined in the GSD file, and is specified for a submodule of a module. By default, the Anybus CompactCom 40 module will not make use of any initial record data, but that can be enabled if needed.

To enable this functionality, the GSD file needs to be modified as specified below. In this example, 2 bytes are written to ADI 67 (ADI 67 corresponds to index 67) during startup of a PROFINET IO connection (the value can be configured by the end user):

```
<RecordDataList>
   <ParameterRecordDataItem Index="67" Length="2" TransferSequence="0">
      <Name TextId="T_ID_EXAMPLE2"/>
      <Ref DataType="Unsigned16" ByteOffset="0" DefaultValue=”0”
         AllowedValues="0..65535" TextId="T_ID_EXAMPLE2_PRM_1"/>
   </ParameterRecordDataItem>
</RecordDataList>
```

It is recommended that the above GSD information is placed directly after the “</IOData>” keyword for the module for which the data is associated.

If more than one ADI needs to be set, the keyword “ParameterRecordDataItem”is duplicated.

Please note that TextId’s (“T_ID_xxx” above) need to be added to the “<ExternalTextList>” section of the GSD file (once for each language defined).

```
<Text TextId="T_ID_EXAMPLE2" Value="Config parameter 1"/>
<Text TextId="T_ID_EXAMPLE2_PRM_1" Value="Parameter value description"/>
```

4.2. Network Identity

By default, the module identifies itself as a generic Anybus implementation as follows:

Device ID	0010h (Anybus CompactCom 40 PROFINET)
Vendor ID	010Ch (HMS Industrial Networks)
Station Type	The Station Type will be assigned a value according to this priority: If the PROFINET IO Object (F6h) is implemented, the value equals that of attribute #3 in instance #1. If the PROFINET IO Object is not implemented, the value of attribute #9 (Product name) in the Application Object (FFh), instance #1, will be used (if implemented). If none of these attributes are implemented, the value equals Anybus CompactCom default value (“CompactCom 40 PIR”).

It is recommended to customize the identity information so that the Anybus module appears as a vendor specific implementation rather than a generic Anybus product.

See also...

- Identification & Maintenance (I&M) (page 34)
- PROFINET IO Object (F6h) (page 228)
- Application Object (FFh) in Anybus CompactCom Software Design Guide

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:

Ethernet Interface Settings	The module is locked to 100 Mbit full duplex operation as required by PROFINET.
IP Settings	These settings must be set in order for the module to be able to participate on the network. Normally set by the IO Controller.
SMTP Account Settings	These settings must be set in order for the module to be able to send e-mail messages.
PROFINET Station Name	The module needs to be assigned a Station Name in order to participate on PROFINET. Normally set from the network.

See also...

- Identification & Maintenance (I&M) (page 34)
- Secure Web Server (HTTPS) (page 43)
- Network Configuration Object (04h) (page 129)
- Secure HICP (Secure Host IP Configuration Protocol) (page 266)

4.5. Network Data Exchange

4.5.1. Application Data Instances (ADIs)

ADIs can be accessed acyclically from the network by means of Record Data read/write services. If addressed through a given API and Index range, the module translates the service into standard object requests towards the Application Data Object. If the host application responds with an error to such a request, that error will be translated to PROFINET standard.

The following parameters affect the addressing of ADIs on PROFINET:

Application Process Instance (API)

API 0 (zero) provides access to data in the Application Data Object, i.e. the ADIs. Acyclic requests towards other APIs will be forwarded to the PROFINET IO Object (F6h) (page 228) by means of the 'Get_Record' and 'Set_Record'-commands.

The remainder of this section assumes API 0 (zero).

Slot & subslot

The Slot and subslot values have no impact on the actual addressing of ADIs, except that the actual Slot and subslot need to be populated with a module/submodule. This is always true for the DAP (Device Access Point), which occupies Slot #0/subslot #1. Other Slot/subslot values can also be used provided that the implementation populates it with a module/submodule.

Index

There is a 1:1 correlation between ADI and index as long as the index number is less than - or equal to - 7FFFh. Index 0 (zero) is not associated with an ADI and cannot be used.

API	Slot	Subslot	Index	ADI	Comments
0	0	1	0000h	-	(not associated with ADIs)
			0001h	1	Device Access Point (DAP)
			0002h	2	
			...	...	
			7FFFh	32767	
			8000h...FFFFh	-	(not associated with ADIs)
	X (>0)	Y	0000h		
			0001h	1	Conditional; X and Y must be populated
			0002h	2	
			...	...	
			7FFFh	32767	
			8000h...FFFFh	-	(not associated with ADIs)
>0	-	-	-	-	See Application Process Instances (API) (page 27)

See also...

- Caveats (page 24)
- Application Process Instances (API) (page 27)
- PROFINET IO Object (F6h) (page 228)

4.5.2. Process Data

Mapping an ADI to Write Process Data results in PROFINET input data, and mapping an ADI to Read Process Data results in PROFINET output data. Consistency over all I/O data mapped on PROFINET is guaranteed.

See also...

- Real Identification (RI) (page 30)

IMPORTANT

The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs).

4.5.3. Caveats

The length parameter in the Record Data request specifies the number of bytes to read/write.

- When reading more data than the actual size of the ADI, the response will only contain the actual ADI data, i.e. no padding on the data is performed by the module.
- When writing to an ADI, the length parameter is not checked by the module, i.e. the host application must respond with an error if the length differs from the actual size of the requested ADI.

See also..

- Application Process Instances (API) (page 27)

4.6. 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 (page 25)
- File Transfer Protocol (WebDAV) (page 81)
- Secure Web Server (HTTPS) (page 43)
- JSON (page 54)

4.7. 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 (page 25)
- E-mail Client (page 85)
- JSON (page 54)
- SMTP Client Object (09h) (page 160)

4.8. 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).

4.8.1. 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.

4.8.2. 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) (page 81)
- Secure Web Server (HTTPS) (page 43)
- E-mail Client (page 85)
- Anybus File System Interface Object (0Ah) (page 163)
- Application File System Interface Object (EAh) (page 216)

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

The following operations will erase one or more flash segments:

- Deleting, moving or renaming a file or directory
- Writing or appending data to an existing file
- Formatting the file system

4.8.3. 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. PROFINET Implementation Details

5.1. General Information

This chapter covers PROFINET specific details in the Anybus implementation. Note that the use of such functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specification. 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 GSD file must be changed to reflect all changes.

Implementation overview:

Conformance Class	The Anybus module complies to conformance class C.
Performance Characteristics	• 100 Mbps, full duplex with autonegotiation enabled as default • Real Time (RT) communication, 250 µs cycle time • Isochronous Real Time (IRT) communication, 250 µs cycle time
Device Model	• One IO Device instance • The IO Device instance includes an Application Process referenced by its identifier (API). API 0 (zero) is implemented by default. • The API implements one or more slots • Each Slot implements one or more subslots • Each subslot may implement one or more Channels
Slots & Subslots	Up to 128 subslots in total.
IO Data	1440 bytes of IO data in each direction, including status bytes (4 bytes for DAP submodules + 1 byte per additional submodule)

See also...

- General Station Description (GSD) (page 18)

5.2. Application Process Instances (API)

As mentioned previously, acyclic requests towards API 0 are forwarded to the Application Data Object.

Cyclic data exchange is by default carried out through API 0 (i.e. the Anybus associates modules and submodules with API 0).

See also...

- Application Data Instances (ADIs) (page 23)

5.3. Application Relationships (AR)

On PROFINET, a connection between an IO Controller/Supervisor and an I/O device (in this case the Anybus) is called Application Relationship (AR). The Anybus module supports multiple simultaneous Application Relationships, allowing multiple IO Supervisors to access its data and functions.

The host implementation can either ignore this functionality altogether, in which case the Anybus module will handle it automatically, or integrate the establishment and handling of Application Relationships into the host firmware.

Application Relationships are managed through the following functions:

- AR_Check_Ind (for command details see PROFINET IO Object (F6h) (page 228))
- Expected_Ident_Ind (for command details see PROFINET IO Object (F6h) (page 228))
- AR_Abort_Ind (for command details see PROFINET IO Object (F6h) (page 228))
- AR_Abort (for command details see Network PROFINET IO Object (0Eh) (page 167))

5.4. PROFINET System Redundancy S2

System redundancy relies on the concept of multiple connections to a device to maintain the system in the event of a failure. System redundancy for PROFINET is available in different setups. The S2 functionality, implemented in the Anybus CompactCom 40 PROFINET IRT IIoT Secure, makes it possible for two PROFINET controllers, one primary and one backup, to connect to the device. If the primary controller fails, the backup controller instantly takes over the communication without any disturbance on the network. Running S2 requires specific controllers supporting this feature.

S2 redundancy is enabled in by setting instance #1, attribute #28, in the PROFINET IO Object (F6h), for details see PROFINET IO Object (F6h) (page 228).

5.4.1. Activating S2 System Redundancy in the GSD File

For each DAP, the following additions must be made to an S2 compliant configuration in the GSD file:

- SystemRedundancy element attributes according to:

```
<SystemRedundancy DeviceType=”S2” MaxSwitchOverTime=”45”
S2MaxInputOnBackupDelay=”0” RT_InputOnBackupAR_Supported=”true”
NumberOfAR_Sets=”1”/>
```

- The attribute PrmBeginPrmEndSequenceSupported according to:

```
<DeviceAccessPointItem ... PrmBeginPrmEndSequenceSupported=”true”/>
```

- The attribute PDEV_CombinedObjectSupported according to:

```
<InterfaceSubmoduleItem ... PDEV_CombinedObjectSupported=”true”/>
```

- The attribute NumberOfAR (equal two times NumberOfAR_Sets) according to:

```
<ApplicationRelations ... NumberOfAR=”2”/>
```

For S2 System Redundancy, MinDeviceInterval has been set to "32" in the GSD file, equivalent to a minimum cycle time of 1 ms. This has been done to account for the potentially big load of operating two I/O AR:s.

```
<DeviceAccessPointItem ... MinDeviceInterval="32" .../>
```

Depending on the process data size and layout exchanged on the network in a specific application, it may be possible to set MinDeviceInterval to a lower value (but not lower than 8).

5.5. Real Identification (RI)

5.5.1. General Information

During the establishment of an IO Connection towards the Anybus CompactCom 40 PROFINET IRT IIoT Secure, the configuration derived from the IO Controller (i.e. the Expected Identification) and the actual configuration in the Anybus CompactCom 40 PROFINET IRT IIoT Secure (i.e. the Real Identification or RI) are compared.

The RI configuration is either handled by the module (default), or by the host application. In either case the GSD file has to be customized to correspond to the configuration.

ADI Based Configuration (Default)

By default (i.e. if the application does not issue API_Add, Plug_Module, Plug_Submodule), the Anybus CompactCom 40 PROFINET IRT IIoT Secure handles the plugging of modules and submodules automatically in accordance with the mapped Process Data as follows:

- A DAP is plugged into Slot 0 (zero)
- Modules are added in consecutive order (based on the order of the mapping commands)
- All modules belong to API 0 (zero)

The Anybus CompactCom 40 PROFINET IRT IIoT Secure internally creates module/submodule identifiers as described in the picture and the example below. The GSD file has to be customized to define the same modules/ submodules with the same identifiers as the Anybus CompactCom 40 PROFINET IRT IIoT Secure has created internally.

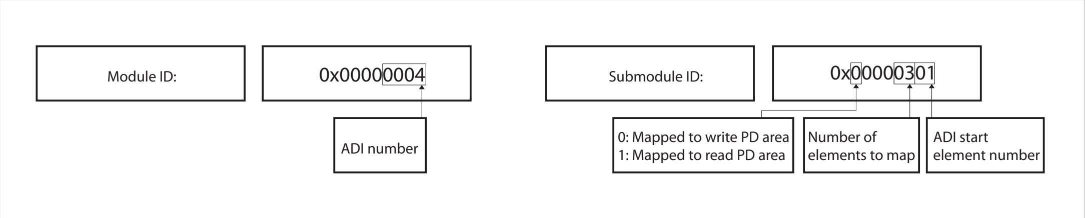


Figure 18.

Example (100BASE-TX DAP):

ADI #	Type	Resulting Real Identification		
		Module/Submodule ID	Slot/Subslot	IO Data Size (bytes)
-	-	0x80010000/0x00000001 0x80010000/0x00000002 0x80010000/0x00000003 0x80010000/0x00000003	0/1 (Device Access Point) 0/0x8000 (Interface) 0/0x8001 (Port1) 0/0x8002 (Port2)	-
6	1 SINT16 Wr	0x00000006/0x00000100	1/1	2
3	2 UINT8 Rd	0x00000003/0x10000200	2/1	2
3	1 UINT8 Rd	0x00000003/0x10000104	2/2	1
9	1 SINT32 Rd	0x00000009/0x10000100	3/1	4
10	1 SINT16 Wr	0x0000000A/0x00000100	4/1	2

See also...

- Application Process Instances (API) (page 27)
- Flowchart —Establishment of Real Identification (RI) (page 263)

Modular Device Configuration

The RI, when implementing a modular device, is generated from the Module IDs and process data mappings as defined in the Modular Device Object (ECh). For more information see Modular Device (page 39).

Custom Configuration (Advanced Users)

Optionally it is possible to override the default configuration during the SETUP state by means of the PROFINET specific commands API_add, Plug_Module, Plug_Submodule, and Plug_Submodule_Ext. This way, the host application can define exactly how ADIs are represented on PROFINET by defining custom modules and submodules. These commands need to be sent after the process data is mapped, and before sending setup complete to the Anybus CompactCom 40 PROFINET IRT IIoT Secure.

See also...

- Network PROFINET IO Object (0Eh) (page 167)
- Flowchart —Establishment of Real Identification (RI) (page 263)

5.5.2. Configuration Mismatch

General

A configuration mismatch occurs when the Real Identification (RI) does not match the Expected Identification. Depending on how the RI configuration is established, the Anybus CompactCom 40 PROFINET IRT IIoT Secure will first try to resolve the mismatch as described in the applicable section below (Resolving Mismatch for default configuration or custom configuration). If this attempt to resolve the mismatch fails, the Anybus CompactCom 40 PROFINET IRT IIoT Secure will provide indications as described in Further Actions to Resolve Mismatch (page 32).

Resolving Mismatch (Default Configuration)

If the Real Identification has been established according to the default mode, the Anybus CompactCom 40 PROFINET IRT IIoT Secure will try to remap the Real Identification to match the Expected Identification.

> **IMPORTANT**
> The application must have implemented support for the remap commands in the Application Data object (FEh), for remap to be possible.

As the Module ID contains the ADI number and the Submodule ID describes I/O direction and element section, all information required to perform a remap is available.

> **NOTE**
> The application must be able to respond with the data type for every ADI, or a remap is not possible.

The remap is performed as follows:

- A request is sent to remap the read area. If this request is rejected no change is made to the process data map and the Real Identification.
- If the read area remap succeeds, a write remap request is sent. If this is rejected, the process data map is in an inconsistent state and new remap commands are sent that will remove all mappings.
- If both the read and the write remap requests succeed a new Real Identification will be built to match the Expected Identification.

See also ...

- Default Configuration Mismatch (page 264) (flowchart)
- Application Data Object (FEh) (see Anybus CompactCom 40 Software Design Guide)

Resolving Mismatch (Custom Configuration)

If a configuration mismatch occurs for a custom configuration, the Anybus CompactCom 40 PROFINET IRT IIoT Secure will issue the command Expected_Ident_Ind to the host application. If the host application intends to change the Real Identification based on the Expected Identification, it responds with “Block” and performs the required Pull/Plug operations, before sending an Ident_Change_Done command to the Network PROFINET IO Object (0Eh).

See also ....

- Custom Configuration mismatch (page 265) (flowchart)
- Expected_Ident_Ind, see command details in PROFINET IO Object (F6h) (page 228)
- Ident_Change_Done, see command details in Network PROFINET IO Object (0Eh) (page 167)

Further Actions to Resolve Mismatch

If the mismatch remains unsolved, either for default or custom configuration, the following will be performed by the Anybus CompactCom 40 PROFINET IRT IIoT Secure to find a solution that will make exchange of valid data possible:

Incomplete output mapping:	If the controller tries to connect to fewer output submodules than are plugged by the application, the controller will not be granted ownership of any output submodule. The Anybus state is set to ERROR and the LEDs will indicate configuration mismatch. Matching input submodules will be owned by controller and work normally.
Incomplete input mapping:	The controller may choose to connect to a subset of the available input submodules without any restrictions. Anybus state is set to PROCESS_ACTIVE and no error is indicated on any LED.
Mismatch of submodule(s):	As long as all of the output submodules of the Real Identification are present and matching in the Expected Identification, the Anybus state is set to PROCESS_ACTIVE. However, if there is any mismatch among the other submodules the LEDs will indicate configuration mismatch.

See also...

- The Remap_ADI_Write_Area and Remap_ADI_Read_Area commands in the Application Data Object (FEh), found in Anybus CompactCom 40 Software Design Guide.

5.6. Diagnostics

5.6.1. Standard Diagnostics

PROFINET IO uses alarms when informing the IO Controller of diagnostic entries. In the Anybus implementation, it is possible for the application to create alarms via diagnostic entries by means of the Diagnostic Object (02h).

Up to 5 diagnostic instances can be created by the host application. An additional 6th instance can always be created in event of a major unrecoverable fault.

Creating a diagnostic instance is done by issuing the command Create. If the module is in state IDLE or PROCESS_ACTIVE, the created instance will be communicated on the network as an “appear”-alarm. If the module is in another state, the PLC will be notified in the connect response by a module diff block.

Deleting a diagnostic instance is done by issuing the command Delete. This will trigger a “disappear”-alarm on the network. Supply the instance ID that was returned by the create-command.

Every diagnostic instance has a severity level and an event code associated to it. Major unrecoverable events will cause the module to disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic alarm or a Generic Diagnostic alarm, depending on the Event Code, according to the table below.

Severity	Event code != network specific	Event code = network specific
Minor, recoverable	Channel Diagnostic Alarm	Generic Diagnostic Alarm (See Extended Diagnostics (page 33))
Minor, unrecoverable		
Major, recoverable		
Major, unrecoverable	Anybus enters Exception state	

NOTE
Process alarms can not be created.

See also..

- Diagnostic Object (02h) (page 122)

5.6.2. Extended Diagnostics

Using the network specific event code (FFh) creates a Generic Diagnostic Alarm on the network. This type of alarm can carry extended diagnostic information and more details about the source of the problem.

Generic Diagnostic Alarm instances can be tagged with a source API and slot- and subslot number. It can also contain additional network specific diagnostic data like:

- Standard channel diagnosis with manufacturer extension
- Manufacturer specific channel diagnosis
- Manufacturer specific channel diagnosis with manufacturer specific extension that provide also network specific diagnostic data

For more information, see Details: Network Specific Data (page 123).

5.7. Identification & Maintenance (I&M)

5.7.1. General Information

Identification & Maintenance (I&M) provides a standard way of gathering information about an I/O device. The I&M information can be accessed by the IO Controller by means of acyclic Record Data Read/Write services.

The application should provide application specific I&M0 information during start-up. See PROFINET IO Object (F6h) (page 228) for more information.

It is possible for the application to handle I&M records. Activate this using the IM_Options command. See Network PROFINET IO Object (0Eh) (page 167) for more information.

While I&M0 information describes the application, the I&M5 information describes the communication unit i.e. the Anybus CompactCom 40 PROFINET IRT IIoT Secure, to the PROFINET network. Most of the information is provided by the module, but the application can change the order ID and the IM annotation (attributes #25 and #26, see PROFINET IO Object (F6h) (page 228)). I&M5 is enabled by default, but can be disabled using attribute #27 (IM5 enabled), see PROFINET IO Object (F6h) (page 228). If the example GSD file is used, I&M5 has to be disabled there aswell, se below.

NOTE
I&M5 is not available for Anybus IP. See below how to disable I&M5 in the GSD file for Anybus IP.

Default I&M0 information:

IM Manufacturer ID	010Ch (HMS Industrial Networks)
IM Order ID	“CompactCom 40 PIR”
IM Serial Number	(unique serial number, set during manufacturing)
IM Hardware Revision	(Anybus hardware revision ID, set during manufacturing)
IM Software Revision	(Anybus software revision, set during manufacturing)
IM Revision Counter	(Revision counter)
IM Profile ID	0000h (Generic Device)
IM Profile Specific Type	0004h (No profile)
IM Version	0101h
IM Supported	For submodules belonging to a “non–zero API”, the returned value is zero. For submodules belonging to API 0, the returned value is 000Eh (IM0-3 supported)

Disabling I&M5 in the GSD File

NOTE
I&M5 is always disabled and can not be enabled on Anybus IP. It has to be disabled in the example GSD file, if this is to be used.

I&M5 is enabled by default in the GSD file.

The settings for I&M are located at the Device Access Point, within `<VirtualSubmoduleItem...>`. To disable I&M5 , change GSDML entry `<IM5_Supported=“true”>` to `<IM5_Supported=“false”>`.

5.7.2. I&M Data Structures

The I&M records uses the following data structures.

Record	Content	Size	Description
I&M0	Manufacturer Id	2 bytes	PROFINET IO Object (F6h), attribute #2 (Vendor ID/I&M Vendor ID)
	Order Id	20 bytes	PROFINET IO Object (F6h), attribute #8 (I&M Order ID)
	Serial number	16 bytes	The content will be assigned in the following priority order: 1. PROFINET IO Object (F6h), attribute #9 (I&M Serial number) 2. Application Object (FFh), attribute #3 (Serial number) 3. Default value, assigned during manufacturing
	Hardware revision	2 bytes	The content will be assigned in the following priority order: 1. PROFINET IO Object (F6h), attribute #10 (I&M Hardware revision) 2. Application Object (FFh), attribute #11 (Hardware version) 3. Default value, assigned during manufacturing
	Software revision	4 bytes	The content will be assigned in the following priority order: 1. PROFINET IO Object (F6h), attribute #11 (I&M Software revision) 2. Application Object (FFh), attribute #10 (Firmware version) 3. Default value, assigned during manufacturing
	Revision counter	2 bytes	PROFINET IO Object (F6h), attribute #12 (I&M Revision counter)
	Profile Id	2 bytes	PROFINET IO Object (F6h), attribute #13 (I&M Profile ID)
	Profile specific type	2 bytes	PROFINET IO Object (F6h), attribute #14 (I&M Profile specific type)
	IM version	2 bytes	0101h (Internal, constant value)
	IM supported	2 bytes	For submodules belonging to a “non–zero API”,the returned value is zero. For submodules belonging to API 0, the returned value is 000Eh (IM0-3 supported)
I&M1	Tag Function	32 bytes	Default: All bytes set to blanks
	Tag Location	22 bytes	Default: All bytes set to blanks
I&M2	Installation date	16 bytes	Default: All bytes set to blanks
I&M3	Descriptor	54 bytes	Default: All bytes set to blanks
I&M5	IM_Annotation	64 bytes	PROFINET IO object; attribute #26 (IM Annotation). If the application doesn’t support the attribute: Anybus CompactCom 40 PROFINET IRT IIoT
	IM_OrderID	64 bytes	PROFINET IO object; attribute #25 (IM Module Order ID). If the application doesn’t support the attribute: Anybus CompactCom 40 PROFINET IRT IIoT
	VendorIDHigh	1 byte	01h
	VendorIDLow	1 byte	0Ch
	Serial number	16 bytes	Serial number of the Anybus CompactCom
	Hardware revision	2 bytes	Hardware version of the Anybus CompactCom
	Software revision	4 bytes	Product version of the Anybus CompactCom

See also..

- PROFINET IO Object (F6h) (page 228)
- Anybus CompactCom 40 Software Design Guide, Application Object (FFh)

5.8. Asset Management

Asset management provides means to collect information on non PROFINET automation components, that are connected to PROFINET networked devices. These components are not part of the PROFINET system, but the collected information will facilitate troubleshooting and exchange of faulty components. The host application can describe up to 32 different non PROFINET components in the instances of the Asset Management Object.

See also...

- Asset Management Object (E5h) (page 211)

5.8.1. Activating Asset Management in the GSD File

Asset management is not a default feature, and the example GSD files distributed do not contain the keyword “AssetManagement”. To activate it, the user must implement the Asset Management object for the device to accept the Asset Management Read Record from the network. In addition the keyword “AssetManagement” must be added to the GSD.

The settings for Asset Management have to be added at the end of the Device Access Point, directly after `</Graphics>`, or the GSD checker tool will invalidate the file:

```
...
<DeviceAccessPointItem...>
  ....
  <Graphics>
    <GraphicItemRef Type=”DeviceSymbol” GraphicItemTarget”1”/>
  </Graphics>
  <AssetManagement/>
</DeviceAccessPointItem>
...
```

5.9. Address Conflict Detection (ACD)

The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports Address Conflict Detection (ACD). This mechanism involves the following two aspects:

- Initial probing: before using an IP address, the module issues ARP probes to see if the address is already in use (three probes with a 100 ms delay).
- Address announcement: after the initial probing, the module issues ARP announcements.

If an IP address conflict is detected, IP address error will be indicated on the Network Status LED. The module will use address 0.0.0.0. A new address can be configured via the Anybus IPconfig tool.

IMPORTANT

If Fast Start Up is used, ACD initial probing is automatically disabled to ensure a fast startup. Address announcement is still used, as it will not affect the actual startup time.

To enable/disable ACD, seeEthernet Host Object (F9h) (page 250).

5.10. PROFIenergy Profile

The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports the PROFIenergy profile, according to the PROFIenergy Technical Specification, rev. 1.2. This profile makes it possible for a user to temporarily put a device in energy saving mode, e.g. during a lunch break or during weekends. The amount of power used by machines, when they are not in active use is thus reduced. Each device can be set individually to the energy saving mode that is the most optimal depending on the length of the production stop. Operators in factories, for example, can easily set all devices at the same time, in the, for each device, optimal energy saving mode.

The application defines the time for how long the device will stay in energy saving mode, and the device decides which mode will be the most appropriate. Transitions between the “Ready to operate” mode and all saving modes are mandatory, while transitions between different energy saving modes are optional. The transition from “Ready to operate mode” to “Power off mode” is not mandatory, as repowering the device may mean hands on restart.

5.10.1. Implementation

The PROFIenergy profile is implemented in the Anybus CompactCom 40 PROFINET IRT IIoT Secure according to the state machine described in the PROFIenergy Technical Specification (available from PROFIBUS International). PROFIenergy commands arriving from the network will be translated into the Anybus CompactCom implementation as follows:

PROFIenergy command	Anybus CompactCom 40 implementation
Start_Pause	Translated into a StartPause command to instance #0 in the Energy Control Object.
Start_Pause_with_time_response	Same as Start_Pause but will do several reads of the Energy Control object, and the currently used energy saving mode to complement the response.
End_Pause	Translated into a EndPause command to instance #0 in the Energy Control Object.
List_Energy_Saving_Modes	Instances present in the Energy object will be listed.
Get_Mode	Supported attributes for the requested mode/instance will be read.
PEM_Status	Results in several reads of Energy Control object and the currently used energy saving mode.
PEM_Status_with_Ext1	
PE_Identify	Returns a static list of the supported, and also mandatory, PROFIenergy commands.
Get_Measurement_List	Returns the list of supported measurement values from the Energy Measurement object.
Get_Measurement_Values	Returns the actual measurement values from the Energy Measurement object.
Get_measurement_List_with_object_number	Returns the list of supported measurement values from the Energy Measurement object.
Get_Measurement_Values_with_object_number	Returns the actual measurement values from the Energy Measurement object.
Query_Version	Return the supported PROFIenergy version.
Info_Sleep_Mode_WOL	Not supported.
Go_Sleep_Mode_WOL	

The PROFIenergy profile is valid for all sub-slots and is accessed through index 80A0h. Maximum number of instances/modes in the Energy Control Object is 8.

See also...

- Energy Measurement Object (E4h) (page 199)
- Energy Control Object (F0h) (page 222)

5.11. PROFIsafe

The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports the PROFIsafe profile. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over PROFINET using an add on Safety Module, e.g. the IXXAT Safe T100/PS. For more information about this module, see IXXAT Safe T100 Manual, available at www.ixxat.com.

The Safety Module can be located in any valid slot. This is true for when the configuration is handled by the module, as well as when the configuration is handled by the application.

For an application to support PROFIsafe, the Functional Safety Object has to be implemented in the application. Slots are assigned using the Command Add_Safety_Module to the Network PROFINET IO object (0Eh). The safe communication is enabled in the host application Functional Safety Object (E8h).

The Anybus CompactCom serial channel is used for functional safety communication. If this channel is used for the host application, a second separate serial channel is implemented for the functional safety communication, see Anybus CompactCom Hardware Design Guide.

See also...

Functional Safety Module Object (11h) (page 184)

Functional Safety Object (E8h) (page 215)

PROFINET IO Object (F6h) (page 228)

Command details: Add_Safety_Module in Network PROFINET IO Object (0Eh) (page 167)

Anybus CompactCom Hardware Design Guide

6. Modular Device

6.1. General

The modular device concept is useful when creating a configuration based on reusable modules, for example when modelling a backplane device with pluggable modules.

The first slot (slot 0) is always occupied by the Device Access Point (DAP) module, which represents the head node where the Anybus CompactCom is located. It is not allowed to add a process data mapping for any ADI within the range of slot 0. The remaining slots are assigned fixed ranges of ADIs, set up using the Modular Device Object. A GSD module can represent different ADIs, depending on which slot it occupies, making it possible to create a configuration based on reusable modules. This is not possible in ADI based RI mode.

See also ...

- “ Modular Device Object (ECh)” (see Anybus CompactCom 40 Software Design Guide)

6.2. Modular Device RI

The RI, when implementing a modular device, is generated from the Module IDs as defined in the Modular Device Object (ECh) (see the Anybus CompactCom 40 Software Design Guide) and from the process data mappings as shown in the example below.

Module ID:	0x0000004				
Submodule ID:	0x00010203				
	0x0				0: Mapped to Write PD area 1: Mapped to Read PD area
		001			ADI index
			02		Number of elements to map
				03	ADI start element #

A set of conditions must be fulfilled for the Anybus CompactCom to use the Modular Device RI:

- The Modular Device object implemented must be of revision 1.
- Modular device object attributes #11 and #12 must be readable and valid
  - Number of slots must be: 0 < NumSlots <= 256
  - ADIs per slot must be: 0 < AdisPerSlot <= 4096
- The list with Module Type IDs must contain “NumSlots” items.

6.3. Remap

For a host application not supporting remap, the modules must contain a fixed set of submodules (GSD attribute FixedInSubslots), where the process data mapping of each slot is performed in accordance with the GSD definition of each particular module type. See Modular Device Example (page 40), where slot 1 and slot 3 have the same module ID and also matching process data mapping commands.

If remap is supported by the application the GSD modules may be designed in a flexible manner with pluggable submodules (GSD attributes AllowedInSubslots / UsedInSubslots). If the configuration results in an expected identification that differs from the real identification the Anybus CompactCom will attempt to resolve the mismatch according to Configuration Mismatch (page 31).

6.4. Safety Module

A safety module, such as the IXXAT Safe T100/PS may be used within the modular device concept. If enabled, the safety module can be plugged into any valid slot and the host application will not be allowed to add a process data mapping for any ADI within the range this slot. Please note that this module can only be used once and that should be reflected in the GSD file e.g. by setting AllowedInSlots/UsedInSlots/FixedInSlots to “[safety module slot no]”.

6.5. Record Data

In non-transparent record data mode, ADIs are addressed acyclically, based on slot and index in a one-to-one relationship, according to ADI number = Slot * ADIsPerSlot + Index. Since the address space is not divided per slot/module, it is recommended to use transparent record data mode when implementing the modular device concept. This way the host application will be informed which slot, subslot and index is addressed by the controller.

6.6. Modular Device Example

The following example shows a device with two defined types of modules (in addition to the DAP). In its current configuration the device has Module Type B plugged in slots 1 and 3, while Module Type C is plugged in slot 2. No safety module is used so process data can be added to all slots except slot 0.

6.6.1. Mapping Commands

Please note that the mapping commands below omit parts not relevant for the example.

Slot 1:

Map_ADI_Write_Ext_Area(ADI=257, FirstIndex=0, NumElems=8, DataTypes=(UINT8,))

Map_ADI_Read_Ext_Area(ADI=300, FirstIndex=3, NumElems=2, DataTypes=(UINT16,))

Map_ADI_Read_Ext_Area(ADI=304, FirstIndex=0, NumElems=1, DataTypes=(INT32,))

Slot 2:

Map_ADI_Read_Ext_Area(ADI=513, FirstIndex=0, NumElems=4, DataTypes=(UINT8,))

Map_ADI_Write_Ext_Area(ADI=700, FirstIndex=0, NumElems=2, DataTypes=(UINT16,))

Map_ADI_Write_Ext_Area(ADI=708, FirstIndex=0, NumElems=16, DataTypes=(UINT32,))

Map_ADI_Read_Ext_Area(ADI=760, FirstIndex=0, NumElems=3, DataTypes=(BIT4, PAD2, BIT2))

Slot 3:

Map_ADI_Write_Ext_Area(ADI=769, FirstIndex=0, NumElems=8, DataTypes=(UINT8,))

Map_ADI_Read_Ext_Area(ADI=812, FirstIndex=3, NumElems=2, DataTypes=(UINT16,))

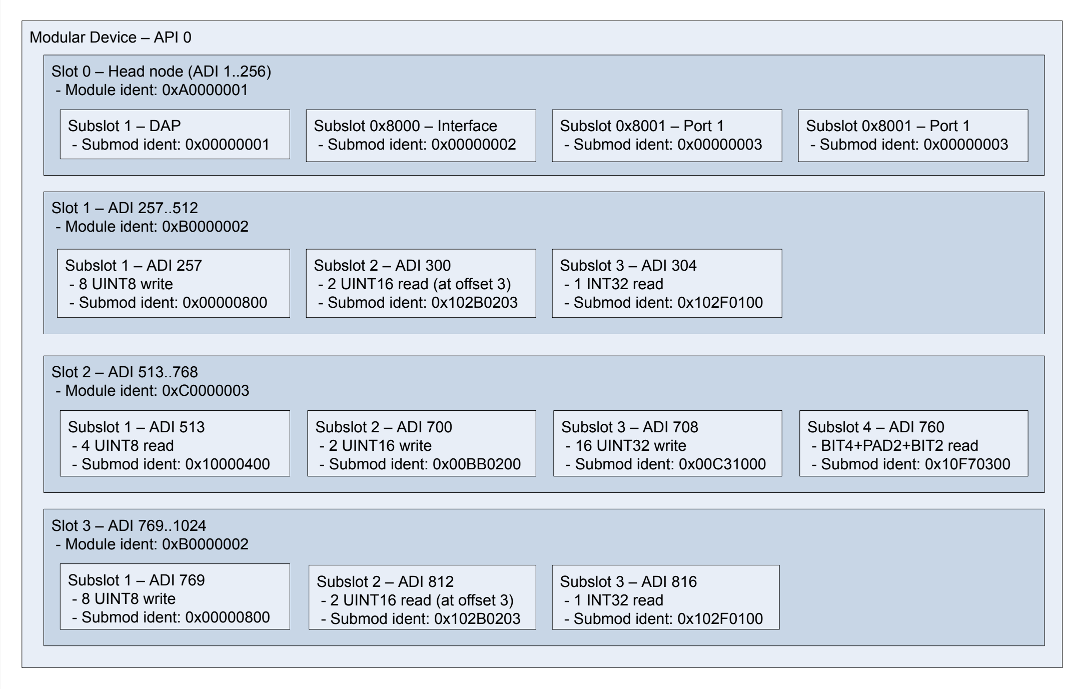
Map_ADI_Read_Ext_Area(ADI=816, FirstIndex=0, NumElems=1, DataTypes=(INT32,))

6.6.2. List of All Module IDs:

The response to the Get_List command (list type 01h) contains the following modules:

1. 0xA0000001 (DAP)
2. 0xB0000002 (Module type B)
3. 0xC0000003 (Module type C)
4. 0xB0000002 (Module type B)

6.6.3. Device Layout



Modular Device – API 0

Slot 0 – Head node (ADI 1..256)
- Module ident: 0xA0000001

Subslot 1 – DAP
- Submod ident: 0x00000001

Subslot 0x8000 – Interface
- Submod ident: 0x00000002

Subslot 0x8001 – Port 1
- Submod ident: 0x00000003

Subslot 0x8001 – Port 1
- Submod ident: 0x00000003

Slot 1 – ADI 257..512
- Module ident: 0xB0000002

Subslot 1 – ADI 257
- 8 UINT8 write
- Submod ident: 0x00000800

Subslot 2 – ADI 300
- 2 UINT16 read (at offset 3)
- Submod ident: 0x102B0203

Subslot 3 – ADI 304
- 1 INT32 read
- Submod ident: 0x102F0100

Slot 2 – ADI 513..768
- Module ident: 0xC0000003

Subslot 1 – ADI 513
- 4 UINT8 read
- Submod ident: 0x10000400

Subslot 2 – ADI 700
- 2 UINT16 write
- Submod ident: 0x00BB0200

Subslot 3 – ADI 708
- 16 UINT32 write
- Submod ident: 0x00C31000

Subslot 4 – ADI 760
- BIT4+PAD2+BIT2 read
- Submod ident: 0x10F70300

Slot 3 – ADI 769..1024
- Module ident: 0xB0000002

Subslot 1 – ADI 769
- 8 UINT8 write
- Submod ident: 0x00000800

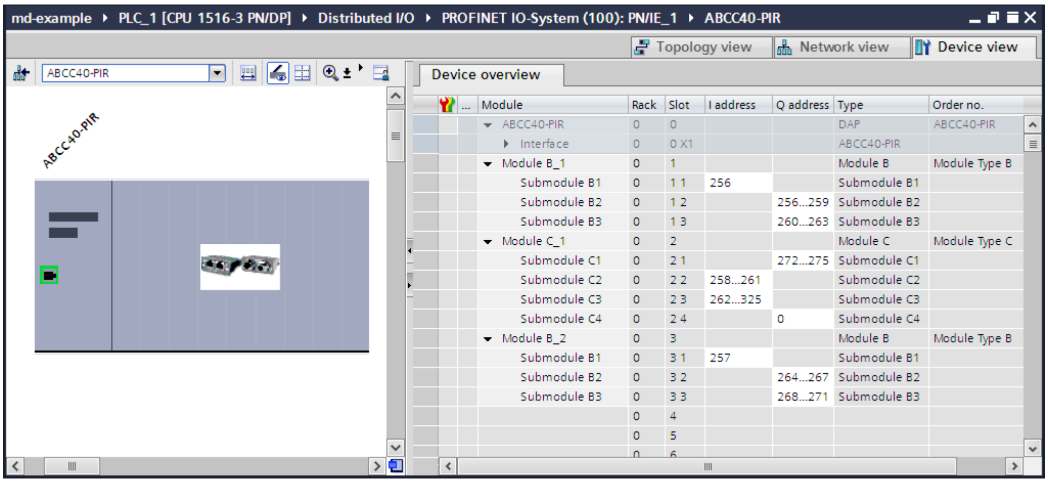
Subslot 2 – ADI 812
- 2 UINT16 read (at offset 3)
- Submod ident: 0x102B0203

Subslot 3 – ADI 816
- 1 INT32 read
- Submod ident: 0x102F0100

Figure 19.

6.6.4. Example View in TIA Portal

Note that the GSD instantiates the same module, Module Type B in both slot 1 and slot 3.



md-example ▸ PLC_1 [CPU 1516-3 PN/DP] ▸ Distributed I/O ▸ PROFINET IO-System (100): PN/IE_1 ▸ ABCC40-PIR
Topology view
Network view
Device view
ABCC40-PIR
ABCC40-PIR
Device overview

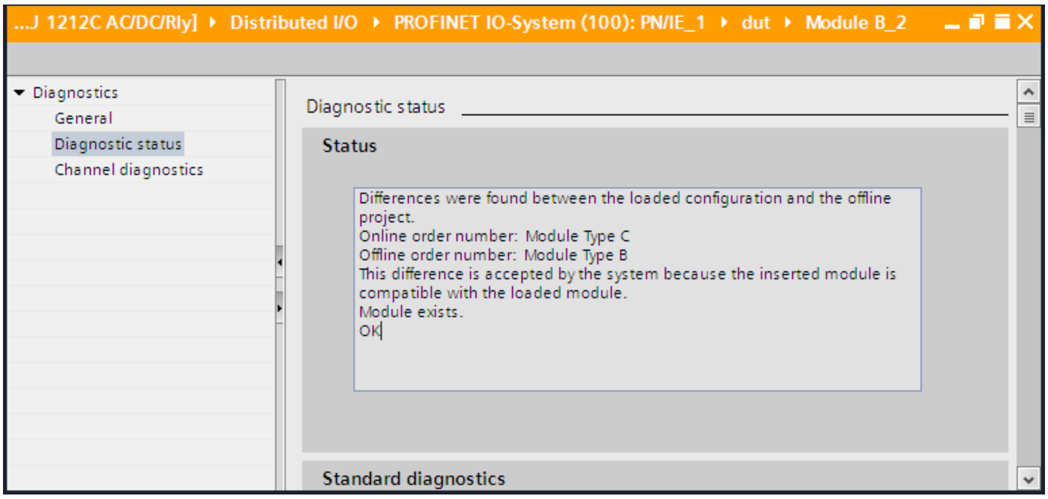
Module	Rack	Slot	I address	Q address	Type	Order no.
ABCC40-PIR	0	0			DAP	ABCC40-PIR
Interface	0	0 X1			ABCC40-PIR	
Module B_1	0	1			Module B	Module Type B
Submodule B1	0	1 1	256		Submodule B1	
Submodule B2	0	1 2		256...259	Submodule B2	
Submodule B3	0	1 3		260...263	Submodule B3	
Module C_1	0	2			Module C	Module Type C
Submodule C1	0	2 1		272...275	Submodule C1	
Submodule C2	0	2 2	258...261		Submodule C2	
Submodule C3	0	2 3	262...325		Submodule C3	
Submodule C4	0	2 4		0	Submodule C4	
Module B_2	0	3			Module B	Module Type B
Submodule B1	0	3 1	257		Submodule B1	
Submodule B2	0	3 2		264...267	Submodule B2	
Submodule B3	0	3 3		268...271	Submodule B3	
	0	4				
	0	5				
	0	6				

Figure 20.

6.7. Recommendations

6.7.1. I&M Data

Identification & Maintenance data may be used by the engineering tool to assist in resolving mismatching configurations. By comparing the OrderID from the expected GSD module with the OrderID from the I&M data of the actual, plugged module, the user can adapt the configuration to match the real identification. To ease integration and tracking of assets it is highly recommended to implement transparent I&M data for modules. When implementing I&M0 data for modules, one of the submodules in each module must be selected as a module representative. This submodule shall be listed as I&M0 Carrier and Module Representative in the I&M0 Filter Data of the IM_Options command.



Differences were found between the loaded configuration and the offline project.
Online order number: Module Type C
Offline order number: Module Type B
This difference is accepted by the system because the inserted module is compatible with the loaded module.
Module exists.
OK

Figure 21.

NOTE

After a successful process data remap the entire RI is regenerated and any I&M0 Filter Data must be reconfigured. Refer to the Expected_Ident_Ind command for finding out the new RI.

7. Secure Web Server (HTTPS)

7.1. 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.

IMPORTANT
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) (page 81)
- JSON (page 54)
- Certificates (page 6)
- Ethernet Host Object (F9h) (page 250)

7.2. 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 (page 7)

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)

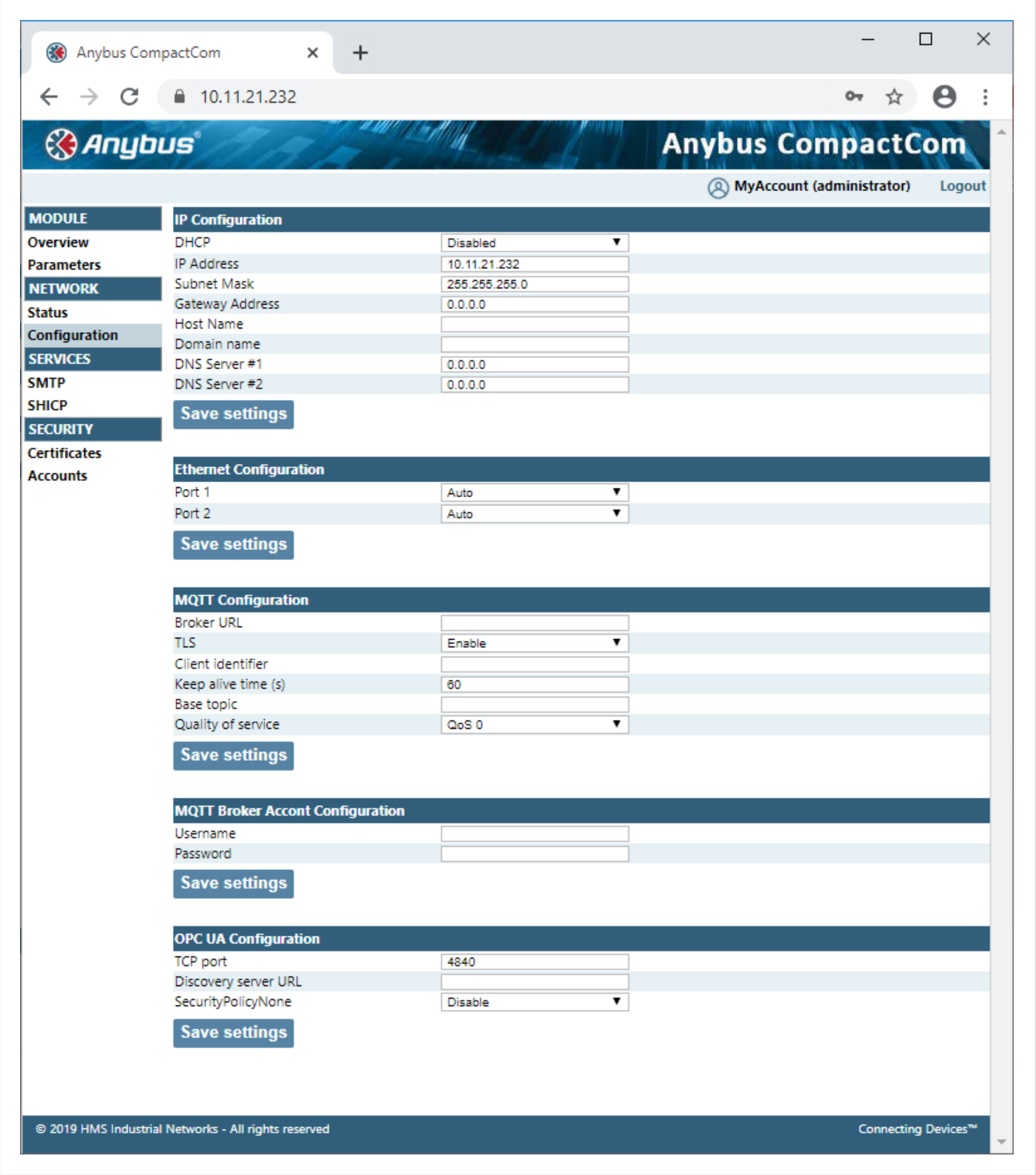
If a complete customized web system is designed and no files in the vfs folder are to be used, it is recommended to turn off the virtual file system completely, see the File System Interface Object.

See also ...

- File System (page 25)
- Anybus File System Interface Object (0Ah) (page 163)

7.2.1. 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

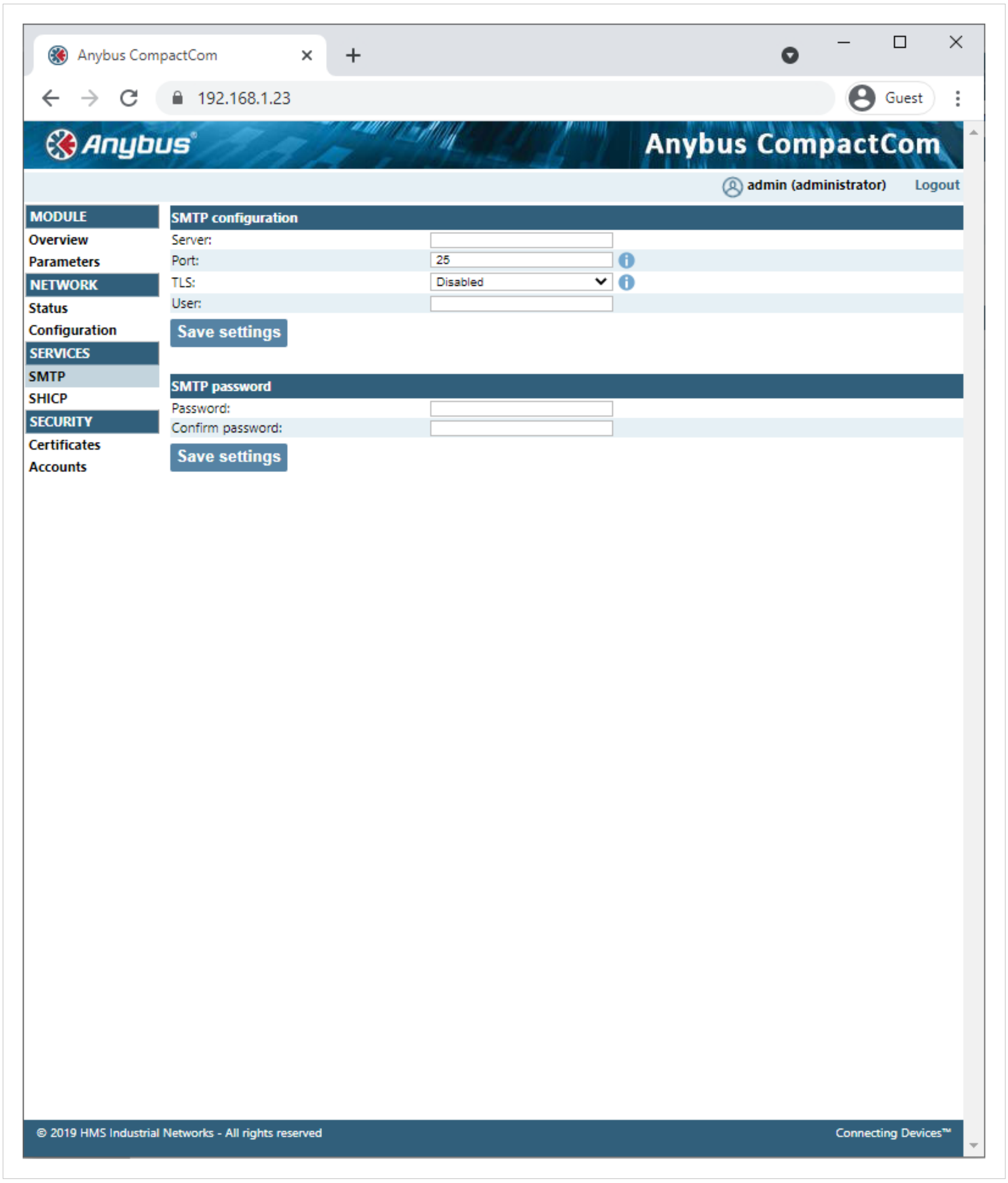
Ethernet Configuration	
Port 1	Auto
Port 2	Auto

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

Figure 22.



SMTP configuration

Server:	
Port:	25
TLS:	Disabled
User:	

SMTP password

Password:	
Confirm password:	

Figure 23.

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

7.2.2. 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
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.

7.3. 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 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\`	

Content	Description

`$all:\web\img\`
`$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) (page 81)
- JSON (page 54)
- Ethernet Host Object (F9h) (page 250)

7.3.1. 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

7.4. 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)

7.5. 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.

7.6. 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 (page 55) (JSON) for more information.

8. JSON

8.1. 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

8.1.1. 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.

8.1.2. Access

IMPORTANT
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.

8.1.3. 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.

8.1.4. 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.

8.2. 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 (page 55).

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.

8.3. Supported JSON functions

The table below lists all supported JSON functions in the Anybus CompactCom 40 PROFINET IRT 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

8.4. JSON API

8.4.1. 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 (page 80) for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see Hex Format Explained (page 80) 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 (page 80) for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see Hex Format Explained (page 80) for more information. May be NULL of no maximum value is present.
default	String	Hex formatted string, see Hex Format Explained (page 80) 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 (page 80) 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
}
```

8.4.2. 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	Vendor 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,
}
```

8.4.3. 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 }
```

8.4.4. Services

api/services/config/smtp.json

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

NOTE
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 (page 133) 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],
]
```

8.4.5. 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 }
}
```

8.4.6. 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 certificate This 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 CompactCom 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 }
       }
}
```

8.4.7. 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"
}
```

8.4.8. 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"
}
```

8.4.9. 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"
  }
```

8.4.10. 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
```

8.5. 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>
```

9. 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 PROFINET IRT 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) (page 250)

9.1. 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`

NOTE
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
```

9.2. WebDAV

If a trusted certificate is configured for HTTPS, the file system of the Anybus CompactCom 40 PROFINET IRT IIoT Secure can be mapped as a network drive on your PC. See File Transfer Protocol (WebDAV) (page 81) for more information.

1. Select Map network drive on your PC.

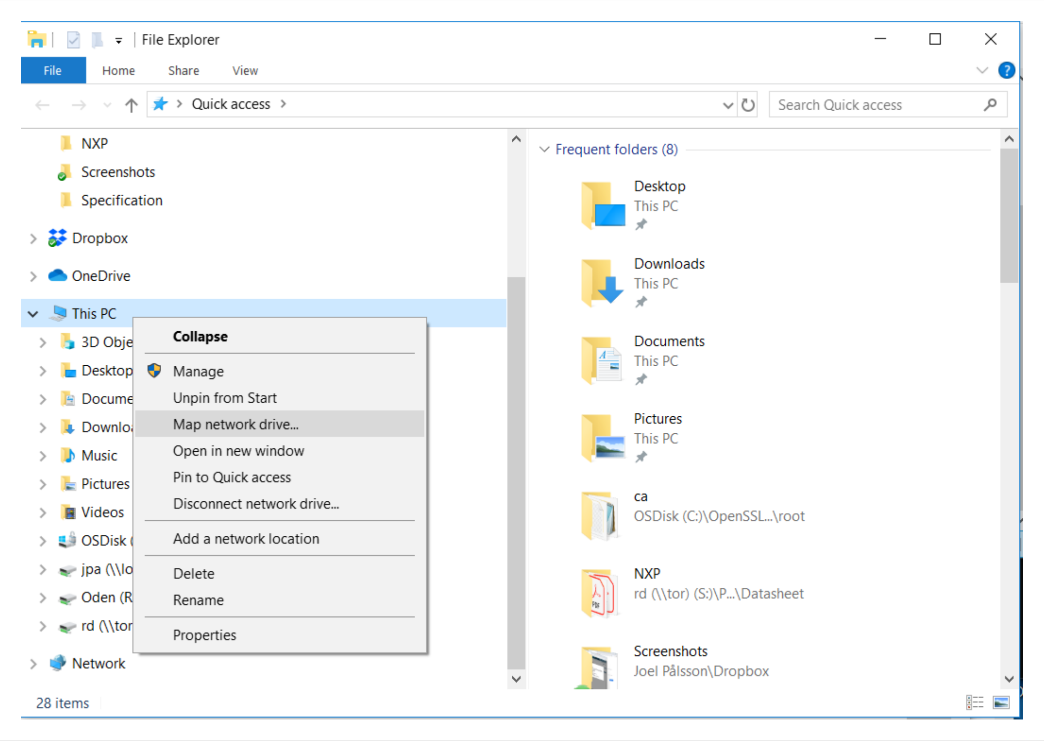
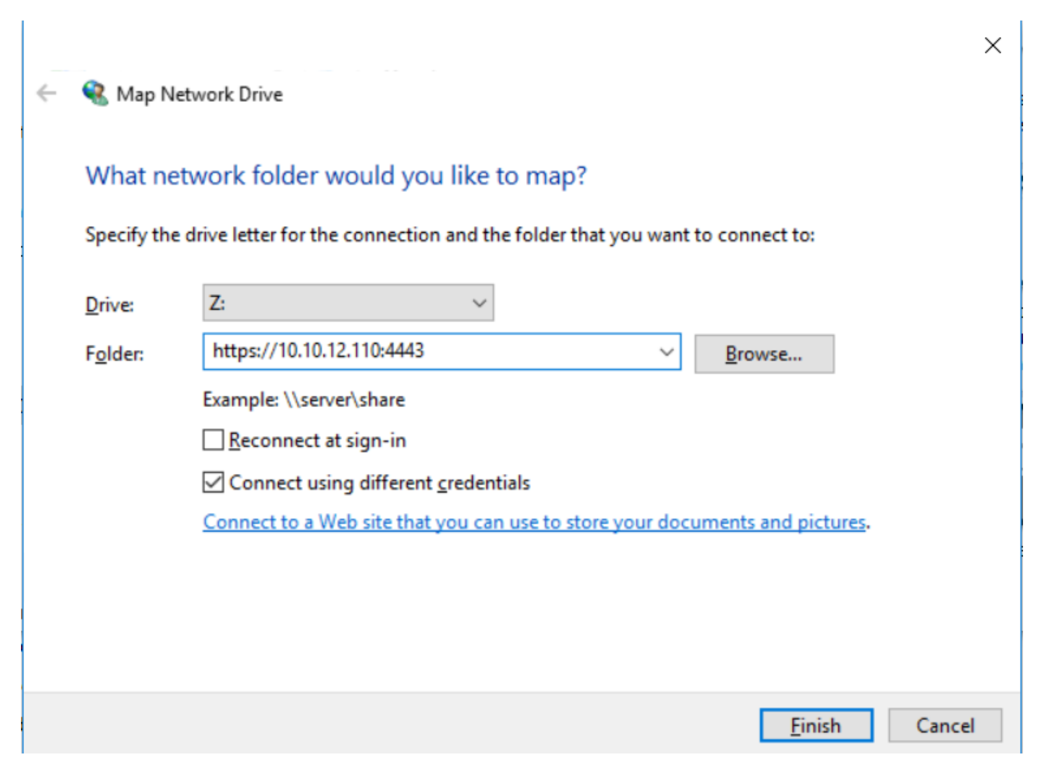


Figure 24.

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



Map Network Drive

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

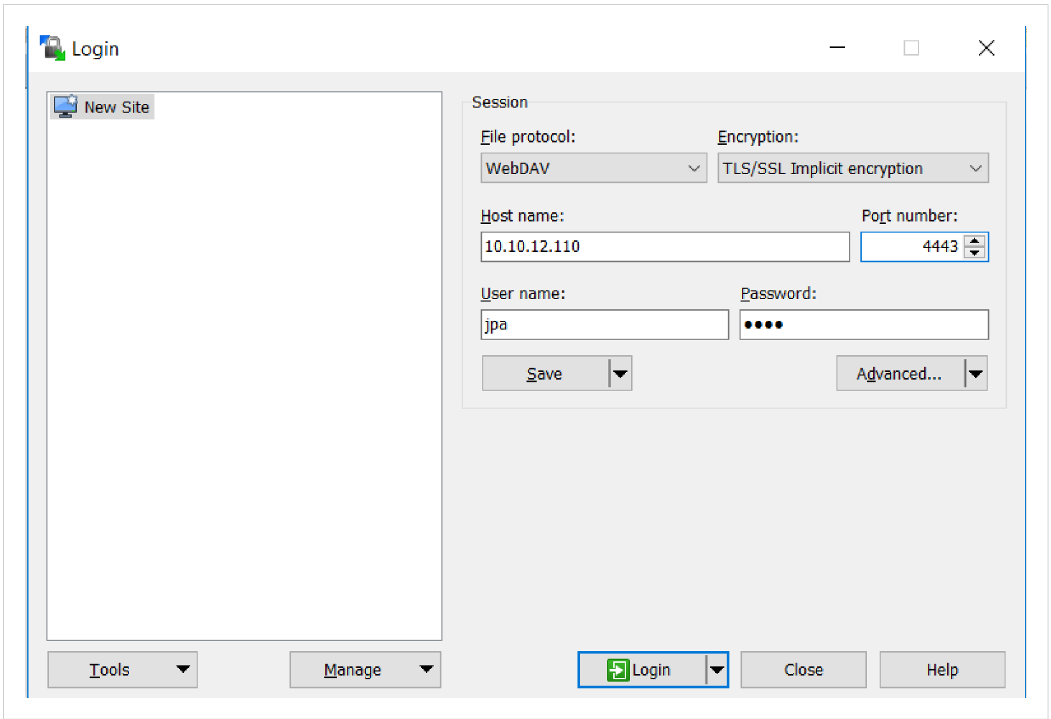
□ Reconnect at sign-in

☑ Connect using different credentials

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

Figure 25.

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.



File protocol: WebDAV
Encryption: TLS/SSL Implicit encryption
Host name: 10.10.12.110
Port number: 4443
User name: jpa
Password: ••••

Figure 26.

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

10. E-mail Client

10.1. 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.

10.2. 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).

11. SNMP Agent

11.1. General

Simple Network Management Protocol (SNMP, see RFC1157 standard) is used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. A management agent is installed in the managing station, and exchanges data via get and set requests.

11.2. Community Strings

SNMP uses community strings for authentication. SNMP support and default community strings are required for PROFINET comformance. For security reasons it is strongly recommended to change the values of these strings. See attributes #20 and #21 in the Ethernet Host Object (F9h) (page 250).

11.3. Management Information (MIB)

A MIB is a device database that is accessed by an SNMP agent. The Anybus CompactCom 40 PROFINET IRT IIoT Securesupports standardized MIBs: LLDP-MIB and MIB-II. Standardized MIBs are defined in RFC standards and contain variables that are divided into so called groups. The host application can change the values of some of the variables for the MIB-II.

11.4. MIB_II

The MIB-II of the Anybus CompactCom 40 PROFINET IRT IIoT Secure contains the system- and interfaces group. The following tables show the variables according to the MIB-II standard (RFC1213) for monitoring the device status. The access authorizations refer to access via the SNMP protocol.

11.4.1. System Group Variables

Variable	Access Authorizations	Description	Source of Origin
sysDescr	Read only	Description of the device. Data type: DisplayString (only printable ASCII characters). Max 255 characters.	Internal
sysObjectID	Read only	N/A. Value=0	Internal
sysUpTime	Read only	Time since last power up (in hundredths of a second)	Internal
sysContact	Read/Write	Identification of the contact person for the device, including contact information. Data type: Displaystring. Max 255 characters. Factory default setting: empty string	Internal
sysName	Read/Write	Name of the device. Data type: Displaystring. Max 255 characters. Factory default setting: empty string	Internal
sysLocation	Read/Write	Physical location of the device (IM Tag Location). Data type: (DisplayString). Max 255 characters. Factory default setting: empty string	Internal
sysServices	Read only	Functionality of the device. Value=74, which indicates that the device has functionality that represents layers 2(switch), 4(TCP) and 7(Application) in the OSI model.	Internal

11.4.2. Interfaces Group Variables

Access authorizations for all variables are read only with values from internal sources. The number in brackets refers to the port number (1 - Port 1, 2 - Port 2, 3 - Internal port)

If nothing else is specified, the value of a variable is 0

Variable	Data Type	Value	Description
ifNumber	integer	3	Number of network interfaces present. Constant
IfIndex(1..3)	integer	ifIndex(1) = 1 ifIndex(2) = 2 ifIndex(3) = 3	Unique value for each interface. Constant
ifDescr(1..3)	octetstring	ifDescr(1) = “port-001” ifDescr(2) = “port-002” ifDescr(3) = “port-internal”	Information about the interface. ifDescr(1) must equal "port-001"and ifDescr(2) = "port-002" to be compatible with the STEP7 topology scanner.
ifType(1..3)	integer	6 (“Ethernet-csmacd”)	Type of interface
ifMtu(1..3)	integer	1500	Size of largest datagram that can be sent/received on the interface, specified in octets
ifSpeed(1..3)	gauge	0 or 100 000 000	Data transfer rate of the Ethernet port in bits per second. The speed is only shown for ports where the link status is "up".
ifPhysAddress(1..3)	octetstring		MAC address for the ports
ifAdminStatus(1..3)	integer	1 (“up”)	Desired state of the Ethernet port
ifOperStatus(1..3)	integer	1 (“up”) or 2 (“down”)	Current operating state of the Ethernet port. (Link = "up", No link = "down".)
ifLastChange(1..3)	timeticks	Time when state changed, except ifLastChange(3) = 0	Time (since start-up) when the port changed to its current state, see previous variable. Indicated in multiples of hundredths of a second
ifInOctets(1..3)	counter	ifInOctets(1..3) = Number of octets	Total number of octets received on the interface, including framing characters
ifInUcastPkts(1..3)	counter	ifInUcastPkts(1..3) = Number of unicast packets	Number of subnetwork-unicast packets delivered to a higher-layer protocol
ifInNUcastPkts(1..3)	counter	ifInNUcastPkts(1..3) = Number of non-unicast packets	Number of non-unicast (i.e. subnetwork-broadcast or subnetwork-multicast) delivered to a higher-layer protocol.
ifInDiscards(1..3)	counter	ifInDiscards(1..3) = number of discarded packets	Number of inbound packets which were discarded, without any error detected, not to be delivered to a higher-layer protocol. (One reason to discard packages might be to free up buffer space)
ifInErrors(1..3)	counter	ifInErrors(1..3) = number of error packets	Number of inbound packets with errors
ifInUnknownProtos(1..3)	counter	ifInUnknownProtos(1..3) = Number of unknown packets	Number of packets received via the interface, discarded because of an unknown or unsupported protocol.
ifOutOctets(1..3)	counter	ifOutOctets(1..3) = Number of octets	Total number of octets transmitted out from the interface, including framing characters
ifOutUcastPkts(1..3)	counter	ifOutUcastPkts(1..3) = Number of unicast packets	Total number of packets that higher-level protocols requested to be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.
ifOutNUcastPkts(1..3)	counter	ifOutNUcastPkts(1..3) = Number of non-unicast packets	Total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e. a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.
ifOutDiscards(1..3)	counter	ifOutDiscards(1..3) = Number of discarded packets	Number of outbound packets which were discarded without any error detected, not to be transmitted. (One reason to discard packages might be to free up buffer space)
ifOutErrors(1..3)	counter	ifOutErrors(1..3) = Number of error packets	Number of outbound packets that could not be transmitted due to errors
ifOutQLen(1..3)	gauge	ifOutQLen(1..3) = Number of packets in queue	Length of the output packet queue (in packets),
ifSpecific(1..3)	objid	.0.0	Reference to MIB definitions specific to the particular media being used to realize the interface. Here no reference is available, so a fixed value is used for all ports.

12. Media Redundancy Protocol (MRP)

12.1. General

Media Redundancy Protocol (MRP) is a PROFINET specific ring protocol ensuring redundancy in the network, which can significantly decrease network downtime. It is a token based ring protocol with a master-slave hierarchy.

All the nodes in the PROFINET network part of the ring are connected using ring topology (that is, the last node is connected directly to the first node). If, at any point, the connection between two nodes would break, the data will flow the other way instead, guaranteeing that data can be sent to/from the IO Controller to the IO Device(s). The self-healing time is approximately 200 ms.

The Media Redundancy Master (MRM) is responsible for checking the functional capability of the ring network, by sending out cyclic tokens. The Media Redundancy Clients (MRC) basically work as switches that pass on the tokens. The Anybus CompactCom supports acting as a Media Redundancy Client (MRC). It also supports propagating link change to the Media Redundancy Master.

If port 2 of the device is inactivated in the Ethernet Host Object (F9h), support for MRP is removed. See Ethernet Host Object (F9h) (page 250) , instance attribute #13, for more information.

12.2. GSDML Entries

MRP functionality is enabled by default in the GSD file. The settings for MRP are located at the Device Access Point (DAP). within the `<InterfaceSubmoduleItem` …>. The Anybus CompactCom is defined as “Client” with the keyword `<MediaRedundancy SupportedRole=“Client”/>`.

For each physical port there are two keywords in the `<PortSubmoduleItem` …> section `<PortSubmoduleItem` … `SupportsRingportConfig=“true”IsDefaultRingport=“true”` … `/>`. These are set to “true” by default. To disable MRP, these two shall be set to“false”.

13. OPC UA

13.1. 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 (page 91).

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 (page 102).

OPC UA 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) (page 194)
- “Application Object (FFh)” (see Anybus CompactCom 40 Software Design Guide)

13.2. Configuration

13.2.1. 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

13.2.2. 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 x – Execute method 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,x:4	The role operator gets read and browse access to all namespaces and write and execute 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)

13.3. 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.

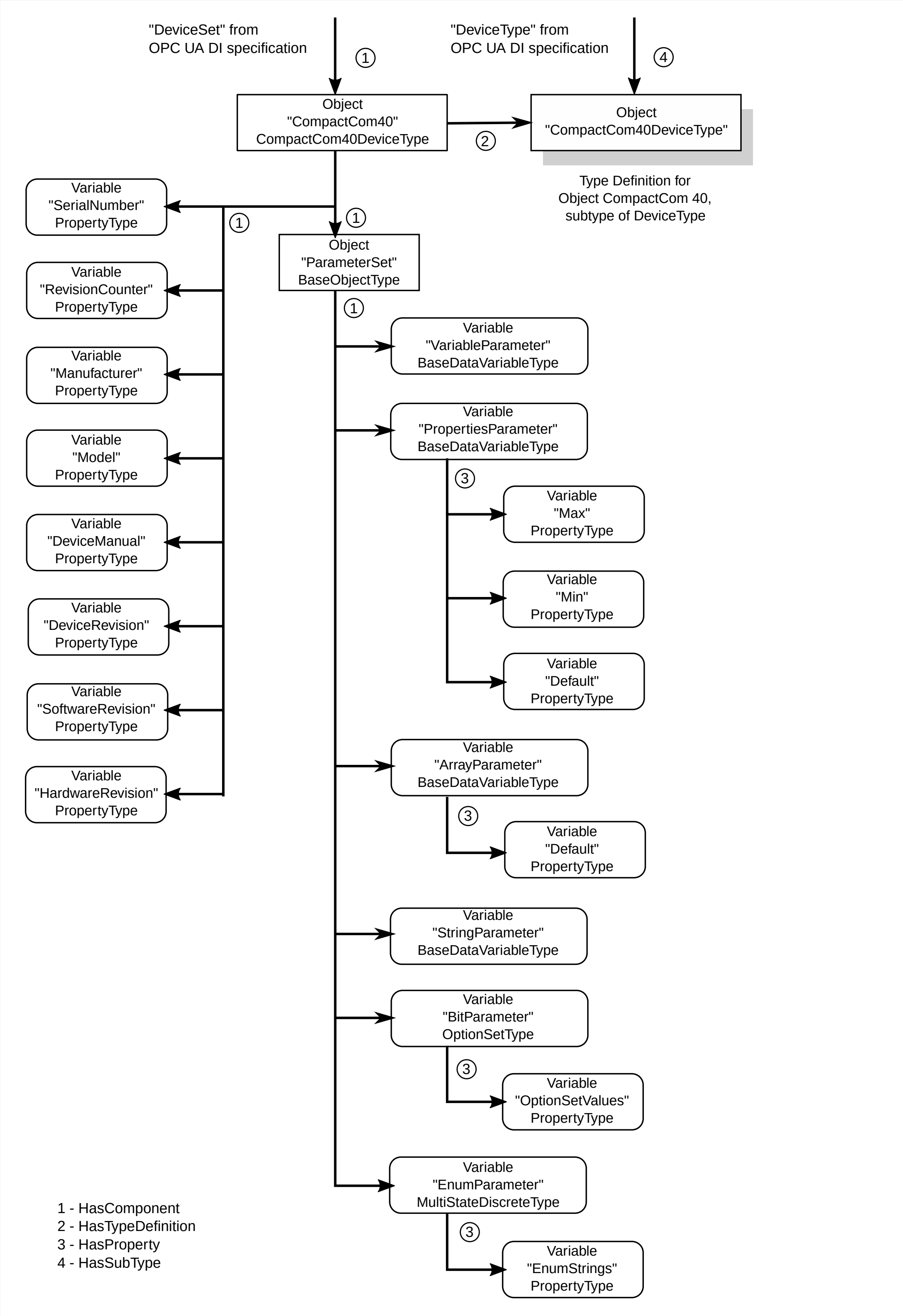

"DeviceSet" from
OPC UA DI specification
1
"DeviceType" from
OPC UA DI specification
4
Object
"CompactCom40"
CompactCom40DeviceType
2
Object
"CompactCom40DeviceType"
Type Definition for
Object CompactCom 40,
subtype of DeviceType
1
1
Object
"ParameterSet"
BaseObjectType
1
Variable
"SerialNumber"
PropertyType
Variable
"RevisionCounter"
PropertyType
Variable
"Manufacturer"
PropertyType
Variable
"Model"
PropertyType
Variable
"DeviceManual"
PropertyType
Variable
"DeviceRevision"
PropertyType
Variable
"SoftwareRevision"
PropertyType
Variable
"HardwareRevision"
PropertyType
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

Figure 27.

13.3.1. 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	CompactCom40DeviceType	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

13.3.2. 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.
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.

13.3.3. 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	
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 (page 94)	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 (page 96)
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 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 (page 99).
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 (page 96))
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 (page 96))
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 (page 96).

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 (page 94)	
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)
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 (page 91).

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

13.4. 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

13.4.1. 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 (page 94). 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 (page 94). 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	Method calls will be forwarded to the host application, see Command Details: Method_Call (page 196).
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.

13.4.2. 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.
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.

13.4.3. 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

13.5. 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.

IMPORTANT

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.

13.6. 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.

13.6.1. 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

13.6.2. 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 (page 89).

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.

13.6.3. 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
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 (page 7).

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.

IMPORTANT

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.

13.7. 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.

13.7.1. 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.

13.8. 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.

13.8.1. 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	-

13.8.2. 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) (page 194)
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) (page 194)
samplingInterval Min	1000 ms	Configurable via the OPC UA Host Object (E3h). See OPC UA Object (E3h) (page 194)
samplingInterval Max	86400000 ms	24h
queueSize Min	1	-
queueSize Max	1	-
maxPublishReqPerSession	10	-

13.8.3. 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	
MaxNodesPerMethodCall	5	Part of the OperationLimitsType

14. 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.

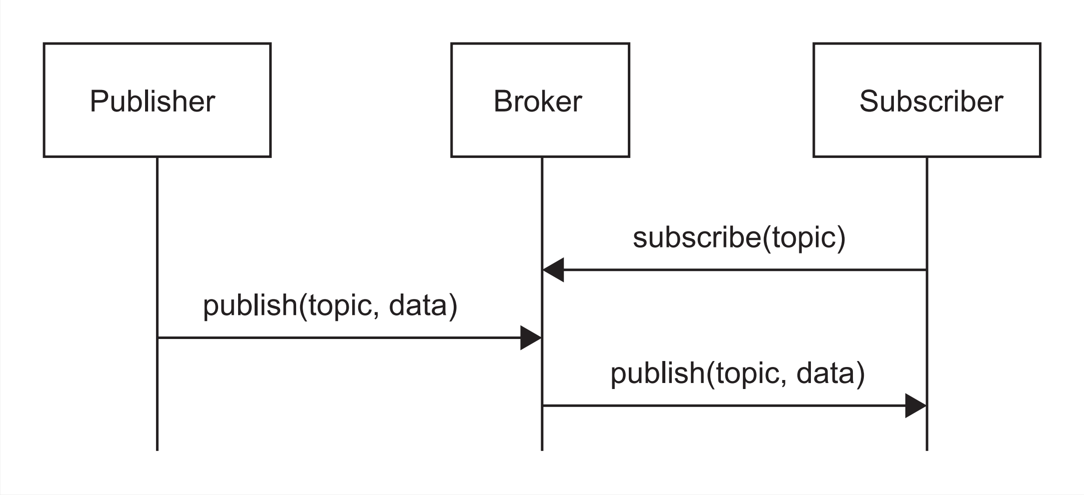


Figure 28.

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

14.1. 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

14.2. 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 (page 7).

14.3. 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.

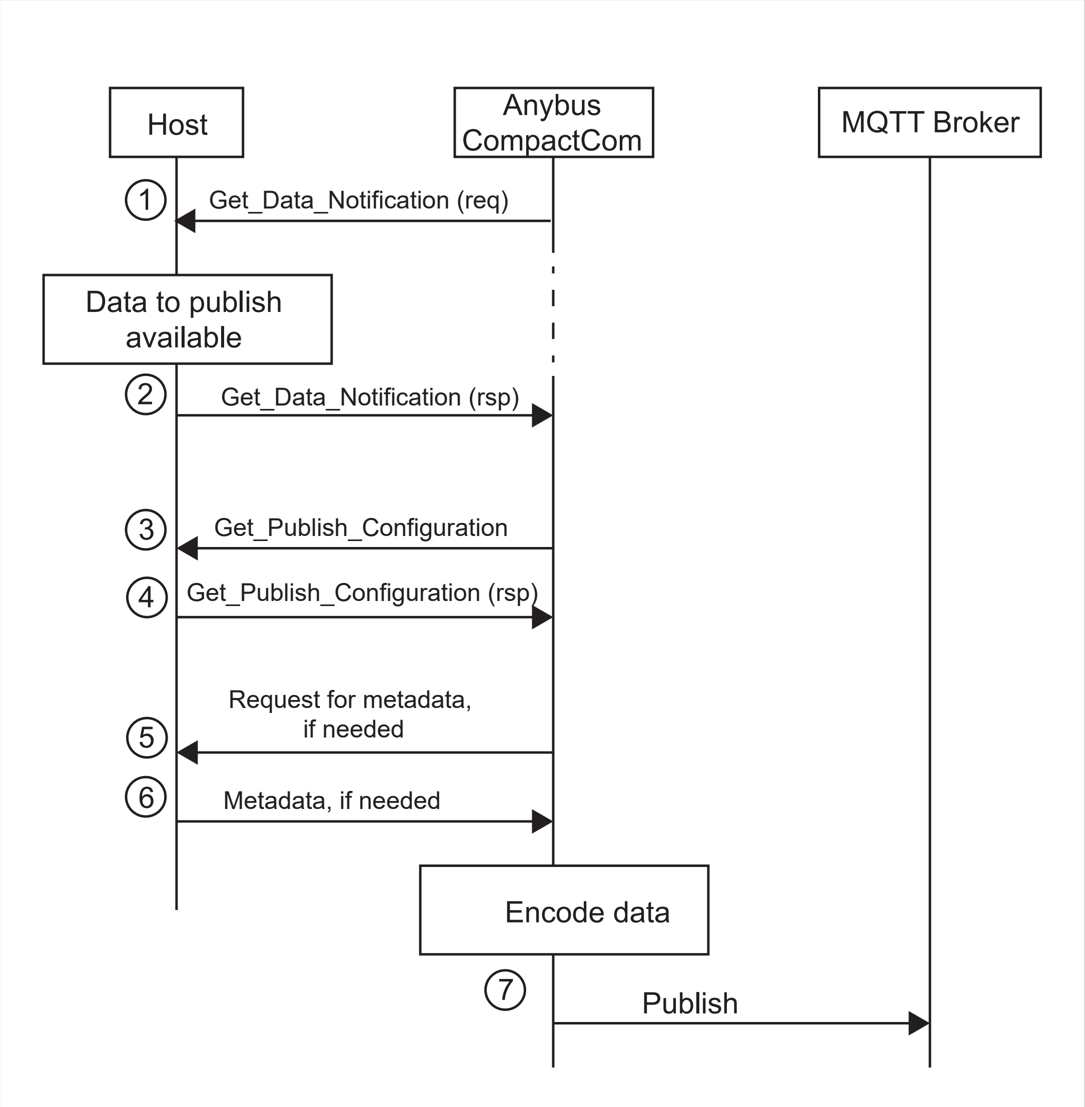

Host
Anybus CompactCom
MQTT Broker
1
Get_Data_Notification (req)
Data to publish available
2
Get_Data_Notification (rsp)
3
Get_Publish_Configuration
4
Get_Publish_Configuration (rsp)
Request for metadata, if needed
5
6
Metadata, if needed
Encode data
7
Publish

Figure 29.

14.3.1. 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.

14.3.2. Dataset Encoding

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

NOTE
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
}
```

14.4. 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

15. Anybus Module Objects

15.1. General Information

Standard Objects:

- Anybus Object (01h) (page 121)
- Diagnostic Object (02h) (page 122)
- Network Object (03h) (page 125)
- Network Configuration Object (04h) (page 129)

Network Specific Objects:

- Socket Interface Object (07h) (page 143)
- SMTP Client Object (09h) (page 160)
- Anybus File System Interface Object (0Ah) (page 163)
- Network Ethernet Object (0Ch) (page 164)
- Network PROFINET IO Object (0Eh) (page 167)
- Functional Safety Module Object (11h) (page 184)
- Time Object (13h) (page 189)

15.2. Anybus Object (01h)

15.2.1. Category

Basic

15.2.2. Object Description

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

15.2.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String

15.2.4. Object Attributes (Instance #0)

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

15.2.5. 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:	Value:	Color:
			UINT8 (LED1A)	01h	Green
			UINT8 (LED1B)	02h	Red
			UINT8 (LED2A)	01h	Green
			UINT8 (LED2B)	02h	Red
13... 16	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.	
17	Virtual attributes	Get/Set			
18	Black list/White list	Get/Set			
19	Network time	Get	UINT64	0 (Not supported)	

15.3. Diagnostic Object (02h)

15.3.1. Category

Extended

15.3.2. 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.

15.3.3. Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute

15.3.4. 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")

15.3.5. 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	NW specific data	Get	Array of UINT8	Optional network specific information, see below.
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	

Major unrecoverable events cause the module to disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic entry/alarm or a Generic Diagnostic entry/alarm, depending on the Event Code:

Event Code	Result		
0...FEh	Module issues a Channel Diagnostic entry/alarm. The Event Code will be translated and represented as the Channel Error Type as follows:		
	Code:	Event (Anybus):	Channel Error Type (PROFINET):
	10h	Generic Error	Error
	20h	Current	Short circuit
	21h	Current, device input side	Short circuit
	22h	Current, inside the device	Short circuit
	23h	Current, device output side	Short circuit
	30h	Voltage	Overvoltage
	31h	Mains Voltage	Overvoltage
	32h	Voltage inside the device	Overvoltage
	33h	Output Voltage	Overvoltage
	40h	Temperature	Overtemperature
	41h	Ambient Temperature	Overtemperature
	42h	Device Temperature	Overtemperature
	50h	Device Hardware	Error
	60h	Device Software	Error
	61h	Internal Software	Error
	62h	User Software	Error
	63h	Data Set	Error
	70h	Additional Modules	Error
	80h	Monitoring	Error
	81h	Communication	Error
	82h	Protocol Error	Error
	90h	External Error	Error
	F0h	Additional Functions	Error
FFh	Module issues a Generic Diagnostic entry/alarm based on network specific data. See details below.		

15.3.6. Details: Network Specific Data

Network specific diagnostic data serves as the payload in the PROFINET diagnostic alarm. The data contains an identifier (UserStructureIdentifier) that describes the structure of the data.

The following identifier values are supported:

- 8000h (Channel Diagnostic)
- 8002h (Extended Channel Diagnostic)
- 8003h (Qualified Channel Diagnostic)

Byte	Contents
1	UserStructureIdentifier, low byte
2	UserStructureIdentifier, high byte
3... 16	Data

15.3.7. Object Error Codes

Code	Error
03h	API does not exist
04h	No module inserted in the specified slot
05h	No submodule inserted in the specified subslot
06h	Slot number specified is out-of-range
07h	Subslot number specified is out-of-range
08h	Failed to add the channel diagnostic entry
09h	Failed to send the channel diagnostic alarm
0Ah	Channel number out-of-range
0Bh	ChannelPropType out-of-range
0Ch	ChannelPropDir out-of-range
0Dh	ChannelPropAcc out-of-range
0Eh	ChannelPropMaintReq out-of-range
0Fh	ChannelPropMaintDem out-of-range
10h	UserStructIdent out-of-range
11h	ChannelErrType out-of-range
FFh	Unknown error

15.4. Network Object (03h)

15.4.1. Category

Basic

15.4.2. Object Description

PROFINET specific information for this object is given below. For more information regarding this object, consult the general Anybus CompactCom 40 Software Design Guide.

15.4.3. 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

15.4.4. Object Attributes (Instance #0)

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

15.4.5. Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	00ADh (PROFINET IRT) + IIoT
2	Network type string	Get	Array of CHAR	“PROFINET IRT”
3	Data format	Get	ENUM	01h (MSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area. (Consult the general 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 01h Illegal value 02h Wrong data size 03h Illegal response 04h Missing MAC address (Only valid for Anybus IP) 05h Command timeout (an expected command not received within time limit) 06h The implementation of the Modular Device Object in the host application is not correct. Either Number of ADIs per module exceeds 4095 or an error response is received on the Get_List command. 07h Wrong version in PROFIenergy object 08h Wrong number of instances in PROFIenergy object

15.4.6. Command Details: Map_ADI_Write_Ext_Area

Details

Command Code:	12h
Valid for:	Instance

Description

This command is only supported by Anybus CompactCom 40 devices.

This command is equivalent to Map_ADI_Write_Area, but can map more than 256 bytes of data. It supports mapping fractional byte size types, and it can be used to map only specific parts of an ADI.

It maps an ADI as Write Process Data. If successful, the response data contains the offset, in bits, for the mapped ADI from the start of the Write Process Data area.

- Mapping an ADI more than once (i.e. map it multiple times to the Read- or Write Process Data, or map it to both the Read- and Write Process Data) is not accepted by all networks.
- It is not allowed to mix mapping commands Map_ADI_Read/Write_Area and Map_ADI_Read/Write_Ext_Area within one area (Read/Write).
- It is recommended to only map one item for each mapping command during initial development, since data area offset is only given for the first mapping item, and all mapping items may be rejected using one single error code.
- All mapped elements, except those of types BIT1-BIT7 and PADx, must be byte aligned.
- The only implicit padding done is from the very last mapped item up to byte alignment, since the process data needs to be of byte size when the setup is complete.
- Explicit padding is done either through available ADI elements of PADx type, or through the imaginary ADI 0, which is assumed to be an array with 255 elements of type PAD1. Explicit padding of process data is the only correct use of ADI 0. Padding bits might not be visible on the network.
- This command may permanently alter the state of the Anybus CompactCom 40 PROFINET IRT IIoT Secure even though the command is returned with an error. Network specific restrictions may lead to n mapping items to be accepted, but with an error on mapping item n+1. If so, the mappings up to and including n will be accepted, but all other mapping items, starting with n+1, are rejected. The number of accepted mappings is declared in CmdExt[0] of the answer.
- Certain Anybus implementations allow the network to remap the Process Data during runtime. (Consult the general Anybus CompactCom 40 Software Design Guide, Application Data Object (FEh) for further information.)

See also...

- Anybus CompactCom 40 Software Design Guide, Application Object (FFh)

> **IMPORTANT**
> Error control is only performed on the command parameters. The Anybus module does not verify the correctness of these parameters by a read of the actual ADI attributes.

- Command details:

Field	Contents
CmdExt[0]	The number of mapping items to add (0-217)
CmdExt[1]	Reserved. Set to 0
MsgData[0-1]	New mapping item 1: ADI number
MsgData[2]	New mapping item 1: Number of elements in the ADI
MsgData[3]	New mapping item 1: Index to the first element to map (0-254)
MsgData[4]	New mapping item 1: Number of consecutive elements to map (1-255)
MsgData[5]	New mapping item 1: Number of type descriptors (1-255)
MsgData[6..n]	New mapping item 1: Array of type specifiers for each mapped element
...	Repeat MsgData[0-n] (as above) for mapping item 2 and onwards.

- Response details (Success):

Field	Contents
CmdExt[0]	The number of accepted mapping items (0-217)
MsgData[0]	Bit offset of the mapped ADI from the start of the Write Process Data (Least significant byte)
MsgData[1]	Bit offset of the mapped ADI from the start of the Write Process Data
MsgData[2]	Bit offset of the mapped ADI from the start of the Write Process Data
MsgData[3]	Bit offset of the mapped ADI from the start of the Write Process Data (Most significant byte)

- Response details (Error):

Error	Contents	
Invalid CmdExt[0]	The number of accepted mapping items, before an error occurred	
Invalid State	Mapping of ADIs is only allowed in the SETUP state	
Object Specific Error	Object specific error, see MsgData[1] for details:	
	01h: Invalid data type	The data type is not valid for Process Data
	02h: Invalid number of elements	The number of elements is not valid (zero, or too many elements)
	03h: Invalid total size	The requested mapping is denied because the resulting total data size would exceed the maximum permissible (depending on network type)
	06h: Invalid map command sequence	The order in which the commands were received is invalid
	07h: Invalid mapping command	Inconsistencies in the command makes it impossible to parse
	08h: Bad alignment	The alignment rules for process data are not followed
	09h: Invalid use of ADI 0	ADI 0 is an array (255 elements) of type PAD1
	FFh: Network specific restriction	Mapping an ADI other than the previous (non-padding) ADI, must be done on a byte boundary. Bits from multiple ADIs cannot be mapped to the same byte. Indicated by network specific error code 01h (total error response is FFh FFh 01h).

Error control is only performed on the command parameters. The Anybus module does not verify the correctness of these parameters by a read of the actual ADI attributes.

15.4.7. Command Details: Map_ADI_Read_Ext_Area

Details

Command Code:	13h
Valid for:	Instance

Description

This command is only supported by Anybus CompactCom 40 devices.

This command is equivalent to Map_ADI_Read_Area, but can map more than 256 bytes of data.

It is identical to Map_ADI_Write_Ext_Area, described above, except that it maps ADIs to Read Process Data.

15.5. Network Configuration Object (04h)

15.5.1. Category

Extended

15.5.2. 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.

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

See also...

- Communication Settings (page 22)
- E-mail Client (page 85)
- Ethernet Host Object (F9h) (page 250)

IMPORTANT
Allowing the following instances to be set by the host application during start-up will inhibit the possibility to pass conformance tests.

15.5.3. Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

15.5.4. Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
3	Number of instances	Get	UINT16	22	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.)

15.5.5. 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 Multilingual Strings (page 141))
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)

IMPORTANT

This attribute should not be set by the application at every power on, as this would cause certification problems.

15.5.6. 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 Multilingual Strings (page 141))
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)

IMPORTANT

This attribute should not be set by the application at every power on, as this would cause certification problems.

15.5.7. 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 Multilingual Strings (page 141))
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)

IMPORTANT

This attribute should not be set by the application at every power on, as this would cause certification problems.

15.5.8. 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 Multilingual Strings (page 141))
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 Multilingual Strings (page 141)) Value String Meaning 00h “Disable” DHCP disabled (default) 01h “Enable” DHCP enabled
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

IMPORTANT

Do not set this unless the end user explicitly would like to turn DHCP on. Normally the PROFINET IO Controller assigns the IP address.

15.5.9. 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 Multilingual Strings (page 141))
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)

15.5.10. 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 Multilingual Strings (page 141))
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)

15.5.11. 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 Multilingual Strings (page 141))
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

15.5.12. 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 Multilingual Strings (page 141))
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

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

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP server" (Multilingual, see page Multilingual Strings (page 141))
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 (page 133) 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.

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.

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

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“SMTP user” (Multilingual, see page Multilingual Strings (page 141))
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 user name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account user name, 64 characters

15.5.15. 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 Multilingual Strings (page 141))
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

15.5.16. Instance Attributes (Instances #16 - #19)

(Reserved)

15.5.17. Instance Attributes (Instance #20, Station Name)

The Station Name identifies the Anybus module on PROFINET. If this value is changed by the host application during runtime, a reset is required in order for changes to have effect. Changes made through DCP will have immediate effect, however.

The Station Name field shall be coded as data type CHAR with 1 to 240 characters. The definition of RFC 5890 and the following syntax applies:

- 1 or more labels, separated by [.]
- Total length is 1 to 240
- Label length is 1 to 63
- Labels consist of [a-z, 0-9, -]
- Labels do not start with [-]
- Labels do not end with [-]
- The first label must not have the form “port-xyz” or “port-xyz-abcde”, where a, b, c, d, e, x, y, z = 0...9, to avoid similarity with the field AliasNameValue
- Station names must not have the form n.n.n.n, where n = 0...999

IMPORTANT
If the Station name does not pass validation by the Anybus module, an error response with error code 0Ch will be returned to the application and the Station name will not be changed/saved.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Station name” (Multilingual, see page Multilingual Strings (page 141))
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	F0h (240 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	The current Station name 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.

IMPORTANT
This attribute shall normally not be set by the application. The station name is normally set by the end user via the network. The host application shall use this attribute when the end user has the possibility to edit the station name through the application, and chooses to do so.

This attribute should not be set by the application at every power on, as this would cause certification problems.

15.5.18. Instance Attributes (Instance #21, F-Address)

This instance holds the F-Address, which is the PROFIsafe address for the safety module. This instance has no effect unless the attribute #1 (Safety enabled) is set to TRUE in the Functional Safety host object (instance #1). If the attribute is set to FALSE in the Functional Safety host object the application is advised to hide this instance to the end-user. See Functional Safety Object (E8h) (page 215)

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	“F-Address” (Multilingual, see page Multilingual Strings (page 141))
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one elements)
4	Descriptor	Get	UINT8	03h (read/write access)
5	Value	Get/Set	UINT16	F-Address set by the host application. Range: 1 - 65534 (Default: 1)
6	Configured value	Get/Set	UINT16	The configured value that will be used after restart

15.5.19. 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 Multilingual Strings (page 141))
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a restart. Actual OPC UA TCP port Range: 1 - 65535 (Default: 4840)
6	Configured value	Get	UINT16	The configured value that will be used after restart

15.5.20. 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”
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.
6	Configured value	Get	CHAR[80]	The configured value always equals the actual value (Attribute 5)

15.5.21. 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 (page 108).

IMPORTANT

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 Multilingual Strings (page 141))
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 Multilingual Strings (page 141)) 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

15.5.22. 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 Multilingual Strings (page 141))
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

15.5.23. 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 Multilingual Strings (page 141))
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.

15.5.24. 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 Multilingual Strings (page 141))
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 in seconds. Default: 60. Value set to this attribute will be used in the next connection attempt.
6	Configured value	Get	UINT16	Configured keep alive value.

15.5.25. Instance Attributes (Instance #53, MQTT Username)

This instance holds the MQTT username.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"MQTT Username" (Multilingual, see page Multilingual Strings (page 141))
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.

15.5.26. 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 Multilingual Strings (page 141))
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 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.

15.5.27. 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 Multilingual Strings (page 141))
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	255 (Max length of the MQTT Base topic 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.

15.5.28. 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 Multilingual Strings (page 141))
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.

15.5.29. 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 Multilingual Strings (page 141))
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

15.5.30. 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
8	Comm 2	Komm 2	Comu 2	Connessione 2	Comm 2
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
20	Station name	Stationsname	Nom. Estacion	Nome Stazione	Nom Station
21	F-Address	F-Adresse	Dirección-F	Indirizzo-F	F-Adresse
40	OPC TCP Port	OPC TCP Port	OPC Puerto	OPC Puerto	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
42	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
42	Endpoint	Endpoint	Endpoint	Endpoint	Endpoint
42	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

15.5.31. Command Details: Reset

Category

Details

Command Code:	05h
Valid for:	Object Instance

Description

A reset command to this object will result in that all instances are set to their default values.

It is optional to implement support for this command.

- Command Details

Field	Comments
CmdExt[0]	00h (Reserved)
CmdExt[1]	01h (Factory default reset)

- Response Details
  (No data)

15.6. Socket Interface Object (07h)

15.6.1. Category

Extended

15.6.2. 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 (page 158).

NOTE

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.

15.6.3. 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)

15.6.4. 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

15.6.5. 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)

15.6.6. 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)	

15.6.7. 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)

15.6.8. 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)	

15.6.9. 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: Mode: 00h Shutdown receive channel 01h Shutdown send channel 02h Shutdown both receive- and send channel

- Response Details
  (no data)

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

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.
2. Receive data on socket until error message Object specific error (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.

15.6.10. 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)

15.6.11. 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)

15.6.12. 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)

15.6.13. 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 (page 158)).

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 (page 158)
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 (page 158)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation (page 158)
Data[0...n]	Received data	-

15.6.14. 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 (page 158)).

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 (page 158)
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 (page 158)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation (page 158)
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	

15.6.15. 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 (page 158)).

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 (page 158)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see Message Segmentation (page 158))
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)	

15.6.16. 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 (page 158)).

- 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 (page 158)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see Message Segmentation (page 158)
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)	

15.6.17. 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)

15.6.18. 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)

15.6.19. 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	

15.6.20. 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.

15.6.21. 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.

NOTE

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)

15.7. SMTP Client Object (09h)

15.7.1. Category

Extended

15.7.2. Object Description

This object groups functions related to the SMTP client.

15.7.3. Supported Commands

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

15.7.4. 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

15.7.5. 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”

15.7.6. 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

15.7.7. 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)

15.7.8. 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)

15.7.9. 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)

15.7.10. 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)

15.8. Anybus File System Interface Object (0Ah)

15.8.1. Category

Extended

15.8.2. 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.

15.9. Network Ethernet Object (0Ch)

15.9.1. Category

Extended

15.9.2. 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 with the MAC address given in attribute #4. This information can e.g be presented on internal web pages, if present, using the JSON script language.

NOTE

Instance attributes #1 - #3 are reserved and used for backwards compatibility with earlier applications.

15.9.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

15.9.4. 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

15.9.5. Instance Attributes (Instances #1 - #3)

#	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) (page 250))
2	Port 1 MAC Address	Get	Array of UINT8	Reserved, used for backwards compatibility. (MAC address for port 1 (mandatory for the LLDP protocol)) (See also Ethernet Host Object (F9h) (page 250))
3	Port 2 MAC Address	Get	Array of UINT8	Reserved, used for backwards compatibility. (MAC address for port 2 (mandatory for the LLDP protocol)) (See also Ethernet Host Object (F9h) (page 250))
4	MAC Address	Get	Array of UINT8	MAC address of the port.
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. **Note:** This attribute is not valid for the internal port, i.e. instance #1 of the object

15.9.6. 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

15.9.7. 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

15.10. Network PROFINET IO Object (0Eh)

15.10.1. Category

Extended

15.10.2. Object Description

When the application maps ADIs to process data during start-up, the Anybus CompactCom 40 PROFINET IRT IIoT Secure will create the module configuration as described in Real Identification (RI) (page 30). The modules in the GSDML file must then be described in the same way. The GSDML file provided by HMS provides a few examples based on this way of describing modules.

If the end-user wishes to define modules in another way the application must provide the module configuration to the Anybus CompactCom 40 PROFINET IRT IIoT Secure. This is achieved by using the following commands:

- API_Add
- Plug_Module
- Plug_Submodule
- Plug_Submodule_Ext

These commands need to be sent after the process data is mapped, and before sending setup complete to the Anybus CompactCom 40 PROFINET IRT IIoT Secure.

Example:
Initially, the application maps ADIs as process data by calling all or some of the functions below:

- Map_ADI_Write_Area (10h)
- Map_ADI_Read_Area (11h)
- Map_ADI_Write_Ext_Area (12h)
- Map_ADI_Read_Ext_Area (13h)

Modules and submodules are now created based on this information as described in Real Identification (RI) (page 30).

1. Call API_Add to add an API.
2. Call Plug_Module to add a module to the API.
3. Call Plug_Submodule one or more times to add submodules to the module.
4. Repeat steps 2 and 3 to add modules to the API.

After the configuration is complete, call setup complete.

See also ...

- Flowchart —Establishment of Real Identification (RI) (page 263)

Removing and Exchanging Modules and Submodules

If the RI has been created by the host application through custom configuration, there are ways of removing modules and plug new modules during runtime.

The application will be notified by the command Cfg_Mismatch_Ind for every submodule that does not match. This information will also be provided in the command Expected_Ident_Ind. The application can then decide to remove the plugged module by issuing the command Pull_Module. This will remove the whole module and its submodules. Then, based on the information received from either Cfg_Mismatch_Ind or Expected_Ident_Ind, the application can adopt to the PLC configuration by issuing new Plug_Module, Plug_Submodule and Plug_Submodule_Ext commands.

See also ...

- Configuration Mismatch (page 31)
- Custom Configuration mismatch (page 265) (flowchart)

15.10.3. Supported Commands

Object:	Get_Attribute
	Plug_Module (see below)
	Plug_Submodule (see below)
	Plug_Submodule_Ext (see below)
	Pull_Module (see below)
	Pull_Submodule (see below)
	API_Add (see below)
	Appl_State_Ready (see below)
	AR_Abort (see below)
	IM_Options (see below)
	Ident_Change_Done (see below)
	Add_Safety_Module (see below)
Instance:	Get_Attribute

15.10.4. Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of Char	“Network PROFINET IO”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance number	Get	UINT16	0001h

15.10.5. Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Value
1	OnLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of on-line transitions
2	OffLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of off-line transitions
3	(reserved)			
4	Last AbortInd ReasonCode	Get	UINT16	Reason code for most recent Abort indication See command details for command AR_Abort
5	AddedApis	Get	UINT16	Returns the number of APIs added (including API 0)
6	ApiList	Get	Array of UINT32	First element will always be zero and the second element will contain an additional API. Length of the array is determined by parameter "AddedApis".
7	EstablishedArs	Get	UINT16	The number of Application Relationships currently established
8	ArList	Get	Array of UINT16	Array of Application Relationship handles. Length of array is determined by parameter "EstablishedArs".
9	-	-	-	-
10	Port 1 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 1 MAC address See also... - Ethernet Host Object (F9h) (page 250)
11	Port 2 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 2 MAC address See also... - Ethernet Host Object (F9h) (page 250)

15.10.6. Command Details: Plug_Module

Category

Extended

Details

Command Code:	10h
Valid for:	Object Instance

Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification.

NOTE
It is only permitted to issue this command if **API_Add** has been issued first.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API)
Data[1]	API (low word, high byte)	See also... - Application Data Instances (ADIs) (page 23)
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Destination slot for module.
Data[5]	SlotNr (high byte)	Range: 0... 0x7FFF
Data[6]	ModIdent (low word, low byte)	Module identified as stated in the GSD-file
Data[7]	ModIdent (low word, high byte)	
Data[8]	ModIdent (high word, low byte)	
Data[9]	ModIdent (high word, high byte)	

See also...

- Real Identification (RI) (page 30) (Configuration Mismatch (page 31))

15.10.7. Command Details: Plug_Submodule

Category

Extended

Details

Command Code:	11h
Valid for:	Object Instance

Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification. In such case, the Anybus will automatically issue a **Plug** or **Plug Wrong Submodule**-alarm to the IO Controller.

A submodule plugged with this command can hold IO data to the master, from the master or data in both directions. It is also possible to plug submodules which do not carry any data at all.

The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports up to 128 submodules in total.

NOTE

In case the slot number in the command is set to 0 (zero), the ADI number must also be 0 (zero), since slot 0 cannot hold any actual data.

It is only permitted to issue this command if **API_Add** has been issued first.

The **Interface**- and **Port** submodules have to be plugged in order to pass certification tests.

The interface and port submodule can only be plugged during the SETUP-state. Any attempt to plug these submodules during runtime will result in error.

No more than one safety submodule can be plugged at any time. If, during NW_INIT, it is determined that no safety module is attached, and a safety submodule is plugged in state SETUP, an exception will be raised. If an attempt to plug a safety submodule is done during runtime, when no safety module is attached, the request will be rejected.

- Command Details

Field	Contents	Comments
CmdExt[0]	Bit 0: Submodule type	0: Normal I/O submodule 1: Safety submodule (no ADI mapping, set Data[0..5] to 0)
	Bit 1-7: (reserved, set to zero)	(reserved for future use)
CmdExt[1]	(reserved, set to zero)	(reserved for future use)
Data[0]	ADI number (low byte), Read	Reference to the ADI previously mapped with Map_ADI_Read_Area.
Data[1]	ADI number (high byte), Read	
Data[2]	ADI element, Read	Reference to the element of the ADI mapped with Map_ADI_Read_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255 ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[3]	ADI number (low byte), Write	Reference to the ADI previously mapped with Map_ADI_Write_Area.
Data[4]	ADI number (high byte), Write	
Data[5]	ADI element, Write	Reference to the element of the ADI mapped with Map_ADI_Write_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255 ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[6]	API (low word, low byte)	Application Process Instance (API) See also... - Application Process Instances (API) (page 27)
Data[7]	API (low word, high byte)	
Data[8]	API (high word, low byte)	
Data[9]	API (high word, high byte)	
Data[10]	SlotNr (low byte)	Destination slot for submodule. Range: 0... 0x7FFF
Data[11]	SlotNr (high byte)	
Data[12]	SubSlotNr (low byte)	Destination subslot for submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[13]	SubSlotNr (high byte)	
Data[14]	SubModIdent (low word, low byte)	Submodule identifier as stated in the GSD-file. If the submodule is a safety module only high word is used. Low word is specified by the safety module.
Data[15]	SubModIdent (low word, high byte)	
Data[16]	SubModIdent (high word, low byte)	
Data[17]	SubModIdent (high word, high byte)	

See also...

- Real Identification (RI) (page 30) (Configuration Mismatch (page 31))

15.10.8. Command Details: Plug_Submodule_Ext

Category

Extended

Details

Command Code:	19h
Valid for:	Object Instance

Description

This is an extended version of the **Plug_Submodule** command. This command may be called during start-up to specify the Real Identification. It additionally features the possibility to associate a submodule with several consecutive ADI elements. (The **Plug_Submodule** command only allows association with one ADI element or all ADI elements.)

This command can also be called during operation if there are changes to the Real Identification. A **Plug** or **Plug Wrong Submodule**-alarm is automatically sent to the master as a result of this action.

A submodule plugged with this command can hold IO data to the master, from the master or data in both directions. It is also possible to plug submodules which do not carry any data at all.

The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports up to 128 submodules in total.

NOTE

In case the slot number in the command is set to 0 (zero), the ADI number must also be 0 (zero), since slot 0 cannot hold any actual data.

It is only permitted to issue this command if **API_Add** has been issued first.

The **Interface**- and **Port** submodules have to be plugged in order to pass certification tests.

The interface and port submodule can only be plugged during the SETUP-state. Any attempt to plug these submodules during runtime will result in error.

It is not recommended to mix **Plug_Submodule** and **Plug_Submodule_Ext** commands.

No more than one safety submodule can be plugged at any time. If, during NW_INIT, it is determined that no safety module is attached, and a safety submodule is plugged in state SETUP, an exception will be raised. If an attempt to plug a safety submodule is done during runtime, when no safety module is attached, the request will be rejected.

- Command Details

Field	Contents	Comments
CmdExt[0]	Bit 0: Submodule type	0: Normal I/O submodule 1: Safety submodule (no ADI mapping, set Data[0..7] to 0)
	Bit 1-7: (reserved, set to zero)	(reserved for future use)
CmdExt[1]	(reserved, set to zero)	(reserved for future use)
Data[0]	ADI number (low byte), Read	Reference to the ADI previously mapped with Map_ADI_Read_Area.
Data[1]	ADI number (high byte), Read	
Data[2]	First ADI element, Read	Reference to the first ADI element associated with the submodule. Range: 0 to 255 - ADI element (0 is the first element)
Data[3]	Number of consecutive ADI elements, Read	Number of consecutive elements associated with the submodule. Range: 1 to 255 (Number of elements)
Data[4]	ADI number (low byte), Write	Reference to the ADI previously mapped with Map_ADI_Write_Area.
Data[5]	ADI number (high byte), Write	
Data[6]	ADI element, Write	Reference to the first ADI element associated with the submodule. Range: 0 to 255 - ADI element (0 is the first element)
Data[7]	Number of consecutive ADI elements, Write	Number of consecutive elements associated with the submodule. Range: 1 to 255 (Number of elements)
Data[8]	API (low word, low byte)	Application Process Instance (API) See also... - Application Process Instances (API) (page 27)
Data[9]	API (low word, high byte)	
Data[10]	API (high word, low byte)	
Data[11]	API (high word, high byte)	
Data[12]	SlotNr (low byte)	Destination slot for submodule. Range: 0... 0x7FFF
Data[13]	SlotNr (high byte)	
Data[14]	SubSlotNr (low byte)	Destination subslot for submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[15]	SubSlotNr (high byte)	
Data[16]	SubModIdent (low word, low byte)	Submodule identifier as stated in the GSD file. If the submodule is a safety module only high word is used. Low word is specified by the safety module.
Data[17]	SubModIdent (low word, high byte)	
Data[18]	SubModIdent (high word, low byte)	
Data[19]	SubModIdent (high word, high byte)	

See also...

- Real Identification (RI) (page 30) (Configuration Mismatch (page 31))

15.10.9. Command Details: Pull_Module

Category

Extended

Details

Command Code:	12h
Valid for:	Object Instance

Description

This command removes a module from the configuration. Can be issued at any time. During runtime, it can be called in case there are changes to the Real Identification. The Anybus CompactCom 40 PROFINET IRT IIoT Secure then automatically issues a **Pull** or **Pull Module** alarm to the master.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API) (page 27)
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of module. Range: 0... 0x7FFF
Data[5]	SlotNr (high byte)	

15.10.10. Command Details: Pull_Submodule

Category

Extended

Details

Command Code:	13h
Valid for:	Object Instance

Description

This command removes a submodule from the configuration and can be issued at any time. During runtime, it can be called in case there are changes to the Real Identification. The Anybus CompactCom 40 PROFINET IRT IIoT Secure then automatically issues a **Pull** alarm to the master.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API) (page 27)
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of submodule. Range: 0... 0x7FFF
Data[5]	SlotNr (high byte)	
Data[6]	SubSlotNr (low byte)	Subslot number of submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[7]	SubSlotNr (high byte)	

15.10.11. Command Details: API_Add

Category

Extended

Details

Command Code:	14h
Valid for:	Object Instance

Description

By default, the module only supports API 0 (zero). If additional APIs are to be supported, or if the host application shall handle plugging/unplugging of modules and submodules, this command must be used to specify the API implementation. Note that if using this command, it is mandatory to declare API 0 (zero) prior to defining other APIs or plugging/unplugging modules/submodules. API numbers are assigned by PROFIBUS & PROFINET International (PI).

NOTE

This command may only be issued prior to setting the **Setup Complete**-attribute in the Anybus Object.

This command clears the default Real Identification created by the Anybus module while mapping ADIs to Process Data. Therefore, issuing this command effectively makes it mandatory to specify the actual Real Identification by means of the **Plug_Module** and **Plug_Submodule**-commands.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API) (page 27)
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	(reserved, set to zero)	(reserved)
Data[5]		
Data[6]		
Data[7]		

See also...

- Application Process Instances (API) (page 27)

15.10.12. Command Details: Appl_State_Ready

Category

Extended

Details

Command Code:	15h
Valid for:	Object Instance

Description

This command is only applicable if the host application implements support for **End_Of_Prm_Ind**, and signals to the module (and in turn the I/O Controller) that the host application is ready for data exchange.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

See also...

- Application Process Instances (API) (page 27)
- End_Of_Prm_Ind, command details in PROFINET IO Object (F6h) (page 228)

15.10.13. Command Details: AR_Abort

Category

Extended

Details

Command Code:	16h
Valid for:	Object Instance

Description

This command indicates to the module that the current application relationship shall be aborted.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

See also...

- Application Process Instances (API) (page 27)
- AR_Check_Ind, command details in PROFINET IO Object (F6h) (page 228)
- Expected_Ident_Ind, command details in PROFINET IO Object (F6h) (page 228)
- Appl_State_Ready, command details in PROFINET IO Object (F6h) (page 228)

15.10.14. Command Details: IM_Options

Category

Extended

Details

Command Code:	18h
Valid for:	Object Instance

Description

During startup, this command can be called to specify if I&M data for Slot0 (DAP) and/or Slot > 0 should be forwarded transparently by the Anybus CompactCom module. Additionally, it provides a way for the application to specify the I&M0 Filter Data.

I&M0 Filter Data is composed of three blocks: I&M0 Carrier Data, Module Representative Data and Device Representative Data (see table below). A submodule can belong to several blocks.

I&M0 Filter Data Contents

Content	Description
I&M0 Carrier Data	List of all submodules being a carrier of discrete I&M data Block is mandatory to support according to the PROFINET specification
Module Representative Data	List of all submodules acting as module representative Block is optional to support according to the PROFINET specification
Device Representative Data	List of at least one submodule where I&M1, I&M2, and I&M3 can be written Block is mandatory to support according to the PROFINET specification

When transparent I&M data for slot > 0 is enabled, the application must store I&M data for modules located in slots > 0 to nonvolatile memory. In this case, all modules that carry discrete I&M data shall be included in the **I&M0 Carrier Data**. (The Anybus CompactCom 40 PROFINET IRT IIoT Secure will include the DAP submodule (located in slot 0, subslot 1) in the **I&M0 Carrier Data** and **Device Representative Data**.)

When transparent I&M data for slot 0 is enabled, the application must store I&M data for slot 0 to nonvolatile memory. The DAP submodule is by default included in the **I&M0 Carrier Data** and **Device Representative Data** but may be removed from any of the blocks using the **IM_Options** command.

See also...

- Flowchart — I&M Record Data Handling (page 262).
- Get Record, command details in PROFINET IO Object (F6h) (page 228)
- Set_Record, command details in PROFINET IO Object (F6h) (page 228)

- Command Details

  If the I&M0 Filter Data is of no interest, the Data Field is left out (command length = 0).

  The command may contain one or several I&M0 Filter Data entries. The maximum amount of entries depends on the application. For a 256 bytes message channel the maximum amount of entries is 51 (256 / 5 = 51). For a 1524 bytes message channel, the maximum amount of entries is 304 (1524 / 5 = 304).

IMPORTANT

For submodules to be listed in the I&M0 Filter data, the command must be sent when the Real Identification has been determined. This means that for the “ADI Based RI” method, the command must be sent when the module has shifted to WAIT_PROCESS state. For the “Application specific RI” method the command can be sent in SETUP state but after the plugging of modules/submodules is finished (Plug_Module/Plug_Submodule).

The table below contains two entries as an example.

Field	Contents	Comments	Example
CmdExt[0]	IM_Transparent	Bitmask for enabling transparent I&M data See table below	02h
CmdExt[1]	Reserved		
Data[0]	SlotNr (low byte)	Location of the submodule that shall be part of the I&M0 Filter Data	Entry #1 - module with discrete I&M data in slot 1, subslot 1
Data[1]	SlotNr (high byte)		
Data[2]	SubSlotNr (low byte)		
Data[3]	SubSlotNr (high byte)		
Data[4]	IM0_Filter_Data	Bitmask specifying which I&M0 Filter Data block(s) the submodule shall belong to See table below	01h
Data[5]	SlotNr (low byte)		Entry #2 - module with discrete I&M data in slot 2, subslot 1
Data[6]	SlotNr (high byte)		
Data[7]	SubSlotNr (low byte)		
Data[8]	SubSlotNr (high byte)		
Data[9]	IM0_Filter_Data		01h

IM_Transparent		
Bit	Value	Description
0	0	
	1	Transparent I&M data for SlotNr = 0 (DAP)
1	0	
	1	Transparent I&M data for SlotNr not equal to 0
2 - 7	-	Reserved

IM0_Filter_Data		
Bit	Value	Description
0	0	
	1	Submodule will be included in the I&M0 Carrier Data block
1	0	
	1	Submodule will be included in the Device Representative Data block
2	0	
	1	Submodule will be included in the Module Representative Data block
3 - 7	-	Reserved

15.10.15. Command Details: Ident_Change_Done

Category

Extended

Details

Command Code:	1Ah
Valid for:	Object Instance

Description

This command shall be sent to the module when the host application has finished its adaptations of the Real Identification.

It is optional to implement support for this command, except for that it must be issued if the host application previously has responded with **Block** to the command Expected_Ident_Ind (1Bh).

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

- Response Details
  (No data)

15.10.16. Command Details: Add_Safety_Module

Category

Extended

Details

Command Code:	17h
Valid for:	Object Instance

Description

Usage of this command is conditional.

- If API_Add has been issued, ADD_Safety_Module must be issued. Any slot can be specified (1... 7FFFh).
- If API_Add has not been issued, ADD_Safety_Module is rejected. A safety module will be placed in slot 1.

The command must be called during start-up, before setup is complete. The Safety Module can be located in a valid slot (1...7FFFh). In addition to the slot number, the 16 most significant bits of the 32 bit module identification number for the Safety Module are specified. The 16 least significant bits are specified by the Safety Module itself.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	SlotNr (low byte)	Number of the slot where to insert the Safety Module. Valid values are 1... 7FFFh.
Data[1]	SlotNr (high byte)	
Data[2]	ModIdent_High (low byte)	Module identifier as stated in the GSD file (highest 16 bits).
Data[3]	ModIdent_High (high byte)	

- Response Details
  See“ Object Specific Error Codes” below.

15.10.17. Object Specific Error Codes

Code	Meaning
01h	The ADI has not been mapped with command Map_ADI_Write_Area
02h	The ADI has not been mapped with command Map_ADI_Read_Area
03h	Element does not exist for the ADI
04h	This ADI/element is already mapped
05h	API 0 must be added first
06h	API does not exist
07h	Trying to add an API already present
08h	There is no room for any more APIs
09h	Module in slot 0 cannot have any IO data
0Ah	Prior to plugging the requested module/submodule, slot 0 must be populated with a module and a submodule (in subslot 1)
0Bh	Slot occupied
0Ch	subslot occupied
0Dh	No module inserted in the specified slot
0Eh	No submodule inserted in the specified subslot
0Fh	Slot number specified is out-of-range
10h	subslot number specified is out-of-range
11h	The AR handle provided is not valid
12h	There is no application ready pending
13h	Unknown error (PROFINET IO stack denied the request)
14h	Max number of submodules have already been plugged
15h	Safety module has not been plugged
16h	ADI data type constraint
17h	Safety Module already plugged

15.11. Functional Safety Module Object (11h)

15.11.1. Category

Extended

15.11.2. 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.

15.11.3. Supported Commands

Object:	Get_Attribute Error_Confirmation Set_IO_Config_String Get_Safety_Output_PDU Get_Safety_Input_PDU
Instance:	Get_Attribute

15.11.4. 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

15.11.5. 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

15.11.6. 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 Module 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)

15.11.7. 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)

15.11.8. 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 PROFINET IRT 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

15.11.9. 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 PROFINET IRT 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

15.11.10. 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).	-

15.12. Time Object (13h)

15.12.1. Category

Extended

15.12.2. 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.

15.12.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

15.12.4. 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 (page 189). Reserved: Should not be used.

15.12.5. Instance Attributes (Instance #n)

Instance 1 is dedicated to PROFINET. Instance 2 is dedicated to OPC UA Discovery server timestamp.

> **IMPORTANT**
> The value of instance 1 is only valid if connected to a PROFINET synch master.
>
> Instance 2 is only available if OPC UA is activated with attribute 1 in OPC UA Object. See OPC UA Object (E3h) (page 194).

#	Name	Access	Data Type	Description
1	Protocol	Get	ENUM	Enumeration identifier of the time protocol. See Time Protocols (page 189) 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 (page 189) for protocol formats.

15.12.6. Time Protocols

Enum value	Priority	Protocol	Format	Epoch
4	0	PROFINET	64 bit tick value (bus cycle counter) Resolution: 31250 nanoseconds	N/A
8	2	OPC UA Discovery server timestamp	64 bit signed number of 100 nanosecond intervals	January 1, 1601

16. Host Application Objects

16.1. General Information

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

Standard Objects

- MQTT Host Object (E2h) (page 191)
- OPC UA Object (E3h) (page 194)
- Energy Measurement Object (E4h) (page 199)
- Functional Safety Object (E8h) (page 215)
- “ Application File System Object (EAh)” (see Anybus CompactCom 40 Software Design Guide)
- “ Modular Device Object (ECh)” (see Anybus CompactCom 40 Software Design Guide)
- Sync Object (EEh) (page 217)
- Energy Control Object (F0h) (page 222)
- “Application Data Object (FEh)” (see Anybus CompactCom 40 Software Design Guide)
- “ Application Object (FFh)” (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- Asset Management Object (E5h) (page 211)
- PROFINET IO Object (F6h) (page 228)
- Ethernet Host Object (F9h) (page 250)

16.2. MQTT Host Object (E2h)

16.2.1. Category

Extended

16.2.2. Object Description

This object implements MQTT functionality for the host application.

See also ...

- MQTT (page 112)

16.2.3. Supported Commands

Object:	Get_Attribute Get_Publish_Configuration
Instance:	Get_Attribute

16.2.4. 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

16.2.5. 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 length: 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 (page 112).

- 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: Meaning: TRUE: Publish the dataset with the retain bit set FALSE: 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.

16.3. OPC UA Object (E3h)

16.3.1. Category

Extended

16.3.2. Object Description

This object implements OPC UA functionality for the host application.

See also ...

- OPC UA (page 89)

16.3.3. Supported Commands

Object:	Get_Attribute Method_Call
Instance:	Get_Attribute

16.3.4. 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

16.3.5. 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.*

Table 1. 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

16.3.6. Command Details: Method_Call

Category

Extended

Details

Command Code:	10h
Valid for:	Object

Description

This command is used to forward the Call service of the OPC UA Method service set to the host application. This command is only relevant if an application defined information model is used.

An OPC UA Method performs an operation defined by the application designer when being called. To provide input to the operation and to return a result of the operation a method can define input and output arguments in property nodes.

The input and output arguments of a method call are lists of Variants (union of all OPC UA built in types), see the OPC UA Specification for more information. All OPC UA data types are derived from an OPC UA built in type. It is therefore possible to always trace the built in type of an OPC UA type. The arguments will be translated to Anybus types using the translation table of OPC UA built in types specified in Translation of Data Types (page 96).

Request format

The Method_Call request provides the NodeId of the method being called, the NodeId of the object the method is being called on, and a list of input arguments.

Parameter	Data type	Description
CmdExt[0]	UINT8	Reserved, ignore
CmdExt[1]	UINT8	Reserved, ignore
Data[0-X]	Anybus NodeId	NodeId of the Method being called For more information, see Anybus NodeId type (page 198).
	Anybus NodeId	NodeId of the Object the Method is being called on For more information, see Anybus NodeId type (page 198).
	UINT32	Number of input arguments
	UINT32	Number of output arguments
	Array of Anybus Argument	Input arguments For more information, see Anybus Argument type (page 197).

Response format

If the operation performed by the method was successful the response holds a list of output arguments to be returned. If the operation was non-successful the response will contain either an Anybus error code or an object specific code. The Anybus error code is translated to an OPC UA Status Code according to the table below. If Anybus error code is set to 0xFF (Object specific error code) the host application sends the OPC UA status code transparent through the Object specific error code parameter.

Table 2. Success response

Parameter	Data type	Description
CmdExt[0]	UINT8	Reserved, set to zero
CmdExt[1]	UINT8	Reserved, set to zero
Data[0-X]	Array of Anybus Argument	Output arguments An Array of structures describing output arguments.

Table 3. Error response

Parameter	Data type	Description
CmdExt[0]	UINT8	Reserved, set to zero
CmdExt[1]	UINT8	Reserved, set to zero
Data[0]	UINT8	Anybus error code
Data[1-4]	UINT32	Object specific error code: OPC UA status code **NOTE** This parameter should only be present if Anybus error code = 0xFF.

Anybus Argument type

Element	Data type	Description
Argument type	ENUM	0: Scalar 1: Array 2: Struct (for future use)
Number of elements	UINT8	Argument type = Scalar: Always 1 Argument type = Array: Defines number of values Argument type = Struct: Defines number of the elements in the struct
Data types	Array of UINT8	Argument type = Scalar and Array: One single UINT8 Argument type = Struct: Number of data types defined by "Number of elements" Defines the data type of the value. BitFieldMaskDataType is not supported. For translation of data types between Anybus and OPC UA, see Translation of Data Types (page 96).
Sub elements	Array of UINT16	Argument type = Scalar and Array: One single UINT16 Argument type = Struct: Number of sub elements defined by "Number of elements" Defines Number of subelements in a string if data type is OCTET or CHAR. For other data types, always 1.
Value	Any	

Anybus NodeId type

This is an Anybus representation of the OPC UA NodeId type. For more information, see the OPC UA Specification from the OPC Foundation.

Element	Data type	Description
Namespace index	UINT16	Namespace index of the NodeId
Identifier type	UINT16	NodeId type of Method NodeId 0 = Numeric 3 = String 4 = GUID 5 = Bytestring
Identifier	Union of:	Determined by Identifier type element
	UINT32	Numeric identifier
	Struct of: UINT32 CHAR[]	String identifier String length in bytes specified in the first element. Padded with one byte if length is odd.
	Struct of: UINT32 UINT16 UINT16 OCTET[8]	GUID identifier
	Struct of: UINT32 OCTET[]	ByteString identifier ByteString length in bytes specified in the first element. Padded with one byte if length is odd.

Anybus error code to OPC UA status code translation

See Translation of Data Types (page 96) for common translation of Anybus error codes. In the table below follows the error codes which depends on function specific translation.

Anybus error code	OPC UA Status code
Unsupported object	Bad_InternalError
Unsupported instance	Bad_InternalError
Unsupported command	Bad_InternalError
Invalid Cmd Ext 0	Bad_InternalError
Invalid Cmd Ext 1	Bad_InternalError
Object specific error	Transparent OPC UA status code specified in Error response (page 197).

In case the output arguments in the response does not have the right format according to the information model Bad_InternalError will be returned and the error counter Discarded Responses in the Anybus object will be increased.

16.4. Energy Measurement Object (E4h)

16.4.1. Category

Extended

16.4.2. Object Description

This object defines a standardized way of reporting different types of measurement values (current, voltage, power, energy etc.) from a measuring device.

Each instance, logical device, represents a group of measurement values for one separate measurement device. Physically, measuring devices might be connected to a higher level network directly, or indirectly via a coupling device. To enable host applications to implement either a direct or an indirect measuring device, the Anybus CompactCom object model implies a 1:1 relationship between measuring device (a.k.a. logical device) and instance number, where each instance attribute represent a group of measurement values (see list of instance attributes below). The host application does not have to support all instance attributes listed.The command Get_Attribute_Measurement_List is used towards the host application, to read out which measurement values are supported by a device.

The table below shows an example of how the instances of the Anybus CompactCom object model maps to PROFIenergy. PROFIenergy uses the Object number to address a logical device, while the Measurement ID refers to the actual measurement value.

Instance number	Attribute number	PROFINET		Description
(logical device)	(measurement value)	Obj nr	Measurement ID	
1	1	0	1,2,3,31	Logical device #1 #1-Voltage PH-N #4-Voltage PH-PH #10-Current
	4		4,5,6,32	
	10		7,8,9,33	
2	1	1	1,2,3,31	Logical device #2 #1-Voltage PH-N #4-Voltage PH-PH #10-Current
	4		4,5,6,32	
	10		7,8,9,33	
3	13	2	10,11,12,36	Logical device #3 #13-Apparent Power #16-Active Power #19-Reactive Power
	16		13,14,15,34	
	19		16,17,18,35	

See also...

- PROFIenergy Profile (page 37)

16.4.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Get_Attribute_Measurement_List

16.4.4. Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Energy Measurement”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	One instance per supported measurement device
4	Highest instance no.	Get	UINT16	Max 8, depending of how many instances have been implemented

16.4.5. Instance Attributes (Instance #1 - #8)

Each instance contains a set of attributes, where each attribute represents a group of measurement values. Each value can be associated with a timestamp, indicating when the recording took place. If the timestamp is set to 0, the value is not associated with a timestamp. The timestamp is given as number of milliseconds elapsed since 00.00.00 Coordinated Universal Time (UTC), Thursday, January 1st 1970, not counting leap seconds. Example:

```
2017-02-01 12:16:30 = 1 485 951 390 000 ms
```

It is advised that the host application store attributes #29 - #31 in non-volatile memory.

- Abbreviations used:
  - TS – Timestamp
  - Avg – Average
  - PH – Phase (L1–Line 1, L2 –Line 2, L3 – Line 3)
  - N – Neutral, G – Ground
- “-” indicates the measurement value has no representation on PROFIenergy.

The attributes can be divided in different categories, depending on the type of the variable they represent. See the table immediately following this one, for more information.

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
1	Get	VoltagePH-N	UINT64 FLOAT UINT32	0: L1-N TS 1: L1-N Value 2: Reserved	1
			UINT64 FLOAT UINT32	3: L2-N TS 4: L2-N Value 5: Reserved	2
			UINT64 FLOAT UINT32	6: L3-N TS 7: L3-N Value 8: Reserved	3
			UINT64 FLOAT UINT32	9: Avg L-N TS 10: Avg L-N Value 11: Reserved	31
2	Get	Min VoltagePH-N	UINT64 FLOAT UINT32	0: L1-N TS 1: L1-N Value 2: Reserved	70
			UINT64 FLOAT UINT32	3: L2-N TS 4: L2-N Value 5: Reserved	71
			UINT64 FLOAT UINT32	6: L3-N TS 7: L3-N Value 8: Reserved	72
			UINT64 FLOAT UINT32	9: Avg L-N TS 10: Avg L-N Value 11: Reserved	92
3	Get	Max VoltagePH-N	UINT64 FLOAT UINT32	0: L1-N TS 1: L1-N Value 2: Reserved	40
			UINT64 FLOAT UINT32	3: L2-N TS 4: L2-N Value 5: Reserved	41
			UINT64 FLOAT UINT32	6: L3-N TS 7: L3-N Value 8: Reserved	42
			UINT64 FLOAT UINT32	9: Avg L-N TS 10: Avg L-N Value 11: Reserved	62
4	Get	VoltagePH-PH	UINT64 FLOAT UINT32	0: L1-L2 TS 1: L1-L2 Value 2: Reserved	4
			UINT64 FLOAT UINT32	3: L2-L3 TS 4: L2-L3 Value 5: Reserved	5
			UINT64 FLOAT UINT32	6: L3-L1 TS 7: L3-L1 Value 8: Reserved	6
			UINT64 FLOAT UINT32	9: Avg L-L TS 10: Avg L-L Value 11: Reserved	32

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
5	Get	Min VoltagePH-PH	UINT64 FLOAT UINT32	0: L1-L2 TS 1: L1-L2 Value 2: Reserved	73
			UINT64 FLOAT UINT32	3: L2-L3 TS 4: L2-L3 Value 5: Reserved	74
			UINT64 FLOAT UINT32	6: L3-L1 TS 7: L3-L1 Value 8: Reserved	75
			UINT64 FLOAT UINT32	9: Avg L-L TS 10: Avg L-L Value 11: Reserved	93
6	Get	Max VoltagePH-PH	UINT64 FLOAT UINT32	0: L1-L2 TS 1: L1-L2 Value 2: Reserved	43
			UINT64 FLOAT UINT32	3: L2-L3 TS 4: L2-L3 Value 5: Reserved	44
			UINT64 FLOATUINT32	6: L3-L1 TS 7: L3-L1 Value 8: Reserved	45
			UINT64 FLOAT UINT32	9: Avg L-L TS 10: Avg L-L Value 11: Reserved	63
7	Get	Voltage PH-G	UINT64 FLOAT UINT32	0: L1-G TS 1: L1-G Value 2: Reserved	-
			UINT64 FLOAT UINT32	3: L2-G TS 4: L2-G Value 5: Reserved	
			UINT64 FLOAT UINT32	6: L3-G TS 7: L3-G Value 8: Reserved	
			UINT64 FLOAT UINT32	9: Avg L-G TS 10: Avg L-G Value 11: Reserved	
8	Get	Min VoltagePH-G	UINT64 FLOAT UINT32	0: L1-G TS 1: L1-G Value 2: Reserved	-
			UINT64 FLOAT UINT32	3: L2-G TS 4: L2-G Value 5: Reserved	
			UINT64 FLOAT UINT32	6: L3-G TS 7: L3-G Value 8: Reserved	
			UINT64 FLOAT UINT32	9: Avg L-G TS 10: Avg L-G Value 11: Reserved	
9	Get	Max VoltagePH-G	UINT64 FLOAT UINT32	0: L1-G TS 1: L1-G Value 2: Reserved	-

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
			UINT64	3: L2-G TS	
			FLOAT	4: L2-G Value	
			UINT32	5: Reserved	
			UINT64	6: L3-G TS	
			FLOAT	7: L3-G Value	
			UINT32	8: Reserved	
			UINT64	9: Avg L-G TS	
			FLOAT	10: Avg L-G Value	
			UINT32	11: Reserved	
10	Get	Current	UINT64	0: L1 TS	7
			FLOAT	1: L1 Value	
			UINT32	2: Reserved	
			UINT64	3: L2 TS	8
			FLOAT	4: L2 Value	
			UINT32	5: Reserved	
			UINT64	6: L3 TS	9
			FLOAT	7: L3 Value	
			UINT32	8: Reserved	
			UINT64	9: Avg L TS	33
			FLOAT	10: Avg L Value	
			UINT32	11: Reserved	
			UINT64	12: N TS	-
			FLOAT	13: N Value	
			UINT32	14: Reserved	
11	Get	Min Current	UINT64	0: L1 TS	76
			FLOAT	1: L1 Value	
			UINT32	2: Reserved	
			UINT64	3: L2 TS	77
			FLOAT	4: L2 Value	
			UINT32	5: Reserved	
			UINT64	6: L3 TS	78
			FLOAT	7: L3 Value	
			UINT32	8: Reserved	
			UINT64	9: Avg L TS	94
			FLOAT	10: Avg L Value	
			UINT32	11: Reserved	
			UINT64	12: N TS	-
			FLOAT	13: N Value	
			UINT32	14: Reserved	
12	Get	Max Current	UINT64	0: L1 TS	46
			FLOAT	1: L1 Value	
			UINT32	2: Reserved	
			UINT64	3: L2 TS	47
			FLOAT	4: L2 Value	
			UINT32	5: Reserved	
			UINT64	6: L3 TS	48
			FLOAT	7: L3 Value	
			UINT32	8: Reserved	
			UINT64	9: Avg L TS	64
			FLOAT	10: Avg L Value	
			UINT32	11: Reserved	

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
			UINT64 FLOAT UINT32	12: N TS 13: N Value 14: Reserved	-
13	Get	Apparent Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	10
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	11
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	12
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	36
14	Get	Min Apparent Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	79
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	80
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	81
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	97
15	Get	Max Apparent Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	49
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	50
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	51
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	67
16	Get	Active Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	13
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	14
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	15
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	34

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
17	Get	Min Active Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	82
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	83
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	84
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	95
18	Get	Max Active Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	52
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	53
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	54
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	65
19	Get	Reactive Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	16
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	17
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	18
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	35
20	Get	Min Reactive Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	85
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	86
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	87
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	96
21	Get	Max Reactive Power	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	55

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	56
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	57
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	66
22	Get	Power factor	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	19
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	20
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	21
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	37
23	Get	Min Power factor	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	88
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	89
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	90
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	98
24	Get	Max Power factor	UINT64 FLOAT UINT32	0: L1 TS 1: L1 Value 2: Reserved	58
			UINT64 FLOAT UINT32	3: L2 TS 4: L2 Value 5: Reserved	59
			UINT64 FLOAT UINT32	6: L3 TS 7: L3 Value 8: Reserved	60
			UINT64 FLOAT UINT32	9: Total TS 10: Total Value 11: Reserved	68
25	Get	Frequency	UINT64 FLOAT UINT32	0: Line frequency TS 1: Line frequency Value 2: Reserved	30
26	Get	Min Frequency	UINT64 FLOAT UINT32	0: Line frequency TS 1: Line frequency Value 2: Reserved	91

Attr ID#	Access Rule	Name	Data Type	Element number:description	PROFIenergy Measurement ID
27	Get	Max Frequency	UINT64 FLOAT UINT32	0: Line frequency TS 1: Line frequency Value 2: Reserved	61
28	Get	Field rotation	UINT64 FLOAT UINT32	0: Field rotation TS 1: Field rotation Value 2: Reserved	-
29	Get	Total Active energy(Sum = Consumed – Generated)	UINT64 FLOAT UINT32	0: Consumed TS 1: Consumed Value 2: Reserved	200
			UINT64 FLOAT UINT32	3: Generated TS 4: Generated Value 5: Reserved	201
			UINT64 FLOAT UINT32	6: Sum TS 7: Sum Value 8: Reserved	205
30	Get	Total Reactive energy(Sum = Consumed – Generated)	UINT64 FLOAT UINT32	0: Consumed TS 1: Consumed Value 2: Reserved	202
			UINT64 FLOAT UINT32	3: Generated TS 4: Generated Value 5: Reserved	203
			UINT64 FLOAT UINT32	6: Sum TS 7: Sum Value 8: Reserved	206
31	Get	Total Apparent energy	UINT64 FLOAT UINT32	0: Consumed TS 1: Consumed Value 2: Reserved	-
			UINT64 FLOAT UINT32	3: Generated TS 4: Generated Value 5: Reserved	-
			UINT64 FLOAT UINT32	6: Sum TS 7: Sum Value 8: Reserved	204

The table below describes how the generic values for the different variable categories are represented on PROFINET.

Attr #	Variable category	Unit	Comments
1-9	Voltage	V	RMS value Absolute value
10-12	Current	A	RMS value Positive values indicate consumed current; negative values indicate generated current.
13-15	Apparent Power	VA	Positive values indicate consumed apparent power; negative values indicate generated apparent power.
16-18	Active Power	W	Positive values indicate consumed active power; negative values indicate generated active power.
19-21	Reactive Power	VAR	Positive values indicates inductive reactive power; negative values indicates capacitive reactive power.
22-24	Power factor		Range:-1.0 to 1.0. Positive values indicate leading power factor; negative values indicate lagging power factor.
25-27	Frequency	Hz	Absolute value
28	Field Rotation		0: no rotation recognized (not supported) 1: Clock-wise -1: Counter clock-wise
29	Total Active energy	Wh	For the Import- and Export variables an absolute value is used.For the Sum variable a positive value indicates consumed energy; a negative value indicates generated energy.
30	Total Reactive energy	VARh	For the Consumed and Generated variables an absolute value is used. For the Sum variable a positive value indicates consumed reactive energy (inductive load); a negative value indicates generated reactive energy (capacitive load).
31	Total Apparent energy	VAh	Absolute value

16.4.6. Command Details: Get_Attribute_Measurement_List

Details

Command Code	10h
Valid for:	Instance

Description

This command is used to gather all supported attribute elements of an instance. If several instances are supported, the command will be sent numerous times (implied by object attribute Highest instance number). Based on the response(s) the Anybus CompactCom will provide information to the network indicating which measurement values that are supported by a device.

The command response shall contain all attributes up to the last supported attribute number. For unsupported attributes within this range the bit field shall be set to zero.

The table below shows an example of the expected response for a (multiple phase) device supporting several measurement values for attributes #1, #4 and #10.

Data[0-1]	0x000F	Attribute #1 - measurement values 1-4 supported (i.e. L1-L2-L3-Avg)
Data[2-3]	0x0000	Attribute #2 - attribute not supported
Data[4-5]	0x0000	Attribute #3 – attribute not supported
Data[6-7]	0x000F	Attribute #4 - measurement values 1-4 supported (i.e. L1-L2-L3-Avg)
Data[8-9]	0x0000	Attribute #5 – attribute not supported
Data[10-11]	0x0000	Attribute #6 – attribute not supported
Data[12-13]	0x0000	Attribute #7 – attribute not supported
Data[14-15]	0x0000	Attribute #8 – attribute not supported
Data[16-17]	0x0000	Attribute #9 – attribute not supported
Data[18-19]	0x000F	Attribute #10 – measurement values 1-4 supported (i.e. L1-L2-L3-Avg)

The table below shows an example of the expected response for a (single phase) device supporting one measurement value for attributes #1, #4 and #10.

Data[0-1]	0x0001	Attribute #1 - measurement value 1 supported (i.e. L1)
Data[2-3]	0x0000	Attribute #2 - attribute not supported
Data[4-5]	0x0000	Attribute #3 – attribute not supported
Data[6-7]	0x0001	Attribute #4 - measurement value 1 supported (i.e. L1)
Data[8-9]	0x0000	Attribute #5 – attribute not supported
Data[10-11]	0x0000	Attribute #6 – attribute not supported
Data[12-13]	0x0000	Attribute #7 – attribute not supported
Data[14-15]	0x0000	Attribute #8 – attribute not supported
Data[16-17]	0x0000	Attribute #9 – attribute not supported
Data[18-19]	0x0001	Attribute #10 – measurement values 1 supported (i.e. L1)

- Command Details

Field	Contents
CmdExt[0 - 1]	Reserved, set to 00h

- Response Details

Field	Contents	Description
Data[0 - 1]	Attribute #1	Data type: BITS16 Bit 0 – First element Bit 1 – Second element ... Bit 15 – Last element Bit = 0: element not supported Bit = 1: element supported
Data[2 - 3]	Attribute #2	
Data[...]	...	
Data[(N*2-2) – (N*2-1)]	Attribute #N (last supported attribute number)	

16.5. Asset Management Object (E5h)

16.5.1. Category

Extended

16.5.2. Object Description

This object implements asset management information, i.e. identification information about one or several non PROFINET components of a device. Each component is represented by one object instance. The application collects identification information and assembles it in the instance attributes for each object instance.

16.5.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

16.5.4. Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Asset management”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	Max 32. Depends on number of non PROFINET components.
4	Highest instance no.	Get	UINT16	Max 32. Depends on number of non PROFINET components.

16.5.5. Instance Attributes (Instance #1 - #32)

Asset management data for a non PROFINET component is represented by one of three different formats depending on the complexity of the component. For a complex component based on both hardware and software the format AM_Full information comply. For hardware-only components the format AM_HardwareOnly complies, and for software-only components the AM_FirmwareOnly format is used.

Which format to use for a specific component is selected by attribute AM_InfoType.

Both AM_HardwareOnly and AM_FirmwareOnly are subsets of AM_Full. If AM_HardwareOnly is selected the attributes AM_SoftwareRevision and IM_Software_Revision are of no interest and thus not read by the Anybus CompactCom. In the same manner, if AM_FirmwareOnly is selected the attributes AM_HardwareRevision and IM_Hardware_Revision are of no interest and thus not read by the Anybus CompactCom.

#	Name	Access	Data Type	Default Value	Comment
1	AM info Type	Get	UINT8	-	Asset management data representation 0: AM_FullInfo 1: AM_HardwareOnlyInfo; attributes AM Software Revision and IM Software Revision are irrelevant 2: AM_FirmwareOnlyInfo; attributes AM Hardware Revision and IM Hardware Revision are irrelevant
2	IM Unique Identifier	Get	Array of UINT8	0 (unused)	128 bit UUID. Manufacturer created unique defined identifier. Used as source identification if issuing an event using reporting system or delivering asset management information ISO/IEC 9834-8:2014 specifies the format and generation rules that enable users to produce 128-bit unique identifiers. Number of elements in array: 16.
3	Location Type	Get	UINT8	-	Asset management includes two concepts how to locate an asset. See section on AM_Location below for more information. 1: Twelve level tree format (LT) 2: Slot- and SubslotNumber format (SS)
4	Location LT	Get	Array of UINT16	-	Mandatory if Location Type is “1” (LT). If not present any request will increase the error counter and the instance will not be reported on the network. See section on AM_Location below for usage. Maximum number of elements in array: 12.
5	Location SS	Get	struct of UINT16 BeginSlot UINT16 BeginSubslot UINT16 EndSlot UINT16 EndSubslot	-	Mandatory if Location Type is “2” (´SS). If not present any request will increase the error counter and the instance will not be reported on the network. See section on AM_Location below for usage. Maximum number of elements in array: 12.
6	IM Annotation	Get	Array of CHAR	64 blanks	Manufacturer defined description, The value shall be set to a manufacturer specific identifier, and can be used for the name of the component. Maximum number of elements in array: 64.
7	IM Order ID	Get	Array of CHAR	64 blanks	Manufacturer defined order id Maximum number of elements in array: 64.
8	IM Serial Number	Get	Array of CHAR	16 blanks	Manufacturer defined serial number Maximum number of elements in array: 16.
9	AM Device Identification	Get	Array of UINT16	-	The content of this attribute can be used for coding of device, vendor and organization identification. See details below. Number of elements in array: 4.

#	Name	Access	Data Type	Default Value	Comment
10	AM Type Identification	Get	UINT16	0	0000h Unspecified 0001h PLC 0002h PC based station 0003h IO Module or IO Submodule 0004h Communication Module or Communication Submodule 0005h Interface Module or Interface Submodule 0006h Active Network Infrastructure Component 0007h Media converter 0100h - 7FFFh Shall be defined by the manufacturer of the reporting entity Other (reserved)
11	AM Software Revision	Get	Array of CHAR	64 blanks	Manufacturer defined software revision. Maximum number of elements in array: 64. Example: “V1.3.0” If attribute #12, IM Software Revision, is supported, this attribute should not be implemented.
12	IM Software Revision	Get	struct of CHAR 3*UINT8	V0.0.0	Manufacturer defined software revision.
13	AM Hardware Revision	Get	Array of CHAR	64 blanks	Manufacturer defined hardware revision Maximum number of elements in array: 64 If attribute #14, IM Hardware Revision, is supported, then this attribute should not be implemented.
14	IM Hardware Revision	Get	UINT16	0	Manufacturer defined hardware revision 0000h-FFFFh (FFFFh indicates availability of profile specific information)

16.5.6. AM_Location

Asset management include two concepts of how to locate an asset; “Twelve level tree format” and “Slot- and Subslot number format”.

Location Concept: Slot- and Subslot Number

In case of a remote IO device the location of a component (e.g. rack module) is clearly defined by the slot of the IO device. Other components may span several slots, such as a backplane, or several subslots, such as a terminal block. This means each asset can be located by “BeginSlotNumber/BeginSubslotNumber” and “EndSlotNumber/EndSubslotNumber”.

For this concept the following parameters have to be considered:

Array index	Description
0	BeginSlotNumber
1	BeginSubslotNumber
2	EndSlotNumber
3	EndSubslotNumber

NOTE

It is important that the defined slot-/subslot numbers (i.e. BeginSlotNumber, EndSlotNumber, BeginSubslotNumber and EndSubslotNumber) are present in the Real Identification. This is checked at conformance testing.

Location Concept: Twelve Level Tree Format

In case of a remote IO device the location of a component is clearly defined by the level information. The Twelve level tree format may be used for modular devices with hierarchical device structure. This means each asset can be located by a level number 0-11 (e.g. x.y.z level notation). If LT format is used a value must be provided at least for level 0. If a shorter array is supplied value 03FFh is implied for the missing levels.

For this concept the following parameters have to be considered:

Array index	Description	Value
0	Level 0	0000h - 03FEh: Address information to identify a reported node 03FFh: Level not used 0400h-0xFFFFh: Reserved
1	Level 1	
2	Level 2	
3	Level 3	
4	Level 4	
5	Level 5	
6	Level 6	
7	Level 7	
8	Level 8	
9	Level 9	
10	Level 10	
11	Level 11	

16.5.7. AM_DeviceIdentification

Array index	Description
0	Organization (0=PI) 0: Must be used 1–FFFFh: reserved
1	Vendor ID
2	Device ID
3	Device Sub ID: 0: Must be used 1–FFFFh: reserved

16.6. Functional Safety Object (E8h)

16.6.1. Category

Extended

16.6.2. Object Description

IMPORTANT
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.

16.6.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

16.6.4. 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

16.6.5. 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)				
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.

16.7. Application File System Interface Object (EAh)

16.7.1. Category

Extended

16.7.2. 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.

16.8. Sync Object (EEh)

16.8.1. Category

Extended

16.8.2. Object Description

This object contains the host object sync settings.

The implementation of this object is optional. If it is not implemented the Anybus CompactCom 40 PROFINET IRT IIoT Secure can not be used for a synchronous application.

If there is any problem with the configuration of the sync functionality as a whole, the application must indicate this in the application status register. The Anybus CompactCom will then change states and inform the problem to the PROFINET IO Controller, see Application Status Register (page 259).

See also ...

- Anybus CompactCom 40 Software Design Guide, “Sync”
- Anybus CompactCom 40 Software Design Guide, “Sync Object”

16.8.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

16.8.4. Object Attributes (Instance #0)

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

16.8.5. Anybus CompactCom Sync and PROFINET Isochronous Mode

The sync functionality is described differently in the PROFINET network specification than how it is described in the specification for Anybus CompactCom in general. See the Anybus CompactCom 40 Software Design Guide for a detailed description of the Anybus CompactCom sync functionality.

This section describes the correlation between the specifications.

The figure below shows a timing diagram for PROFINET isochronous mode.

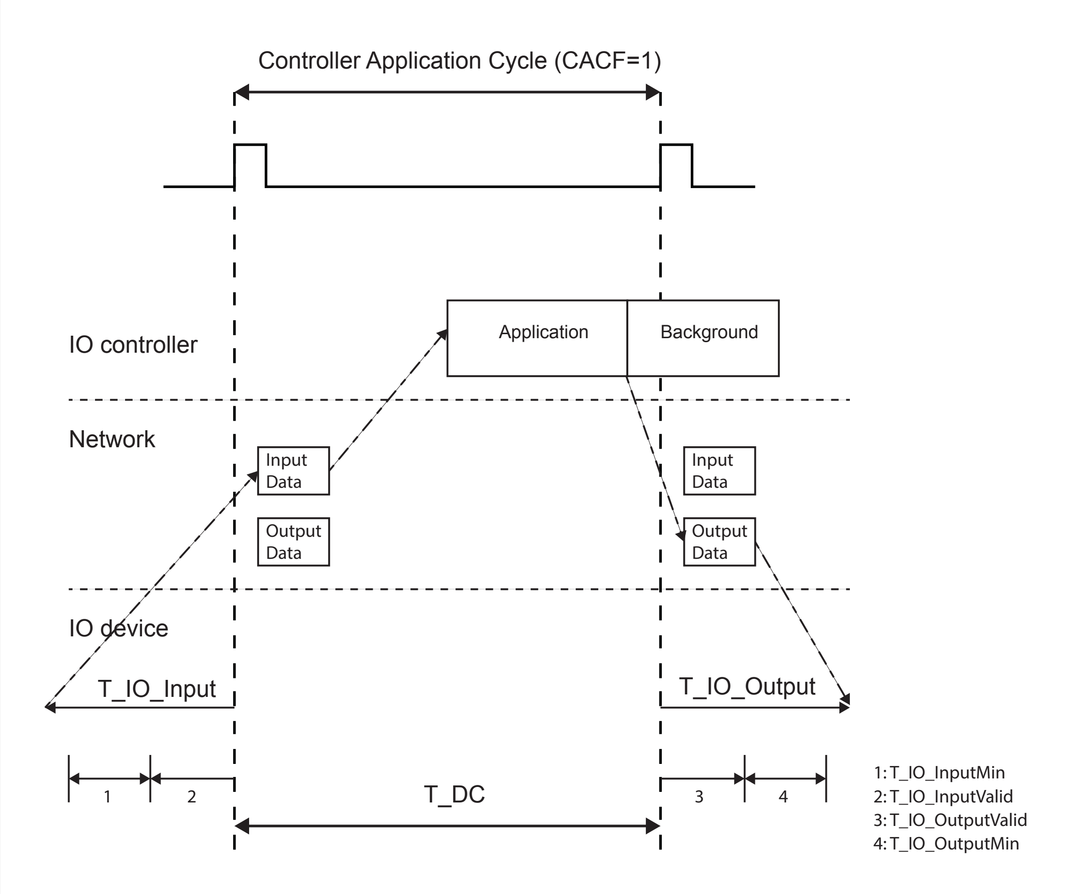


Figure 30.

T_IO_OutputMin

T_IO_OutputMin consists of two delays:

- The delay added by the Anybus CompactCom. This is the time from when the message is available on the network until it is available to the application. This delay is 0 in the current implementation.
- The delay added by the application. This is the time it takes for the application from when it is notified that new process data has arrived, to when the process data is copied and the output is valid. This delay must be measured by the application designer. The measured value shall be written to Attribute #4 (Output Processing).

The value of T_IO_OutputMin must be entered in the GSD file for every submodule supporting synchronous operation. See example GSD entries below.

T_IO_InputMin

T_IO_InputMin consists of two delays:

- The delay added by the Anybus CompactCom. This is the time from when the message is available in the device until it is available on the network. This delay is 12 µs in the current implementation.
- The delay added by the application. This is the time it takes for the application from when inputs are captured, to when the input data is available to the Anybus CompactCom. This delay must be measured by the application designer. The measured value shall be written to Attribute #5 (Input Processing).

The value of T_IO_InputMin must be entered in the GSD file for every submodule supporting synchronous operation. See example GSD entries below.

T_IO_Output

T_IO_Output is equivalent to attribute #2 (Output Valid). The value is configured from the IO controller.

T_IO_Input

T_IO_Input is the configured Cycle Time (T_DC) minus the Input Capture time (Attribute #3). The value is configured from the IO controller.

T_DC

T_DC = Cycle Time (Attribute #1)

16.8.6. GSDML Entries

The following must be added to the GSDML file for modules supporting isochronous operation:

```
IsochroneModeInRT_Classes=”RT_CLASS_3” in the <InterfacesubmoduleItem>
<IsochroneMode T_DC_Base="8" T_DC_Min="1" T_DC_Max="16"
T_IO_Base="1000"    T_IO_OutputMin="X" T_IO_InputMin="Y"/>
```

16.8.7. Instance Attributes (Instance #1)

#	Name	Access	Corresponding term for PROFINET	Data Type	Value
1	Cycle time	Get/Set	T_DC	UINT32	Application cycle time in nanoseconds
2	Output valid	Get/Set	T_IO_Output	UINT32	Output valid point relative to SYNC events, in nanoseconds Default value: 0
3	Input capture	Get/Set	T_DC - T_IO_Input	UINT32	Input capture point relative to SYNC events, in nanoseconds Default value: 0
4	Output processing	Get	T_IO_OutputMin	UINT32	Minimum required time, in nanoseconds, between RDPDI interrupt and “Output valid”
5	Input processing	Get	T_IO_InputMin - 12 µs	UINT32	Maximum required time, in nanoseconds, from “Input capture” until write process data has been completely written to the Anybus CompactCom module
6	Min cycle time	Get	T_DC_Min	UINT32	Minimum cycle time supported by the application, in nanoseconds
7	Sync mode	Get/Set	-	UINT16	This attribute is used to select synchronization mode. It enumerates the bits in attribute 8 0: Nonsynchronous mode. (Default value if nonsynchronous mode is supported) 1: Synchronous mode 2 - 65535: Reserved. Any attempt to set sync mode to an unsupported value shall generate an error response
8	Supported sync modes	Get	-	UINT16	A list of the synchronization modes the application supports. Each bit corresponds to a mode in attribute 7 Bit 0: 1 = Nonsynchronous mode supported Bit 1: 1 = Synchronous mode supported (Required for synchronous application) Bit 2 - 15: Reserved (0)
9	Control Task cycle factor	Get/Set	CACF (Controller Application Cycle Factor)	UINT16	If the synchronous control task operates at a cycle that is longer than the data cycle (see attribute #1, Cycle time) then this attribute provides a scaling factor for the cycle time such that: Control task duration = Control task cycle factor x Cycle time. The information may be used e.g. to interpolate output values, if required by the process. Note that synchronization to the Control Task cycle must be done by the host application. Default: 1

The figure below shows an example of Control Task cycle and Cycle time, with Control Task cycle factor = 3:

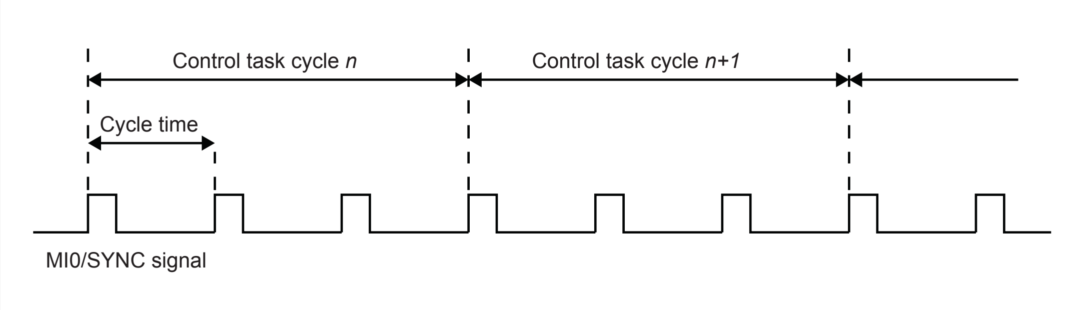


Figure 31.

16.9. Energy Control Object (F0h)

16.9.1. Category

Extended

16.9.2. 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.

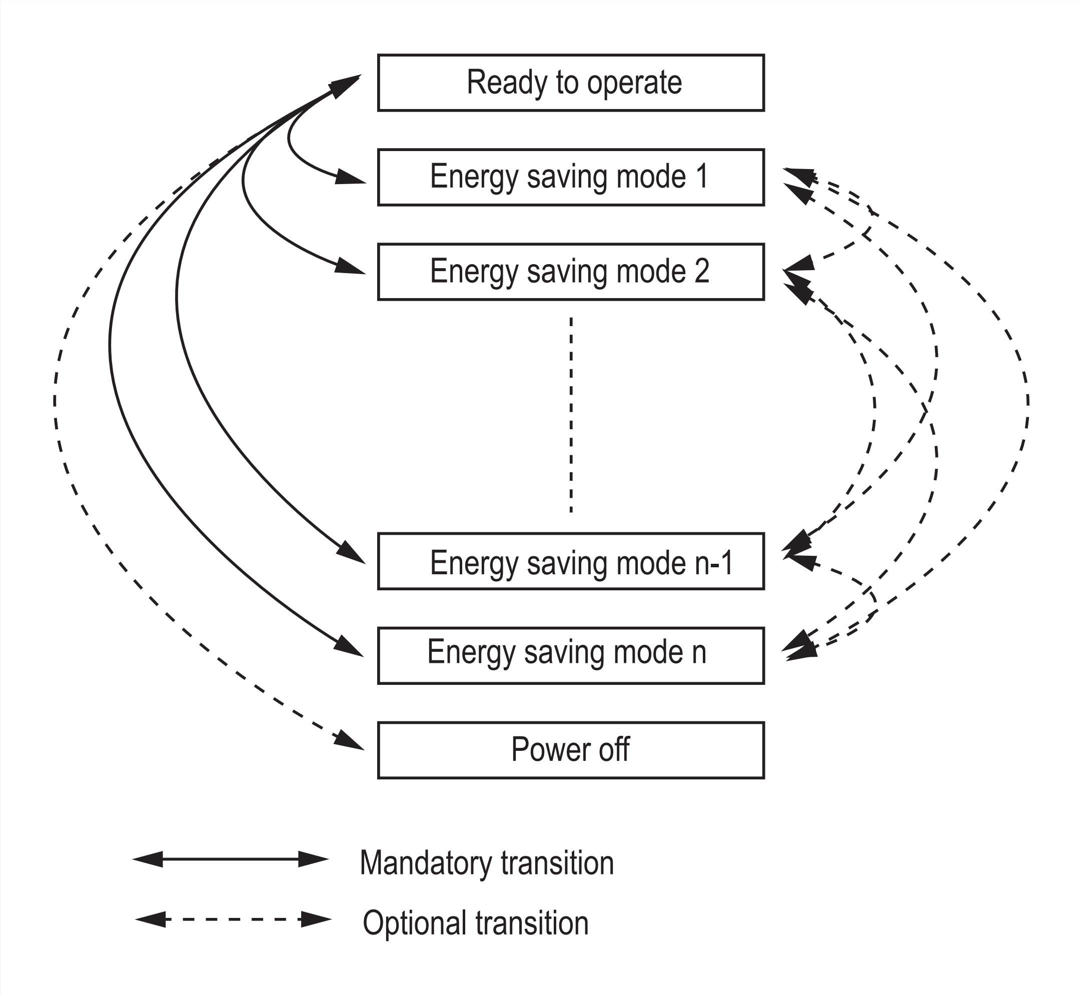


Figure 32.

16.9.3. Supported Commands

Object:	Get_Attribute StartPause EndPause Preview_Pause_Time (not PROFINET)
Instance:	Get_Attribute

16.9.4. 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

16.9.5. 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

16.10. PROFINET IO Object (F6h)

16.10.1. Category

Basic, extended

16.10.2. Object Description

This object implements PROFINET IO related settings in the host application.

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

See also...

- Network PROFINET IO Object (0Eh) (page 167)
- Flowchart — Record Data Access (page 261)
- Anybus CompactCom 40 Software Design Guide, “Error Codes”

16.10.3. Supported Commands

Object:	Get_Attribute (01h) Get_Record (10h, see below) Set_Record (11h, see below) AR_Check_Ind (14h, see below) Cfg_Mismatch_Ind (15h, see below) Expected_Ident_Ind (1Bh, see below) End_Of_Prm_Ind (17h, see below) AR_Abort_Ind (19h, see below) Indicate_Device (1Eh, see below) Prm_Begin_Ind (1Fh, see below)
Instance:	Get_Attribute (01h)

16.10.4. Object Attributes (Instance #0)

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

16.10.5. Instance Attributes (Instance #1)

Basic

#	Name	Access	Data Type	Default	Comment
1	Device ID	Get	UINT16	0010h	Identifies the device. (Assigned by manufacturer) If not implemented, the default value will be used.
2	Vendor ID (I&M Manufacturer ID)	Get	UINT16	010Ch (HMS Industrial Networks)	(Assigned by PROFIBUS & PROFINET International (PI)) If not implemented, the default value will be used.
3	Station Type	Get	Array of CHAR	“CompactCom 40 PIR”	Characterizes the device. Assigned by manufacturer); up to 25 characters. If not implemented, the value of attribute #9 (Product name) in the Application Object (FFh), will be used. If this attribute is not implemented, the default value will be used.
8	I&M Order ID	Get	Array of CHAR	“CompactCom 40 PIR”	I&M0 Parameter: Order ID of device; up to 20 characters. If not implemented, the default value will be used.
9	I&M Serial Number	Get	Array of CHAR	Assigned during manufacturing	I&M0 Parameter: Serial number of device; up to 16 characters. If not implemented, the value of attribute #3 (Serial number) in the Application Object (FFh), will be used. If this attribute is not implemented, the default value will be used.

GSD Entries

The GSDML entries below must match the values of the corresponding attributes in the PROFINET IO object.

- Attributes #1 and #2 correspond to the following entry in the GSDML file:
  `<DeviceIdentity VendorID=“0x010C” DeviceID=“0x0010”>`
- Attribute #8 correspond to the following entry in the GSDML file:
  `<OrderNumber Value=“CompactCom 40 PIR”/>`

Extended

- If an attribute is not implemented, the default value will be used.
- The Anybus module in itself does not alter its behavior based on the value of attributes #13 and #14. The host application has to implement the corresponding functionality.
- The module is preprogrammed with a valid Mac address. To use that address, do not implement attributes #17 and #18.

#	Name	Access	Data Type	Default Value	Comment
4	MaxAr	Get	UINT32	0000 0003h	Max. no.of simultaneous ARs. (Range 1... 4) **Note 1**: With S2Enabled = True, MaxAr needs to be at least 2. GSDML attribute NumberOfDeviceAccessAR must be set to MaxAr - 2. **Note 2**: With S2Enabled = False, MaxAr needs to be at least 1. GSDML attribute NumberOfDeviceAccessAR must be set to MaxAr - 1.
5	(Reserved)	-	-	-	Reserved for future use
6					
7	Record Data Mode	Get	UINT8	00h	This setting affects how Record Data requests are treated, and holds a bit field as follows: Bit 0:Index 0... 7FFFh: 0: Normal Mode 1: Transparent Mode Bit 1:Index AFF0h... AFFFh: Reserved (replaced with the command IM_Options, see Network PROFINET IO Object (0Eh) (page 167))
10	I&M Hardware Revision	Get	UINT16	Hardware Rev.	I&M0 Parameter: Hardware revision of device; FFFFh indicates availability of profile specific information If not implemented, the value of attribute #11 (Hardware version) in the Application Object (FFh), will be used. If this attribute is not implemented, the default value will be used.
11	I&M Software Revision	Get	Struct of CHAR 3*UINT8	Software Rev.	I&M0 Parameter: Software revision of device. Byte: Value: Meaning: 0: “V” Official release “R” Revision “P” Prototype “U” Under test “T” Test Device 1: 0... 255 Major Version 2: 0... 255 Minor Version 3: 0... 255 Internal Change Please note that version V255.255.255 indicates availability of profile specific information. If not implemented, the value of attribute #10 (Firmware version) in the Application Object (FFh), will be used. If this attribute is not implemented, the default value will be used.
12	I&M Revision Counter	Get	UINT16	0000h	I&M0 Parameter: Revision counter of device; a changed value marks a change of the hardware or its parameters.

#	Name	Access	Data Type	Default Value	Comment
13	I&M Profile ID	Get	UINT16	0000h (Non Profile Device)	I&M0 Parameter: If the application supports a specific profile, the Profile ID must be defined here. It shall equal the API number for the profile. Please note: this will not change the behavior of the module, since it does not handle profiles.
14	I&M Profile Specific Type	Get	UINT16	Generic Profile: 0004h (Communication Module)	I&M0 Parameter: If the application supports a specific profile, the profile specific type must be defined here (value is defined by the profile). Please note: this will not change the behavior of the module, since it does not handle profiles.
15-16	(Reserved)	-	-	-	Reserved, not used
17	Port 1 MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Do not implement this attribute if the preprogrammed Mac address is to be used.
18	Port 2 MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Do not implement this attribute if the preprogrammed Mac address is to be used.
19-23	(Reserved)	-	-	-	Reserved, not used
24	Custom Station Name	Get	Array of CHAR	Product specific. (default: “CompactCom-40-PIR”)	When using shift register mode, this value is used to create the station name. If for example the attribute has the value abcc40-pir, the station name will be abcc40-pir-XYZ where XYZ is the value of DIP2. Valid characters are letters “a” to “z”, numbers “0” to “9”, hyphens and periods. The attribute value must begin with a letter. Maximum number of elements in array: 58 For more information, see the Anybus CompactCom M40 Hardware Design Guide.
25	IM Module Order ID	Get	Array of CHAR	Anybus CompactCom 40 PROFINET IRT IIoT	I&M5 parameter. Order ID of the communication interface. Maximum number of elements in array is 64. Not available for Anybus IP.
26	IM Annotation	Get	Array of CHAR	Anybus CompactCom 40 PROFINET IRT IIoT	I&M5 parameter Description of the communication interface. Maximum number of elements in array: 64. Not available for Anybus IP.
27	IM5 Enabled	Get	BOOL	True	Enable I&M5 support. True: I&M5 enabled False: I&M5 disabled

#	Name	Access	Data Type	Default Value	Comment
28	S2 Enabled	Get	BOOL	False	Enable S2 system redundancy. True: S2 enabled False: S2 disabled
29	S2 Primary AR handle	Set	UINT16	0	AR handle of currently primary AR when S2 is enabled (attribute #28). A switchover (Backup to Primary) or loss of Primary triggers an update of the AR handle indicating change of primary AR. The handle may be used by the application to reference AR specific commands; e.g. AR_Check_Ind that contains IP address and Station name of the IO controller. The value 0 indicates “No Primary AR” or “No S2 connection”.

16.10.6. Command Details: Get_Record

Category

Extended

Details

Command Code:	10h
Valid for:	Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Read request towards an API other than 0 (zero).
- Module receives a Record Data Read request towards API 0, but the record in question is handled in Transparent Mode.
  See instance attribute #7 for more information about Transparent Mode.

See Flowchart — Record Data Access (page 261) for more information.

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	Length (low byte)	Range: 1 - 1524
MsgData[11]	Length (high byte)	MsgData[11] is only available if the length value exceeds 255.

- Response Details (Success)

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0... n]	Data (up to 1524 bytes)	Data to be returned in the Record Data Read response.

- Response Details (Error)

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See Details: Error Code 1 (page 249)
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- Details for the command Set_Record (below)
- Flowchart — Record Data Access (page 261)

16.10.7. Command Details: Set_Record

Category

Extended

Details

Command Code:	11h
Valid for:	Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Write request towards an API other than 0 (zero).
- Module receives a Record Data Write request towards API 0, but the record in question is handled in Transparent Mode
  See instance attribute #7 for more information about Transparent Mode.

See Flowchart — Record Data Access (page 261) for more information.

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	(reserved)	(set to zero)
MsgData[11.. n]	Data (up to 1512 bytes)	Data from the Record Data Write request.

- Response Details (Success)
  (no data)
- Response Details (Error)

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See Details: Error Code 1 (page 249)
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- Command details for “Get_Record”
- Flowchart — Record Data Access (page 261)

16.10.8. Command Details: AR_Check_Ind

Category

Extended

Details

Command Code:	14h
Valid for:	Object Instance

Description

The module issues this command to inform the host application that an Application Relationship (AR) is to be established. It is optional to implement support for this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	IP address (low word, low byte)	IP address of the remote station (IO Controller/Supervisor)
MsgData[1]	IP address (low word, high byte)	
MsgData[2]	IP address (high word, low byte)	
MsgData[3]	IP address (high word, high byte)	
MsgData[4]	AR Type (low byte)	Indicates the type of AR as follows: Value: Meaning: 0001h IOCARSingle (RT connection to single IO Connector) 0006h IOSAR (Used in combination with ARProperties.DeviceAccess for connections without process data) 0010h IOCARSingle using RT_CLASS_3 (IRT connection to single IO Controller) 0020h IOCARSR (Variant of IOCARSingle indicating usage of System Redundance)
MsgData[5]	AR Type (high byte)	
MsgData[6]	AR Properties (low word, low byte)	Bit-field indicating the properties of the AR as described in the table below.:
MsgData[7]	AR Properties (low word, high byte)	
MsgData[8]	AR Properties (high word, low byte)	
MsgData[9]	AR Properties (high word, high byte)	
MsgData[10]	Remote station name length	Length of remote station name, in bytes
MsgData[11.. n]	Remote station name	Remote station name (IO Controller/Supervisor)

AR Properties:

Bit(s)	Value / Description
0-2	State: 0: (reserved) 1: Active 2-7: (reserved)
3	Supervisor take over allowed: 0: Not allowed 1: Allowed
4	Parameterization server: 0: Externak PrmServer 1: CM Initiator
5-7	(reserved)
8	Device Access: 0: Only submodules from ExpectedSubmoduleBlock are accessible 1: Submodule access is controlled by IO device application
9-10	Companion AR: 0: Single AR 1-3: (reserved)
11	AcknowledgeCompanion AR: 0: No companion AR 1: (reserved)
12-27	(reserved)

Bit(s)	Value / Description
28	Time aware system: 0: Non time aware 1: Time aware
29	Combined object container (COC) with StartupMode = Legacy 0: No COC 1: (reserved) with StartupMode = Advanced 0: No COC 1: Usage of COC required
30	Startup mode: 0: Legacy 1: Advanced
31	Pull module alarm allowed: 0: Subslot number 0 shall indicate pulling of module in conjunction with AlarmType(=Pull) 1: Sublsot number 0-9FFFh shall indicate pulling of submodule in conjunction with AlarmType(=Pull)

- Response Details
  (no data)

16.10.9. Command Details: Cfg_Mismatch_Ind

Category

Extended

Details

Command Code:	15h
Valid for:	Object Instance

Description

The module issues this command to inform the host application that the configuration in the IO Controller (i.e. the Expected Identification) does not match the configuration defined by the host application (i.e. the Real Identification).

It is optional to implement support for this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of mismatch
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of mismatch
MsgData[7]	Subslot (high byte)	
MsgData[8]	Expected Module Identifier (low word, low byte)	Module identifier (as stated in the GSD file) derived from the IO Controller configuration
MsgData[9]	Expected Module Identifier (low word, high byte)	
MsgData[10]	Expected Module Identifier (high word, low byte)	
MsgData[11]	Expected Module Identifier (high word, high byte)	
MsgData[12]	Expected Submodule Identifier (low word, low byte)	Submodule identifier (as stated in the GSD file) derived from the IO Controller configuration
MsgData[13]	Expected Submodule Identifier (low word, high byte)	
MsgData[14]	Expected Submodule Identifier (high word, low byte)	
MsgData[15]	Expected Submodule Identifier (high word, high byte)	

- Response Details
  (no data)

16.10.10. Command Details: Expected_Ident_Ind

Category

Extended

Details

Command Code:	1Bh
Valid for:	Object Instance

Description

The module issues this command to inform the host application of the Expected Identification (Module/ Submodule List) that the IO Controller will use for the established AR.

Note that this information may be split in multiple segments, which means that this command will be issued multiple times by the module, each time containing different parts of the configuration.

Expected_Ident_Ind is similar to AR_Info_Ind but uses a different segmentation protocol that shall be used for the 40 series concept, see Anybus CompactCom 40 Software Design Guide (Message Segmentation).

For very large configurations where the Expected Identification cannot fit into one message, this segmentation protocol will be used. If the number of modules/sub-modules exceeds the capabilities of the Anybus CompactCom the message will be truncated. The size of the configuration can be up to 2370 bytes at maximum number of modules/sub-modules.

It is optional to implement support for this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	Reserved	For segmented messages the CmdExt[1] byte has been reserved for segment bits, see Anybus CompactCom 40 Software Design Guide (Message Segmentation)..
CmdExt[1]	Cmd segment bits	
MsgData[0]	AR handle (low byte)	Handle for the Application Relationship.
MsgData[1]	AR handle (high byte)	
MsgData[2.. n]	Data field	The first two bytes in the initial block of the Data field indicates the number of modules in the configuration. Each module is represented by a Module block, followed by a number of Submodule blocks (provided that the module in question contains submodules). See “Data Format” below for coding of the data field.

- Response Details

Field	Contents	Comments
CmdExt[0]	Reserved	(set to zero)
CmdExt[1]		
MsgData[0]	Continue/Block	Value: Meaning: 0 Continue. The host application will not perform any changes to the Real Identification, A connect response may be sent on the network immediately following this response. 1 Block. The host application will perform changes in the Real identification in response to the received Expected identification. A connect response will not be sent on the network until Ident_Change_Done is sent by the host application Returning any error response or providing no data/ too much data will give the ssame result as responding with Continue (0). The timeout is 300 seconds. A “Continue” response or the Ident_Change_Done command should be sent well within this time limit.

Data Format

When all data has been received, the resulting data shall be interpreted as follows:

Type	Name	Description
UINT16	iNbrApi	Number of APIs in configuration
UINT32	iApiNbr	Initial module block including API number and number of module blocks in the API.
UINT16	iNbrMod	
UINT16	iSlotNbr	Module block (8 bytes), see below.
UINT16	iNbrSubMod	
UINT32	lModIdent	
UINT16	iSubSlotNbr	Submodule block (10 bytes), see below.
UINT32	lSubModIdent	
UINT16	iInDataLength	
UINT16	iOutDataLength	

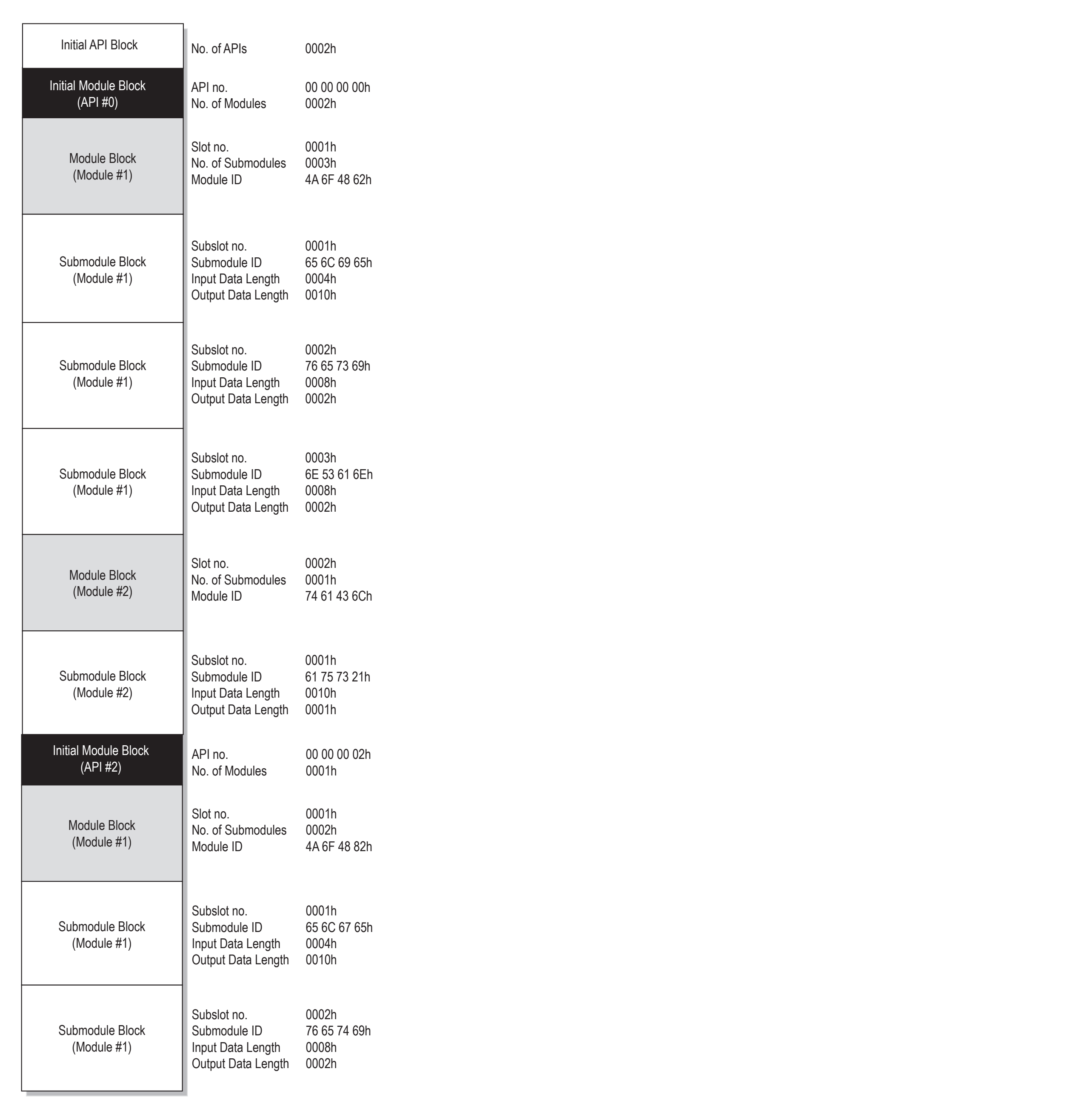
The initial API block (iNbrApi) defines the number of APIs in the configuration.

Each API has an initial module block, that includes information on the API number (iApiNbr) and the number of modules (or slots) in the API.

Each module starts with a module block, which holds the slot number, the number of submodules (or subslots) and the module identity number.

Finally each submodule block holds subslot number, submodule identification number, input and output data lengths.

Example



Initial API Block	No. of APIs	0002h
Initial Module Block (API #0)	API no. No. of Modules	00 00 00 00h 0002h
Module Block (Module #1)	Slot no. No. of Submodules Module ID	0001h 0003h 4A 6F 48 62h
Submodule Block (Module #1)	Subslot no. Submodule ID Input Data Length Output Data Length	0001h 65 6C 69 65h 0004h 0010h
Submodule Block (Module #1)	Subslot no. Submodule ID Input Data Length Output Data Length	0002h 76 65 73 69h 0008h 0002h
Submodule Block (Module #1)	Subslot no. Submodule ID Input Data Length Output Data Length	0003h 6E 53 61 6Eh 0008h 0002h
Module Block (Module #2)	Slot no. No. of Submodules Module ID	0002h 0001h 74 61 43 6Ch
Submodule Block (Module #2)	Subslot no. Submodule ID Input Data Length Output Data Length	0001h 61 75 73 21h 0010h 0001h
Initial Module Block (API #2)	API no. No. of Modules	00 00 00 02h 0001h
Module Block (Module #1)	Slot no. No. of Submodules Module ID	0001h 0002h 4A 6F 48 82h
Submodule Block (Module #1)	Subslot no. Submodule ID Input Data Length Output Data Length	0001h 65 6C 67 65h 0004h 0010h
Submodule Block (Module #1)	Subslot no. Submodule ID Input Data Length Output Data Length	0002h 76 65 74 69h 0008h 0002h

Figure 33.

In this example, the configuration contains two APIs with the following properties:

- API #0 contains two modules, the first with three submodules, the second with one submodule
- API #2 contains one module with two submodules

16.10.11. Command Details: End_Of_Prm_Ind

Category

Extended

Details

Command Code:	17h
Valid for:	Object Instance

Description

The module may issue this command to indicate to the host application that the parameterization phase is completed. It is optional to implement support for this command.

If implemented, the host application may, depending on the response issued to this command, be required to issue ‘Appl_State_Ready’ at a later stage to indicate that it is ready for data exchange. If not implemented, this is handled automatically by the module.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API) - Only valid if subslot > 0
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number affected by the command - Only valid if subslot > 0
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number affected by the command Value: Meaning: 0: Command applies to all modules in the configuration other: Command applies only to the specified slot/subslot
MsgData[7]	Subslot (high byte)	

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
MsgData[0]	Subslot (high byte)	Value: Meaning: 0: Ready for Data Exchange 1: Not ready for data exchange (Appl_State_Ready must be issued at a later stage)

See also...

- Appl_State_Ready, details in Network PROFINET IO Object (0Eh) (page 167)

16.10.12. Command Details: AR_Abort_Ind

Category

Extended

Details

Command Code:	19h
Valid for:	Object Instance

Description

This command is optional. The module issues this command to indicate to the host application that an Application Relationship (AR) is aborted (by the application or any other source).

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	Reason code (low byte)	Reason code for the offline transition
MsgData[1]	Reason code (high byte)	

Value:	Reason:
0	No reason (unknown reason)
3	Out of memory
4	Add provider or consumer failed
5	Miss (consumer)
6	Cmi timeout
7	Alarm-open failed
8	Alarm-send.cnf(-)
9	Alarm-ack-send.cnf(-)
10	Alarm-data too long
11	Alarm.ind(err)
12	Rpc-client call.cnf(-)
13	Ar-abort.req
14	Re-run aborts existing
15	Got release.ind
16	Device passivated
17	
18	Protocol violation
19	NARE error
20	RPC-Bind error
21	RPC-Connect error
22	RPC-Read error
23	RPC-Write error
24	RPC-Control error
25	Forbidden pull or plug after check.rsp and before in-data.ind
26	AP removed
27	Link down
28	Could not register multicast-mac
29	Not synchronized (cannot start companion-ar)
30	Wrong topology (cannot start companion-ar)
31	Dcp, station-name changed
32	Dcp, reset to factory-settings
33	Cannot start companion AR because of parameter error

- Response Details
  (no data)

16.10.13. Command Details: Indicate_Device

Category

Extended

Details

Command Code:	1Eh
Valid for:	Object Instance

Description

This command is optional. The module issues this command to inform the application that the DCP command Set Control/Signal has been received on the network. This is used by engineering tools to identify the node on the network. The application should flash its dedicated Network Status LED for 3 seconds after receiving this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		

- Response Details
  (no data)

16.10.14. Command Details: Prm_Begin_Ind

Category

Extended

Details

Command Code:	1Fh
Valid for:	Object Instance

Description

This command is optional. The module issues this command to indicate to the host application that a parameterization phase, outside of the connection establishment, is initiated. During the parameterization phase the host application may receive Set_Record commands targeting the submodules addressed by the Prm_Begin_Ind command.

- Command Details

Msg field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
Data[0]	API (low word, low byte)	Application Process Instance Only valid if Sub-slot > 0
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	Slot (low byte)	Slot number affected by the command Only valid if Sub-slot > 0
Data[5]	Slot (high byte)	
Data[6]	Sub-slot (low byte)	Sub-slot number affected by the command 0 — Command applies to all modules in the configuration >0 — Command applies to the specified slot/sub-slot
Data[7]	Sub-slot (high byte)	

- Response Details
  (no data)

16.10.15. Details: Error Code 1

The error codes below shall be used when providing error responses to the following commands:

- Get_Record
- Set_Record
- Get_IM_Record
- Set_IM_Record

High nibble (bits 4... 7)		Low nibble (bits 0... 3)	
ErrorClass	**Meaning**	**ErrorCode**	**Meaning**
0... 9	Reserved	(reserved)	(reserved)
10	Application	0	Read error
		1	Write error
		2	Module error
		3... 6	(reserved)
		7	Busy
		8	Version conflict
		9	Feature not supported
		10... 15	User specific
11	Access	0	Invalid index
		1	Write length error
		2	Invalid slot/subslot
		3	Type conflict
		4	Invalid area
		5	State conflict
		6	Access denied
		7	Invalid range
		8	Invalid parameter
		9	Invalid type
		10	Backup
		11... 15	User specific
12	Resource	0	Read constrain conflict
		1	Write constrain conflict
		2	Resource busy
		3	Resource unavailable
		4... 7	(reserved)
		8... 15	User specific
13... 15	User specific	(user specific)	User specific

See also...

- Command Details: Get_Record (page 234)
- Command Details: Set_Record (page 236)

16.11. Ethernet Host Object (F9h)

16.11.1. Object Description

This object implements Ethernet features in the host application.

16.11.2. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

16.11.3. 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

16.11.4. 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 SHICP	Get	BOOL	True (Enabled)	Enable/Disable SHICP
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.
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.

#	Name	Access	Data Type	Default Value	Comment
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 SHICP	Get	BOOL	0 = False	Enables the option to reset the module from SHICP.
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.
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)

#	Name	Access	Data Type	Default Value	Comment
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

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

16.11.6. DHCP Option 61 (Client Identifier)

NOTE
Only valid for EtherNet/IP devices

The DHCP Option 61 (Client Identifier) allows 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.

Appendix 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.

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.

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.

Appendix B. Compatibility to Standard Anybus CompactCom 40

The Anybus CompactCom 40 PROFINET IRT IIoT Secure is not compatible to standard Anybus CompactCom. The differences are summarized in the table below.

Issue	Description
User account configuration	"web_access.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 (page 7).
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) (page 43).
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 (page 54).
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) (page 43).
OPC UA	File format changed for opcua.cfg.

Appendix C. Anybus Implementation Details

1. SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of PROFINET, this bit is set when one or more IO connections are established.

2. Anybus State Machine

The table below describes how the Anybus State Machine relates to the PROFINET network.

Anybus State	Implementation	Comment
WAIT_PROCESS	The Anybus stays in this state until an IO connection with an IO controller is opened.	-
ERROR	Configuration data mismatch, initial parameter error, or error reported from application status register.	See Application Status Register (page 259) for more information.
PROCESS_ACTIVE	IO connection established with IO controller and valid output data has been provided at least once.	-
IDLE	IO controller with which an IO connection is established is in STOP mode, the IO controller has not provided valid output at least once, or error reported from application status register..	• Some IO controllers will not provide valid output data in the first cycles following a successful connection. • See Application Status Register (page 259) for more information.
EXCEPTION	Turn module status LED red, to indicate major fault, turn network status LED off, and hold Ethernet MAC in reset.	Some kind of unexpected behavior, for example watchdog timeout. See also instance 1, attribute 7 in Network Object (03h) (page 125).

3. Application Status Register

The application status register is primarily used in SYNC applications. It is used in applications where the network in question supports the ability to indicate critical process data errors to the master. If the application sets an error status to the application status register, the module cannot operate in the PROCESS_ACTIVE state. The Anybus CompactCom will accept and handle the below listed status codes written by the application.

Value	Error	Description	Anybus CompactCom 40 PROFINET IRT IIoT Secure Response
0000h	No error	Ready for transition to state PROCESS_ACTIVE (Default)	If Isochronous mode is active, the Anybus CompactCom monitors write process data updates according to the table below (Input time failure).
0001h	Not yet synchronized	Not ready for transition to state PROCESS_ACTIVE	Alarm: Application not synchronized • The Anybus CompactCom is set to state IDLE • IOxS is set to BAD.
0002h	Sync configuration error	A problem with the current attribute values in the Sync object prevents the transition to state PROCESS_ACTIVE	Alarm: Sync configuration error • The Anybus CompactCom is set to state IDLE • IOxS is set to BAD.
0003h	Read process data configuration error	A problem with the current read process data mapping prevents the transition to state PROCESS_ACTIVE	Alarm: Output configuration error • The Anybus CompactCom is set to state IDLE • IOxS is set to BAD.
0004h	Write process data configuration error	A problem with the current write process data mapping prevents the transition to state PROCESS_ACTIVE	Alarm: Input configuration error • The Anybus CompactCom is set to state IDLE • IOxS is set to BAD.
0005h	Synchronization loss	The application has lost synchronization lock If the Anybus CompactCom is in the state PROCESS_ACTIVE, it will change to the state ERROR	Alarm: Application not synchronized • The Anybus CompactCom is set to state ERROR • IOCS is set to BAD.
0006h	Excessive data loss	The application has detected a significant loss of process data from the network If the Anybus CompactCom is in the state PROCESS_ACTIVE, it will change to the state ERROR	
0007h	Output error	Application malfunction If the Anybus CompactCom is in the state PROCESS_ACTIVE, it will change to the state ERROR	Alarm: Output disabled • The Anybus CompactCom is set to state ERROR • IOxS is set to BAD.

The Anybus CompactCom produces synchronous mode related diagnosis as shown in the table below, where the cause is reported from the Application status register.

Cause	ChannelErrorType	ExtChannelErrorType	Description
0007h	0018h	-	Output disabled
0001h or 0005h or 0006h	7000h (vendor specific – “Isochronous mode error”)	0001h	Application not synchronized
0002h		0002h	Sync configuration error
0003h		0003h	Output configuration error
0004h		0004h	Input configuration error
0000h	8004h (Isochronous mode mismatch)	8001h	Input time failure is reported if the application status register reports 0000h and Isochronous Mode is active and no new write process data update occurs before input processing.

For GSDML entries, see the SYNC example GSDML file that is included in the configuration files package, that can be downloaded from the Anybus CompactCom 40 PROFINET IRT IIoT Secure page at www.anybus.com/technical-support.

4. Application Watchdog Timeout Handling

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

Appendix D. Flowcharts

1. Flowchart — Record Data Access

This flowchart illustrates how Record Data requests are handled by the Anybus module.

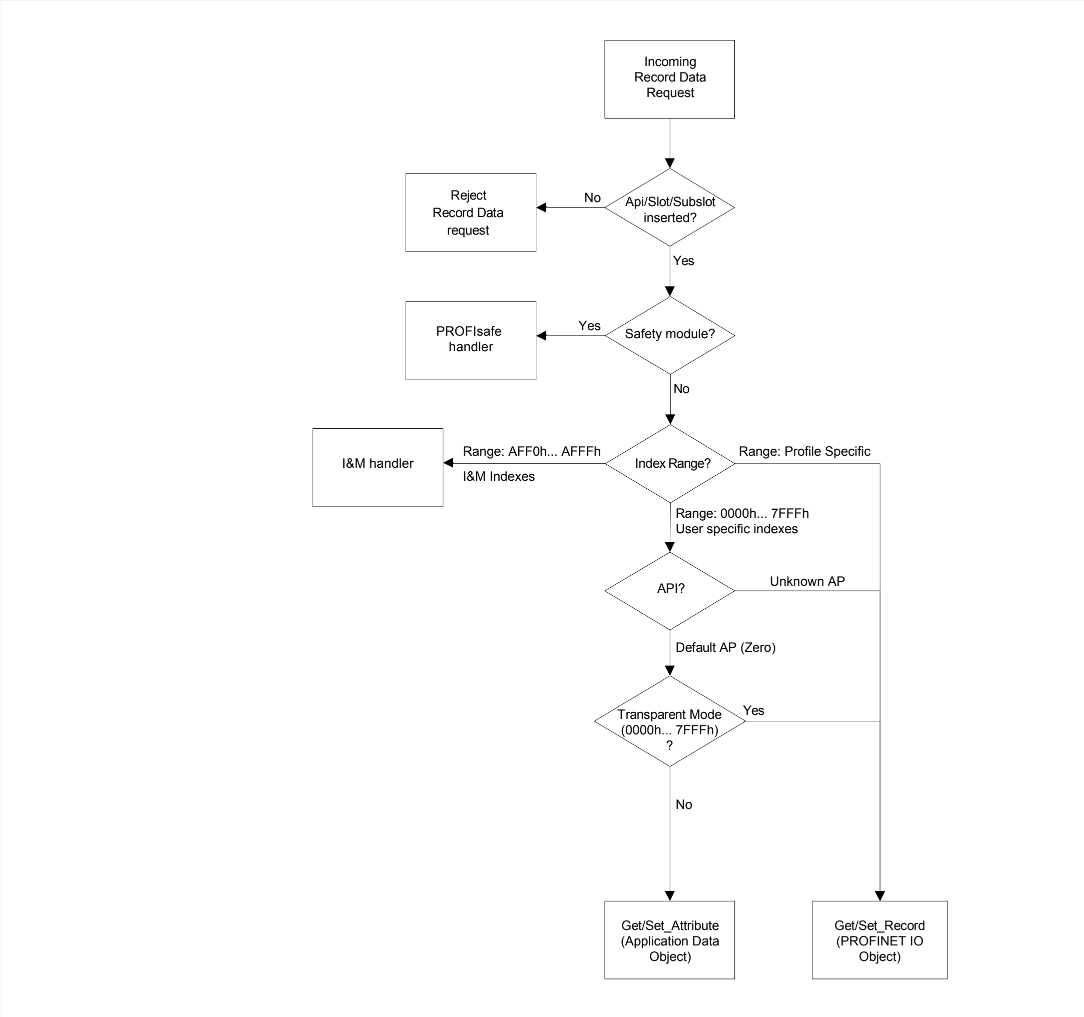


Figure D.1.

See also...

- Application Data Instances (ADIs) (page 23)
- PROFINET IO Object (F6h) (page 228)
- Details for command Get_Record in the PROFINET IO Object (F6h) (page 228)
- Details for command Set_Record in the PROFINET IO Object (F6h) (page 228)

2. Flowchart — I&M Record Data Handling

This flowchart illustrates how I&M Record Data requests are handled by the Anybus module.

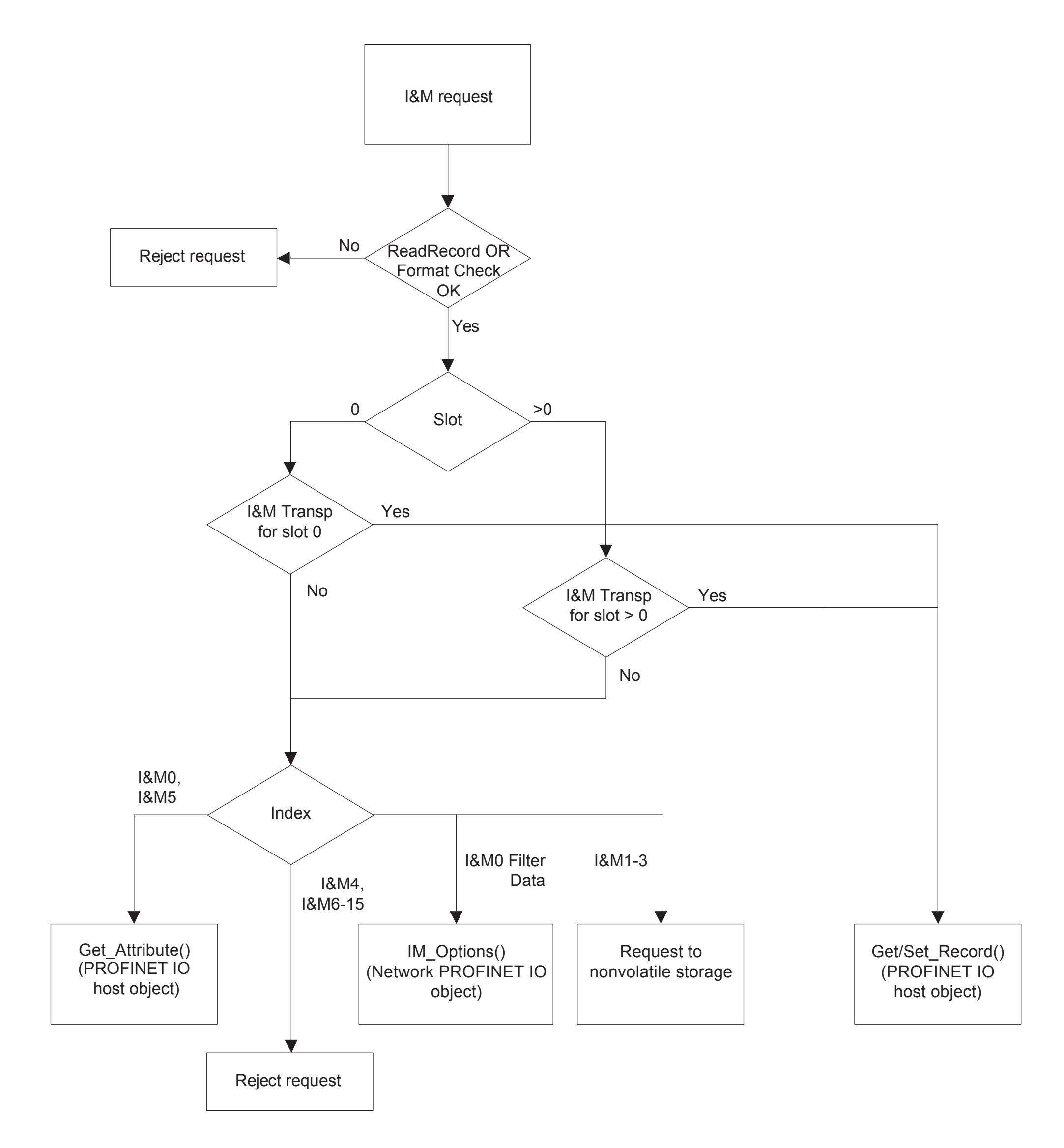

I&M request
ReadRecord OR Format Check OK
No
Reject request
Yes
Slot
0
>0
I&M Transp for slot 0
Yes
No
I&M Transp for slot > 0
Yes
No
Index
I&M0, I&M5
I&M4, I&M6-15
I&M0 Filter Data
I&M1-3
Get_Attribute() (PROFINET IO host object)
IM_Options() (Network PROFINET IO object)
Request to nonvolatile storage
Get/Set_Record() (PROFINET IO host object)
Reject request

Figure D.2.

- PROFINET IO Object (F6h) (page 228)
- Details for command Get_Record in the PROFINET IO Object (F6h) (page 228)
- Details for command Set_Record in the PROFINET IO Object (F6h) (page 228)

3. Flowchart —Establishment of Real Identification (RI)

This flowchart illustrates the establishment of the Real Identification.

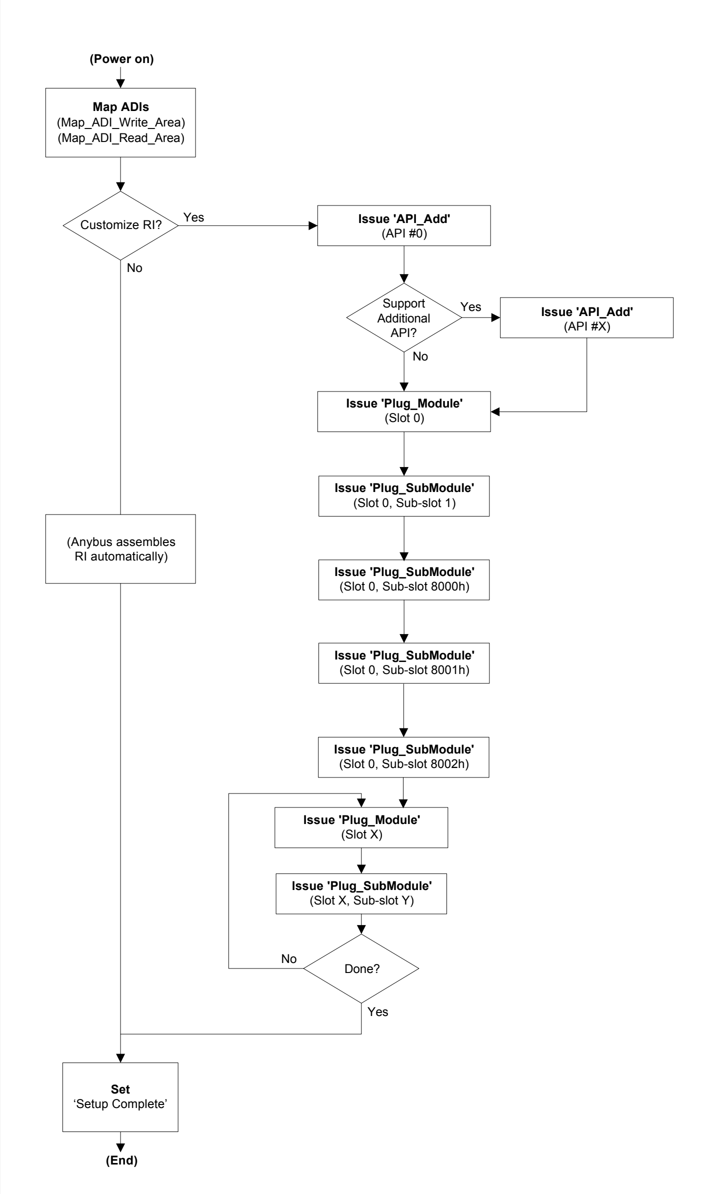


Figure D.3.

- Process Data (page 23)
- Real Identification (RI) (page 30)
- Details for command Set_Record in the PROFINET IO Object (F6h) (page 228)

4. Flowcharts — Handling of Configuration Mismatch

4.1. Default Configuration Mismatch

This flowchart shows how the Anybus CompactCom automatically handles a configuration mismatch when the Real Identification has been established by the default configuration method.

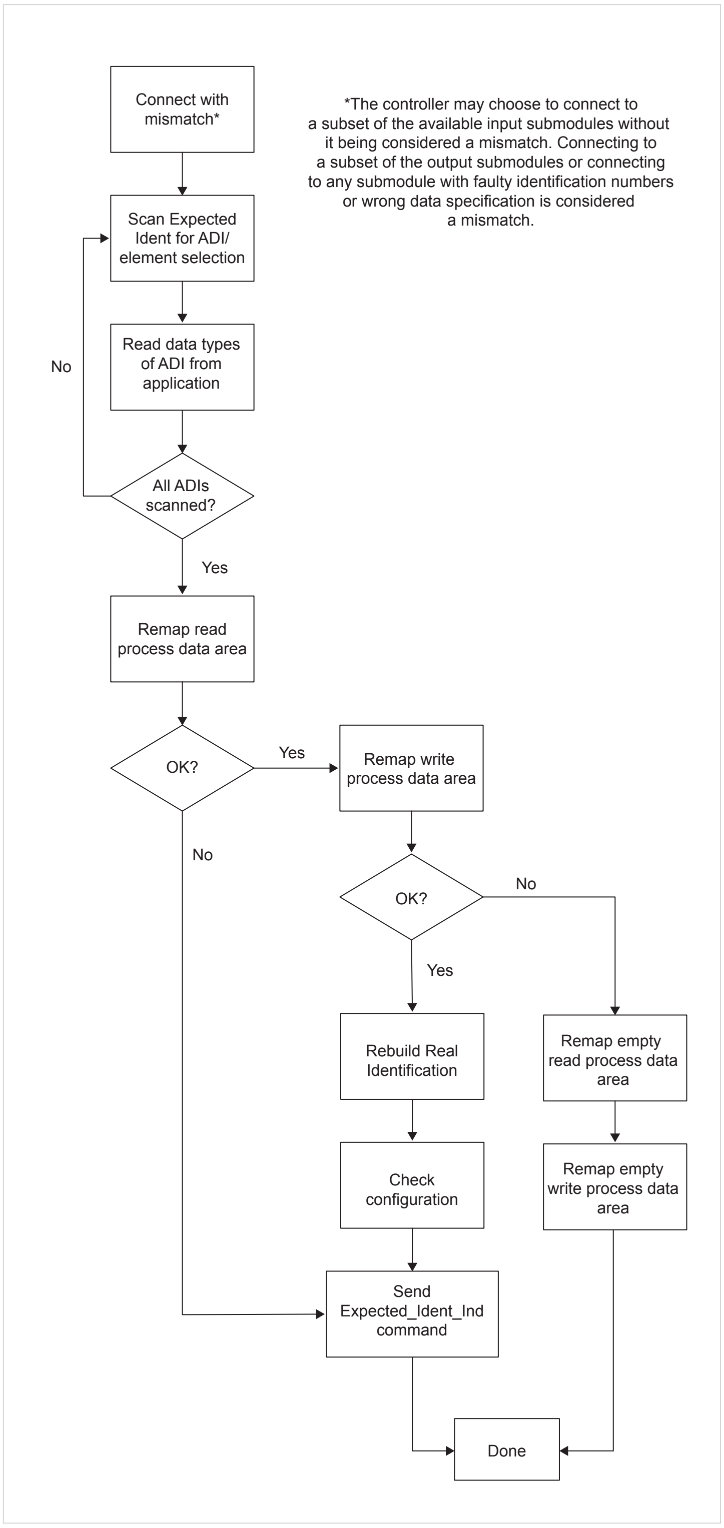

Connect with mismatch*
*The controller may choose to connect to a subset of the available input submodules without it being considered a mismatch. Connecting to a subset of the output submodules or connecting to any submodule with faulty identification numbers or wrong data specification is considered a mismatch.
Scan Expected Ident for ADI/ element selection
Read data types of ADI from application
No
All ADIs scanned?
Yes
Remap read process data area
OK?
Yes
Remap write process data area
No
OK?
No
Yes
Rebuild Real Identification
Remap empty read process data area
Check configuration
Remap empty write process data area
Send Expected_Ident_Ind command
Done

Figure D.4.

4.2. Custom Configuration mismatch

This flowchart shows how to handle a configuration mismatch when the Real Identification has been established by the host application (custom configuration).

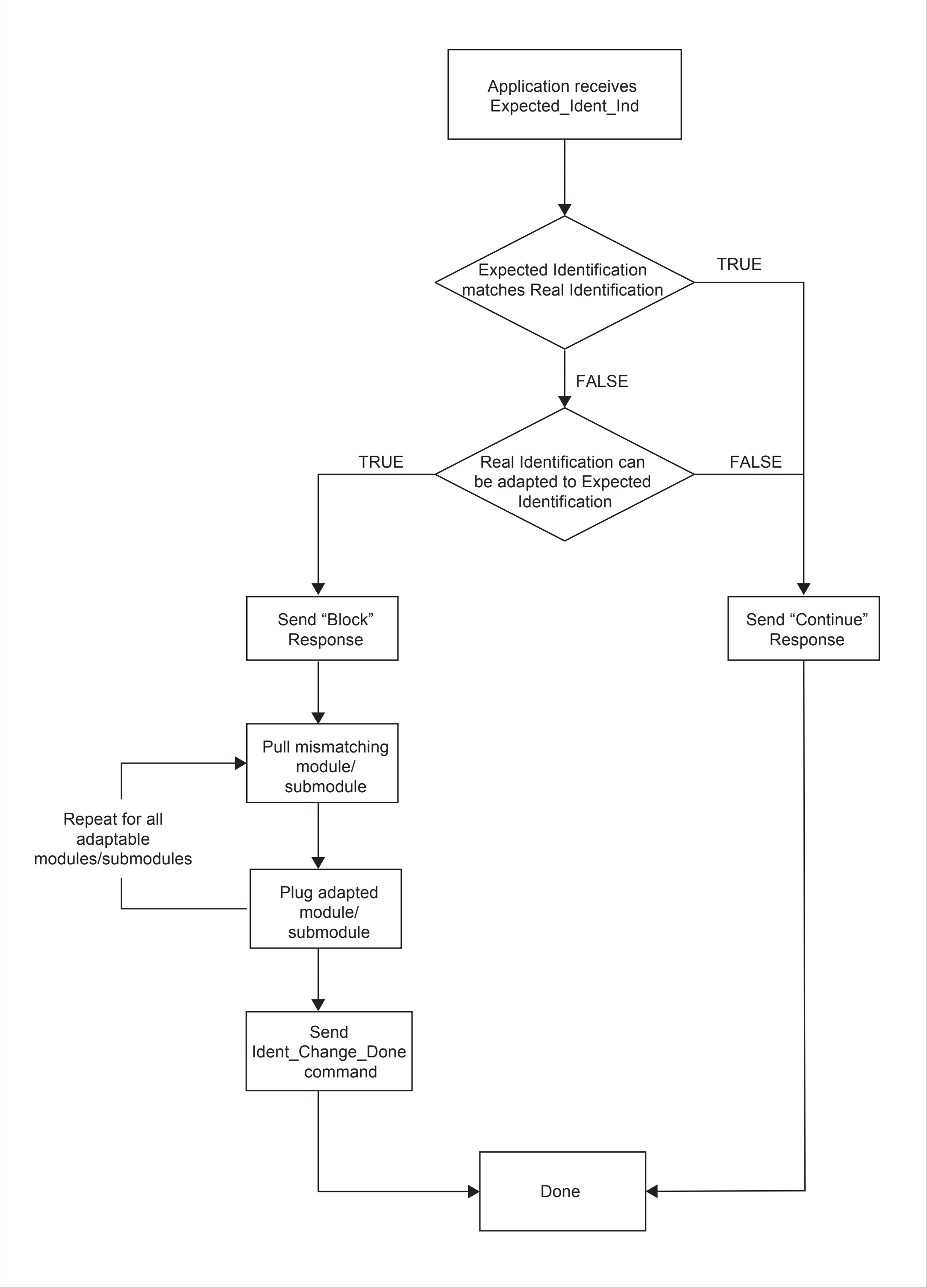

Application receives Expected_Ident_Ind
Expected Identification matches Real Identification
TRUE
FALSE
Real Identification can be adapted to Expected Identification
TRUE
FALSE
Send "Block" Response
Send "Continue" Response
Pull mismatching module/ submodule
Repeat for all adaptable modules/submodules
Plug adapted module/ submodule
Send Ident_Change_Done command
Done

Figure D.5.

Appendix E. Secure HICP (Secure Host IP Configuration Protocol)

1. General

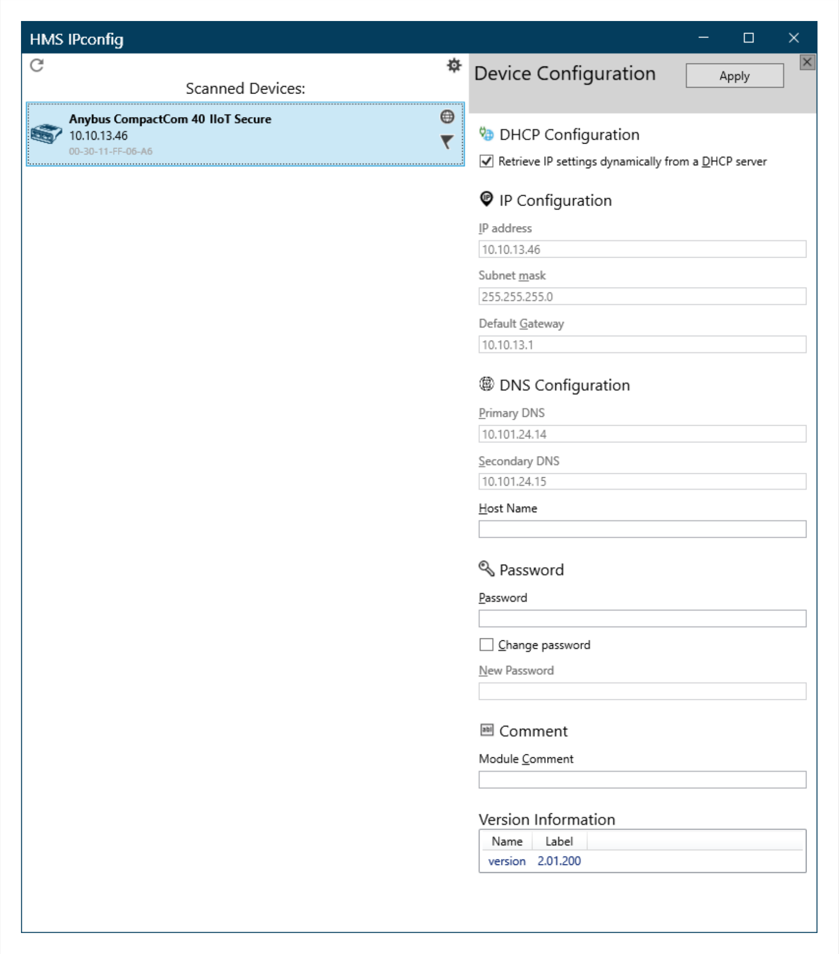
The Anybus CompactCom 40 PROFINET IRT IIoT Secure supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the support pages at HMS Networks, www.anybus.com/support. 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).

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.



Anybus CompactCom 40 IIoT Secure
10.10.13.46
00-30-11-FF-06-A6

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

Figure E.1.

Validate the new settings by clicking **Set**, or click **Cancel** to cancel all changes.

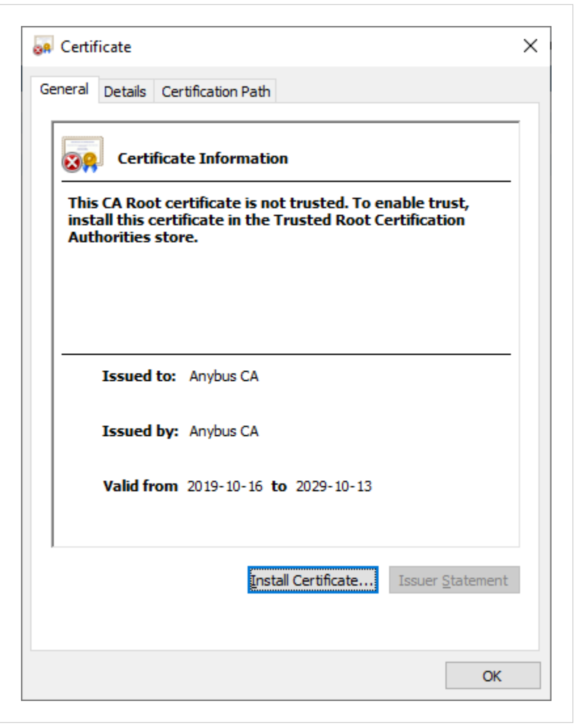
For Anybus CompactCom 40 PROFINET IRT IIoT Secure, you can not set a password to protect the configuration from unauthorized access in this view. The password is set from the internal web pages.

IMPORTANT
It is strongly recommended to password protect the IP configuration.

Appendix F. 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....”



This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

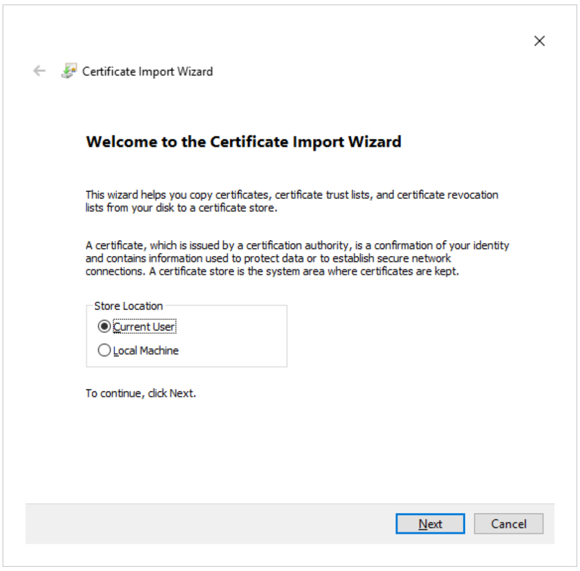
Issued to: Anybus CA

Issued by: Anybus CA

Valid from 2019-10-16 **to** 2029-10-13

Figure F.1.

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

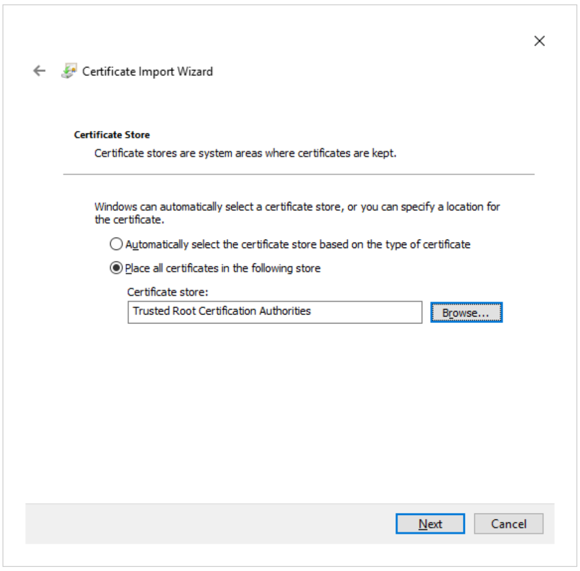
● Current User

○ Local Machine

To continue, click Next.

Figure F.2.

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 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.

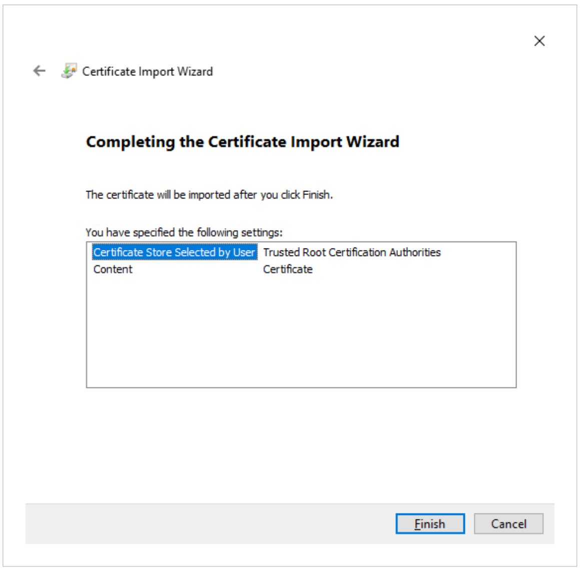
○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store: Trusted Root Certification Authorities

Figure F.3.

5. Check that the settings are correct before you complete the wizard.



Completing the Certificate Import Wizard

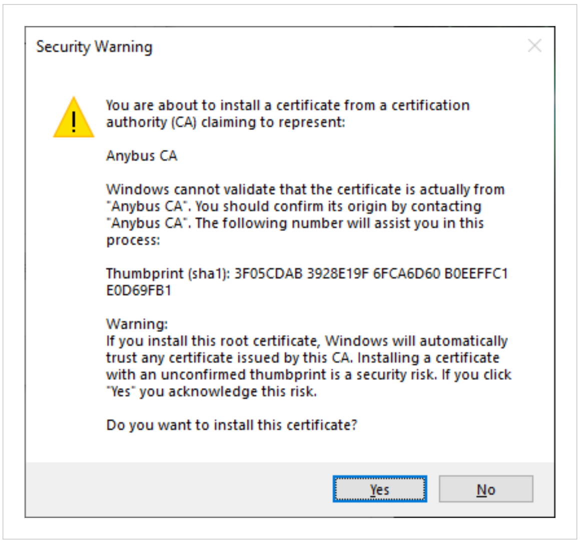
The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certification Authorities
Content	Certificate

Figure F.4.

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.

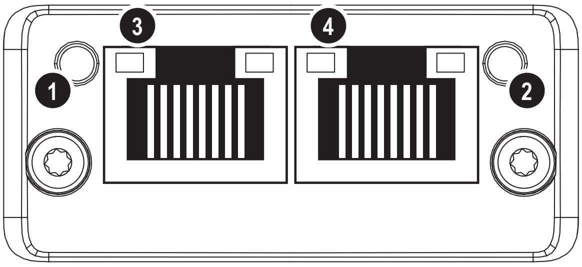
Do you want to install this certificate?

Figure F.5.

Appendix G. Technical Specification

1. Front View

1.1. Front View (PROFINET IRT, Ethernet Connectors)

#	Item	
1	Network Status LED	
2	Module Status LED	
3	Link/Activity LED (port 1)	
4	Link/Activity LED (port 2)	

1.2. Network Status LED

LED State	Description	Comments
Off	Offline	• No power • No connection with IO Controller
Green	Online (RUN)	• Connection with IO Controller established • IO Controller in RUN state
Green, 1 flash	Online (STOP)	• Connection with IO Controller established • IO Controller in STOP state or IO data bad • IRT synchronization not finished
Green, blinking	Blink	Used by engineering tools to identify the node on the network
Red	Fatal event	Major internal error (this indication is combined with a red module status LED)
Red, 1 flash	Station Name error	Station Name not set
Red, 2 flashes	IP address error	IP address not set
Red, 3 flashes	Configuration error	Expected Identification differs from Real Identification

1.3. Module Status LED

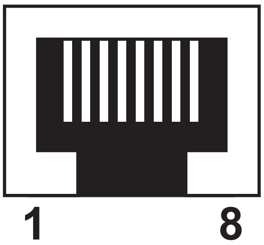
LED State	Description	Comments
Off	Not Initialized	No power OR Module in SETUP or NW_INIT state.
Green	Normal Operation	Module has shifted from the NW_INIT state.
Green, 1 flash	Diagnostic Event(s)	Diagnostic event(s) present
Red	Exception error	Device in state EXCEPTION.
	Fatal event	Major internal error (this indication is combined with a red network status LED)
Alternating Red/Green	Firmware update	Do NOT power off the module. Turning the module off during this phase could cause permanent damage.

1.4. LINK/Activity LED

LED State	Description	Comments
Off	No Link	No link, no communication present
Green	Link	Ethernet link established, no communication present
Green, flickering	Activity	Ethernet link established, communication present

1.5. Ethernet Interface (RJ45 connectors)

The Ethernet interface operates at 100 Mbit, full duplex, as required by PROFINET.

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	

For information on how to connect the PROFINET cable, see Functional Earth (FE) Requirements (page 272)

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.

> **NOTE**
> The shield of the RJ45 connector is not connected directly to FE. As all nodes in a PROFINET network have to share chassis ground connection, the PROFINET cable shield has to be connected to the chassis ground at each node in the network.

For further information, see PROFINET Installation Guideline for Cabling and Assembly, available for download at www.profibus.com.

3. Power Supply

3.1. Supply Voltage

The Anybus CompactCom 40 PROFINET IRT IIoT Secure requires a regulated 3.3 V power source as specified in the general Anybus CompactCom 40 Hardware Design Guide.

3.2. Power Consumption

The Anybus CompactCom 40 PROFINET IRT IIoT Secure is designed to fulfil the requirements of a Class C module. The current hardware design consumes up to 550 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.

4. Environmental Specification

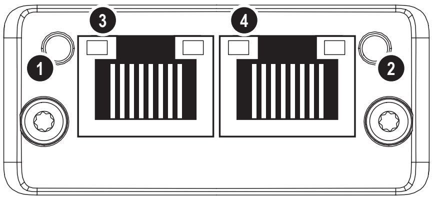
Consult the Anybus CompactCom 40 Hardware Design Guide for further information.

5. EMC Compliance

Consult the Anybus CompactCom 40 Hardware Design Guide for further information.

Appendix H. Anybus CompactCom AIDA LED Mode

1. Front View

#	Default Mode (see Front View (page 271))	AIDA Mode (see description below)	
1	Network Status LED	Bus Failure LED	
2	Module Status LED	System Failure LED	
3	Link/Activity LED (port 1)	Link (port 1)	
4	Link/Activity LED (port 2)	Link (port 2)	

2. Bus Failure LED

LED State	Description	Comments
Off	Ok	No problem
Red	Fatal event	Major internal error (this indication is combined with a red system failure LED)
	Station Name error	Station Name not set
	IP address error	IP address not set
	Configuration error	Expected Identification differs from Real Identification
	Online (STOP)	Connection with IO Controller established • IO Controller in STOP state or IO data BAD • IRT synchronization not finished
	Connection error	No connection with IO Controller

3. System Failure LED

LED State	Description	Comments
Off	Not Initialized	No problems detected
Red	Exception error	Device in state EXCEPTION.
	FATAL event	Major internal error (this indication is combined with a red bus failure LED)
	Diagnostic Event(s)	Diagnostic event(s) present

4. Port Link LED

LED State	Description	Comments
Off	No Link	No link, no communication present
Green	Link	Ethernet link established
Green, blinking	DCP Identify	Used by engineering tools to identify the node on the network 500 ms on, 500 ms off during 3 seconds

The signal is also available in the host application connector:

AIDA indication	Signal Name, Application Interface Connector
Link on port 1	nLED3A
Link on port 2	nLED4A

The Port Link signals are always available in the host application connector in all operating modes except 16 bit parallel. For more information consult the Anybus CompactCom M40 Hardware Design Guide (module) or the Anybus CompactCom B40-1 Design Guide (brick).

When Transparent Ethernet is activated, the LED status signals are only available in the LED status register.

5. Port Activity

AIDA LED State	Description
Off	No Activity
Orange, flashing	Activity

There is no LED on the front of the Anybus CompactCom 40 PROFINET IRT IIoT Secure showing this indication, but the signal is available in the host application connector according to this table:

AIDA indication	Signal Name, Host Application Connector
Activity on port 1	nLED3B
Activity on port 2	nLED4B

The Port Activity signals are always available in the host application connector in all operating modes except 16 bit parallel. For more information consult the Anybus CompactCom M40 Hardware Design Guide (module) or the Anybus CompactCom B40-1 Design Guide (brick).

When Transparent Ethernet is activated, the LED status signals are only available in the LED status register.

Appendix I. Conformance Test Guide

1. General

When using the default settings of all parameters, the Anybus CompactCom 40 PROFINET IRT IIoT Secure is precertified for network compliance. This precertification is done to ensure that your product *can* be certified.

Changes in the parameters in the example GSD file, supplied by HMS Networks, will require a certification. A vendor ID can be obtained from PNO and is compulsory for certification. This chapter provides a guide for successful conformance testing your product, containing the Anybus CompactCom 40 PROFINET IRT IIoT Secure, to comply with the demands for network certification set by the PNO.

Independent of selected operation mode, the actions described in this appendix have to be accounted for in the certification process. The identity of the product needs to be changed to match your company and device.

For multiple-API implementations, e.g. PROFIdrive, one of the submodules belonging to the “non-zero API” must be inserted in the I&M0 Filter Data by the application, using the command IM_Options.

IMPORTANT

This appendix provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take.

Please contact HMS Networks at www.anybus.com/support for more information.

2. Reidentifying Your Product

After successful setting of the **Setup Complete** attribute in the Anybus Object (01h), the Anybus module asks for identification data from the host PROFINET IO Object (F6h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
PROFINET IO Object (F6h), Instance 1	#1,Device ID	With this attribute you set the Device ID of the device	Device ID: 0010h	Device ID: YYYYh	This information must match the keys of the “DeviceIdentity” of the GSD-file. Note that the GSD file keyword “VendorName” must correspond to the Vendor ID value.
PROFINET IO Object (F6h), Instance 1	#2,Vendor ID	With this attribute you set the Vendor ID of the device	Vendor ID: 010Ch (HMS)	Vendor ID: XXXXh	
PROFINET IO Object (F6h), Instance 1	#3,Station Type	With this attribute you set the station type of the device	“CompactCom 40 PIR”	“Cust-PNIO-Dev”	Optional if Application Object, Instance 1, Attribute 9 "Product Name" is implemented. In this case the product name common for all networks will be used as Station Type/Device Type on PROFINET.
PROFINET IO Object (F6h), Instance 1	#8, I&M Order ID	With this attribute you set the Order ID that is used in the I&M data	“CompactCom 40 PIR”	“Cust-PNIO-Dev”	This information must match the keys of the “OrderNumber” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#10, I&M Hardware Revision	With this attribute you set the I&M Hardware Revision	(Hardware Rev.)	“0002h”	Optional. This information must match the keys of the “HardwareRelease” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#11, I&M Software Revision	With this attribute you set the I&M Software Revision	(Software Rev.)	“V2.5.3”	Optional. This information must match the keys of the “SoftwareRelease” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#13, I&M Profile ID	With this attribute you set the I&M Profile ID	0xF600 (Generic device)	API number of profile.	Optional, Only relevant for multi-API implementations like PROFIdrive.
PROFINET IO Object (F6h), Instance 1	#14, I&M Profile Specific type	With this attribute you set the I&M Profile Specific typ	0x0004 (Communication Module)	Profile specific type, defined by profile.	Optional, Only relevant for multi-API implementations like PROFIdrive.

2.1. Additional GSD File Information

The GSD file keyword “ProductFamily” shall correspond to the vendor’s name of the device.

The GSD file keyword “MainFamily” lists the kinds of devices for which the product shall be listed. As of GSD specification v2.3, the following “families” are available:

“General”, “Drives”, “Switching Devices”, “I/O”, “Valves”, “Controllers”, “HMI”,“Encoders”, “NC/RC”, “Gateway”, “PLCs”, “Ident Systems”, “PA Profiles”, “Network Components”, “Sensors”.

3. Factory Default Reset

Reset command to Application Object (FFh) must be supported

When PROFINET IO modules are delivered, they are required to be in their “Factory Default” state. For PROFINET devices this means that their Station Name is empty (“‘’”) and that the IP suite is not assigned (IP 0.0.0.0). When a Factory Default Reset command is received from the network, the Anybus module will erase all IP and Station Name information and inform the host application that a hardware reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host. For more details, please consult the Anybus CompactCom Software Design Guide.

4. IP Address

Normally the IP numbers of PROFINET IO devices are assigned via the PROFINET network via DCP (Discovery and Configuration Protocol). HMS Networks recommends not using the Network Configuration Object (04h, instances #3 - #6) during the initialization phase for PROFINET modules, unless the end user has requested the IP address to be set to a specific value (by for example using a keypad). The reason is that when a factory default reset command is received from the PROFINET network (via DCP) the node must be available after a hardware reset with the default IP-address (0.0.0.0).

If the IP suite is set using the Network Configuration Object (04h), the key word AddressAssignment = “DCP;LOCAL” must be added to the GSD file in the section DeviceAccessPointItem:

```
<DeviceAccessPointItem AddressAssignment=“DCP;LOCAL”...>
```

5. Station Name

Normally the Station Name of a PROFINET device is assigned by the end user via the PROFINET DCP protocol. HMS recommends not using the Station Name instance in the Network Configuration Object during the initialization phase for PROFINET modules. If this attribute is used, it is recommended that it is sent explicitly when the end user changes the Station Name with e.g. a keypad. The reason is that when a factory default reset command is received from the PROFINET network (via DCP), the node must be available after a hardware reset with the default Station Name (“”””).

NOTE

The Anybus module will forward all information about the connection being established to the IO Controller, as commands to the host PROFINET IO Object (F6h). Even though the host application might not need this information, a response must always be generated (such as 05h, “Unsupported command”)

6. Documentation Considerations

To obtain a certificate, the following information must be present in the customer's user manual:

1. Behavior of the outputs if IOPS=BAD.
2. Behavior of the outputs if connection is aborted.
3. Behavior of the outputs at power on.

The Anybus CompactCom handles these situations in the following ways:

1. State change to IDLE. The network is informed that the I/O data of the submodule with IOPS=BAD is substituted with zeros (clear). No read process data is updated in the host interface.
2. State change to WAIT_PROCESS. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.
3. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.
4. A shift register application must use the PA signal to clear outputs when the Anybus CompactCom 40 PROFINET IRT IIoT Secure is not in the state PROCESS_ACTIVE.

7. Certification in Generic Anybus Mode

In Generic Anybus Mode (when the command API_add in the Network PROFINET IO Object (0Eh) is not used) there is normally nothing that needs to be considered apart from what is mentioned earlier in this appendix. The example HMS GSD file has to be modified with respect to the process mapping and identity of the product and this requires a certification of the product.

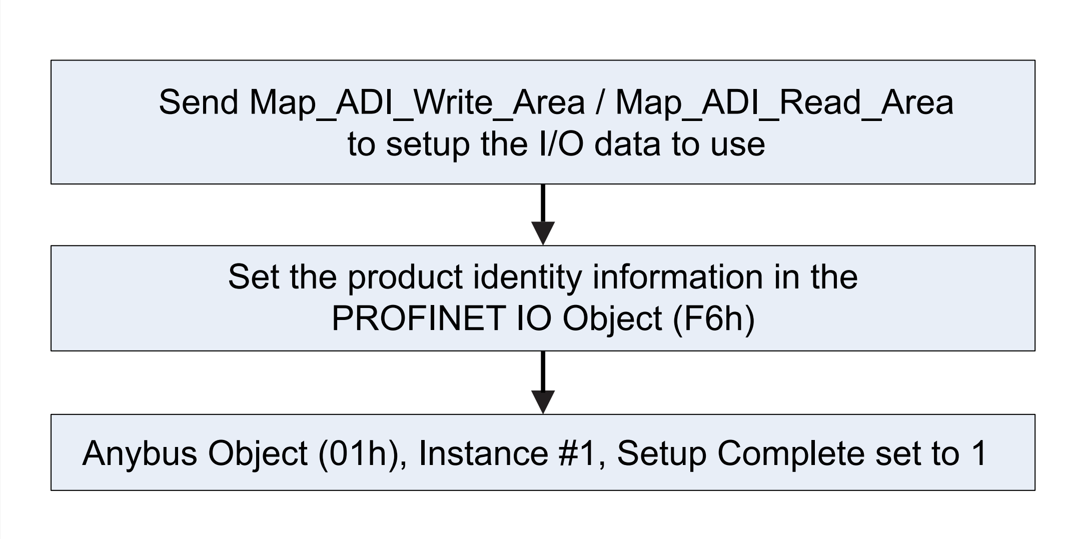

Send Map_ADI_Write_Area / Map_ADI_Read_Area
to setup the I/O data to use
Set the product identity information in the
PROFINET IO Object (F6h)
Anybus Object (01h), Instance #1, Setup Complete set to 1

Figure I.1.

8. Certification in Advanced Mode

In advanced mode (Network PROFINET IO Object (0Eh) is used), the most important thing is to use a Device Access Point (DAP) that conform to PROFINET IO Specification v2.0 or later (DAP2). From specification version 2.0 it is possible to describe the physical Ethernet interface and its ports (PDEV, or Physical Device) with a special mechanism. This is done with special submodules at slot 0 (the module at slot 0 is the access point for the device). HMS recommends following the flow below for setting up a DAP2.

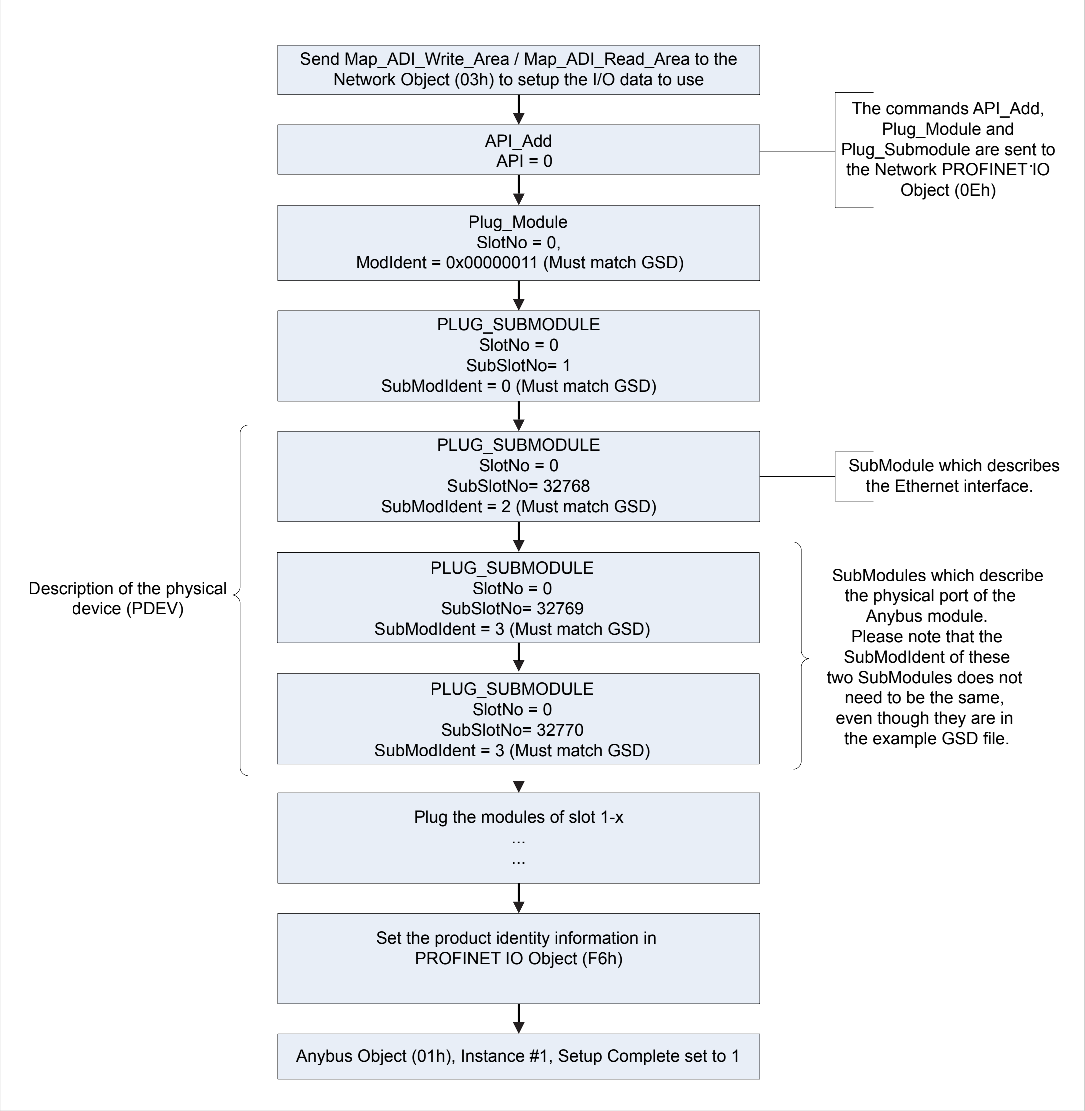

Send Map_ADI_Write_Area / Map_ADI_Read_Area to the Network Object (03h) to setup the I/O data to use
API_Add
API = 0
The commands API_Add, Plug_Module and Plug_Submodule are sent to the Network PROFINET IO Object (0Eh)
Plug_Module
SlotNo = 0,
ModIdent = 0x00000011 (Must match GSD)
PLUG_SUBMODULE
SlotNo = 0
SubSlotNo= 1
SubModIdent = 0 (Must match GSD)
Description of the physical device (PDEV)
PLUG_SUBMODULE
SlotNo = 0
SubSlotNo= 32768
SubModIdent = 2 (Must match GSD)
SubModule which describes the Ethernet interface.
PLUG_SUBMODULE
SlotNo = 0
SubSlotNo= 32769
SubModIdent = 3 (Must match GSD)
PLUG_SUBMODULE
SlotNo = 0
SubSlotNo= 32770
SubModIdent = 3 (Must match GSD)
SubModules which describe the physical port of the Anybus module. Please note that the SubModIdent of these two SubModules does not need to be the same, even though they are in the example GSD file.
Plug the modules of slot 1-x
...
...
Set the product identity information in PROFINET IO Object (F6h)
Anybus Object (01h), Instance #1, Setup Complete set to 1

Figure I.2.

The figure shows how to set up a PROFINET compatible DAP. Please note that for some commands only the relevant parameters are shown.

Please note that the values of “SubModIdent” in the above flowchart are the values of the example HMS GSD file. They can be changed if necessary, but there is no real need for it, the important thing is that it matches the GSD file. To be able to pass the PROFINET conformance test a “DAP2” is mandatory.

9. Changes in GSD File for Conformance Class B

The example GSD file, supplied by HMS Networks, is adapted for testing a Anybus CompactCom 40 PROFINET IRT IIoT Secure for conformance class C. If the implementation does not need the isochronous features of the device, the GSD file can be modified to mirror this. The implementation can then be conformance tested for conformance class B instead. The list below describe the changes needed in the example GSD file to accomplish this.

1. The value of the `ConformanceClass` attribute in the `<CertificationInfo...>` element in each DAP must be changed from “C” to “B”.:
   ```
   <CertificationInfo ConformanceClass="B" ApplicationClass=""
   NetloadClass="III"/>
   ```
2. The value of the `SupportedRT_Classes` attribute in the `<InterfaceSubmoduleItem ...>` element in each DAP must be “RT_CLASS_1”. I.e. “RT_CLASS_3” must be removed:
   ```
   <InterfaceSubmoduleItem ID="Interface" SubslotNumber="32768"
   SubmoduleIdentNumber="0x00000002"
   SupportedRT_Classes="RT_CLASS_1" TextId="T_ID_INTERFACE"
   SupportedProtocols="SNMP;LLDP" DCP_HelloSupported="true"
   PTP_BoundarySupported="true" DCP_BoundarySupported="true"
   DelayMeasurementSupported="true">
   ```
3. The elements `<RT_Class3Properties ...>`, `<SynchronisationMode ...>`, and `<RT_Class3TimingProperties ...>` must be removed from each DAP

10. SYNC Pin Measurements for Conformance Class C Test

For a conformance class C (IRT) test, access to the SYNC pin must be provided to the test lab.

NOTE

The Sync Object (EEh) does not have to be enabled for the SYNC pin to function.

Appendix J. License Information

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

****

Copyright 1986 by Carnegie Mellon.

****

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

****

Copyright (c) 1988 Stephen Deering.

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

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

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

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

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

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

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

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

****

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

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

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

****

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

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 AND

NONINFRINGEMENT. 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 AND NONINFRINGEMENT. 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 AND NONINFRINGEMENT.

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.

****

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 trems.

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) 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 ftpd.c - This file is part of the FTP daemon for lwIP

****

Format - lightweight string formatting library.

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

****

lwIP is licenced under the BSD licence:

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.

****

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.

Author: David van Moolenbroek <david@minix3.org>

****

MD5 routines

L. Peter Deutsch

ghost@aladdin.com

****

The “inih” library is distributed under the New BSD license:

****

open62541 is licensed under the Mozilla Public License v2.0

To obtain customized changes please contact foss@anybus.com.

****

musl as a whole is licensed under the following standard MIT license:

****

PCG Random Number Generation for C.

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

@(#)queue.h 8.5 (Berkeley) 8/20/94

This page is intentionally left blank.