

BRADY IRX200

GETTING STARTED WITH PROFINET



TABLE OF CONTENTS

1. INTRODUCTION.....	4
2. NEEDED TOOLS, FILES AND ACCESSORIES.....	5
2.1. NEEDED TOOLS.....	5
2.2. NEEDED FILES.....	5
2.3. NEEDED ACCESSORIES.....	5
3. SETTING UP THE DEVICE.....	6
3.1. POWERING UP THE READER.....	6
3.2. ACCESSING THE DEVICE ADMIN WEBUI.....	6
3.2.1. OVER USB CONNECTION.....	6
3.2.2. OVER ETHERNET CONNECTION (NORMAL MODE).....	6
3.2.1. USING BRADY SMART DEVICE CONTROL APP.....	7
3.2.2. OBTAINING PASSWORD AND USERNAME.....	7
3.3. SETTING UP THE PROFINET MODE AND IP FROM THE WEBUI.....	8
3.4. NETWORK DISCOVERY AND SETUP USING PRONETA.....	9
4. IRX200 INDUSTRIAL READER CONFIGURATION WITH SIEMENS TIA PORTAL.....	11
4.1. IRX200 HARDWARE CONFIGURATION.....	11
4.1.1. IMPORTING THE GSDML FILE.....	11
4.1.2. ADDING A DEVICE TO THE PROFINET NETWORK AND HARDWARE CATALOG.....	12
4.1.3. DEVICE/MODULE PARAMETERS.....	13
4.1.4. IMPORTING IRP USER DATA TYPES (UDTS AND ENUMS).....	15
4.1.5. IO MAPPING (GETIO / SETIO).....	16
4.2. RUNNING THE SAMPLE PLC APPLICATION.....	18
4.2.1. SAMPLE APPLICATION PROJECT TREE VIEW.....	19
4.2.2. DEMO APPLICATION SIMULATION AND RESULTS.....	22
5. SETTING UP THE PLC DEVELOPMENT ENVIRONMENT - CODESYS.....	40
5.1. CODESYS VERSION.....	40
5.2. INSTALLING THE DEVICE DESCRIPTOR / GSDML.....	40
5.3. RUNNING THE SAMPLE PLC APPLICATION.....	41
5.3.1. IMPORTING THE SAMPLE PROJECT.....	41
5.3.2. STARTING AND CONNECTING TO THE SOFTWARE PLC.....	43
5.3.3. HUMAN MACHINE INTERFACE.....	46
5.4. CREATING A NEW CODESYS PROJECT FOR THE IRX200.....	47
5.4.1. IMPORTING PLCOPENXML.....	56
5.4.2. I/O MAPPING TO ESTABLISH MESSAGE EXCHANGE ON TOP OF PROFINET CYCLIC DATA EXCHANGE.....	57
5.4.3. TESTING THE I/O MAPPING.....	59
5.4.4. ADDING APPLICATION-LEVEL LOGIC TO THE PLC PROGRAM.....	60

6. TROUBLESHOOTING	62
6.1. SOFTWARE PLC (CODESYS CONTROL) NOT RESPONDING.....	62
6.2. DEVICE CONNECTION FAILS	62
6.3. DEVICE KEEPS RECONNECTING.....	63
6.4. LOGGING	63
7. DATA EXCHANGE WITH THE PLC	65
7.1. DEVICE STATUS SUBMODULE	65
7.2. IRP MESSAGE EXCHANGE SUBMODULE	65
7.2.1. PLC INPUT MAPPING (DATA TRANSFER TO PLC).....	66
7.2.2. PLC OUTPUT MAPPING (DATA TRANSFER FROM PLC).....	66
7.2.3. MESSAGE EXCHANGE SEQUENCE	67
7.3. GPIO SUBMODULE	67
7.4. DEVICE PROPERTIES	68
8. IRP – INDUSTRIAL READER PROTOCOL DESCRIPTION	69
8.1. MESSAGE TYPES AND ASSOCIATED RESPONSES	69
8.2. GLOBAL DEFINATIONS.....	70
8.3. ENUM DATA TYPES	71
8.4. DATA STRUCTURES.....	76
9. VERSION HISTORY	87

1. INTRODUCTION

This getting started guide will instruct you thru the phases how to set up the Brady IRX200 and make the required configuration to use the device over the Profinet interface. Brady IRX200 is designed to meet the Profinet CC-B conformance class and can operate down to 1ms cycle times.

2. NEEDED TOOLS, FILES AND ACCESSORIES

This section lists the tools, documents and accessories which are used or referenced during the phases of this getting started guide.

2.1. NEEDED TOOLS

Below tools are used during the phases of this getting started guide.

- Siemens PRONETA 3.5.0.0 discovery tool
- Siemens TIA portal development environment V18
- Brady smart device control application
- CODESYS development environment

2.2. NEEDED FILES

Below files are used during the phases of this getting started guide.

- Brady IRX200 GSDML file
- IRP_PLCopenXML file
- IRP_SiemensXML files
- TIA Portal sample project(s)
- CODESYS sample project(s)

2.3. NEEDED ACCESSORIES

Below accessories are used during the phases of this getting started guide.

- Power supply for the IRX200
- M12-A to USB/IO cable
- M12-X to RJ45 ethernet data cable

3. SETTING UP THE DEVICE

This section covers the setting up phase of the Brady IRX200 reader. Initial network configuration can be done in multiple ways which are explained below.

3.1. POWERING UP THE READER

To power up the IRX200 device you will need to attach the 24V DC power supply to the M12-L 5pin connector (FE-connected). Brady is offering AC/DC power supply adapter or power cable with stripped wires as an accessory.

NOTE! Power supply is not included in the package. Needs to be purchased separately.

3.2. ACCESSING THE DEVICE ADMIN WEBUI

The Brady IRX200 admin web user interface can be accessed in two different ways, through the Ethernet and the USB connections.

3.2.1. OVER USB CONNECTION

The device supports USB RNDIS and CDC-ECM profiles which allow to access the device's admin Web UI through TCP/IP connections. When the USB cable is connected the IP address of the device is 169.254.0.1 by default. Typing that IP address to the web browser will open the login screen of the Web UI.

NOTE! USB cable is not included in the package. Needs to be purchased separately.

3.2.2. OVER ETHERNET CONNECTION (NORMAL MODE)

By default, the network interface is set to DHCP mode, and the MAC address of the interface is indicated on product label on the underside of the device. To access the Web UI, you need to connect the IRX200 and the PC into the same local network having DHCP server.

The connection can be established by typing the IRX200's hostname in the address bar of a web browser if the network and host device support mDNS. The default hostname for IRX200 is:

- irx200-serialnumber

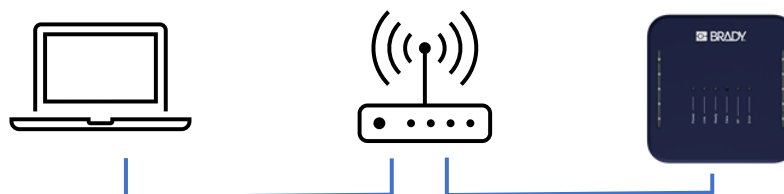


Figure 1. Connecting the IRX200 to PC via router with integrated DHCP server.

When the IP address of the Brady IRX200 is known, typing it in the address bar of a web browser will open the login screen of the Web UI. Applications like the Brady Smart Device Control can be used to discover the address.

3.2.1. USING BRADY SMART DEVICE CONTROL APP

Brady smart device control PC application is a simple tool which can be used to discover devices which are connected to PC via USB or connected to same ethernet network as the PC in question. By clicking a device in the list, a new page is opened which leads to IRX200 devices web UI login page.

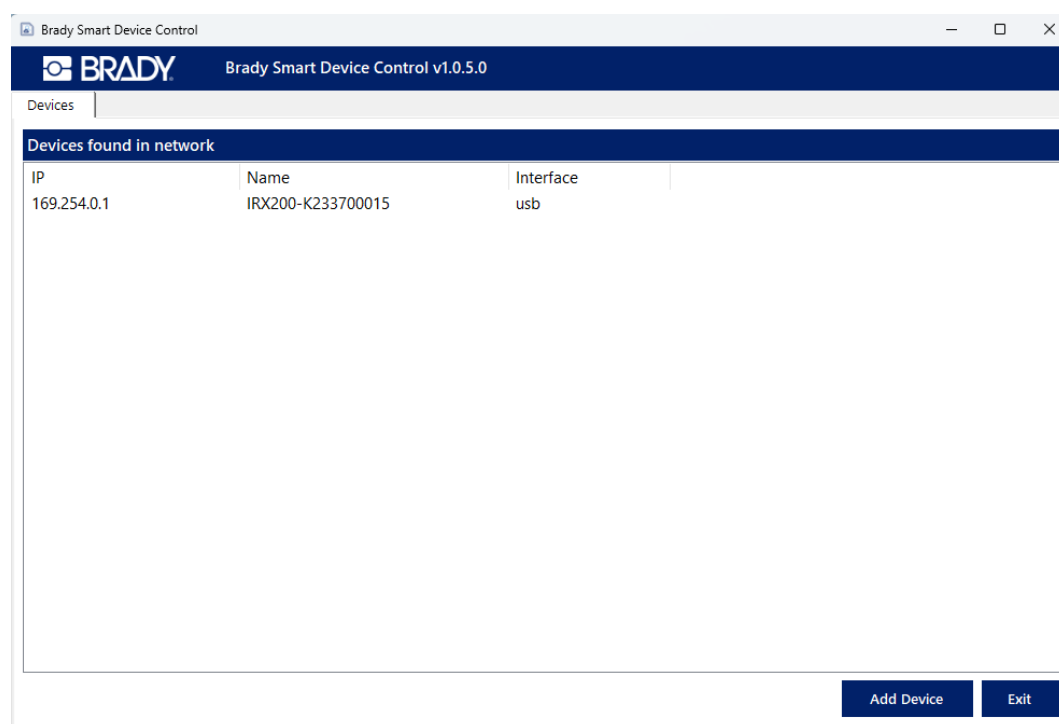


Figure 2. Brady smart device control application view.

3.2.2. OBTAINING PASSWORD AND USERNAME

The default username ("admin") and password are printed on the product type label attached to the back side of the device.



Figure 3. Type label of the IRX200.

Note that a security alert will pop up on the browser the first time that you connect to the device, as the connection is forced to be secure. To make the alert disappear you can manage the security certificates in the device's Web UI Menu -> Network -> Certificates.

3.3. SETTING UP THE PROFINET MODE AND IP FROM THE WEBUI

The Brady IRX200 reader has different operating modes; normal mode, OPC UA and Profinet modes. In normal / OPC UA mode, the device's ethernet port is controlled by the Linux OS and Profinet functionalities are disabled. In this mode, the legacy UHF RFID interface called NurApi can be used to control the UHF RFID engine.

In the Profinet mode, the legacy NurApi interface is not available and control of the ethernet port is handed over to the real-time OS running the Profinet functionality.

The operating mode can be changed from the device's Web UI. Menu -> Industrial -> Protocol. See figure 4 below. Note that reboot of the device is needed before a new selection becomes active. While operating mode is Profinet, a Profinet specific sub-pages appears under the Menu -> Industrial.

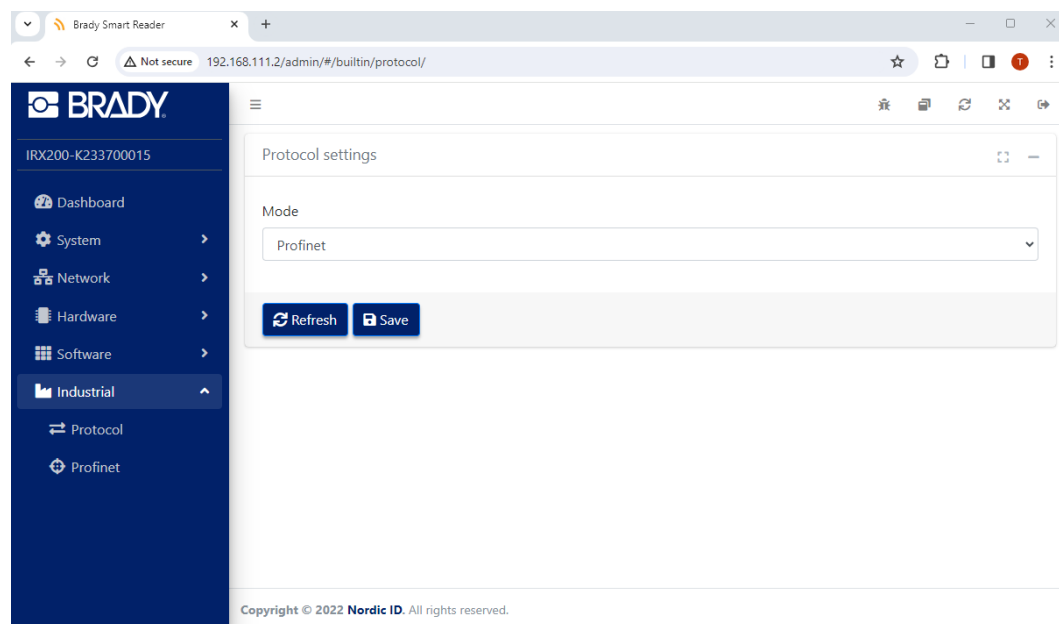


Figure 4. Web UI Industrial page.

From the Profinet Web UI page you can see the basic information of the reader, set the station settings and set the I&M1-3 records. In addition, the IRP interface message byte order setting can be selected.

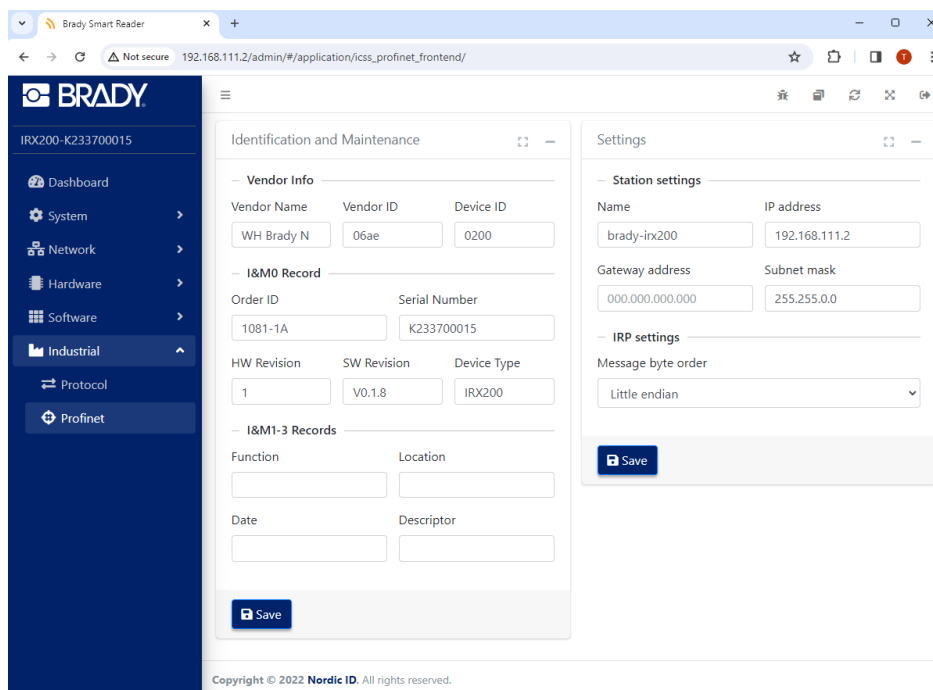


Figure 5. Web UI Profinet page.

3.4. NETWORK DISCOVERY AND SETUP USING PRONETA

Another way to discover and set up the station settings of the Brady IRX200 over the Profinet network is to use Siemens PRONETA discovery tool. To accomplish that you need to do the below steps.

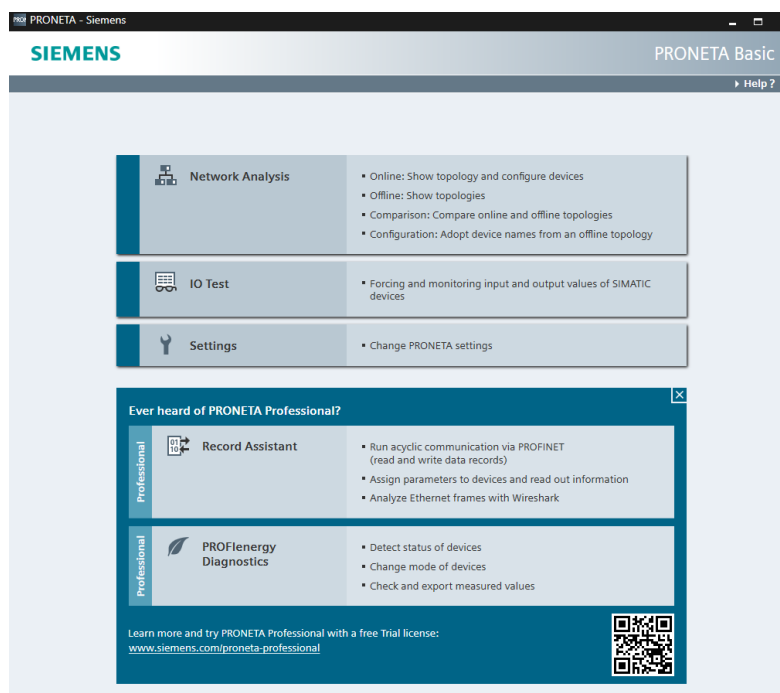


Figure 6. Siemens PRONETA main view.

1. Connect the ethernet data cable between the IRX200 and PC. Set the PC network adapters IP address to static and set the IP address and subnet mask. For example, IPv4 IP to 192.168.111.1 and subnet mask to 255.255.255.0.
2. Install and open the Siemens PRONETA tool.
3. Select “Network Analysis” from the main view.
4. The Brady IRX200 should now appear into view as in picture 6. By right clicking one can open the menu which allows you to interact with the reader. From the “Set network parameters” you can set the IP address and subnet mask to the reader. For example, IP address 192.168.111.2.
5. Optionally, with the PRONETA, you can also flash the devices high visibility LEDs to find the physical device in question, reset the device settings and read/write other I&M1-3 data records.

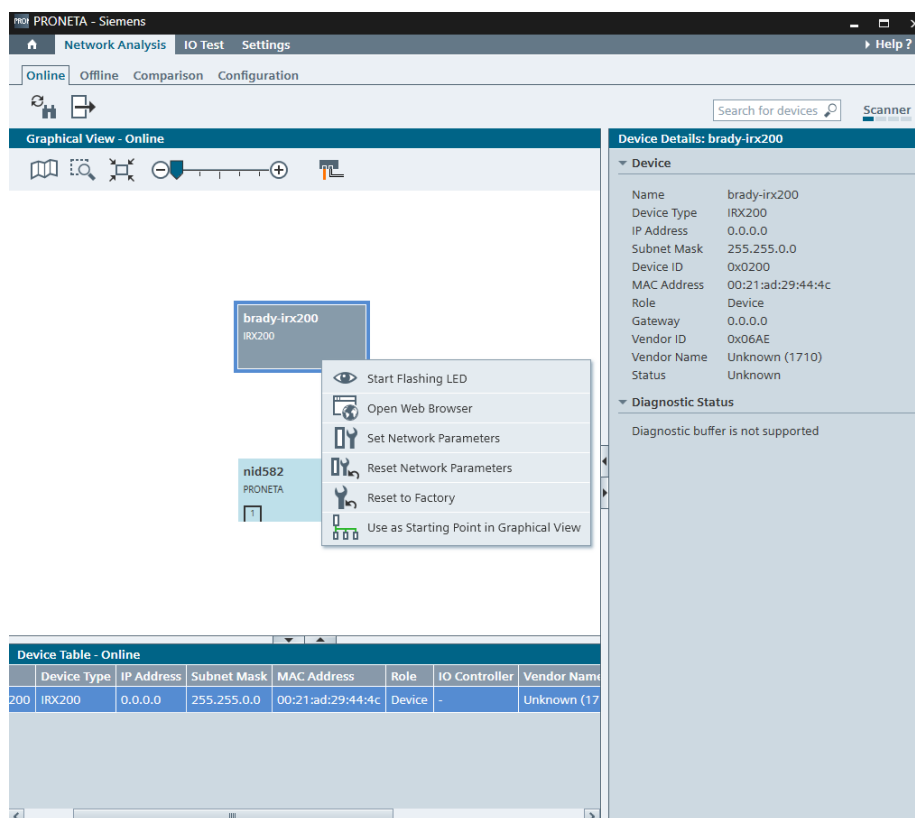


Figure 7. IRX200 discovered and right clicked.

4. IRX200 INDUSTRIAL READER CONFIGURATION WITH SIEMENS TIA PORTAL

This section will guide you on how to establish a communication between the PLC and IRX200 reader using Siemens TIA Portal and Simatic S7 series PLCs. Section 4.1 shows how to set up a new project and import necessary files whereas section 4.2 explains the provided sample's functionalities.

4.1. IRX200 HARDWARE CONFIGURATION

4.1.1. IMPORTING THE GSDML FILE

1. Open TIA portal (sample images are from V18).
2. From the options menu, select Manage General Station Description files (GSD) (Figure 7).

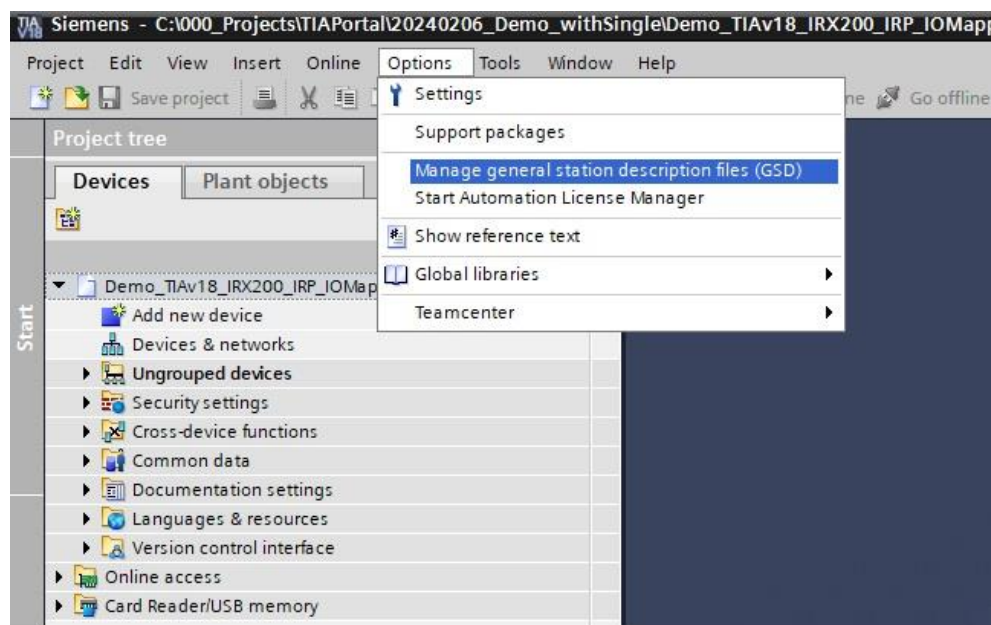


Figure 8. IRX200 configuration with TIA Portal - import the GSDML file.

3. Browse and select the GSDML file.
4. Install the GSDML file (Figure 8).

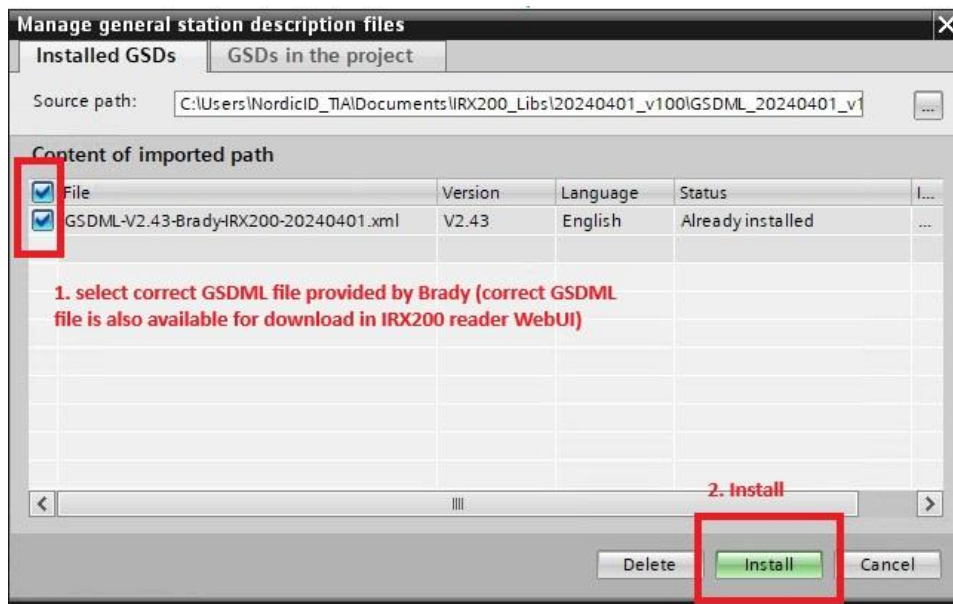


Figure 9. IRX200 configuration with TIA Portal - install the GSDML file.

4.1.2. ADDING A DEVICE TO THE PROFINET NETWORK AND HARDWARE CATALOG

After installing the GDML file, IRX200 is ready to be used in the project. To add the IRX200 to hardware configuration simply do the below:

Hardware Catalog → Other field devices → PROFINET IO → I/O → WH Brady NV → Brady RFID Readers → IRX200.

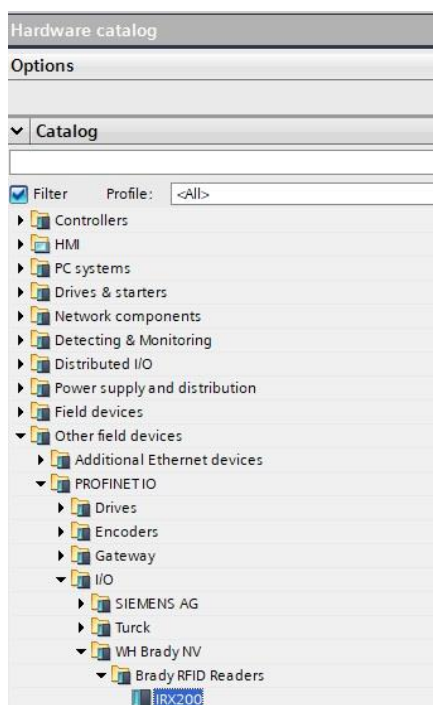
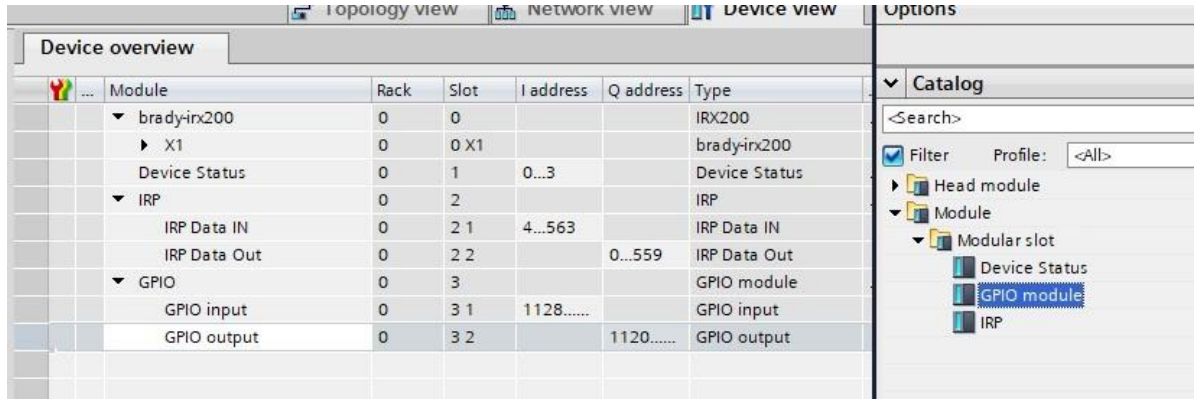


Figure 10. IRX200 configuration with TIA Portal - hardware catalog path.

As described in Section 7 there are 3 submodules available for IRX200 which are fixed in specific sub-slots. These modules can be found under Module à Modular slot category.



Module	Rack	Slot	I address	Q address	Type
brady-irx200	0	0			IRX200
X1	0	0 X1			brady-irx200
Device Status	0	1	0...3		Device Status
IRP	0	2			IRP
IRP Data IN	0	2 1	4...563		IRP Data IN
IRP Data Out	0	2 2		0...559	IRP Data Out
GPIO	0	3			GPIO module
GPIO input	0	3 1	1128.....		GPIO input
GPIO output	0	3 2		1120.....	GPIO output

Catalog

- <Search>
- Filter Profile: <All>
- Head module
- Module
 - Modular slot
 - Device Status
 - GPIO module
 - IRP

Figure 11. IRX200 configuration with TIA Portal – Sub-Modules.

After adding the IRX200 to Devices & networks, then simply connect the IRX200 to the PLC.

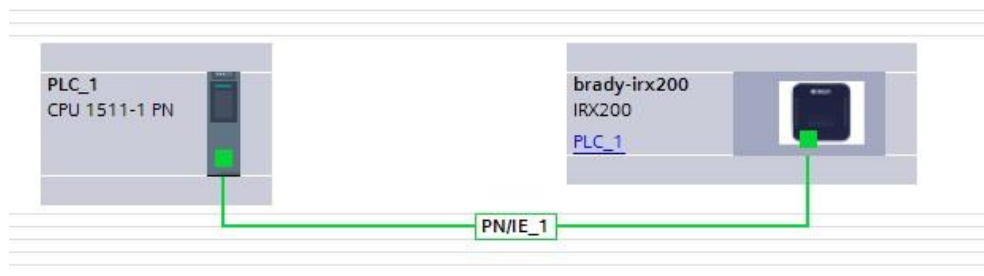


Figure 12. IRX200 configuration with TIA Portal - network connection.

4.1.3. DEVICE/MODULE PARAMETERS

After the project setup and physical connection, you can configure the device name, network settings and module parameters.

To set the device name, enter the same name as configured via device's web UI or using Proneta.

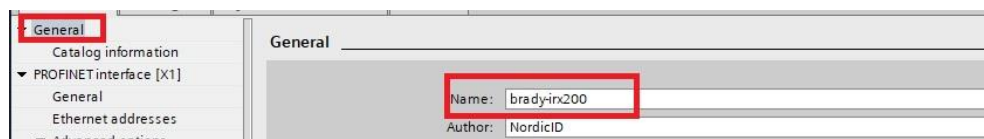


Figure 13. IRX200 configuration with TIA Portal - configure device name.

To configure the network settings, assign the IP and Subnet Mask, as they are necessary to enable communication with the PLC.

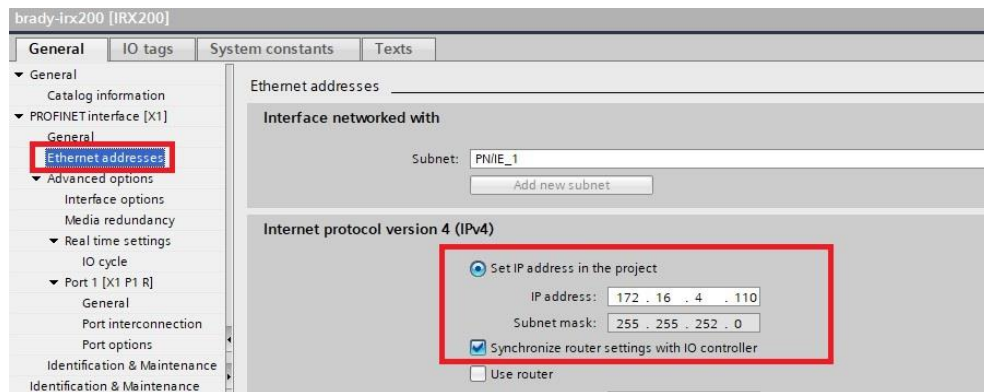


Figure 14. IRX200 configuration with TIA Portal - configure the IP and Subnet Mask.

From the module parameters, you can set the PLC properties override enabled or disabled and select the message byte order setting. If the device properties enabled setting is set to true, these properties settings will override the settings made from the device's web UI.

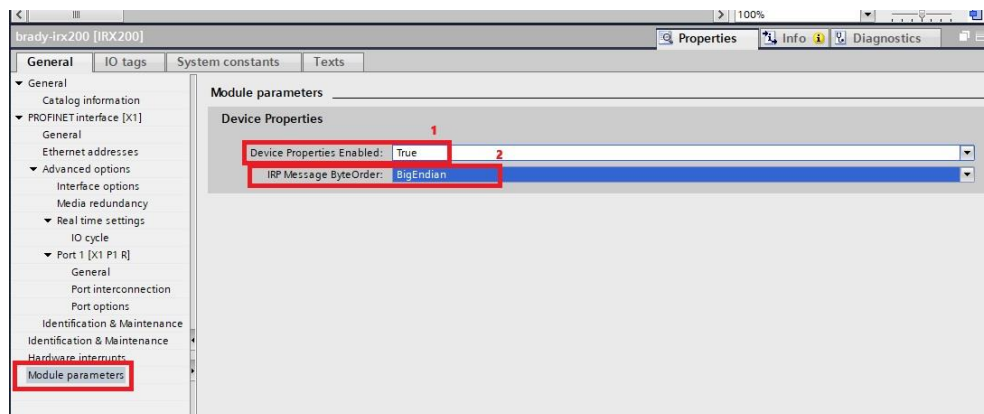


Figure 15. IRX200 configuration with TIA Portal - configure module parameters.

NOTE It is necessary to set IRP Message Byte Order to BigEndian to be able receive/send data correctly. This can be set simply from Module parameters in TIA portal hardware configuration as shown above or, from the IRX200 web UI.

IRP (Industrial reader protocol) input/output module's hardware ID numbers and allocated I/O addresses can be viewed via device view.

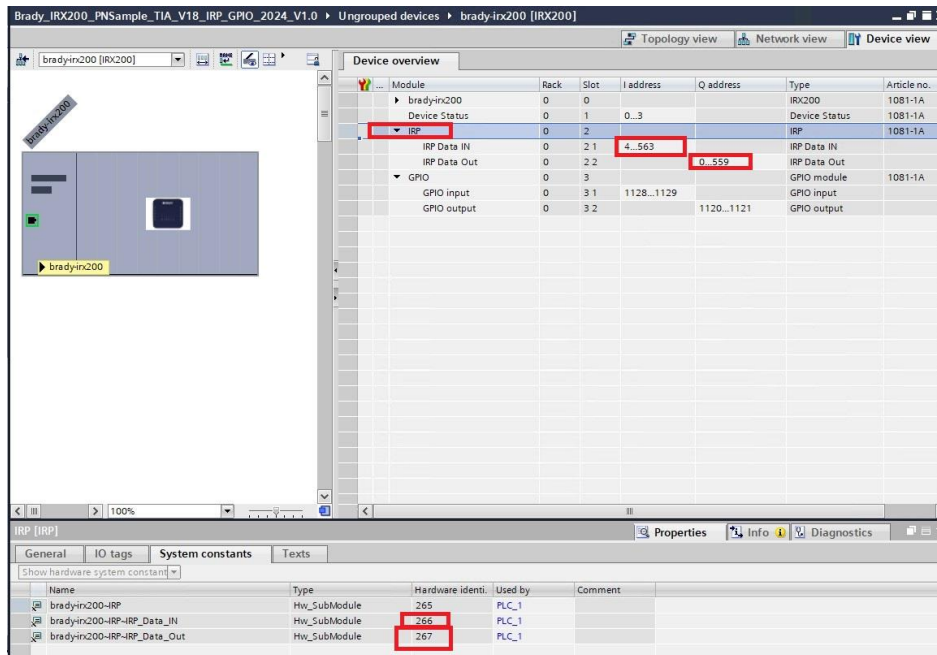


Figure 16. IRX200 configuration with TIA Portal - IRP module I/O's.

4.1.4. IMPORTING IRP USER DATA TYPES (UDTS AND ENUMS)

Import necessary UDTs supplied by Brady for the IRX200 IRP. Correct UDT package is included in software support package.

1. Go on Project Tree → PLC → PLC data types and right click and select Import from directory.

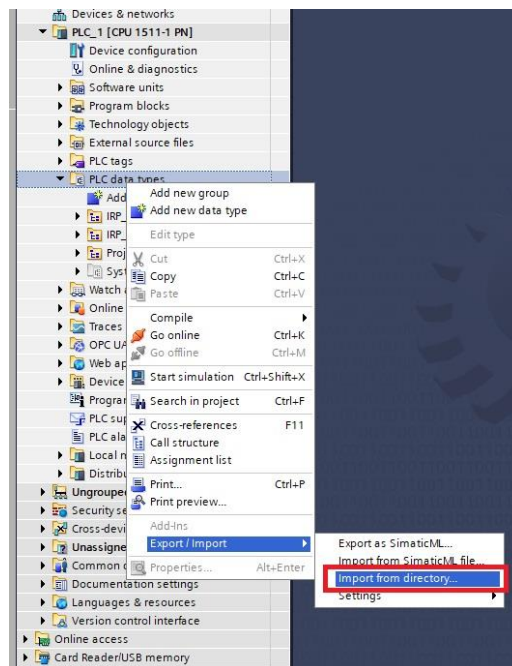


Figure17. IRX200 configuration with TIA Portal - Import from directory.

2. Select main folder to import all ENUMs and UDTs.



Figure 18. IRX200 configuration with TIA Portal - PLC UDTs Main Folder.

3. Wait for Import to finish and then Save and Compile to Project to confirm all the PLC user data types were imported successfully.
- 4.

4.1.5. IO MAPPING (GETIO / SETIO)

Input/Output format details are explained in section 7. Addition to details in that section, the IO mapping and parameters for the TIA portal are shown in below tables under each module's sub-title.

- **Device Status Module**

Inputs	Byte allocation	Description	GETIO Parameters
Device Status	0	Status Byte (* Details are in section 7)	LEN: 4
	1	Reserved for future expansion. Set to zero.	
	2-3	Number of messages on device waiting to be read.	

Table 1. IRX200 configuration with TIA Portal - Device Status input mapping.

- **IRP (Industrial Reader Protocol) Module**

Inputs	Byte allocation	Description	GETIO_PART Parameters
AckSendID	0-1	Device sets to match Send ID when ready to accept new requests	OFFSET: 0 LEN: 2
RecvID	2-3	Device changes value when new message available	OFFSET: 2 LEN: 2
Reserved	4-7	Reserved for future expansion. Set to zero.	OFFSET: 4 LEN: 4
	8-9	Type	OFFSET: 8

RecvMsg (Msg Data)	10-11	MsgID	LEN: 552
	12-13	Revision (IRP protocol Revision)	
	14-15	reserved	
	16-560	Payload (details for different type of payloads are presented in section 7.2)	

Table 2. IRX200 configuration with TIA Portal -IRP input mapping.

Outputs	Byte allocation	Description	SETIO_PART Parameters
AckRecvID	0-1	PLC sets to match Receive ID when ready to accept new messages.	OFFSET: 0 LEN: 2
SendID	2-3	PLC changes value to trigger request execution.	OFFSET: 2 LEN: 2
reserved	4-7	Reserved for future expansion. Set to zero.	OFFSET: 4 LEN: 4
SendMsg (Msg Data)	8-9	Type	OFFSET: 8 LEN: 552
	10-11	ID	
	12-13	Revision (IRP protocol Revision)	
	14-15	reserved	
	16-560	Payload (details for different type of payloads are presented in section 7.2)	

Table 3. IRX200 configuration with TIA Portal - IRP output mapping.

- **GPIO (General Purpose input-output) Module**

Inputs	Byte allocation	Description	GETIO Parameters
GPIO Input	0-1	GPIO Input (* Details are in section 7).	OFFSET: 0 LEN: 2

Table 4. IRX200 configuration with TIA Portal - GPIO input mapping.

Outputs	Byte allocation	Description	SETIO Parameters
GPIO Output	0-1	GPIO Output (* Details are in section 7).	OFFSET: 0 LEN: 2

Table 5. IRX200 configuration with TIA Portal – GPIO output mapping.

4.2. RUNNING THE SAMPLE PLC APPLICATION

This section describes the TIA portal sample project/application, which is provided by the Brady, (Brady_IRX200_PN_Sample_TIA_V18_IRP_GPIO_2024_V1.0) using 2 Brady IRX200 PN modules. Sample application is built with TIA Portal Version V18.

Hardware components that are used in this sample application are:

- Siemens CPU 1511-1 PN (Article Number: 6ES7 511-1AK02-0AB0)
- Brady IRX200 UHF RFID reader x2

Software/file versions used in this project are shown below.

Item	Description	Version
GSDML	General Station Description file associated with the Brady IRX200	GSDML-V2.43-Brady-IRX200-202400401
IRP UDTs & Enums	Industrial Reader Protocol user defined types and enums import files	IRP_SiemensXml
IRX200 firmware versions	IRX200 OS version / PN firmware version	OS FW V1 / PN FW V1

Table 6. TIA Portal sample application's used software/file versions.

The purpose of sample application is to demonstrate the sending and receiving of data to and from the IRX200 readers based on IRP API with all available message types and create event/data logs via data blocks.

4.2.1. SAMPLE APPLICATION PROJECT TREE VIEW

Below is the sample applications project tree view.

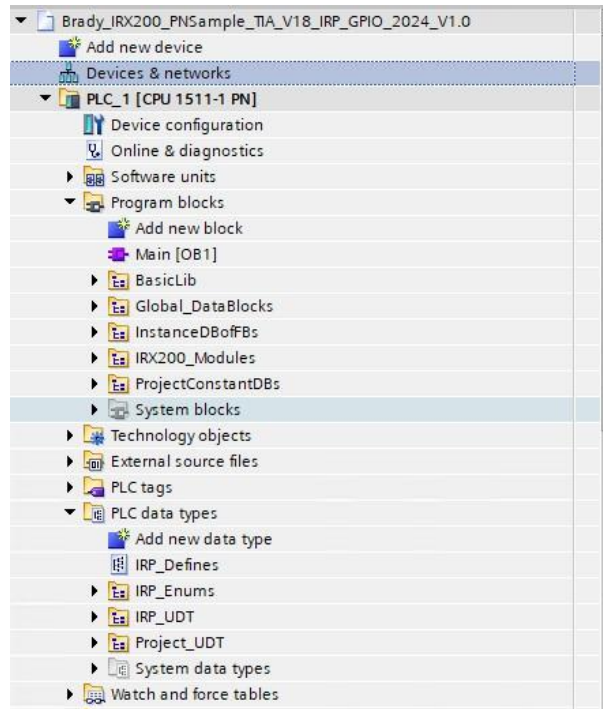


Figure 19. TIA Portal sample application - project tree

In this sample application, created with global data block interfaces, commands, responses and logs can be reviewed via **Global Data Blocks**.

All Global Data Blocks (except the ones used as a project Constants) are located under **Global_DataBlocks** folder. Global Data Blocks and their use case are explained below.

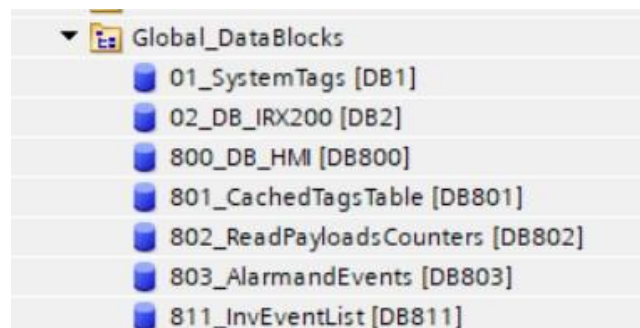


Figure 20. TIA Portal sample application - global data blocks.

- **01_SystemTags (DB1):** To Enable/Disable IRX200 modules, For each module giving station name, monitor and store data array counters, project sequence numbers and initializing bits handled with this DB.

- 02_DB_IRX200 (DB2): For each IRX200 raw input and output values, communication statuses during GET_IO, SET_IO and data mapping.
- 800_DB_HMI (DB800): Used as UI to send requests / data and read data for each IRX200 module.
- 801_CachedTagsTable (DB801): Stored/cached tags EPC data (Same array used for both of IRX200 modules).
- 802_ReadPayloadsCounters (DB802): Received message counters and unsorted messages (Counters and Unsorted List array used as common for both modules).
- 803_AlarmandEvents (DB803): Process alarms and events which included same list for both modules.
- 811_InvEventList (DB811): IRX200 inventory event list (Each module has their own list).

Other sub-folder program blocks are described below.

- BasicLib: Function Blocks for Data conversations, creating logs and strings.



Figure 21. TIA Portal sample application – Used BasicLib's.

IRX200 modules: IRX200 functions and function blocks for IO Mapping, Send/Read Messages triggers and sorting.



Figure 22. TIA Portal sample application - IRX200 modules.

InstanceDBofFBs: Instance data block type data blocks that are used in the sample application.

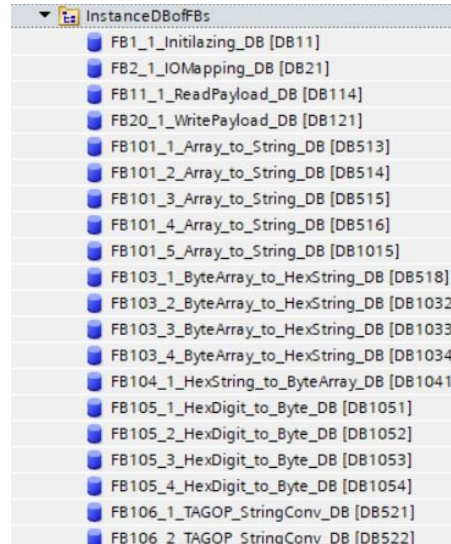


Figure 23. TIA Portal sample application – used InstanceDBofFBs.

ProjectConstants: Created with linked to the **IRP_ENUM** user data types and project specific constants to avoid of using direct values in program.

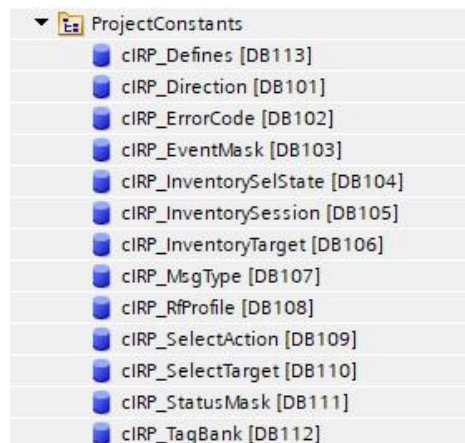


Figure 24. TIA Portal sample application - project constants.

PLC Data Types: **IRP_UDT** and **IRP_ENUM** folders are directly imported files as described in table 3. Sample application specific user data types are collected under **Project_UDT** subfolder. These are optional and application specific user data types.



Figure 25. TIA Portal sample application - PLC data types.

4.2.2. DEMO APPLICATION SIMULATION AND RESULTS

Simulation interface is handled via data block **800_DB_HMI** (DB800). After downloading and getting online to the PLC, open data block **800_DB_HMI** and start monitoring the data block to force simulation values. Below table shows the data structures.

Variable Name	Data Type	Description
ReaderConfigArgs	IRP_MsgPayload_ReaderConfig	Defined in IRP Specification
InventoryReqArgs	IRP_MsgPayload_InventoryReq	Defined in IRP Specification
InventoryReadConfigArgs	IRP_MsgPayload_InventoryReadConfig	Defined in IRP Specification
InventorySelectMaskConfigArgs	IRP_MsgPayload_InventorySelectMaskConfig	Defined in IRP Specification
ReaderCapabilitiesArgs	IRP_MsgPayload_ReaderCapabilities	Defined in IRP Specification
TagOpSelectMaskReq	IRP_MsgPayload_TagOpSelectMaskReq	Defined in IRP Specification
TagOpReadArgs	IRP_MsgPayload_TagOpReadReq	Defined in IRP Specification
TagOpWriteArgs	IRP_MsgPayload_TagOpWriteReq	Defined in IRP Specification
TagOpLockArgs	IRP_MsgPayload_TagOpLockReq	Defined in IRP Specification

TagOpKillArgs	IRP_MsgPayload_TagOpKillReq	Defined in IRP Specification
TagOpReadResp	IRP_MsgPayload_TagOpReadResp	Defined in IRP Specification
HMIStatus	HMI_Status	HMI/Process informative bits
ReaderCapabilitiesReaderIdStr	String[16]	Reader ID human readable string (empty during init)
ReaderCapabilitiesIcssFwVersionStr	String[16]	icss firmware version human readable string (empty during init)
ReaderCapabilitiesRfidFwVersionStr	String[16]	rfid firmware version human readable string (empty during init)
ReaderCapabilitiesModelNameStr	String[16]	Device Model name human readable string (empty during init)
InventorySelectMaskDataHexStr	String	Inventory Select Mask HEX string (empty during init)
TagOpSelectMaskDataHexStr	String	Tag Operation Select Mask HEX String (empty during init) used during Tag Operation Args Update.
TagOpWriteDataHexStr	String	Tag Operations Write Data HEX String format (empty during init) used as a data to be written during TAGOPWrite
ReadRespHexStr	String	Tag Operations Read Data HEX String Format (empty during init) updated via TAGOPRead
SendReq	Int	Trigger any IRP_MsgType Req. Sample application triggers SendRequest everytime value has changed (to send same request then use ReSendReq)
eventListLen	Int	Inventory Event List Len
SelectSpecificTAGIndex	Int	To select specific EPC data from CachedTagList to be used during UpdateTAGOpArgs if SelectSpecificTAGEn is TRUE
SelectSpecificTAGEn	Bool	To select specific EPC data from CachedTagList to be used during UpdateTAGOpArgs if it is TRUE then Application will copy EPC data from CachedTagList[SelectSpecificTAGIndex] during UpdateTAGOpArgs.
UpdateTAGOpArgs	Bool	To Update TagOpSelectMaskReq.MaskData
ReSendReq	Bool	To be able to send IRP Msg without changing the SendReq value

Table 7. TIA Portal sample application - 800_DB_HMI structures.

800_DB_HMI					
	Name	Data type	Sta...	Monitor value	Reta
	Static				
	IRX200HMI_1	"HMIData"			
	ReaderConfigArgs	"IRP_MsgPayload_Re...			
	InventoryReqArgs	"IRP_MsgPayload_In...			
	InventoryReadConfigArgs	"IRP_MsgPayload_In...			
	InventorySelectMaskConfigArgs	"IRP_MsgPayload_In...			
	ReaderCapabilitiesArgs	"IRP_MsgPayload_Re...			
	TagOpSelectMaskReq	"IRP_MsgPayload-Ta...			
	TagOpReadArgs	"IRP_MsgPayload-Ta...			
0	TagOpWriteArgs	"IRP_MsgPayload-Ta...			
1	TagOpLockArgs	"IRP_MsgPayload-Ta...			
2	TagOpKillArgs	"IRP_MsgPayload-Ta...			
3	TagOpReadResp	"IRP_MsgPayload-Ta...			
4	HMIStatus	"HMI_Status"			
5	ReaderCapabilitiesReaderIdStr	String[16]	"	"	
6	ReaderCapabilitiesIcssFwVersionStr	String[16]	"	"	
7	ReaderCapabilitiesRfidFwVersionStr	String[16]	"	"	
8	ReaderCapabilitiesModelNameStr	String[16]	"	"	
9	InventorySelectMaskDataHexStr	String	"	"	
0	TagOpSelectMaskDataHexStr	String	"	"	
1	TagOpWriteDataHexStr	String	"	"	
2	ReadRespHexStr	String	"	"	
3	SendReq	Int	0	0	
4	HMItagIndex	Int	0	0	
5	eventListLen	Int	1	1	
6	SelectSpecificTAGIndex	Int	0	0	
7	ReaderConfigEditNum	USInt	0	1	
8	ReaderConfigEditNumPrev	USInt	0	1	
9	InventorySelectMaskEditNum	USInt	0	1	
0	InventorySelectMaskEditNumPrev	USInt	0	1	
1	TAGReadWriteLen	USInt	0	0	
2	TagOPBankSelection	USInt	1	1	
3	SelectSpecificTAGEn	Bool	false	FALSE	
4	UpdateTAGOpArgs	Bool	false	FALSE	
5	ReSendReq	Bool	false	FALSE	

Figure 26. TIA Portal sample application: Examples - init values.

All the requests are triggered via SendReq variable. PLC triggers/sends requests every time when SendReq value is changed. In case of sending same request again, then simply toggle ReSendReq variable.

SendReq valid values and expected responses are detailed in section 7.2.1 Message Types and Associated Responses (for SendReq use decimal values).

Get Reader Capabilities (request and response).

IRX200HMI_1	"HMIData"		
▶ ReaderConfigArgs	"IRP_MsgPayload_R..."		
▶ InventoryReqArgs	"IRP_MsgPayload_I..."		
▶ InventoryReadCon...	"IRP_MsgPayload_I..."		
▶ InventorySelectMa...	"IRP_MsgPayload_I..."		
▶ ReaderCapabilities...	"IRP_MsgPayload_R..."		
minPowerDbm	DInt	0	1
maxPowerDbm	DInt	0	30
stepDb	DInt	0	1
region	USInt	0	0
maxNumAnten...	USInt	0	2
readerIdLen	USInt	16	10
icssFwVersionL...	USInt	16	5
rfdFwVersionL...	USInt	16	6
modelNameLen	USInt	16	7
▶ readerId	Array[0..15] of USInt		
▶ icssFwVersion	Array[0..15] of USInt		
▶ rfdFwVersion	Array[0..15] of USInt		
▶ modelName	Array[0..15] of USInt		
reserved0	USInt	0	0
reserved1	USInt	0	0
▶ TagOpSelectMaskR...	"IRP_MsgPayload_T..."		
▶ TagOpReadArgs	"IRP_MsgPayload_T..."		
▶ TagOpWriteArgs	"IRP_MsgPayload_T..."		
▶ TagOpLockArgs	"IRP_MsgPayload_T..."		
▶ TagOpKillArgs	"IRP_MsgPayload_T..."		
▶ TagOpReadResp	"IRP_MsgPayload_T..."		
HMIStatus	"HMI_Status"		
ReaderCapabilities...	String[16]	"	'K234500055'
ReaderCapabilitiesI...	String[16]	"	'0.8.1'
ReaderCapabilities...	String[16]	"	'17.2-A'
ReaderCapabilities...	String[16]	"	'1081-1A'
inventoryselectma...	String		
TagOpSelectMask...	String	"	"
TagOpReadResp	String	"	"
IRP_MsgType_GetReaderCapabilitiesReq DEC value	String	"	"
SendReq	Int	0	16
HMI tagIndex	Int	0	0
eventListLen	Int	1	1
SelectSpecificTAGI...	Int	0	0
ReaderConfigEditN...	USInt	0	1
ReaderConfigEditN...	USInt	0	1
InventorySelectMa...	USInt	0	1
InventorySelectMa...	USInt	0	1
TAGReadWriteLen	USInt	0	0
TagOPBankSelector	USInt	1	1
SelectSpecificTAGEn	Bool	false	FALSE
UpdateTAGOpArgs	Bool	false	FALSE
ReSendReq	Bool	false	FALSE

Figure 27. TIA Portal sample application: Examples - get reader capabilities req and response.

Get Reader Configuration (request and Response).

IRX200HMI_1	"HMIData"		
ReaderConfigArgs	"IRP_MsgPayload_Re..."		
tagReadTimeoutMs	UInt	0	500
tagWriteTimeoutMs	UInt	0	500
tagLockTimeoutMs	UInt	0	500
tagKillTimeoutMs	UInt	0	500
rfProfile	USInt	0	1
reserved0	USInt	0	0
reserved1	USInt	0	0
numAntennaConfigs	USInt	0	1
antennaConfigs	Array[0..31] of "IRP_..."		
InventoryReqArgs	"IRP_MsgPayload_In..."		
InventoryReadConfigArgs	"IRP_MsgPayload_In..."		
InventorySelectMaskConfigArgs	"IRP_MsgPayload_In..."		
ReaderCapabilitiesArgs	"IRP_MsgPayload_Re..."		
TagOpSelectMaskReq	"IRP_MsgPayload-Ta..."		
TagOpReadArgs	"IRP_MsgPayload-Ta..."		
TagOpWriteArgs	"IRP_MsgPayload-Ta..."		
TagOpLockArgs	"IRP_MsgPayload-Ta..."		
TagOpKillArgs	"IRP_MsgPayload-Ta..."		
TagOpReadResp	"IRP_MsgPayload-Ta..."		
HMIStatus	"HMI_Status"		
ReaderCapabilitiesReaderIdStr	String[16]	"	'K234500055'
ReaderCapabilitiesIcssFwVersionStr	String[16]	"	'0.1.8'
ReaderCapabilitiesRfidFwVersionStr	String[16]	"	'17.1-D'
ReaderCapabilitiesModelNameStr	String[16]	"	'1081-1A'
InventorySelectMaskDataHexStr	String	"	"
TagOpSelectMaskDataHexStr	String	"	"
TagOpWriteDataHexStr	String	"	"
ReadRespHexStr	String	"	"
SendReq	Int	0	17
rmiTagIndex	Int	0	0
eventListLen	Int	1	1
SelectSpecificTAGIndex	Int	0	0
ReaderConfigEditNum	USInt	0	1
ReaderConfigEditNumPrev	USInt	0	1
InventorySelectMaskEditNum	USInt	0	1

Figure 28. TIA Portal sample application: Examples - get antenna config.

Set Reader Configuration (request and response).

As described in section 7.2(IRP Description), **SetReaderConfigReq** requires **ReaderConfig** payload. In this sample application, **SetReaderConfigReq** and **GetReaderConfigReq** uses the same **ReaderConfig** payload data.

So, if **GetReaderConfigReq** already sent, then **ReaderConfig** data is already filled with default **ReaderConfig** arguments. To change any configuration, simply set values in **ReaderConfigArgs** data (shown in figure 23) and then set **SendReq** value to as described in section 7.2(IRP Description), **IRP_MsgType_SetReaderConfigReq** DEC value.

To track, if the request successfully completed, simply view IRX200 logs from web UI or from data block 803_AlarmandEvents (DB803).

803_AlarmandEvents				
Name	Data...	Star...	Monitor value	Re
Static				
AE_view	Arra...			
AE_view[0]	String	"		
AE_view[1]	String	"	'IRX200-1: Req. Type: GetReaderCapabilitiesReq sent succesfully with MsgID:+40308'	
AE_view[2]	String	"	'IRX200-1: IRP MsgType: GetReaderCapabilitiesResp with success'	
AE_view[3]	String	"	'IRX200-1: Req. Type: GetReaderConfigReq sent succesfully with MsgID:+54549'	
AE_view[4]	String	"	'IRX200-1: IRP MsgType: GetReaderConfigResp with success'	
AE_view[5]	String	"	'IRX200-1: Req. Type: SetReaderConfigReq sent succesfully with MsgID:+36428'	
AE_view[6]	String	"	'IRX200-1: Req. Type: SetReaderConfigReq returned with Errorcode response: 'Success'	
AE_view[7]	String	"		
AE_view[8]	String	"		

Figure 29. TIA Portal sample application - set reader config event log in DB.

Get Inventory Reader Configuration (request and response).

IRX200HMI_1	*HMIData*			
ReaderConfigArgs	*IRP_MsgPayload_R...			
InventoryReqArgs	*IRP_MsgPayload_I...			
InventoryReadCon...	*IRP_MsgPayload_I...			
enabled	USInt	0	0	
bank	USInt	0	0	
reserved0	USInt	0	0	
readLenBytes	USInt	0	0	
addressBytes	UDInt	0	0	
InventorySelectMa...	IRP_MsgPayload_I...			
ReaderCapabilities...	*IRP_MsgPayload_R...			
TagOpSelectMaskR...	*IRP_MsgPayload_T...			
TagOpReadArgs	*IRP_MsgPayload_T...			
TagOpWriteArgs	*IRP_MsgPayload_T...			
TagOpLockArgs	*IRP_MsgPayload_T...			
TagOpKillArgs	*IRP_MsgPayload_T...			
TagOpReadResp	*IRP_MsgPayload_T...			
HMIStatus	*HMI_Status*			
ReaderCapabilities...	String[16]			'K234500055'
ReaderCapabilitiesI...	String[16]			'0.8.1'
ReaderCapabilities...	String[16]			'17.2-A'
ReaderCapabilities...	String[16]			'1081-1A'
InventorySelectMa...	String			
TagOpSelectMask...	String			
TagOpWriteDataH...	String			
ReadRespHexStr	String			
SendReq	Int	0	19	
HMI TagIndex	Int	0	0	
eventListLen	Int	1	1	
SelectSpecificTAGI...	Int	0	0	

Figure 30. TIA Portal sample application - get Inv. reader configuration.

Set Inventory Reader Configuration (request and response).

As described in section 7.2 (IRP Description), SetInventoryReadConfigReq requires InventoryReadConfig payload. In this sample application SetInventoryReadConfigReq and GetInventoryReadConfigReq uses the same InventoryReadConfig payload data.

So, if **GetInventoryReadConfigReq** already sent, then **InventoryReadConfig** data is already filled with default **InventoryReadConfig** arguments.

To change any configuration, simply set values in InventoryReadConfig data (shown in figure 25) and then set SendReq value to as described in section 7.2 (IRP Description), IRP_MsgType_SetInventoryReadConfigReq DEC value.

To track if the request successfully completed, simply view IRX200 logs from web UI or from data block 803_AlarmandEvents (DB803).

Inventory (request and response).

IRX200HMI_1	"HMIData"				
ReaderConfigArgs	"IRP_MsgPayload_ReaderConfig"				
InventoryReqArgs	"IRP_MsgPayload_InventoryReq"				
antennaMask	UDInt	0	0		
cycleTimeMs	UDInt	0	0		
stopTagLimit	UDInt	0	0		
stopCycleLimit	UDInt	0	0		
stopStaticTimeoutMs	UDInt	0	0		
stopNoNewTagsTimeoutMs	UDInt	0	0		
eventMask	UDInt	0	4_294_967_295		
eventTagTimeoutMs	UDInt	0	12600		
eventAntennaMask	UDInt	0	4_294_967_295		
eventRssiThreshold	USInt	0	0		
eventSeenCountInterval	USInt	0	1		
reserved0	USInt	0	0		
reserved1	USInt	0	0		
InventoryHeadConfigArgs	"IRP_MsgPayload_InventoryHeadConfig"				
InventorySelectMaskConfigArgs	"IRP_MsgPayload_InventorySelectMaskConfig"				
ReaderCapabilitiesArgs	"IRP_MsgPayload_ReaderCapabilities"				
TagOpSelectMaskReq	"IRP_MsgPayload_TagOpSelectMaskReq"				
TagOpReadArgs	"IRP_MsgPayload_TagOpReadReq"				
TagOpWriteArgs	"IRP_MsgPayload_TagOpWriteReq"				
TagOpLockArgs	"IRP_MsgPayload_TagOpLockReq"				
TagOpKillArgs	"IRP_MsgPayload_TagOpKillReq"				
TagOpReadResp	"IRP_MsgPayload_TagOpReadResp"				
HMIStatus	"HMI_Status"				
ReaderCapabilitiesReaderIdStr	String[16]	"	"K234500055"		
ReaderCapabilitiesIcs:FwVersionStr	String[16]	"	"0.8.1"		
ReaderCapabilitiesRfidFwVersionStr	String[16]	"	"17.2-A"		
ReaderCapabilitiesModelNameStr	String[16]	"	"1081-1A"		
InventorySelectMaskDataHexStr	String	"	"		
TagOpSelectMaskDataHexStr	String	"	"		
TagOpWriteDataHexStr	String	"	"		
ReadRespHexStr	String	"	"		
SendReq	Int	0	23		
eventListLen	Int	1	12		
SelectSpecificTAGIndex	Int	0	0		
ReaderConfigEditNum	USInt	0	1		
ReaderConfigEditNumPrev	USInt	0	1		
InventorySelectMaskEditNum	USInt	0	1		

Figure 31. TIA Portal sample application - inventory request.

811_InvEventList			
	Name	Data type	Monitor value
1	Static		
2	Events_IRX200_2	Array[0..19] of String	
3	Events_IRX200_1	Array[0..19] of String	
4	Events_IRX200_1[0]	String[128]	" +32 E2002064860902250390E577 InventorySeenCountIntervalEvent seencount:+1322 =>+1323"
5	Events_IRX200_1[1]	String[128]	" +32 E2002064860902250380E576 InventorySeenCountIntervalEvent seencount:+1323 =>+1324"
6	Events_IRX200_1[2]	String[128]	" +126 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1295 =>+1296"
7	Events_IRX200_1[3]	String[128]	" +126 E2002064860902250370E575 InventorySeenCountIntervalEvent seencount:+1324 =>+1325"
8	Events_IRX200_1[4]	String[128]	" +32 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1295 =>+1296"
9	Events_IRX200_1[5]	String[128]	" +32 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1295 =>+1296"
10	Events_IRX200_1[6]	String[128]	" +32 E2002064860902250370E575 InventorySeenCountIntervalEvent seencount:+1323 =>+1324"
11	Events_IRX200_1[7]	String[128]	" +32 E2002064860902250390E577 InventorySeenCountIntervalEvent seencount:+1322 =>+1323"
12	Events_IRX200_1[8]	String[128]	" +32 E2002064860902250380E576 InventorySeenCountIntervalEvent seencount:+1323 =>+1324"
13	Events_IRX200_1[9]	String[128]	" +126 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1295 =>+1296"
14	Events_IRX200_1[10]	String[128]	" +126 E2002064860902250370E575 InventorySeenCountIntervalEvent seencount:+1324 =>+1325"
15	Events_IRX200_1[11]	String[128]	" +126 E2002064860902250390E577 InventorySeenCountIntervalEvent seencount:+1323 =>+1324"
16	Events_IRX200_1[12]	String[128]	" +126 E2002064860902250380E576 InventorySeenCountIntervalEvent seencount:+1324 =>+1325"
17	Events_IRX200_1[13]	String[128]	" +126 E2002064860902250380E576 InventorySeenCountIntervalEvent seencount:+1324 =>+1325"
18	Events_IRX200_1[14]	String[128]	" +126 E2002064860902250360EA78 InventorySeenCountIntervalEvent seencount:+1325 =>+1326"
19	Events_IRX200_1[15]	String[128]	" +236 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1296 =>+1297"
20	Events_IRX200_1[16]	String[128]	" +236 E2002064860902240340EA72 InventorySeenCountIntervalEvent seencount:+1296 =>+1297"
21	Events_IRX200_1[17]	String[128]	" +236 E2002064860902250370E575 InventorySeenCountIntervalEvent seencount:+1325 =>+1326"
22	Events_IRX200_1[18]	String[128]	" +236 E2002064860902250390E577 InventorySeenCountIntervalEvent seencount:+1324 =>+1325"
23	Events_IRX200_1[19]	String[128]	" +236 E2002064860902250380E576 InventorySeenCountIntervalEvent seencount:+1325 =>+1326"

Figure 32. TIA Portal sample application - inventory request event logs.

803_AlarmAndEvents			
	Name	Data type	Monitor value
1	Static		
2	AE_view	Array...	
3	AE_view[0]	String	"IRX200-1 IRP MsgType: InventorySeenCountIntervalEvent with success"
4	AE_view[1]	String	"IRX200-2 IRP MsgType: InventorySeenCountIntervalEvent with success"
5	AE_view[2]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
6	AE_view[3]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
7	AE_view[4]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
8	AE_view[5]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
9	AE_view[6]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
10	AE_view[7]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
11	AE_view[8]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
12	AE_view[9]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
13	AE_view[10]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
14	AE_view[11]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
15	AE_view[12]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"
16	AE_view[13]	String	"IRX200-2: IRP MsgType: InventorySeenCountIntervalEvent with success"
17	AE_view[14]	String	"IRX200-1: IRP MsgType: InventorySeenCountIntervalEvent with success"

Figure 33. TIA Portal sample application - inventory event list.

802_ReadPayloadsCounters				
	Name	Data type	Start value	Monitor v
	Static			
	UnsortedLogs	Array[0..49] ...		
	TotalUnsortedCount	DInt	0	0
	UnsortedIndex	DInt	0	0
	ErrorCodeRespCount	DInt	0	4
	GetReaderCapabilities...	DInt	0	1
	GetReaderConfigResp...	DInt	0	1
	GetInventoryReadCon...	DInt	0	1
	GetInventorySelectMa...	DInt	0	1
	TagOpReadRespCount	DInt	0	0
	GlobalRfOnOffEventC...	DInt	0	0
	GlobalDiagnosticEven...	DInt	0	0
	InventoryAntennaVisi...	DInt	0	11
	InventoryGlobalVisibil...	DInt	0	11
	InventoryRssiDeltaThr...	DInt	0	0
	InventorySeenCountIn...	DInt	0	421
	InventoryCycleTimeEx...	DInt	0	0

Figure 34. TIA Portal sample application - received Msg counters during Inventory.

Tag Operation Select Mask (request and response):

In this sample application there are 2 different ways to create mask data. Both are shown below:

1. By creating mask data with cached tag table item. In this option application sends EPC data from selected **CachedTagTable** item. All the arguments will be set automatically. To apply this selection simply follow the steps below.

IRX200HMI_1	"HMIData"			
ReaderConfigArgs	"IRP_MsgP..."			
InventoryReqArgs	"IRP_MsgP..."			
InventoryReadConfigArgs	"IRP_MsgP..."			
InventorySelectMaskConfigArgs	"IRP_MsgP..."			
ReaderCapabilitiesArgs	"IRP_MsgP..."			
TagOpSelectMaskReq	"IRP_MsgP..."			
enabled	USInt	0	0	
bank	USInt	0	0	
maskLenBits	UInt	0	0	
addressBits	UDInt	0	0	
maskData	Array[0..3...			
TagOpReadArgs	"IRP_MsgP..."			
TagOpWriteArgs	"IRP_MsgP..."			
TagOpLockArgs	"IRP_MsgP..."			
TagOpKillArgs	"IRP_MsgP..."			
TagOpReadResp	"IRP_MsgP..."			
HMIStatus	"HMI_Stat..."			
ReaderCapabilitiesReaderIdStr	String[16]	"	"K234500055"	
ReaderCapabilitiesIcssFwVersionStr	String[16]	"	"0.8.1"	
ReaderCapabilitiesRfidFwVersionStr	String[16]	"	"17.2-A"	
ReaderCapabilitiesModelNameStr	String[16]	"	"1081-1A"	
InventorySelectMaskDataHexStr	String	"	"	
TagOpSelectMaskDataHexStr	String	"	"	
TagOpWriteDataHexStr	String	"	"	
ReadRespHexStr	String	"	"	
SendReq	Int	0	30	4
HMI tagIndex	Int	0	0	
eventListLen	Int	1	20	
SelectSpecificTAGIndex	Int	0	1	1
ReaderConfigEditNum	USInt	0	1	
ReaderConfigEditNumPrev	USInt	0	1	
InventorySelectMaskEditNum	USInt	0	1	
InventorySelectMaskEditNumPrev	USInt	0	1	
TAGReadWriteLen	USInt	0	0	
TagOPBankSelection	USInt	1	1	
SelectSpecificTAGEn	Bool	false	TRUE	2
UpdateTAGOpArgs	Bool	false	FALSE	3
ResendReq	Bool	false	FALSE	

Figure 35. TIA Portal sample application - TAGOP SelectMask EPC data from CachedTagTable.

- Set the value between 0 to 64 to select array element EPC data from data block **801_CachedTagsTable** (DB803).
- Set **SelectSpecificTAGEn** to TRUE
- Toogle the **UpdateTAGOpArgs** (application will set the bit to FALSE at the end of the operation). In this step, the application will fill in the **TagOpSelectMaskReq** values based on selected TAG data. In below figure 31 **SelectSpecificTAGIndex** is set to 1, so EPC Data of **801_CachedTagsTable.CachedTags[1]** is copied to **TagOpSelectMaskReq** values.

Figure 36. TIA Portal sample application – TAGOP Select Mask from CachedTagTable Compare.

- ```

17548850 Request changed! requestId=0x9
> 17548850 _IRP_autoGen_validate_IRP_MsgHeader(0xa04241ec,1):
> 17548850 type->24
> 17548850 id->1075
> 17548850 IRP_autoGen_validate_IRP_MsgPayload_TagOpSelectMaskReq(0xa04241f0,1):
17548850 enabled->1
17548850 bank->1
17548850 maskLenBits->96
17548850 addressBits->32
> 17548850 maskData[12]->'E2801191A50200601C016F2C' EPC HEX String
17548850 received from PN, type 0x18 (TagOpSelectMaskReq)
17548850 BusyLock acquired for TagOpSelectMaskReq
> 17548850 BusyLock freed by TagOpSelectMaskReq
> 17548850 sending to PN id 1075, type 0x40 (ErrorCodeResp)
> 17548850 _IRP_autoGen_validate_IRP_MsgHeader(0xa0425e95,2):
> 17548850 type->64
> 17548850 id->1075 IRP_MsgID can be seen in AlarmandEventLog too
> 17548850 IRP_autoGen_validate_IRP_MsgPayload_ErrorCode(0xa0425e99,2):
> 17548850 errorCode->0 0 : Success
> 17548850 _IRP_tpPN_send() send queued, waiting: 1
17548850 LED_Set(data, green)
17548856 LED_Set(data, off)
> 17548856 Sent queued response 0/1024 with sendId:687

```

**BRADY Worldwide, Inc.** | ©2022 All rights reserved.  
www.bradyeurope.com



2. By creating mask data with manual entries. In this option user has to entry **TagOpSelectMaskReq** values manually and entry the **TagOpSelectMaskDataHexStr** manually and then toggle the **UpdateTAGOpArgs**. In this option, it is also possible to send mask data from different banks.

The entered **TagOpSelectMaskDataHexStr** data will be copied to **MaskData** array based on **TagOpSelectMaskReq.maskLenBits** value (size of array defined with that entry).

Tag Operations: Read (request and response):

As an example to previous operation, after sending the TAGOP select mask req, user could try to read EPC data of selected mask which could verify the mask data and selected tag are matching.

|                                    |              |       |       |                            |
|------------------------------------|--------------|-------|-------|----------------------------|
| TagOpReadArgs                      | *IRP_MsgP... |       |       |                            |
| antennaMask                        | UDInt        | 0     | 0     |                            |
| accessPasswd                       | UDInt        | 0     | 0     |                            |
| bank                               | USInt        | 0     | 1     |                            |
| reserved0                          | USInt        | 0     | 0     |                            |
| reserved1                          | USInt        | 0     | 0     |                            |
| readLenBytes                       | USInt        | 0     | 12    |                            |
| addressBytes                       | UDInt        | 0     | 4     |                            |
| TagOpWriteArgs                     | *IRP_MsgP... |       |       |                            |
| TagOpLockArgs                      | *IRP_MsgP... |       |       |                            |
| TagOpKillArgs                      | *IRP_MsgP... |       |       |                            |
| TagOpReadResp                      | *IRP_MsgP... |       |       |                            |
| HMIStatus                          | *HMI_Stat... |       |       |                            |
| ReaderCapabilitiesReaderIdStr      | String[16]   | "     |       | 'K234500055'               |
| ReaderCapabilitiesIcssFwVersionStr | String[16]   | "     |       | '0.8.1'                    |
| ReaderCapabilitiesRfidFwVersionStr | String[16]   | "     |       | '17.2-A'                   |
| ReaderCapabilitiesModelNameStr     | String[16]   | "     |       | '1081-1A'                  |
| InventorySelectMaskDataHexStr      | String       | "     |       | "                          |
| TagOpSelectMaskDataHexStr          | String       | "     |       | "                          |
| TagOpWriteDataHexStr               | String       | "     |       | "                          |
| ReadRespHexStr                     | String       | "     |       | 'E2801191A50200601C016F2C' |
| SendReq                            | Int          | 0     | 26    |                            |
| HMIIndex                           | Int          | 0     | 0     |                            |
| eventListLen                       | Int          | 1     | 20    |                            |
| SelectSpecificTAGIndex             | Int          | 0     | 1     |                            |
| ReaderConfigEditNum                | USInt        | 0     | 1     |                            |
| ReaderConfigEditNumPrev            | USInt        | 0     | 1     |                            |
| InventorySelectMaskEditNum         | USInt        | 0     | 1     |                            |
| InventorySelectMaskEditNumPrev     | USInt        | 0     | 1     |                            |
| TAGReadWriteLen                    | USInt        | 0     | 0     |                            |
| TagOPBankSelection                 | USInt        | 1     | 1     |                            |
| SelectSpecificTAGEn                | Bool         | false | FALSE |                            |
| UpdateTAGOpArgs                    | Bool         | false | FALSE |                            |
| ReSendReq                          | Bool         | false | FALSE |                            |

Figure 38. TIA Portal sample application - TAGOP Read EPC data.

|                                    |              |       |       |                                    |  |
|------------------------------------|--------------|-------|-------|------------------------------------|--|
| TagOpReadArgs                      | *IRP_MsgP... |       |       |                                    |  |
| antennaMask                        | UDInt        | 0     | 0     |                                    |  |
| accessPasswd                       | UDInt        | 0     | 0     |                                    |  |
| bank                               | USInt        | 0     | 0     |                                    |  |
| reserved0                          | USInt        | 0     | 0     |                                    |  |
| reserved1                          | USInt        | 0     | 0     |                                    |  |
| readLenBytes                       | USInt        | 0     | 0     |                                    |  |
| addressBytes                       | UDInt        | 0     | 0     |                                    |  |
| TagOpWriteArgs                     | *IRP_MsgP... |       |       |                                    |  |
| TagOpLockArgs                      | *IRP_MsgP... |       |       |                                    |  |
| TagOpKillArgs                      | *IRP_MsgP... |       |       |                                    |  |
| TagOpReadResp                      | *IRP_MsgP... |       |       |                                    |  |
| HMIStatus                          | *HMI_Stat... |       |       |                                    |  |
| ReaderCapabilitiesReaderIdStr      | String[16]   | "     | "     | 'K234500055'                       |  |
| ReaderCapabilitiesIcssFwVersionStr | String[16]   | "     | "     | '0.8.1'                            |  |
| ReaderCapabilitiesRfidFwVersionStr | String[16]   | "     | "     | '17.2-A'                           |  |
| ReaderCapabilitiesModelNameStr     | String[16]   | "     | "     | '1081-1A'                          |  |
| InventorySelectMaskDataHexStr      | String       | "     | "     | "                                  |  |
| TagOpSelectMaskDataHexStr          | String       | "     | "     | "                                  |  |
| TagOpWriteDataHexStr               | String       | "     | "     | "                                  |  |
| ReadRespHexStr                     | String       | "     | "     | '0000000000000000402847523A7B3000' |  |
| SendReq                            | Int          | 0     | 20    |                                    |  |
| HMITagIndex                        | Int          | 0     | 0     |                                    |  |
| eventListLen                       | Int          | 1     | 20    |                                    |  |
| SelectSpecificTAGIndex             | Int          | 0     | 1     |                                    |  |
| ReaderConfigEditNum                | USInt        | 0     | 1     |                                    |  |
| ReaderConfigEditNumPrev            | USInt        | 0     | 1     |                                    |  |
| InventorySelectMaskEditNum         | USInt        | 0     | 1     |                                    |  |
| InventorySelectMaskEditNumPrev     | USInt        | 0     | 1     |                                    |  |
| TAGReadWriteLen                    | USInt        | 0     | 0     |                                    |  |
| TagOPBankSelection                 | USInt        | 1     | 1     |                                    |  |
| SelectSpecificTAGEn                | Bool         | false | FALSE |                                    |  |
| UpdateTAGOpArgs                    | Bool         | false | FALSE |                                    |  |
| ReSendReq                          | Bool         | false | FALSE |                                    |  |

1: Set  
TagOpReadArgs to  
read all password  
data

Human readable HEXstring  
of password data will be  
shown here

2: Toggle ReSendReq

Figure 39. TIA Portal sample application - TAGOP Read password data.

Tag Operations: Write (request and response).

|                                    |               |       |       |                                                                                                  |
|------------------------------------|---------------|-------|-------|--------------------------------------------------------------------------------------------------|
| TagOpReadArgs                      | *IRP_MsgP...  |       |       |                                                                                                  |
| TagOpWriteArgs                     | *IRP_MsgP...  |       |       |                                                                                                  |
| antennaMask                        | UDInt         | 0     | 0     | 1: Set the TagOpWriteArgs values to write password data.                                         |
| accessPasswd                       | UDInt         | 0     | 0     |                                                                                                  |
| bank                               | USInt         | 0     | 0     |                                                                                                  |
| reserved0                          | USInt         | 0     | 0     |                                                                                                  |
| writeLenBytes                      | UInt          | 0     | 4     |                                                                                                  |
| addressBytes                       | UDInt         | 0     | 0     |                                                                                                  |
| writeBytes                         | Array[0..5... |       |       |                                                                                                  |
| TagOpLockArgs                      | *IRP_MsgP...  |       |       |                                                                                                  |
| TagOpKillArgs                      | *IRP_MsgP...  |       |       |                                                                                                  |
| TagOpReadResp                      | *IRP_MsgP...  |       |       |                                                                                                  |
| HMIStatus                          | *HMI_Stat...  |       |       |                                                                                                  |
| ReaderCapabilitiesReaderIdStr      | String[16]    | ""    |       | 'K234500055'                                                                                     |
| ReaderCapabilitiesIcssFwVersionStr | String[16]    | ""    |       | '0.8.1'                                                                                          |
| ReaderCapabilitiesRfidFwVersionStr | String[16]    | ""    |       | '17.2-A'                                                                                         |
| ReaderCapabilitiesModelNameStr     | String[16]    | ""    |       | '1081-1A'                                                                                        |
| InventorySelectMaskDataHexStr      | String        | ""    |       | 2: Set the HEX string data to be written to TAG                                                  |
| TagOpSelectMaskDataHexStr          | String        | ""    |       |                                                                                                  |
| TagOpWriteDataHexStr               | String        | ""    |       | '31716132'                                                                                       |
| ReadRespHexStr                     | String        | ""    |       | '00000000000000000000402847523A7B3000'                                                           |
| SendReq                            | Int           | 0     | 27    |                                                                                                  |
| HMITagIndex                        | Int           | 0     | 0     | 3: Set the value IRP_MsgType_TagOpWriteReq DEC value If already set then simply toggle ReSendReq |
| eventListLen                       | Int           | 1     | 20    |                                                                                                  |
| SelectSpecificTAGIndex             | Int           | 0     | 1     |                                                                                                  |
| ReaderConfigEditNum                | USInt         | 0     | 1     |                                                                                                  |
| ReaderConfigEditNumPrev            | USInt         | 0     | 1     |                                                                                                  |
| InventorySelectMaskEditNum         | USInt         | 0     | 1     |                                                                                                  |
| InventorySelectMaskEditNumPrev     | USInt         | 0     | 1     |                                                                                                  |
| TAGReadWriteLen                    | USInt         | 0     | 0     |                                                                                                  |
| TagOPBankSelection                 | USInt         | 1     | 1     |                                                                                                  |
| SelectSpecificTAGEn                | Bool          | false | FALSE |                                                                                                  |
| UpdateTAGOpArgs                    | Bool          | false | FALSE |                                                                                                  |
| ReSendReq                          | Bool          | false | FALSE |                                                                                                  |

Figure 40. TIA Portal sample application - TAGOP Write password data.

Tag Operations: Lock (request and response).

|                                      |              |       |             |              |                    |
|--------------------------------------|--------------|-------|-------------|--------------|--------------------|
| ▼ TagOpLockArgs                      | *IRP_MsgP... |       |             |              |                    |
| ■ antennaMask                        | UDInt        | 0     | 0           |              | 1: TAG lock args   |
| ■ accessPasswd                       | UDInt        | 0     | 829_514_034 |              |                    |
| ■ mask                               | UInt         | 0     | 32          |              |                    |
| ■ action                             | UInt         | 0     | 32          |              |                    |
| ▶ TagOpKillArgs                      | *IRP_MsgP... |       |             |              |                    |
| ▶ TagOpReadResp                      | *IRP_MsgP... |       |             |              |                    |
| ▶ HMIStatus                          | *HMI_Stat... |       |             |              |                    |
| ■ ReaderCapabilitiesReaderIdStr      | String[16]   | "     | "           | 'K234500055' |                    |
| ■ ReaderCapabilitiesIcssFwVersionStr | String[16]   | "     | "           | '0.8.1'      |                    |
| ■ ReaderCapabilitiesRfidFwVersionStr | String[16]   | "     | "           | '17.2-A'     |                    |
| ■ ReaderCapabilitiesModelNameStr     | String[16]   | "     | "           | '1081-1A'    | convert HEX to DEC |
| ■ InventorySelectMaskDataHexStr      | String       | "     | "           | "            |                    |
| ■ TagOpSelectMaskDataHexStr          | String       | "     | "           | "            |                    |
| ■ TagOpWriteDataHexStr               | String       | "     | "           | '31716132'   |                    |
| ■ ReadRespHexStr                     | String       | "     | "           | '31716132'   |                    |
| ■ SendReq                            | Int          | 0     | 28          |              | 2: SendReq         |
| ■ HMITagIndex                        | Int          | 0     | 0           |              |                    |
| ■ eventListLen                       | Int          | 1     | 20          |              |                    |
| ■ SelectSpecificTAGIndex             | Int          | 0     | 1           |              |                    |
| ■ ReaderConfigEditNum                | USInt        | 0     | 1           |              |                    |
| ■ ReaderConfigEditNumPrev            | USInt        | 0     | 1           |              |                    |
| ■ InventorySelectMaskEditNum         | USInt        | 0     | 1           |              |                    |
| ■ InventorySelectMaskEditNumPrev     | USInt        | 0     | 1           |              |                    |
| ■ TAGReadWriteLen                    | USInt        | 0     | 0           |              |                    |
| ■ TagOPBankSelection                 | USInt        | 1     | 1           |              |                    |
| ■ SelectSpecificTAGEn                | Bool         | false | FALSE       |              |                    |
| ■ UpdateTAGOpArgs                    | Bool         | false | FALSE       |              |                    |
| ■ ReSendReq                          | Bool         | false | FALSE       |              |                    |

**Figure 41.** TIA Portal sample application - TAGOP Lock/Unlock (lock the tag).

|                                    |              |       |              |
|------------------------------------|--------------|-------|--------------|
| ▼ TagOpLockArgs                    | *IRP_MsgP... |       |              |
| antennaMask                        | UDInt        | 0     | 0            |
| accessPasswd                       | UDInt        | 0     | 829_514_034  |
| mask                               | UInt         | 0     | 32           |
| action                             | UInt         | 0     | 0            |
| ▶ TagOpKillArgs                    | *IRP_MsgP... |       |              |
| ▶ TagOpReadResp                    | *IRP_MsgP... |       |              |
| ▶ HMIStatus                        | *HMI_Stat... |       |              |
| ReaderCapabilitiesReaderIdStr      | String[16]   | "     | 'K234500055' |
| ReaderCapabilitiesIcssFwVersionStr | String[16]   | "     | '0.8.1'      |
| ReaderCapabilitiesRfidFwVersionStr | String[16]   | "     | '17.2-A'     |
| ReaderCapabilitiesModelNameStr     | String[16]   | "     | '1081-1A'    |
| InventorySelectMaskDataHexStr      | String       | "     | "            |
| TagOpSelectMaskDataHexStr          | String       | "     | "            |
| TagOpWriteDataHexStr               | String       | "     | '31716132'   |
| ReadRespHexStr                     | String       | "     | '31716132'   |
| SendReq                            | Int          | 0     | 28           |
| HMITagIndex                        | Int          | 0     | 0            |
| eventListLen                       | Int          | 1     | 20           |
| SelectSpecificTAGIndex             | Int          | 0     | 1            |
| ReaderConfigEditNum                | USInt        | 0     | 1            |
| ReaderConfigEditNumPrev            | USInt        | 0     | 1            |
| InventorySelectMaskEditNum         | USInt        | 0     | 1            |
| InventorySelectMaskEditNumPrev     | USInt        | 0     | 1            |
| TAGReadWriteLen                    | USInt        | 0     | 0            |
| TagOPBankSelection                 | USInt        | 1     | 1            |
| SelectSpecificTAGEn                | Bool         | false | FALSE        |
| UpdateTAGOpArgs                    | Bool         | false | FALSE        |
| ReSendReq                          | Bool         | false | FALSE        |

**Figure 42.** TIA Portal sample application - TAGOP Lock/Unlock (unlock the tag).



```

18252872 Request changed! recvid=0xf
18252872 _IRP_autoGen_validate_IRP_MsgHeader(0xa04241ec,1):
18252872 type->28
18252872 id->45946
18252872 _IRP_autoGen_validate_IRP_MsgPayload_TagOpLockReq(0xa04241f0,1):
18252872 antennaMask->0x0
18252872 accessPasswd->0x31716132
18252872 mask->32
18252872 action->32
18252872 received from PN, type 0x1c (TagOpLockReq)
18252872 BusyLock acquired for TagOpLockReq
18252923 Antennas switched to 0x0
18252984 BusyLock freed by TagOpLockReq
18252984 sending to PN id 45946, type 0x40 (ErrorCodeResp)
18252984 _IRP_autoGen_validate_IRP_MsgHeader(0xa0425e95,2):
18252984 type->64
18252984 id->45946
18252984 _IRP_autoGen_validate_IRP_MsgPayload_ErrorCode(0xa0425e99,2):
18252984 errorCode->0
18252984 _IRP_tpPN_send() send queued, waiting: 1
18252984 LED_Set(data, green)
18252989 LED_Set(data, off)

```

**Figure 43.** TIA Portal sample application - TAGOP Lock/Unlock IRX200 logs during tag lock.

```

18349929 Request changed! recvid=0x10
18349929 _IRP_autoGen_validate_IRP_MsgHeader(0xa04241ec,1):
18349929 type->28
18349929 id->2083
18349929 _IRP_autoGen_validate_IRP_MsgPayload_TagOpLockReq(0xa04241f0,1):
18349929 antennaMask->0x0
18349929 accessPasswd->0x31716132
18349929 mask->32
18349929 action->0
18349929 received from PN, type 0x1c (TagOpLockReq)
18349929 BusyLock acquired for TagOpLockReq
18349980 Antennas switched to 0x0
18350041 BusyLock freed by TagOpLockReq
18350041 sending to PN id 2083, type 0x40 (ErrorCodeResp)
18350041 _IRP_autoGen_validate_IRP_MsgHeader(0xa0425e95,2):
18350041 type->64
18350041 id->2083
18350041 _IRP_autoGen_validate_IRP_MsgPayload_ErrorCode(0xa0425e99,2):
18350041 errorCode->0
18350041 _IRP_tpPN_send() send queued, waiting: 1
18350041 LED_Set(data, green)
18350046 LED_Set(data, off)

```

**Figure 44.** TIA Portal sample application - TAGOP Lock/Unlock IRX200 logs during tag unlock.

|             |        |   |                                                                               |
|-------------|--------|---|-------------------------------------------------------------------------------|
| AE_view[21] | String | " | IRX200-1: Req. Type: TagOpLockReq returned with success                       |
| AE_view[22] | String | " | 'IRX200-1: Req. Type: TagOpLockReq sent successfully with MsgID:+45946'       |
| AE_view[23] | String | " | 'IRX200-1: Req. Type: TagOpLockReq returned with Errorcode response: Success' |
| AE_view[24] | String | " | 'IRX200-1: Req. Type: TagOpLockReq sent successfully with MsgID:+2083'        |
| AE_view[25] | String | " | 'IRX200-1: Req. Type: TagOpLockReq returned with Errorcode response: Success' |

**Figure 45.** TIA Portal sample app - TAGOP Lock/Unlock PLC logs.



Stop All (request and response).

|                                    |             |   |    |              |
|------------------------------------|-------------|---|----|--------------|
| HMStatus                           | "HM_Status" |   |    |              |
| ReaderCapabilitiesReaderIdStr      | String[16]  | " | "  | 'K234500055' |
| ReaderCapabilitiesIcssFwVersionStr | String[16]  | " | "  | '0.8.1'      |
| ReaderCapabilitiesRfidFwVersionStr | String[16]  | " | "  | '17.2-A'     |
| ReaderCapabilitiesModelNameStr     | String[16]  | " | "  | '1081-1A'    |
| InventorySelectMaskDataHexStr      | String      | " | "  | "            |
| TagOpSelectMaskDataHexStr          | String      | " | "  | "            |
| TagOpWriteDataHexStr               | String      | " | "  | "            |
| ReadRespHexStr                     | String      | " | "  | "            |
| SendReq                            | Int         | 0 | 30 |              |
| HMTAGIndex                         | Int         | 0 | 0  |              |
| eventListLen                       | Int         | 1 | 20 |              |
| SelectSpecificTAGIndex             | Int         | 0 | 0  |              |

**Figure 46.** TIA Portal sample application - StopAll request.

Alarm and Events.

In addition to IRP Msg alarms and events, the IO errors and data mapping errors are also logged to the data block.

| 803_AlarmandEvents |             |           |             |                                                       |
|--------------------|-------------|-----------|-------------|-------------------------------------------------------|
|                    | Name        | Data type | Start value | Monitor value                                         |
| 16                 | AE_view[13] | String    | "           | 'IRX200-1: Input:AckSendId GETIO Error!'              |
| 17                 | AE_view[14] | String    | "           | 'IRX200-1: Input:recvID GETIO Error!'                 |
| 18                 | AE_view[15] | String    | "           | 'IRX200-1: Input:ReadMsg Header + Data GETIO Error!'  |
| 19                 | AE_view[16] | String    | "           | 'IRX200-1: Output:AckRecvID SETIO Error!'             |
| 20                 | AE_view[17] | String    | "           | 'IRX200-1: Output:SendMsg Header + Data SETIO Error!' |
| 21                 | AE_view[18] | String    | "           | 'IRX200-1: Output:SendID SETIO Error!'                |
| 22                 | AE_view[19] | String    | "           | 'IRX200-1: Input:Status GETIO Error!'                 |
| 23                 | AE_view[20] | String    | "           | 'IRX200-1: Input:AckSendId GETIO Error!'              |
| 24                 | AE_view[21] | String    | "           | 'IRX200-1: Input:recvID GETIO Error!'                 |
| 25                 | AE_view[22] | String    | "           | 'IRX200-1: Input:ReadMsg Header + Data GETIO Error!'  |
| 26                 | AE_view[23] | String    | "           | 'IRX200-1: Output:AckRecvID SETIO Error!'             |
| 27                 | AE_view[24] | String    | "           | 'IRX200-1: Output:SendMsg Header + Data SETIO Error!' |
| 28                 | AE_view[25] | String    | "           | 'IRX200-1: Output:SendID SETIO Error!'                |

**Figure 47.** TIA Portal sample application - IO mapping errors (for example, if communication is lost).

## 5. SETTING UP THE PLC DEVELOPMENT ENVIRONMENT - CODESYS

This section will guide you on how to establish a communication between the PLC and IRX200 reader using CODESYS development environment and runtime.

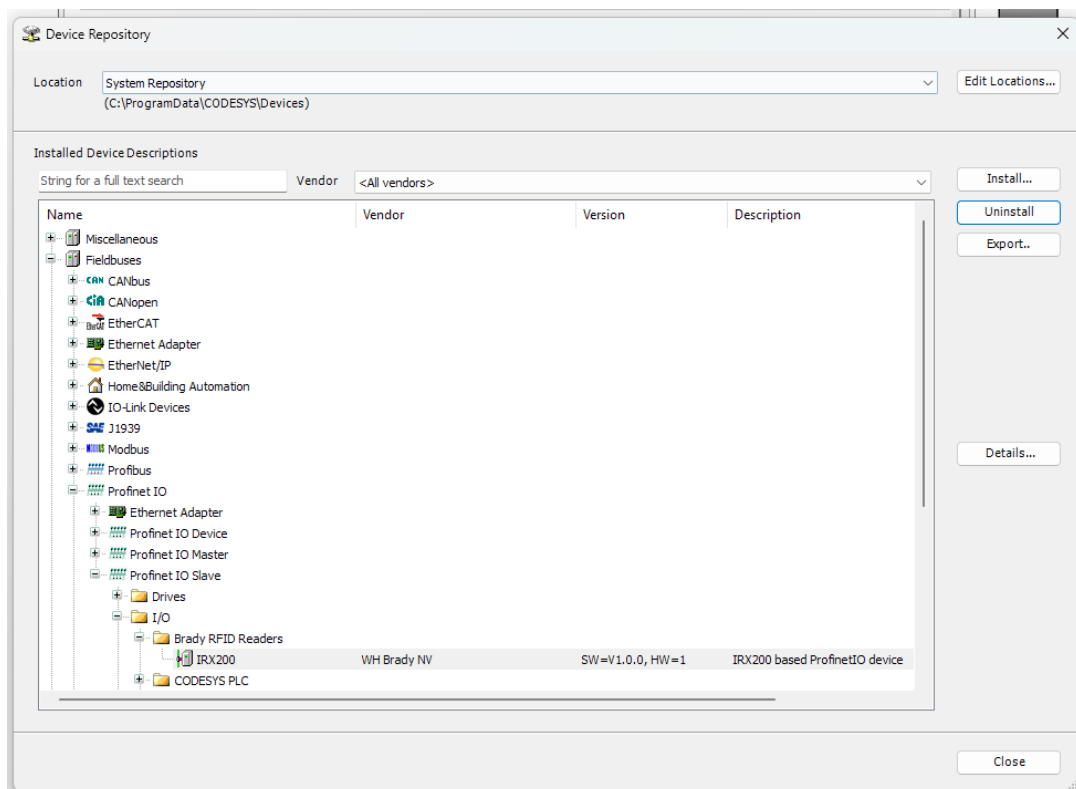
### 5.1. CODESYS VERSION

For new CODESYS projects utilizing the Brady IRX200 reader, any CODESYS version that can accept the provided GSDML file should be compatible.

For the sample application install **CODESYS Development System v3.5.18.30** (shown as “CODESYS V3.5 SP18 Patch 3” in “Help” / “About..”) **or newer**. If a newer version of CODESYS is used, opening the sample project may prompt to update project dependencies to match the installed versions.

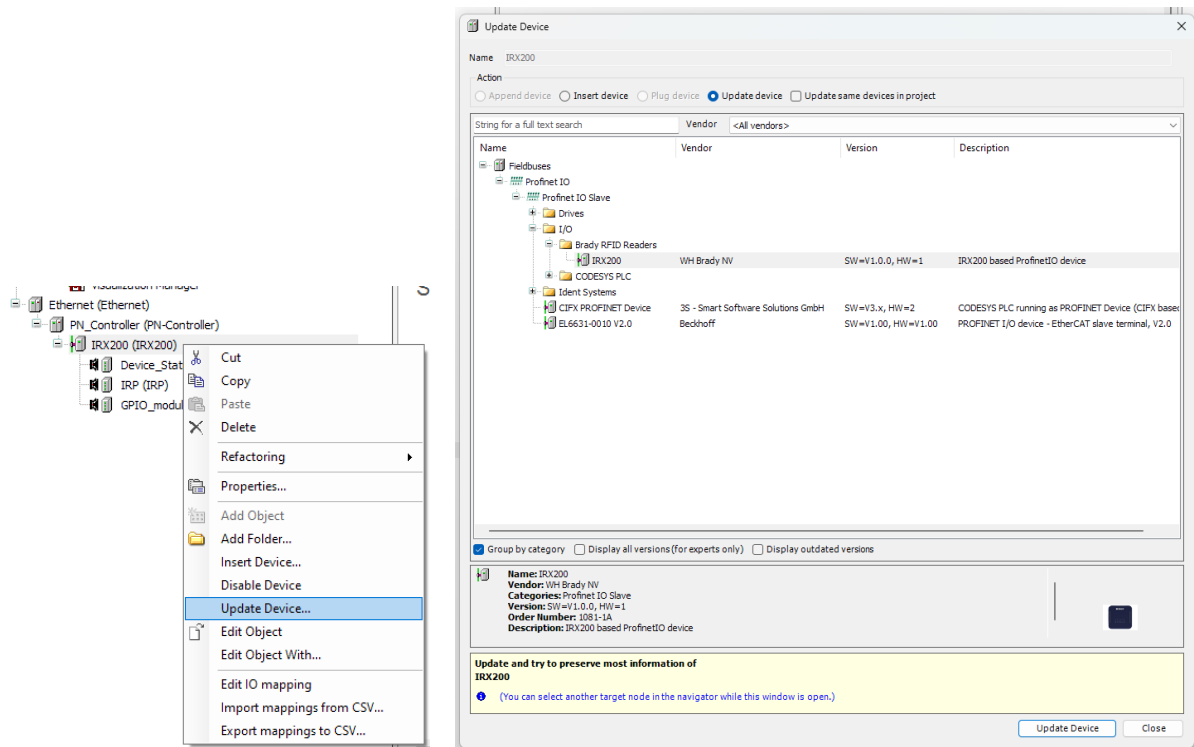
### 5.2. INSTALLING THE DEVICE DESCRIPTOR / GSDML

Under “Tools” / “Device Repository...” click “Install...” and locate **GSDML-V2.43-Brady-IRX200-20240401.xml**. The installed device will appear in the device tree under “**Fieldbuses / Profinet IO / Profinet IO Slave / I/O / Brady RFID Readers**” as “IRX200”. Click “Close”.



**Figure 48.** CODESYS – installing the GSDML file.

If the installed GSDML is to be updated into pre-existing CODESYS projects with IRX200 devices, right click on the device under “Devices” listing, go to “Update Device...” and select the same device path as installed above. Click “Update Device”. Repeat this process for all connected submodules as well (“Device\_Status”, “IRP”, “GPIO\_module” if enabled). This step is not necessary when using the sample project with its included matching GSDML version.



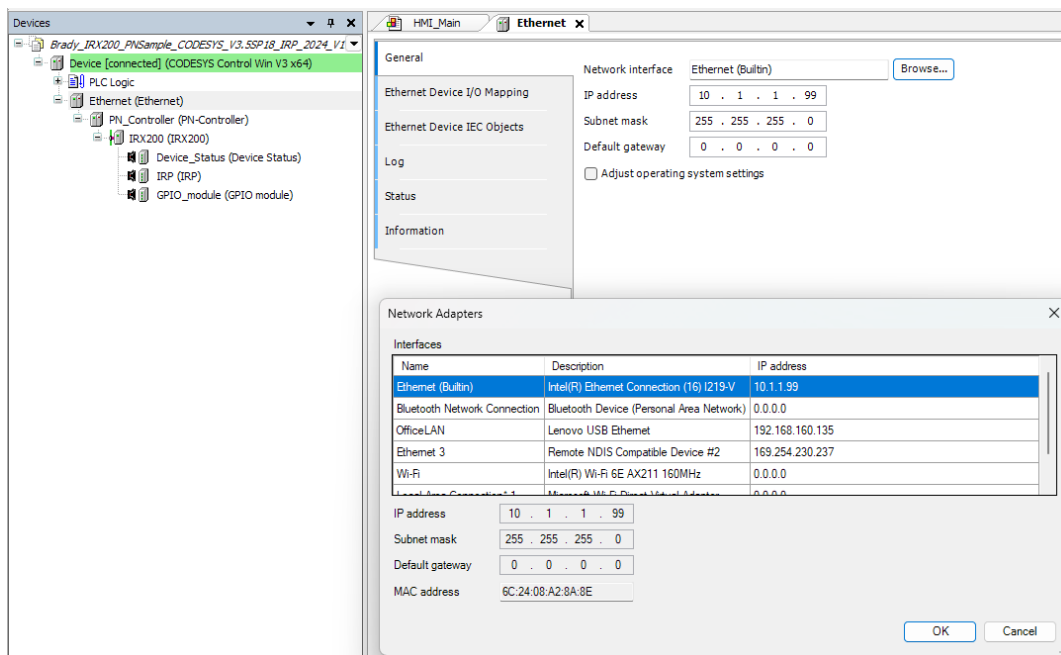
**Figure 49.** CODESYS – updating a device.

## 5.3. RUNNING THE SAMPLE PLC APPLICATION

### 5.3.1. IMPORTING THE SAMPLE PROJECT

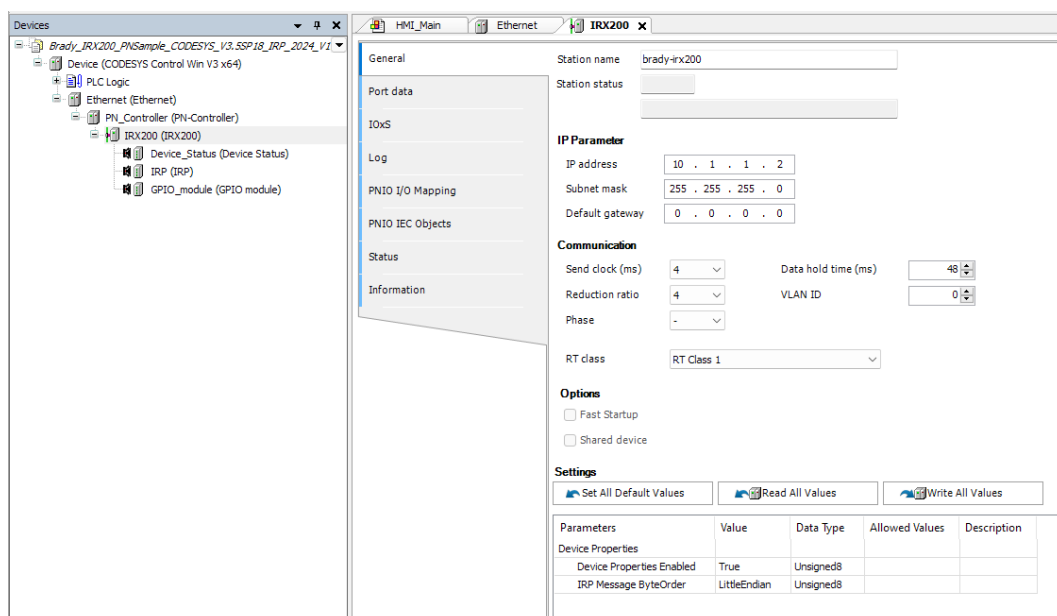
Under “File” / “Open Project..” locate the provided sample project and click “Open”.

Set the required network settings by double clicking “Devices” / “Ethernet (Ethernet)” and clicking “Browse..” under “General” / “Network interface”.



**Figure 50.** CODESYS – selecting the network adapter.

Select the desired Network adapter and click “OK”. Change the adapter’s IP settings if necessary. Double click the “IRX200 (IRX200)” device to verify its IP settings and Station name are correct.



**Figure 51.** CODESYS – adapter’s network settings.

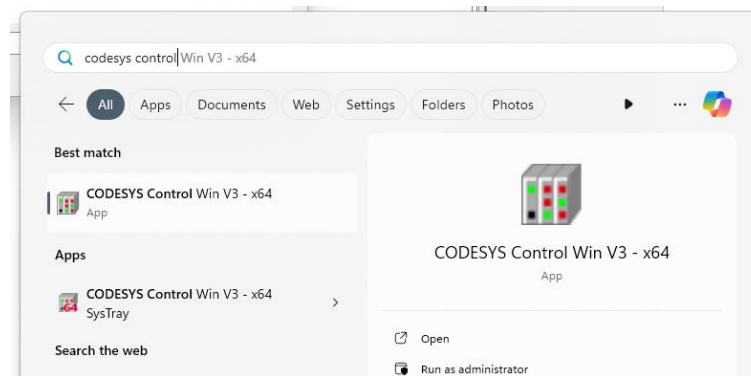
If the reader’s station name has been changed from the default (“brady-irx200”), the matching name must be updated to “Devices” / “IRX200” / “General” / “Station name”.

If the network adapter's IP settings were changed from defaults above, also update the reader's IP settings to the new network.

### 5.3.2. STARTING AND CONNECTING TO THE SOFTWARE PLC

The sample project uses CODESYS Control Win V3 software PLC which comes bundled with the CODESYS Development System installation.

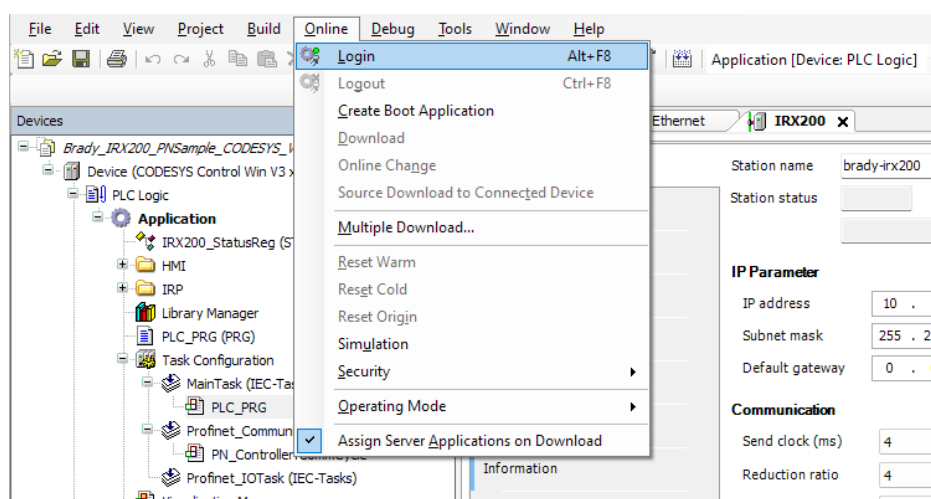
The installer provides a terminal and system tray option for starting the software PLC, we'll use the terminal launcher (upper option in the screenshot below) as it immediately displays useful diagnostic information.



**Figure 52.** CODESYS – selecting the CODESYS control.

Start “CODESYS Control Win V3 – x64 (App)”.

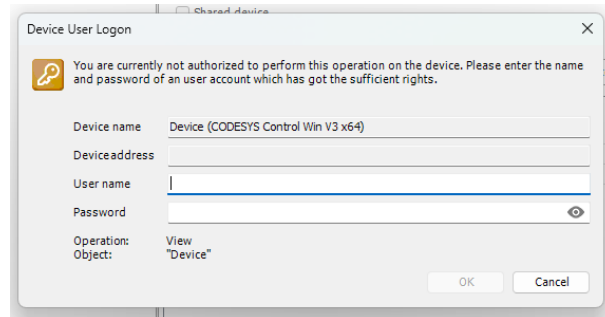
Proceed to login to the PLC within CODESYS by selecting “Online” / “Login”.



**Figure 53.** CODESYS – Login to PLC.

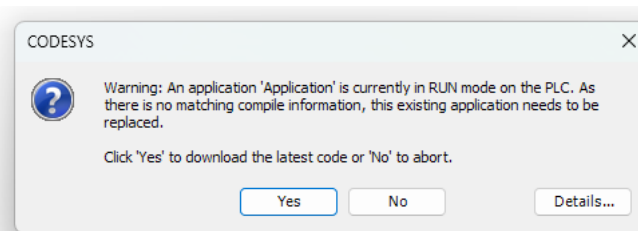
If connecting to the PLC fails at this stage, see “6.1. Software PLC not responding”.

On first login after installation, the PLC will prompt to set a username/password to access the PLC. On all successive logins, enter the credentials set earlier when prompted.



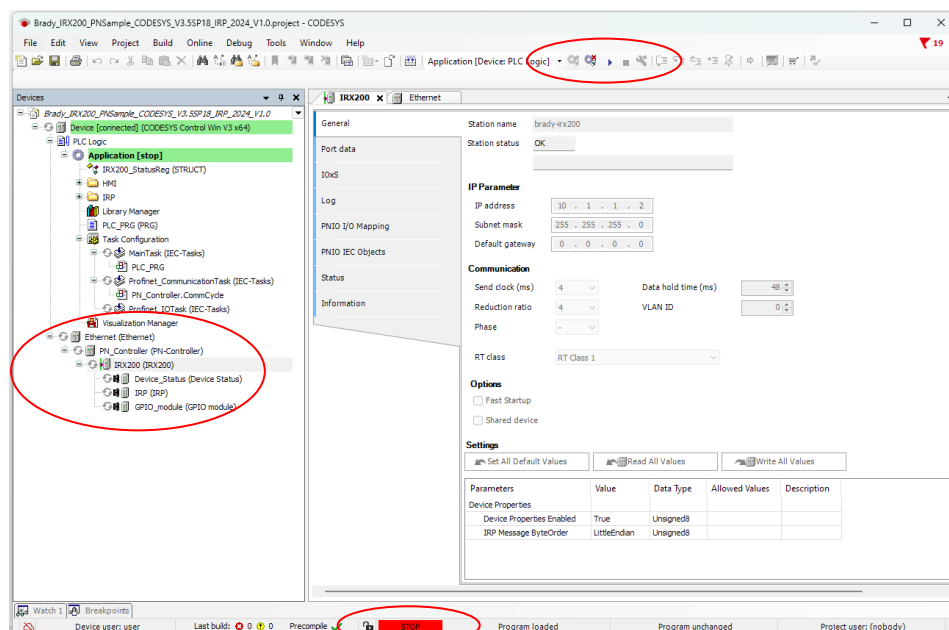
**Figure 54.** CODESYS – CODESYS Control login.

If the following RUN dialog is presented, click “Yes” to restart the PLC.



**Figure 55.** CODESYS – possible warning dialog.

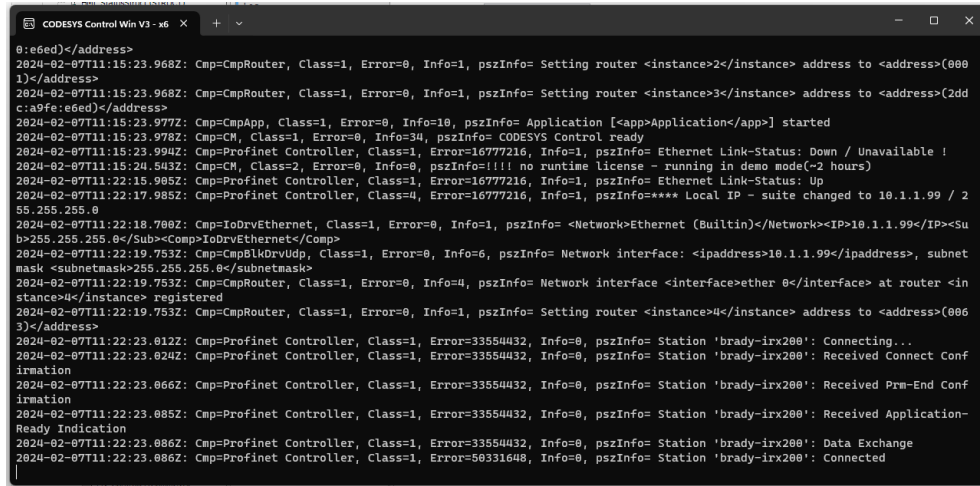
The application is now logged in but stopped.



**Figure 56.** CODESYS – application logged in but stopped.



Start the sample application from “Debug” / “Start”.



```

0:e6ed)</address>
2024-02-07T11:15:23.968Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>2</instance> address to <address>(000
1)</address>
2024-02-07T11:15:23.968Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>3</instance> address to <address>(2dd
c:a9fe:e6ed)</address>
2024-02-07T11:15:23.977Z: Cmp=CmpApp, Class=1, Error=0, Info=10, pszInfo= Application [<app>Application</app>] started
2024-02-07T11:15:23.978Z: Cmp=CM, Class=1, Error=0, Info=34, pszInfo= CODESYS Control ready
2024-02-07T11:15:24.543Z: Cmp=CM, Class=2, Error=0, Info=0, pszInfo=!!!! no runtime license - running in demo mode(-2 hours)
2024-02-07T11:22:15.905Z: Cmp=Profinet Controller, Class=1, Error=16777216, Info=1, pszInfo= Ethernet Link-Status: Up
2024-02-07T11:22:17.985Z: Cmp=Profinet Controller, Class=4, Error=16777216, Info=1, pszInfo=**** Local IP - suite changed to 10.1.1.99 / 2
55.255.255.0
2024-02-07T11:22:18.708Z: Cmp=IoDrvEthernet, Class=1, Error=0, Info=1, pszInfo= <Network>Ethernet (Builtin)</Network><IP>10.1.1.99</IP><Su
b>255.255.255.0</Sub><Cmp>IoDrvEthernet</Cmp>
2024-02-07T11:22:19.753Z: Cmp=CmpBldrvUdp, Class=1, Error=0, Info=6, pszInfo= Network interface: <ipaddress>10.1.1.99</ipaddress>, subnet
mask <subnetmask>255.255.255.0</subnetmask>
2024-02-07T11:22:19.753Z: Cmp=CmpRouter, Class=1, Error=0, Info=4, pszInfo= Network interface <interface>ether 0</interface> at router <in
stance>4</instance> registered
2024-02-07T11:22:19.753Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>4</instance> address to <address>(006
3)</address>
2024-02-07T11:22:23.012Z: Cmp=Profinet Controller, Class=1, Error=33554432, Info=0, pszInfo= Station 'brady-irx200': Connecting...
2024-02-07T11:22:23.024Z: Cmp=Profinet Controller, Class=1, Error=33554432, Info=0, pszInfo= Station 'brady-irx200': Received Connect Conf
irmation
2024-02-07T11:22:23.066Z: Cmp=Profinet Controller, Class=1, Error=33554432, Info=0, pszInfo= Station 'brady-irx200': Received Prm-End Conf
irmation
2024-02-07T11:22:23.085Z: Cmp=Profinet Controller, Class=1, Error=33554432, Info=0, pszInfo= Station 'brady-irx200': Received Application-
Ready Indication
2024-02-07T11:22:23.086Z: Cmp=Profinet Controller, Class=1, Error=33554432, Info=0, pszInfo= Station 'brady-irx200': Data Exchange
2024-02-07T11:22:23.086Z: Cmp=Profinet Controller, Class=1, Error=50331648, Info=0, pszInfo= Station 'brady-irx200': Connected

```

Figure 57. CODESYS – runtime logs.

The reader should appear connected in the PLC log. If not, refer to section “6. Troubleshooting”.

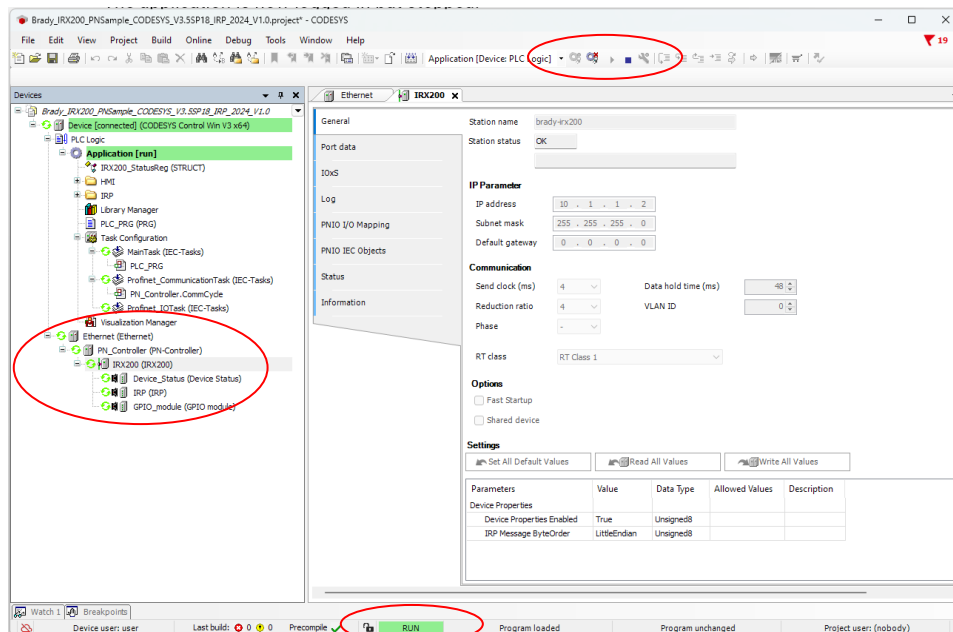
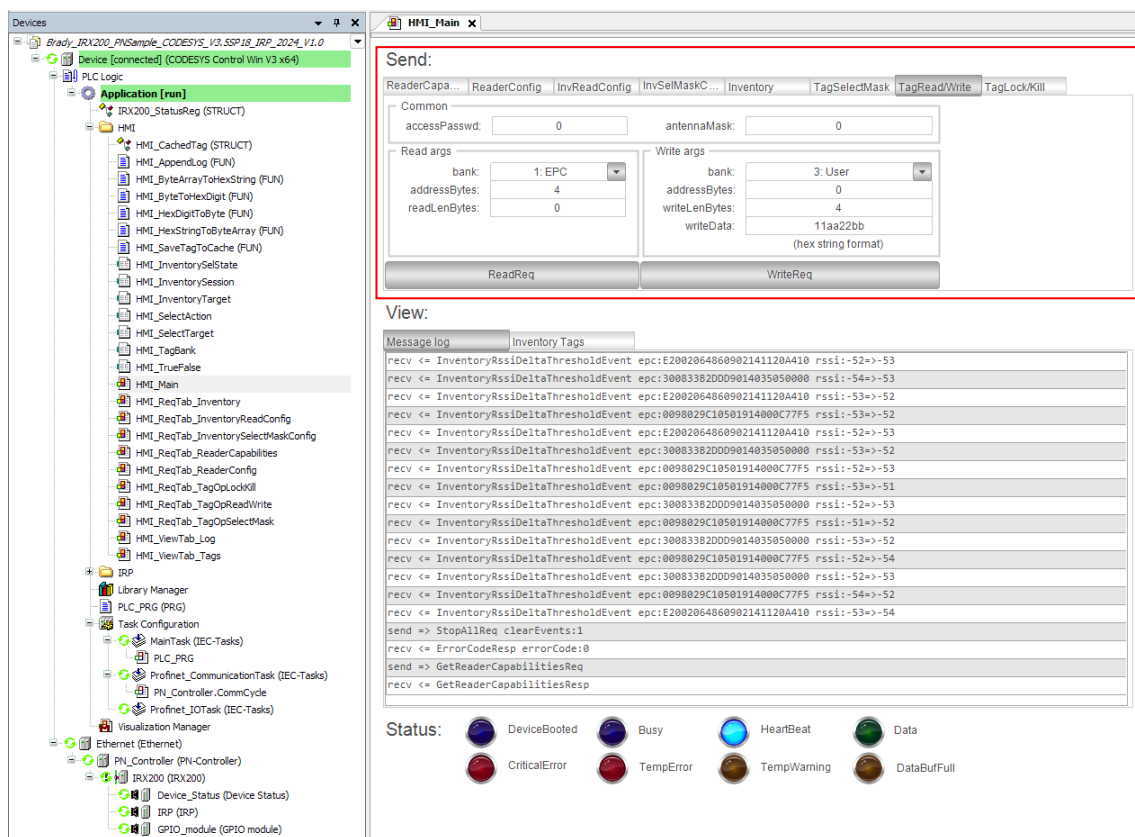


Figure 58. CODESYS – application logged in and running.

The application is now logged in and running.

### 5.3.3. HUMAN MACHINE INTERFACE

The sample application's main user interface can be found under "Devices" / "Application" / "HMI" / "HMI\_Main"



**Figure 59.** CODESYS – HMI main view top section.

The "Send" section of the HMI is for generating requests to the device. In the example above, "accessPasswd" and "antennaMask" apply to both Read and Write requests whereas the arguments specific to ReadReq are listed within "Read args" group.

See section "8. IRP – Industrial reader protocol description" for a full listing of supported request types along with their associated parameters and responses.

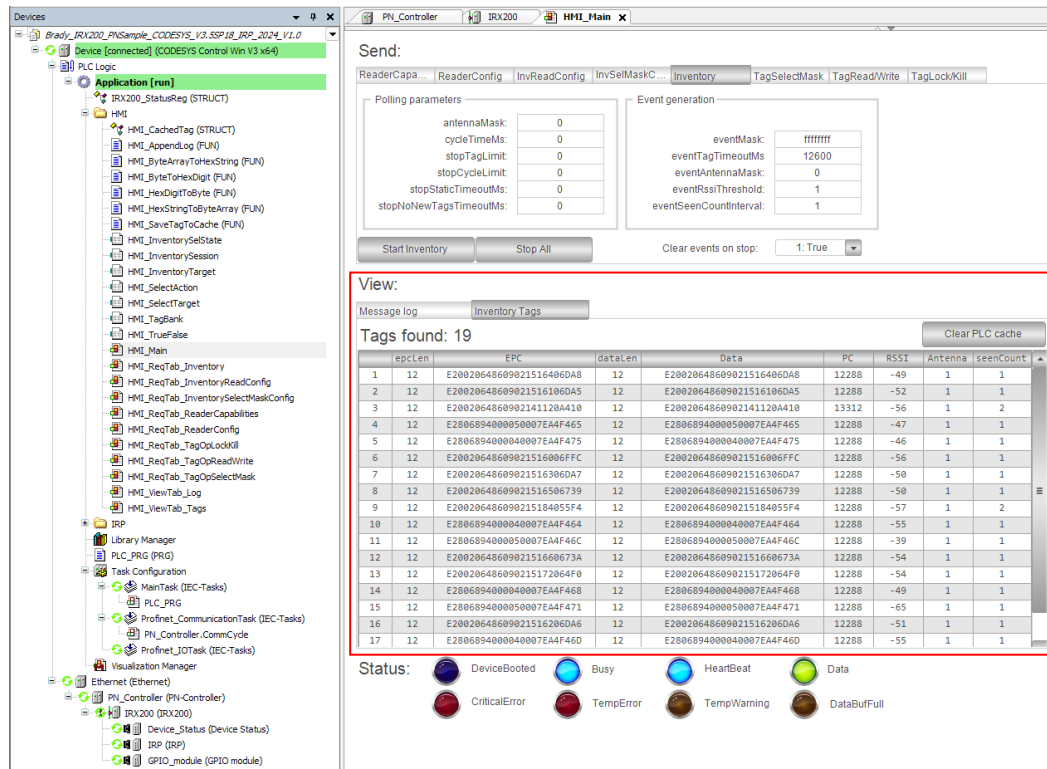


Figure 60. CODESYS – HMI main view middle section.

The “View” section of the HMI displays information on ongoing activity with the sample application. In the example above, the “Inventory Tags” section is getting populated during an ongoing Inventory request. The PLC sample program has an internal tag cache of 64 tags.

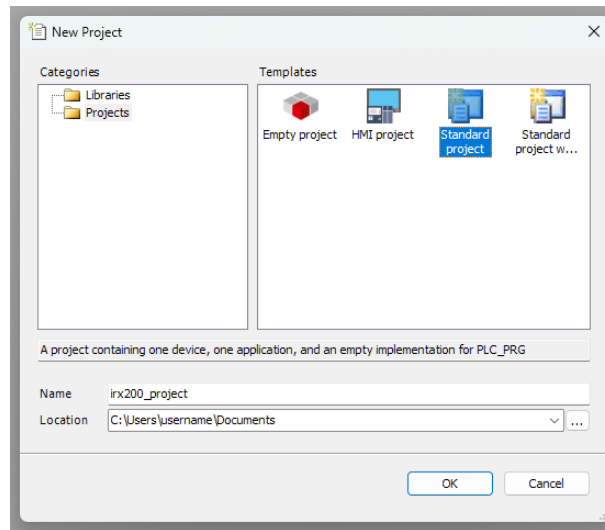
The “Message log” section displays a brief overview on sent and received messages. For more detailed logs see “6.4. Logging” paragraph.

Below the “View” section, the “Status” section displays the current status of the device as read from the Device\_Status submodule. See section “7.1. Device Status submodule” for the definition of each status LED/bit.

## 5.4. CREATING A NEW CODESYS PROJECT FOR THE IRX200

In CODESYS, select “File” / “New Project...”

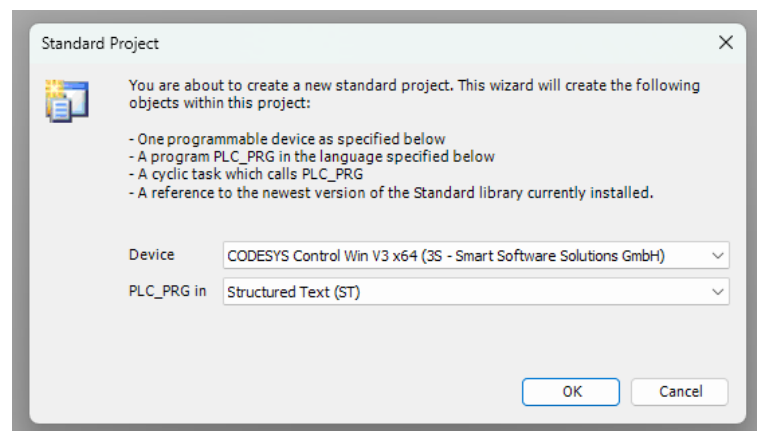
Under “Devices” tree, double click the PLC device, in this example “Device (CODESYS Control Win V3 x64)”, and assign the PLC hostname under “Communication Settings” from the dropdown list on the right.



**Figure 61.** CODESYS – creating a new project.

Select “Standard project”, assign a name and click OK.

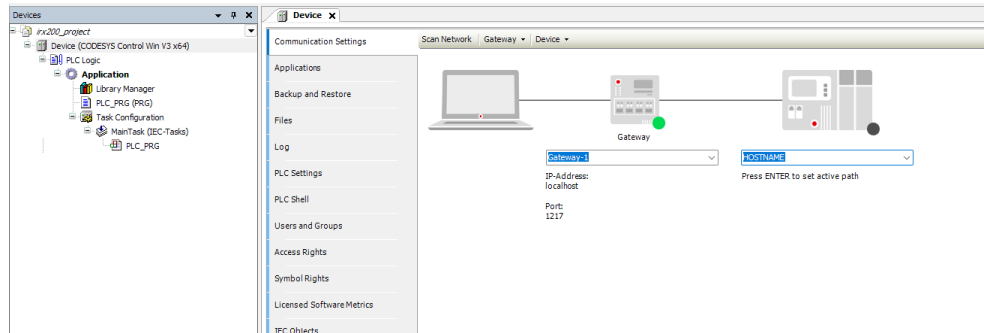
CODESYS will prompt for the PLC type and programming language.



**Figure 62.** CODESYS – PLC type and programming language.

In this example we’ll use CODESYS Control software PLC and Structured Text programming language. Click OK.

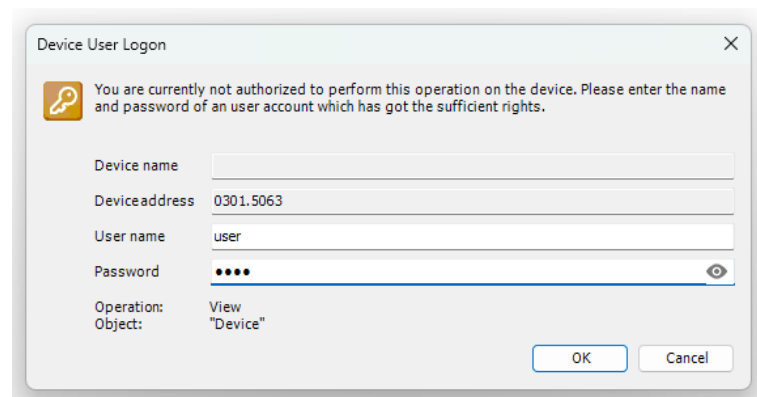
A new standard project has now been created. To continue setting up the project, a connection to the PLC needs to be established. Ensure the PLC is running as described in the **“Running the sample PLC application”** paragraph.



**Figure 63.** CODESYS – devices view.

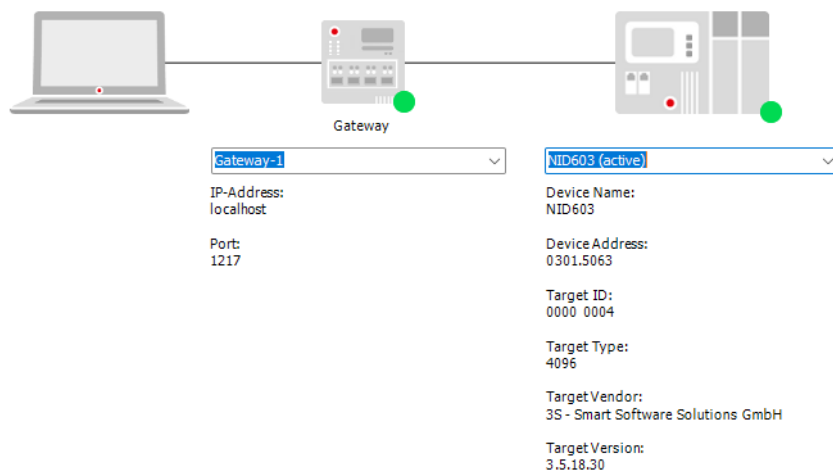
For a local software PLC the default value is likely correct, in which case clicking the PLC hostname and pressing Enter will login to the PLC. If the connection to the PLC is not working, refer to the “**6. Troubleshooting**” paragraph.

When logging into the PLC, a prompt for credentials is displayed. On the very first run, a prompt to create login credentials is displayed instead.



**Figure 64.** CODESYS – device user logon dialog.

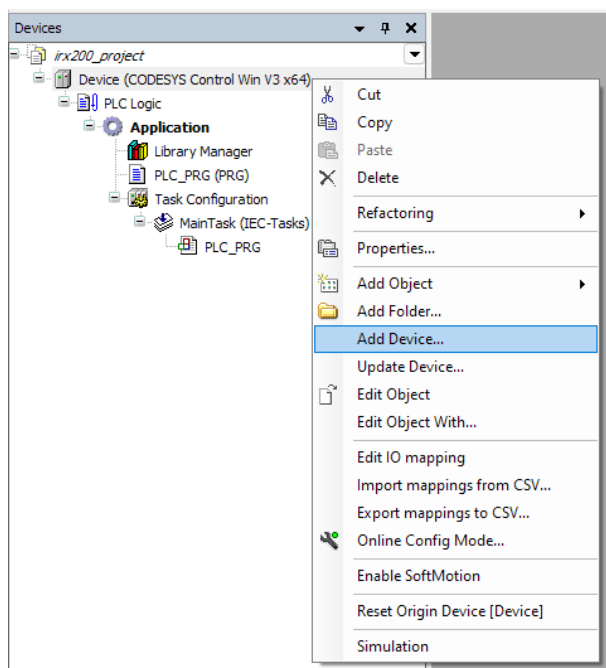
Enter the PLC credentials and click “OK”.



**Figure 65.** CODESYS – device view.

The connection to the PLC is now established.

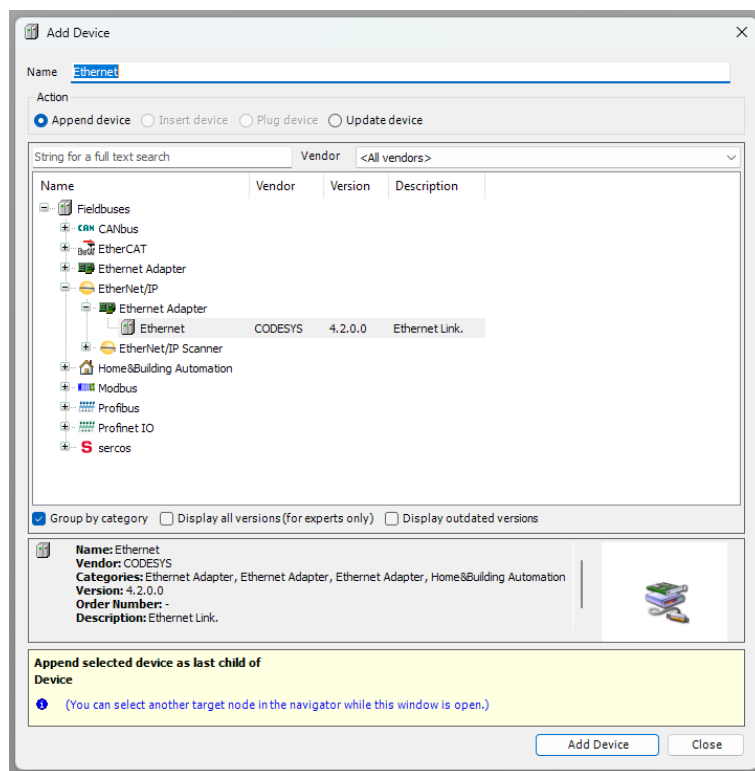
Right click the PLC device, in this example “Device (CODESYS Control Win V3 x64)”, and select “Add Device...”:



**Figure 66.** CODESYS – add device to CODESYS control.

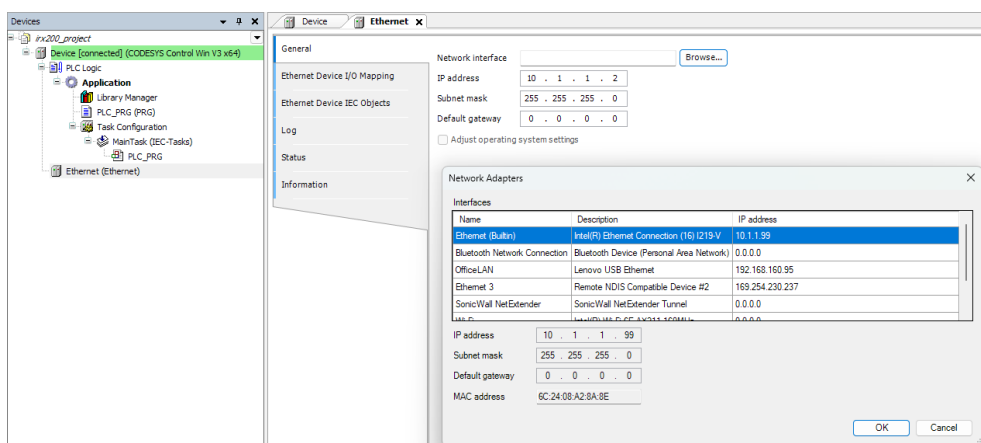
Select “**Fieldbuses / EtherNet/IP / Ethernet Adapter / Ethernet**” and click “Add Device”. Click “Close”.





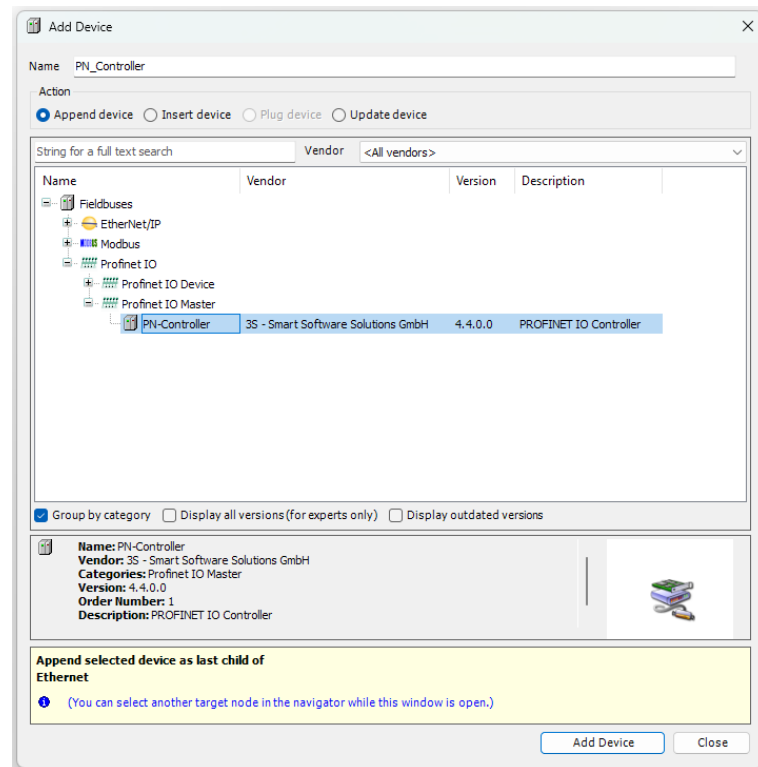
**Figure 67.** CODESYS – adding an ethernet adapter.

Double click the now added “Ethernet (Ethernet)” device and assign the desired IP address and network interface.



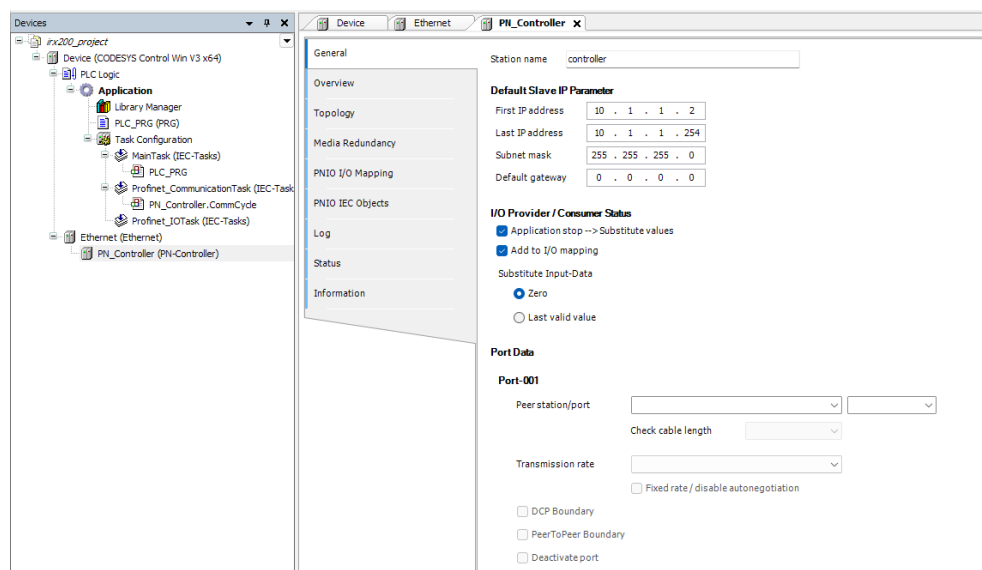
**Figure 68.** CODESYS – network adapter settings.

Right click “Ethernet (Ethernet)” device and select “Add Device...”. Select **Fieldbuses / Profinet IO / Profinet IO Master / PN-Controller** and click “Add Device”. Click “Close”.



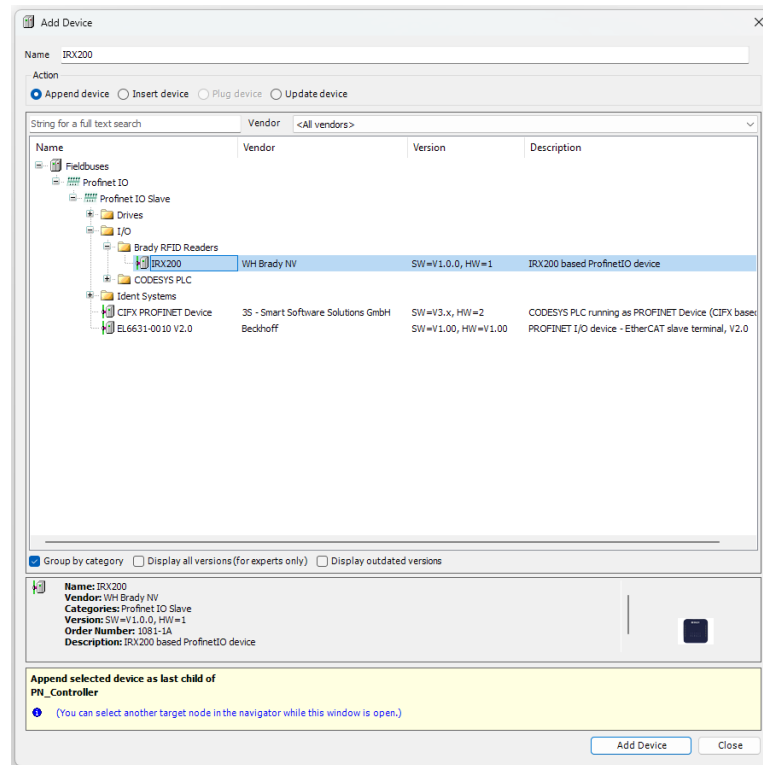
**Figure 69.** CODESYS – adding a PN controller.

Double click the now added “PN\_Controller (PN-Controller)” device to assign the matching IP range for the ProfiNet network.



**Figure 70.** CODESYS – PN controller network settings.

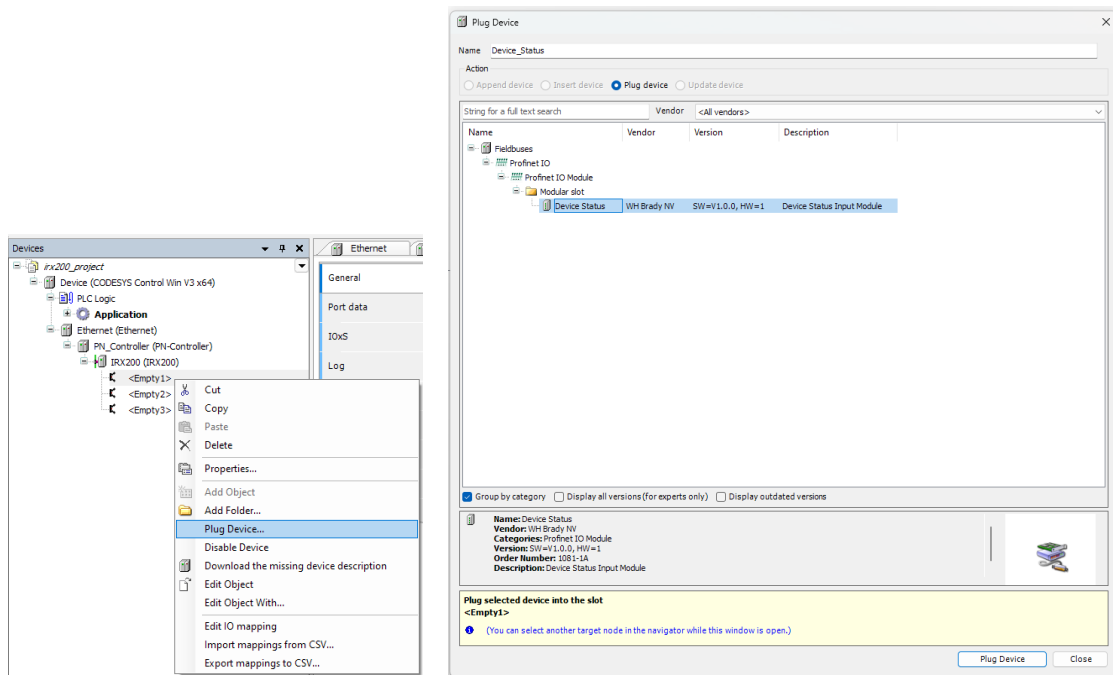
Right click the “PN\_Controller (PN-Controller)” device and click “Add Device”. Select **“Fieldbuses / Profinet IO / Profinet IO Slave / I/O / Brady RFID Readers / IRX200”** and click “Add Device”. Click “Close”.



**Figure 71.** CODESYS – adding a slave device to PN controller.

Once the device has been added, a new entry named “IRX200 (IRX200)” can be found under “PN\_Controller (PN-Controller)” in the Devices tree. Proceed to right click each “<Empty#>” subslot under “IRX200 (IRX200)” in the Devices tree and selecting “Plug Device...”.

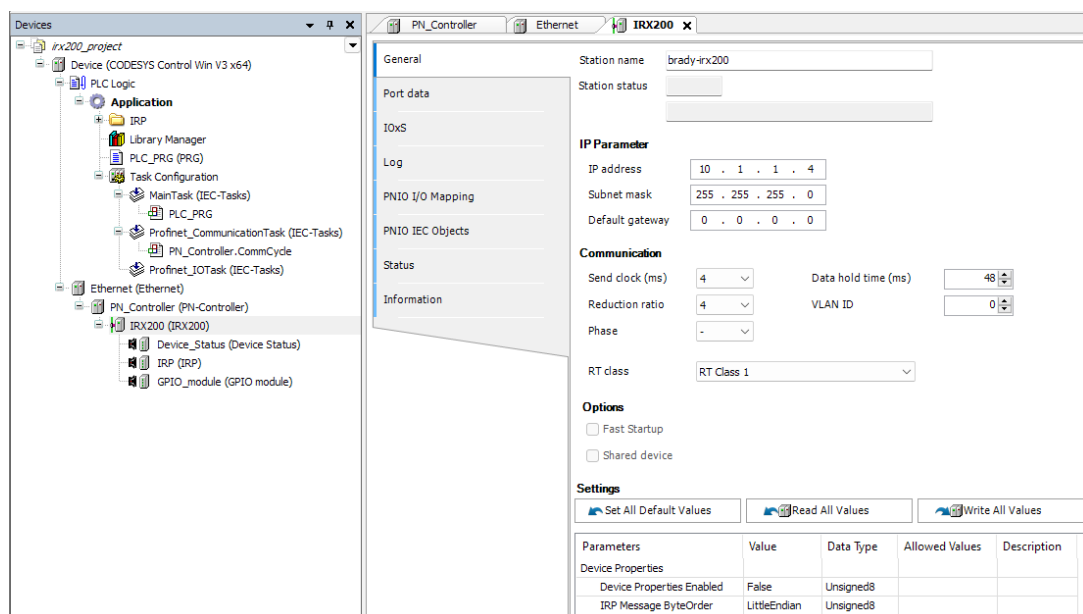
Only one option is provided for each subslot within the following “Add Device” dialog. Select the provided option (“Device\_Status” in the screenshot below) and click “Add Device”. Repeat the process for the remaining subslots desired. The first and second subslots (“Device\_Status” and “IRP”) are required for RFID operations. The third subslot provides “GPIO”, which is optional and may be left empty if GPIO access to the device is not required.



**Figure 72.** CODESYS – Plug Device to subslot.

Once the desired subslots have been added, double click the “IRX200 (IRX200)” device to verify device settings such as station name, IP address and ProfiNet communication settings for the reader. The default station name “brady-irx200” will be correct for readers with factory configuration, but if this has been changed on the reader, the CODESYS project must be updated with the matching name.

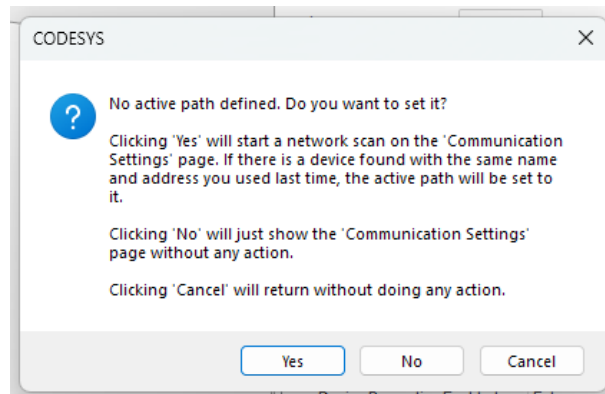
If a software PLC is used as described in this document, the default send clock of 1ms with reduction ratio 1 is likely too fast for the use case. Send clock: 4ms and Reduction ratio: 4 have been found to work reliably in our testing with a software PLC on a regular PC.



**Figure 72.** CODESYS – IRX200 device properties.

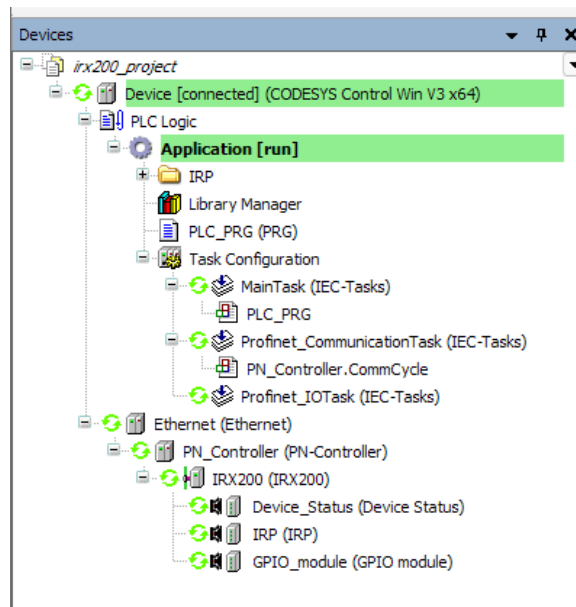
If the project uses the “CODESYS Control Win V3 x64” software PLC, there’s no need to change IRP Message ByteOrder in the Device Properties section of IRX200 configuration. If a BigEndian PLC is used instead, set “Device Properties Enabled” to “True” and “IRP Message ByteOrder” to “BigEndian”.

Now let’s test communication with the reader by selecting “Online” / “Login (F5)”. Enter the PLC credentials when prompted. On first login the following prompt will be displayed.



**Figure 73.** CODESYS – first login prompt.

Select “Yes”. The device should now appear online. If not, refer to the “**6. Troubleshooting**” paragraph.

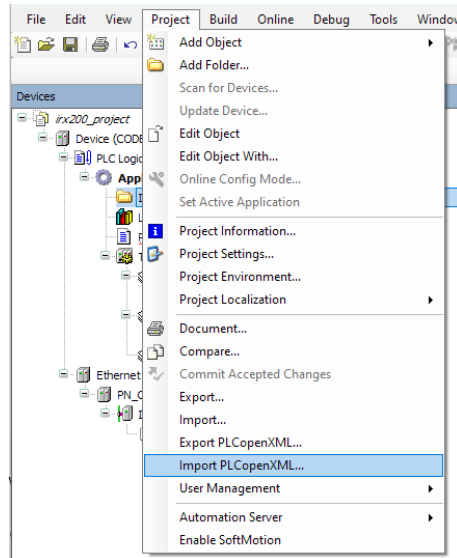


**Figure 74.** CODESYS – devices tree view.

If the communication test was successful, we can now logout via “Online” / “Logout (F8)” menu and proceed to create an initial ST program to run on the PLC.

### 5.4.1. IMPORTING PLCOPENXML

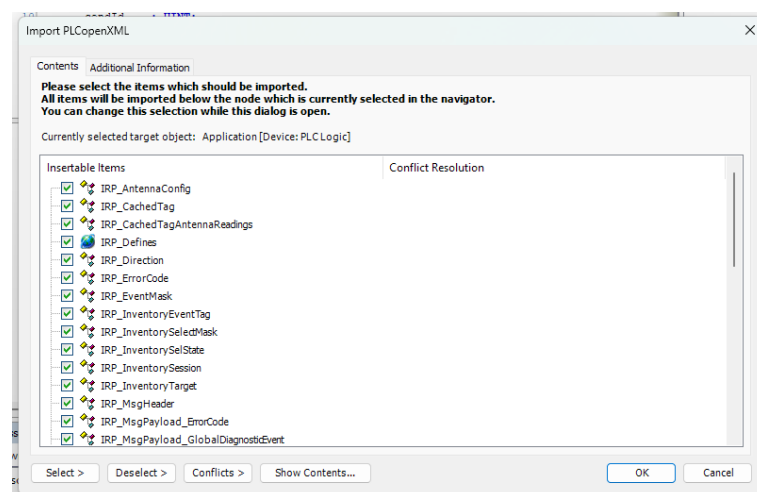
Importing relevant enums and data structures will aid with reading and writing messages. Right click “Application” within the “Devices” tree, select “Add Folder...” and create a folder named “IRP”. Select the newly created folder and proceed to “Project” / “Import PLCOpenXML...”



**Figure 75.** CODESYS – importing the PLCOpenXML.

Browse to select IRP\_PLCOpenXML.xml and click “Open”.

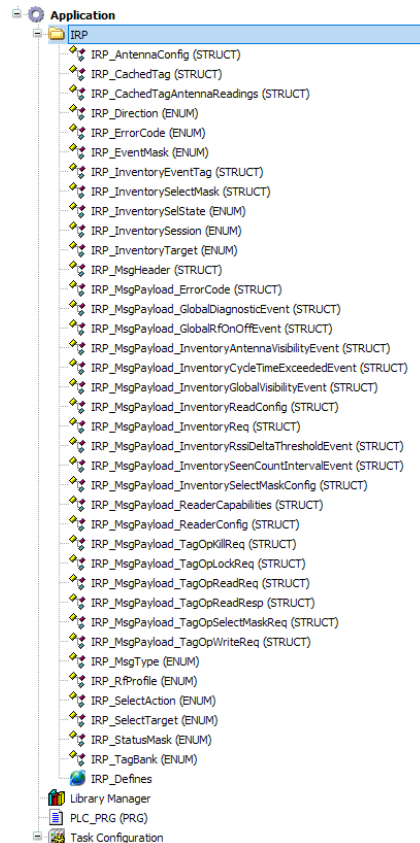
If a different entry was selected in the Devices tree during the import, an error dialog may appear with the message **“There are no objects in the export file which can be imported at the currently selected location”**. In that case, re-import with the newly created “IRP” folder selected.



**Figure 76.** CODESYS – Import PLCOpenXML selection.



CODESYS will prompt to ask which sections from the PLCopenXML file are imported. As everything is selected by default, just click “OK” to import everything. Relevant data structures are now visible under the IRP folder.



**Figure 77.** CODESYS – imported IRP STRUCTs and ENUMs.

#### 5.4.2. I/O MAPPING TO ESTABLISH MESSAGE EXCHANGE ON TOP OF PROFINET CYCLIC DATA EXCHANGE

Double click “PLC\_PRG (PRG)” within the Devices tree. Add the following variables within the VAR section.

```

PROGRAM PLC_PRG
VAR
 initialized : BOOL(FALSE);

 // I/O mapping to GSDML registers
 status : UDINT;
 ackSendId : UINT;
 recvid : UINT;
 recvdData : ARRAY[0..952] OF USINT;
 ackRecvId : UINT;
 sendId : UINT;
 sendData : ARRAY[0..956] OF USINT;

 recvMsg : POINTER TO IRP_MsgHeader;
 sendMsg : POINTER TO IRP_MsgHeader;
 prevRecvId : UINT;
 sendReq : UINT;
END_VAR

```

**Figure 78.** CODESYS - adding the variables to VAR section.

Add the following into the code section of the ST program.

```

IF NOT initialized THEN
 initialized := TRUE;
 recvMsg := ADR(recvdData);
 sendMsg := ADR(sendData);

 // Default to "no action" on startup
 sendMsg^.msgType := IRP_MsgType.Ignore;
 sendReq := IRP_MsgType.Ignore;
END_IF

// Handle input if changed by device
IF recvid <> prevRecvId THEN
 // New message received: add input handling here

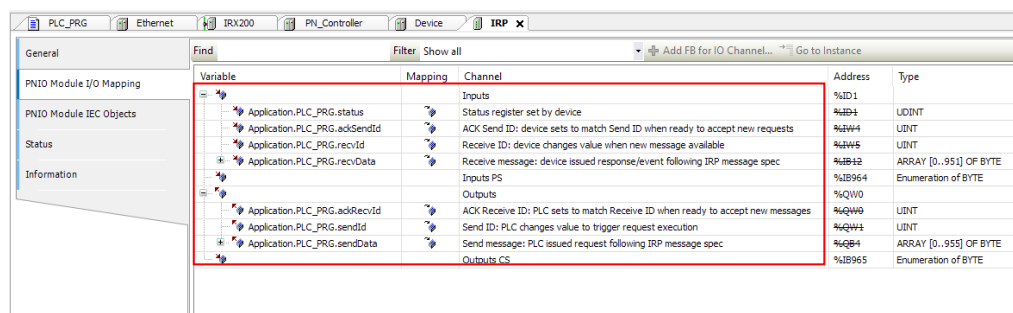
 // Acknowledge received message to accept new input
 prevRecvId := recvid;
 ackRecvId := recvid;
END_IF

// Send new request if device is ready to receive
IF ackSendId = sendId THEN
 IF sendReq <> IRP_MsgType.Ignore THEN
 sendMsg^.msgType := sendReq;
 sendId := sendId + 1; // Mark message for sending
 END_IF
 sendReq := IRP_MsgType.Ignore; // Mark handled request as done
END_IF

```

**Figure 79.** CODESYS - adding code to the ST program.

Double click "IRP (IRP)" device below "IRX200 (IRX200)" device in Devices tree and navigate to "PNIO Module I/O Mapping" to map the newly created variables to the Profinet endpoints as follows.



| Variable                      | Mapping | Channel                                                                        | Address | Type                  |
|-------------------------------|---------|--------------------------------------------------------------------------------|---------|-----------------------|
| Application.PLC_PRG.status    | ↔       | Inputs                                                                         | %ID1    | UDINT                 |
| Application.PLC_PRG.ackSendId | ↔       | Status register set by device                                                  | %ID4    | UDINT                 |
| Application.PLC_PRG.recvid    | ↔       | ACK Send ID: device sets to match Send ID when ready to accept new requests    | %ID5    | UDINT                 |
| Application.PLC_PRG.recvdData | ↔       | Receive ID: device changes value when new message available                    | %IB12   | ARRAY [0..95] OF BYTE |
| Application.PLC_PRG.ackRecvId | ↔       | Receive message: device issued response/event following IRP message spec       | %IB964  | Enumeration of BYTE   |
| Application.PLC_PRG.sendId    | ↔       | Inputs PS                                                                      | %QW0    | UDINT                 |
| Application.PLC_PRG.sendData  | ↔       | Outputs                                                                        | %QW9    | UDINT                 |
| Application.PLC_PRG.sendReq   | ↔       | ACK Receive ID: PLC sets to match Receive ID when ready to accept new messages | %QW4    | UDINT                 |
| Application.PLC_PRG.sendReq   | ↔       | Send ID: PLC changes value to trigger request execution                        | %QB4    | ARRAY [0..95] OF BYTE |
| Application.PLC_PRG.sendReq   | ↔       | Send message: PLC issued request following IRP message spec                    | %IB965  | Enumeration of BYTE   |

**Figure 80.** CODESYS – IO mapping.

Basic I/O mapping is now complete.

### 5.4.3. TESTING THE I/O MAPPING

Login via “Online” / “Login”. Provide credentials when prompted.

Double click the value of sendReq on row 22.

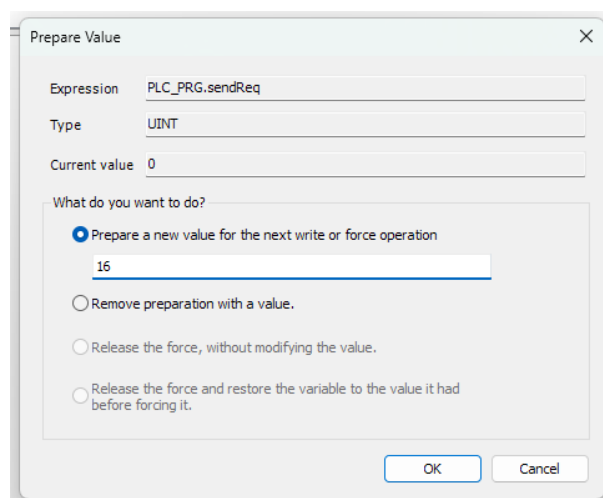
```

1 IF NOT initialized THEN
2 initialized := TRUE;
3 recvMsg[16#000001A10F30CD0E] := ADR(recvData);
4 sendMsg[16#000001A10F30CD07] := ADR(sendData);
5
6 // Default to "no action" on startup
7 sendMsg^.msgType[0] := IRP_MsgType.Ignore[0];
8 sendReq[0] := IRP_MsgType.Ignore[0];
9 END_IF
10
11 // Handle input if changed by device
12 IF recvId[2] <> prevRecvId[2] THEN
13 // New message received: add input handling here
14
15 // Acknowledge received message to accept new input
16 prevRecvId[2] := recvId[2];
17 ackRecvId[2] := recvId[2];
18 END_IF
19
20 // Send new request if device is ready to receive
21 IF ackSendId[0] = sendId[0] THEN
22 IF sendReq[0] < 16 THEN
23 sendMsg^.msgType[0] := sendReq[0];
24 sendId[0] := sendId[0] + 1; // Mark message for sending
25 END_IF
26 sendReq[0] := IRP_MsgType.Ignore[0]; // Mark handled request as done
27 END_IF
28 RETURN

```

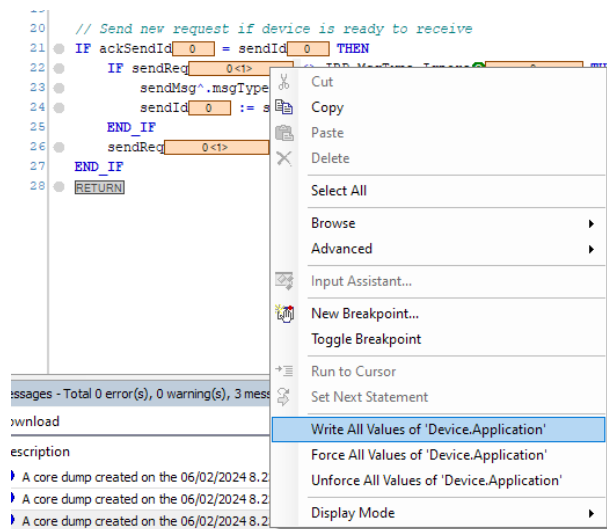
**Figure 81.** CODESYS – assigning the sendReq value.

Assign the value “16” and click “OK”:



**Figure 82.** CODESYS – preparing the value.

sendReq should now show the value as “0 <16>”, which means 0 is the current value but 16 is prepared to be written. Right click and select “**Write All Values of Device.Application**” to apply the prepared value.



**Figure 83.** CODESYS – writing all values of device application.

If the device has been configured to log all IRP message traffic (see “**6.4. Logging**”), the log should now show a received “GetReaderCapabilitiesReq” along with the provided response.

```
nid-sysplugin-nur3 icssprox: <I> 13084446 Request changed! recvid=0x2
nid-sysplugin-nur3 icssprox: <D> 13084446 _IRP_autoGen_validate_IRP_MsgHeader(0xa04257ec,0);
nid-sysplugin-nur3 icssprox: <D> 13084446 type->16
nid-sysplugin-nur3 icssprox: <D> 13084446 id->0
nid-sysplugin-nur3 icssprox: <I> 13084446 received from PN, type 0x10 (GetReaderCapabilitiesReq)
nid-sysplugin-nur3 icssprox: <D> 13084446 _IRP_Handler_GetReaderCapabilitiesReq();
nid-sysplugin-nur3 icssprox: <D> 13084446 sending to PN id 0, type 0x41 (GetReaderCapabilitiesResp)
nid-sysplugin-nur3 icssprox: <D> 13084446 _IRP_autoGen_validate_IRP_MsgHeader(0xa0427495,0);
nid-sysplugin-nur3 icssprox: <D> 13084446 type->65
nid-sysplugin-nur3 icssprox: <D> 13084446 id->0
nid-sysplugin-nur3 icssprox: <D> 13084446 _IRP_autoGen_validate_IRP_MsgPayload_ReaderCapabilities(0xa0427499,0);
nid-sysplugin-nur3 icssprox: <D> 13084446 minPowerDbm->1
nid-sysplugin-nur3 icssprox: <D> 13084446 maxPowerDbm->30
nid-sysplugin-nur3 icssprox: <D> 13084446 stepDb->1
nid-sysplugin-nur3 icssprox: <D> 13084446 region->0
nid-sysplugin-nur3 icssprox: <D> 13084446 maxNumAntennas->2
nid-sysplugin-nur3 icssprox: <D> 13084446 readerIdLen->10
nid-sysplugin-nur3 icssprox: <D> 13084446 icssFwVersionLen->5
nid-sysplugin-nur3 icssprox: <D> 13084446 rfIdFwVersionLen->6
nid-sysplugin-nur3 icssprox: <D> 13084446 modelNameLen->7
nid-sysplugin-nur3 icssprox: <D> 13084446 readerId->'K234500058'
nid-sysplugin-nur3 icssprox: <D> 13084446 icssFwVersion->'0.8.1'
nid-sysplugin-nur3 icssprox: <D> 13084446 rfIdFwVersion->'17.2-C'
nid-sysplugin-nur3 icssprox: <D> 13084446 modelName->'1081-1A'
nid-sysplugin-nur3 icssprox: <D> 13084446 reserved0->0
nid-sysplugin-nur3 icssprox: <D> 13084446 reserved1->0
nid-sysplugin-nur3 icssprox: <D> 13084446 _IRP_tpPN_send() send queued, waiting: 1
```

**Figure 84.** CODESYS – IRX200 device log.

Refer to **7.2 IRP – Industrial Reader Protocol** for the full list of supported requests along with their respective parameters and responses.

#### 5.4.4. ADDING APPLICATION-LEVEL LOGIC TO THE PLC PROGRAM

The comment “New message received: add input handling here” within the PLC ST code marks where new messages from the reader have been received on the PLC.

To send messages to the reader, first populate '**sendMsg.msgPayload**' and optionally '**sendMsg.msghId**'. Finally set '**sendReq**' to the respective **IRP\_MsgType** to trigger sending.

Reading and writing messages may be significantly aided by using IRP data structure pointers to access message contents. For example:

- create a variable "**pErrorCode : POINTER TO IRP\_MsgPayload\_ErrorCode;**"
- in the "add input handling here" section, check to see if the received message type is **IRP\_MsgType.ErrorCodeResp**, and if so, assign "**pErrorCode := ADR(recvMsg^.msgPayload);**" and handle the error using this pointer.

```
// Handle input if changed by device
IF recvId <> prevRecvId THEN
 CASE recvMsg^.msgType OF
 IRP_MsgType.ErrorCodeResp:
 pErrorCode := ADR(recvMsg^.msgPayload);
 IF pErrorCode^.errorCode <> 0 THEN
 // error handling
 END_IF
```

**Figure 85.** CODESYS – input handling.

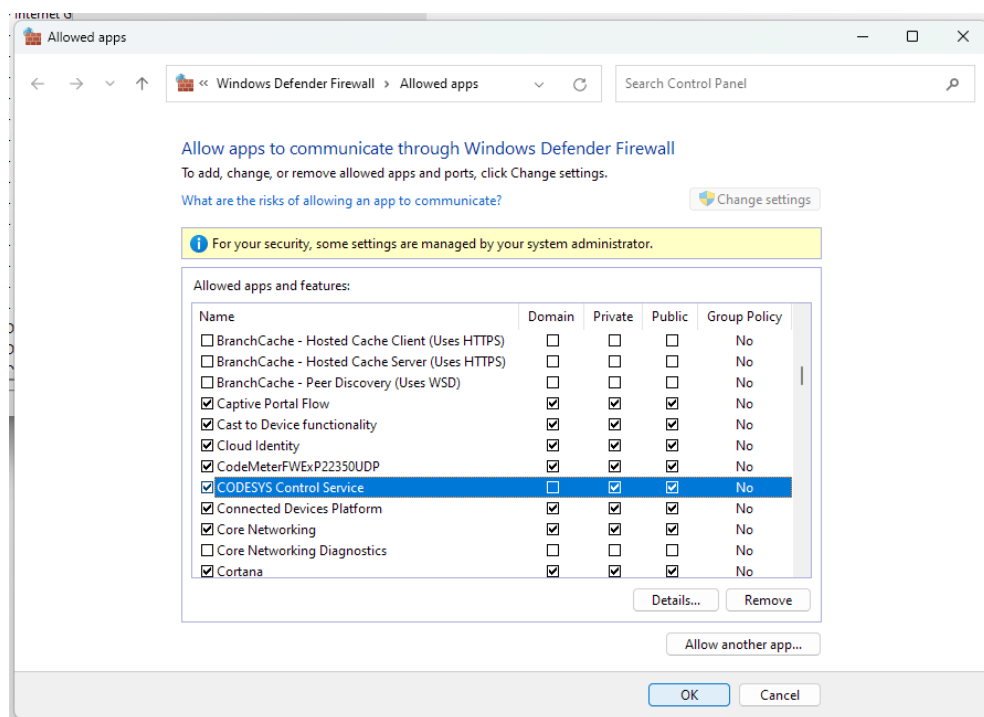
## 6. TROUBLESHOOTING

This section covers most seen issues and how to resolve them.

### 6.1. SOFTWARE PLC (CODESYS CONTROL) NOT RESPONDING

Ensure that the software PLC is running. If started without a license, it will automatically shut down after 2 hours, in which case it needs to be restarted to continue testing.

Ensure that the Windows firewall does not prevent access to the PLC. Start “Firewall & Network Protection”, go to “Allow an app through firewall” / “Change settings” and tick “CODESYS Control Service”.



**Figure 86.** Windows firewall settings.

Click “OK”.

### 6.2. DEVICE CONNECTION FAILS

Ensure that the following requirements stand true:

1. The reader is powered, the network cable is connected, and the “Link” LED is lit on the reader.
2. The PLC is running (if running the software PLC without a license, the PLC will shut down after 2 hours and needs to be manually restarted to resume testing)

3. The station names within the reader's Web UI and the PLC project match.
4. No other device on the same network uses the same station name.
5. Firewall does not prevent messaging with the reader or PLC. See **"6.1 - Software PLC not responding"**.

### 6.3. DEVICE KEEPS RECONNECTING

If the software PLC log shows the device repeatedly reconnecting with "AR consumer DHT expired" messages.

```
pszInfo=**** Station 'brady-irx200': CF81FD0B - Aborted: AR alarm.ind(err): AR consumer DHT expired
pszInfo= Station 'brady-irx200': Connecting...
pszInfo= Station 'brady-irx200': Received Connect Confirmation
pszInfo= Station 'brady-irx200': Received Prm-End Confirmation
pszInfo= Station 'brady-irx200': Received Application-Ready Indication
pszInfo= Station 'brady-irx200': Data Exchange
pszInfo= Station 'brady-irx200': Connected
pszInfo= Station 'brady-irx200':
pszInfo=**** Station 'brady-irx200': CF81FD0B - Aborted: AR alarm.ind(err): AR consumer DHT expired
pszInfo= Station 'brady-irx200': Connecting...
pszInfo= Station 'brady-irx200': Received Connect Confirmation
pszInfo= Station 'brady-irx200': Received Prm-End Confirmation
pszInfo= Station 'brady-irx200': Received Application-Ready Indication
pszInfo= Station 'brady-irx200': Data Exchange
pszInfo= Station 'brady-irx200': Connected
pszInfo= Station 'brady-irx200':
pszInfo=**** Station 'brady-irx200': CF81FD0B - Aborted: AR alarm.ind(err): AR consumer DHT expired
```

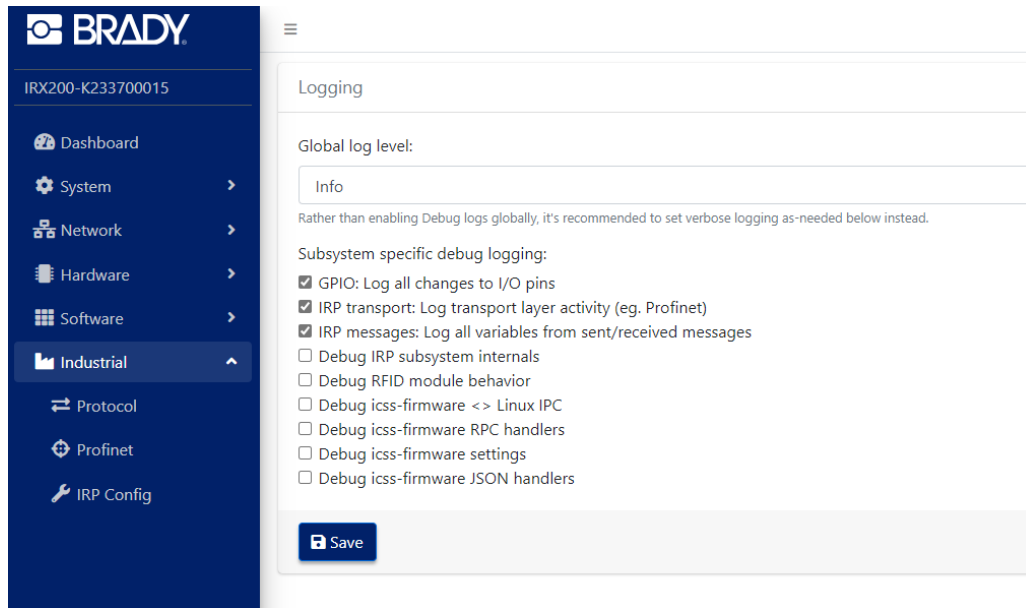
**Figure 87.** PLC log with AR consumer DHT expired message.

The project uses too strict timing for software PLC based implementations to keep up. Increase cycle time and/or reduction ratio until such messages are no longer displayed. If the timing settings are close to the performance limitation of the system running the software PLC, these messages may appear intermittently.

### 6.4. LOGGING

All messages sent and received from the Profinet interface can be logged for troubleshooting purposes and viewed using Web UI. To enable IRP message logging, open the device WebUI and navigate to **Industrial / IRP Config**. Select the desired log scopes such as **"IRP transport: Log transport layer activity"** and **"IRP messages: Log all variables from sent/received messages"**.





**Figure 88.** Web UI IRP Config page for enabling additional debug logging.

Click **Save**. The logs can now be viewed under **System / Log**. The message contents follow “**8. IRP – Industrial Reader Protocol**” specification. These logs would show the reason for why an incoming message was rejected in validation, for example.

If the logged data seen by the reader appears to be in different byte order than sent by the PLC, see section “**7.4. Device Properties**” to change IRP messaging byte order.

## 7. DATA EXCHANGE WITH THE PLC

This section covers the data exchange (PLC Input/Output) description.

### 7.1. DEVICE STATUS SUBMODULE

The Device\_Status submodule defines subslot 1 as an input with the following registers:

- Unsigned16 StatusReg
  - bit0: Busy - high when device is booting or Inventory/TagOp is ongoing. Low when ready to accept new configuration or RF requests.
  - bit1: CriticalError – critical error prevents device from operating properly, may not respond to IRP messages anymore.
  - bit2: SendBufFull – generated messages are being lost as send buffer is full. PLC is not keeping up with receiving messages as fast as they are being produced.
  - bit3: DeviceBooted – set high on boot, cleared by the first IRP message sent by the PLC. May be used to detect if the device has booted.
  - bit4: HeartBeat – toggled every 500ms. May be used to detect connectivity issues between IRX200 and PLC.
  - bit5: TempWarning – temperature warning threshold(80°C) met, overheat mitigation measures throttle RF operation.
  - bit6: TempError – temperature error threshold(95°C) met, RF operations suspended until temperature drops below error threshold.
- Unsigned16 MessagesWaiting - number of messages on IRX200's output buffer waiting to be read by the PLC. The maximum value is 1024.

### 7.2. IRP MESSAGE EXCHANGE SUBMODULE

The IRP submodule provides a transport layer to send and receive messages of up to 960 bytes over PN cyclic data. The payload within these messages are described in the **"8. IRP – Industrial reader protocol description"** section.

All registers listed below follow Profinet/BigEndian byte order, however the byte order for message contents within SendMsg and RecvMsg can be configured via Device properties in PLC or the device's WebUI.

### 7.2.1. PLC INPUT MAPPING (DATA TRANSFER TO PLC)

PLC cyclic input data contains data as described in the tables below.

| PLC Input Data   |              | Data Type               | Offset | Description                                                                                                                                         |
|------------------|--------------|-------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| AckSendID        |              | 2 bytes                 | 0      | Device Sets to match PLC.Output.SendID when ready to accept new requests                                                                            |
| RecvID           |              | 2 bytes                 | 2      | Device changes value when new message available                                                                                                     |
| Reserved         |              | 4 bytes                 | 4      |                                                                                                                                                     |
| (IRP Msg Header) | RecvMsg Type | UInt                    | 8      | One of the IRP_Msg_Type Enum                                                                                                                        |
|                  | MsgID        | UInt                    | 10     | Message specific identification (stamp), responses have the same ID as the request they are responding to. Events always has always has 0 as MsgID. |
|                  | Payload      | Up to 948 bytes of data | 12     | IRP Message payload (Structures described in IRP Specifications)                                                                                    |

**Table 8.** PLC - IRX200 Data Exchange: PLC Input IOMapping.

### 7.2.2. PLC OUTPUT MAPPING (DATA TRANSFER FROM PLC)

PLC cyclic output data contains data as described in the tables below.

| PLC Output Data  |              | Data Type               | Offset | Description                                                           |
|------------------|--------------|-------------------------|--------|-----------------------------------------------------------------------|
| AckRecvID        |              | 2 bytes                 | 0      | PLC Sets to match PLC.Input.RecvID when ready to accept new messages. |
| SendID           |              | 2 bytes                 | 2      | PLC changes the value to trigger request execution                    |
| Reserved         |              | 4 bytes                 | 4      |                                                                       |
| (IRP Msg Header) | SendMsg Type | UInt                    | 8      | One of the IRP_Msg_Type Enum                                          |
|                  | ID           | UInt                    | 10     | Message specific identification (stamp)                               |
|                  | Payload      | Up to 952 bytes of data | 12     | IRP Message payload (Structures described in IRP Specifications)      |

**Table 9.** PLC - IRX200 Data Exchange: PLC Output IOMapping.

### 7.2.3. MESSAGE EXCHANGE SEQUENCE

#### Issuing a request

1. Write SendMsg register in IRP\_MsgHeader format as per IRP description section.
2. Change SendId, this will trigger action on the PN device.
3. Wait for AckSendId to match written request.

#### Reading responses

1. Wait for RecvId to change.
2. Read RecvMsg register in IRP\_MsgHeader format as per IRP description section.
3. Set AckRecvId to match RecvId when ready to accept new messages.

The ID's in SendId and RecvId are unrelated to one another. These values are message specific IDs (stamps).

## 7.3. GPIO SUBMODULE

The GPIO submodule simply defines one 16-bit input register and one 16-bit output register which map to physical IOs. The device will ensure GPIO pin states reflect bits in these registers in both directions.

For the IRX200 device, the registers map as follows:

- bit0: Physical input pin (input)
- bit1: Physical output pin (output)
- bit8: +24V output (output)
- bit9: left LED red channel (output)
- bit10: left LED green channel (output)
- bit11: left LED blue channel (output)
- bit12: right LED red channel (output)
- bit13: right LED green channel (output)
- bit14: right LED blue channel (output)

Writing on an input pin does nothing. Reading an output pin shows the state of the physical output pin and can be used to see when a requested output pin change takes effect.

## 7.4. DEVICE PROPERTIES

A virtual submodule with RecordDataList is provided to allow the PLC to set the device byteorder to match that of the PLC. On CODESYS these settings can be seen when double clicking the IRX200 device.

| Settings                  |              |                 |                |                  |
|---------------------------|--------------|-----------------|----------------|------------------|
| Set All Default Values    |              | Read All Values |                | Write All Values |
| Parameters                | Value        | Data Type       | Allowed Values | Description      |
| Device Properties         |              |                 |                |                  |
| Device Properties Enabled | True         | Unsigned8       |                |                  |
| IRP Message ByteOrder     | LittleEndian | Unsigned8       |                |                  |

**Figure 89.** Devices properties on CODESYS.

**Device Properties Enable:** To enable device properties settings from the PLC. While TRUE, device properties will be set according to values set in the PLC. Device properties default value is FALSE (in which case the device properties are set according to values in the IRX200 Web UI).

**IRP Message ByteOrder:** To switch IRP Message (SendMsg and RecvMsg) between LittleEndian and BigEndian. If Device Properties Enable is set to true, apply this byte order to all IRP message parameters. Otherwise use byte order as set in IRX200 Web UI.

## 8. IRP – INDUSTRIAL READER PROTOCOL DESCRIPTION

This section describes the Brady Industrial Reader Protocol which is a request/response type message interface for Brady IRX200 UHF RFID Readers.

Note that all multi-byte data types may be configured to use LittleEndian or BigEndian byte order. This can be set from the device's WebUI under "Industrial" / "Profinet" or by the PLC via Device Properties (see "7.4. Device properties").

For the online version of the interface description please see the below site:

[https://applicationserver.nordicid.com/irp\\_docs/](https://applicationserver.nordicid.com/irp_docs/)

### 8.1. MESSAGE TYPES AND ASSOCIATED RESPONSES

All message type names refer to entries within IRP\_MsgType enum.

Payload names refer to structs with IRP\_MsgPayload\_ prefix.

| Message type                          | Message payload           | Response type                                              |
|---------------------------------------|---------------------------|------------------------------------------------------------|
| 0x00: Ignore                          | -                         | -                                                          |
| 0x10: GetReaderCapabilitiesReq        | -                         | GetReaderCapabilitiesResp or ErrorCodeResp on error        |
| 0x11: GetReaderConfigReq              | -                         | GetReaderConfigResp or ErrorCodeResp on error              |
| 0x12: SetReaderConfigReq              | ReaderConfig              | ErrorCodeResp                                              |
| 0x13: GetInventoryReadConfigReq       | -                         | GetInventoryReadConfigResp or ErrorCodeResp on error       |
| 0x14: SetInventoryReadConfigReq       | InventoryReadConfig       | ErrorCodeResp                                              |
| 0x15: GetInventorySelectMaskConfigReq | -                         | GetInventorySelectMaskConfigResp or ErrorCodeResp on error |
| 0x16: SetInventorySelectMaskConfigReq | InventorySelectMaskConfig | ErrorCodeResp                                              |
| 0x17: InventoryReq                    | InventoryReq              | ErrorCodeResp                                              |
| 0x18: TagOpSelectMaskReq              | TagOpSelectMaskReq        | ErrorCodeResp                                              |
| 0x1a: TagOpReadReq                    | TagOpReadReq              | TagOpReadResp or ErrorCodeResp on error                    |
| 0x1b: TagOpWriteReq                   | TagOpWriteReq             | ErrorCodeResp                                              |
| 0x1c: TagOpLockReq                    | TagOpLockReq              | ErrorCodeResp                                              |
| 0x1d: TagOpKillReq                    | TagOpKillReq              | ErrorCodeResp                                              |

|                                        |                           |               |
|----------------------------------------|---------------------------|---------------|
| 0x1e: StopAllReq                       | StopAllReq                | ErrorCodeResp |
| 0x40: ErrorCodeResp                    | ErrorCode                 | -             |
| 0x41: GetReaderCapabilitiesResp        | ReaderCapabilities        | -             |
| 0x42: GetReaderConfigResp              | ReaderConfig              | -             |
| 0x43: GetInventoryReadConfigResp       | InventoryReadConfig       | -             |
| 0x44: GetInventorySelectMaskConfigResp | InventorySelectMaskConfig | -             |
| 0x45: TagOpReadResp                    | TagOpReadResp             | -             |
| 0x80: InventoryGlobalVisibilityEvent   |                           |               |
| 0x81: InventoryAntennaVisibilityEvent  |                           |               |
| 0x82: InventoryRssiDeltaThresholdEvent |                           |               |
| 0x83: InventorySeenCountIntervalEvent  |                           |               |

## 8.2. GLOBAL DEFINATIONS

From below tables you will find the global definitions used within IRP.

| Item                   | Value |
|------------------------|-------|
| IRP_REVISION           | 0     |
| IRP_MAX_MSG_LEN        | 524   |
| IRP_MAX_ANTENNAS       | 32    |
| IRP_MAX_EPC_LEN        | 64    |
| IRP_MAX_DATA_LEN       | 64    |
| IRP_MAX_EPCDATA_LEN    | 128   |
| IRP_MAX_SELMASK_LEN    | 32    |
| IRP_MAX_CAPSTR_LEN     | 16    |
| IRP_MAX_INV_SELMASKS   | 8     |
| IRP_MAX_READ_BYTES     | 510   |
| IRP_MAX_WRITE_BYTES    | 510   |
| IRP_MAX_TIMEOUT_MS     | 4000  |
| IRP_DEFAULT_TIMEOUT_MS | 500   |



### 8.3. ENUM DATA TYPES

From below tables you will find the ENUM datatypes used within IRP.

| Name                  | Type     |
|-----------------------|----------|
| IRP_MsgType           | uint16_t |
| IRP_Direction         | uint8_t  |
| IRP_ErrorCode         | uint32_t |
| IRP_EventMask         | uint32_t |
| IRP_InventorySelState | uint8_t  |
| IRP_InventorySession  | uint8_t  |
| IRP_InventoryTarget   | uint8_t  |
| IRP_RfProfile         | uint8_t  |
| IRP_SelectAction      | uint8_t  |
| IRP_SelectTarget      | uint8_t  |
| IRP_StopFlags         | uint8_t  |
| IRP_TagBank           | uint8_t  |
| IRP_PN_StatusMask     | uint32_t |

#### IRP\_MsgType

| Name                                        | Value |
|---------------------------------------------|-------|
| IRP_MsgType_Ignore                          | 0x00  |
| IRP_MsgType_GetReaderCapabilitiesReq        | 0x10  |
| IRP_MsgType_GetReaderConfigReq              | 0x11  |
| IRP_MsgType_SetReaderConfigReq              | 0x12  |
| IRP_MsgType_GetInventoryReadConfigReq       | 0x13  |
| IRP_MsgType_SetInventoryReadConfigReq       | 0x14  |
| IRP_MsgType_GetInventorySelectMaskConfigReq | 0x15  |
| IRP_MsgType_SetInventorySelectMaskConfigReq | 0x16  |
| IRP_MsgType_InventoryReq                    | 0x17  |

|                                              |      |
|----------------------------------------------|------|
| IRP_MsgType_TagOpSelectMaskReq               | 0x18 |
| IRP_MsgType_TagOpReadReq                     | 0x1a |
| IRP_MsgType_TagOpWriteReq                    | 0x1b |
| IRP_MsgType_TagOpLockReq                     | 0x1c |
| IRP_MsgType_TagOpKillReq                     | 0x1d |
| IRP_MsgType_StopAllReq                       | 0x1e |
| IRP_MsgType_ErrorCodeResp                    | 0x40 |
| IRP_MsgType_GetReaderCapabilitiesResp        | 0x41 |
| IRP_MsgType_GetReaderConfigResp              | 0x42 |
| IRP_MsgType_GetInventoryReadConfigResp       | 0x43 |
| IRP_MsgType_GetInventorySelectMaskConfigResp | 0x44 |
| IRP_MsgType_TagOpReadResp                    | 0x45 |
| IRP_MsgType_InventoryGlobalVisibilityEvent   | 0x80 |
| IRP_MsgType_InventoryAntennaVisibilityEvent  | 0x81 |
| IRP_MsgType_InventoryRssiDeltaThresholdEvent | 0x82 |
| IRP_MsgType_InventorySeenCountIntervalEvent  | 0x83 |

#### IRP\_Direction

| Name              | Value | Comment     |
|-------------------|-------|-------------|
| IRP_Direction_IN  | 1     | Tag arrived |
| IRP_Direction_OUT | 0     | Tag left    |

#### IRP\_ErrorCode

| Name                    | Value | Comment                                                            |
|-------------------------|-------|--------------------------------------------------------------------|
| IRP_ErrorCode_SUCCESS   | 0     |                                                                    |
| IRP_ErrorCode_EUNKNOWN  | 1     | Unexpected error received from NurAPI                              |
| IRP_ErrorCode_EREVISION | 2     | Requested revision is newer than installed firmware                |
| IRP_ErrorCode_EINVAL    | 3     | Invalid argument for received message type/revision                |
| IRP_ErrorCode_EBUSY     | 4     |                                                                    |
| IRP_ErrorCode_EINTERNAL | 5     | Internal error, either Inventory task or Nur module not responding |

|                                |    |                |
|--------------------------------|----|----------------|
| IRP_ErrorCode_EHWERR           | 20 |                |
| IRP_ErrorCode_EHIREFLECTIONS   | 21 |                |
| IRP_ErrorCode_ELOWVOLTAGE      | 22 |                |
| IRP_ErrorCode_EOVERTEMP        | 23 |                |
| IRP_ErrorCode_ETAGNOTFOUND     | 40 |                |
| IRP_ErrorCode_ETAGACCESS       | 41 | Wrong password |
| IRP_ErrorCode_ETAGMEMOVERRUN   | 42 |                |
| IRP_ErrorCode_ETAGMEMLOCKED    | 43 |                |
| IRP_ErrorCode_ETAGINSUFPOWER   | 44 |                |
| IRP_ErrorCode_ETAGSELECT       | 45 |                |
| IRP_ErrorCode_ETAGREAD         | 46 |                |
| IRP_ErrorCode_ETAGREADPARTIAL  | 47 |                |
| IRP_ErrorCode_ETAGWRITE        | 48 |                |
| IRP_ErrorCode_ETAGWRITEPARTIAL | 49 |                |
| IRP_ErrorCode_ETAGRESP         | 50 |                |
| IRP_ErrorCode_ETAGNONSPECIFIC  | 51 |                |

#### IRP\_EventMask (values must be maskable bit fields)

| name                                      | Value | Comment                                                                                                      |
|-------------------------------------------|-------|--------------------------------------------------------------------------------------------------------------|
| IRP_EventMask_InventoryGlobalVisibility   | 0x01  | Will be triggered when tag appears and disappears (not seen within a defined timeout).                       |
| IRP_EventMask_InventoryAntennaVisibility  | 0x02  | In addition to above, event will also be triggered when tag changes between antennas.                        |
| IRP_EventMask_InventoryRssiDeltaThreshold | 0x04  | Will be triggered when tag's RSSI value changes more than the defined threshold.                             |
| IRP_EventMask_InventorySeenCountInterval  | 0x08  | Will be triggered when tag has been read as many times as defined by interval value (and multiples of that). |

#### IRP\_InventorySelState

| Name                      | Value | Comment                            |
|---------------------------|-------|------------------------------------|
| IRP_InventorySelState_All | 0     | All tags respond, ignoring SL flag |

|                             |   |                                        |
|-----------------------------|---|----------------------------------------|
| IRP_InventorySelState_NotSL | 2 | Only tags with SL de-asserted responds |
| IRP_InventorySelState_SL    | 3 | Only tags with SL asserted responds    |

#### IRP\_InventorySession

| Item                    | Value | Comment |
|-------------------------|-------|---------|
| IRP_InventorySession_S0 | 0     |         |
| IRP_InventorySession_S1 | 1     |         |
| IRP_InventorySession_S2 | 2     |         |
| IRP_InventorySession_S3 | 3     |         |

#### IRP\_InventoryTarget

| Name                   | Value | Comment                                        |
|------------------------|-------|------------------------------------------------|
| IRP_InventoryTarget_A  | 0     | Query tags with inventoried flag set to A      |
| IRP_InventoryTarget_B  | 1     | Query tags with inventoried flag set to B      |
| IRP_InventoryTarget_AB | 2     | Query tags with inventoried flag set to A or B |

#### IRP\_RfProfile

| Name                   | Value | Comment |
|------------------------|-------|---------|
| IRP_RfProfile_HiSens   | 0     |         |
| IRP_RfProfile_Nominal  | 1     |         |
| IRP_RfProfile_Fast     | 2     |         |
| IRP_RfProfile_HiSpeed  | 3     |         |
| IRP_RfProfile_HiSpeed2 | 4     |         |

#### IRP\_SelectAction

| Name                | Value | Comment                                                                                                                |
|---------------------|-------|------------------------------------------------------------------------------------------------------------------------|
| IRP_SelectAction_A0 | 0     | Matching tags: assert SL or inventoried session flag -> A. Non-matching: de-assert SL or inventoried session flag -> B |

|                     |   |                                                                                                                        |
|---------------------|---|------------------------------------------------------------------------------------------------------------------------|
| IRP_SelectAction_A1 | 1 | Matching tags: assert SL or inventoried session flag -> A. Non-matching: do nothing                                    |
| IRP_SelectAction_A2 | 2 | Matching tags: do nothing. Non-matching: de-assert SL or inventoried session -> B                                      |
| IRP_SelectAction_A3 | 3 | Matching tags: negate SL or invert inventoried session flag (A->B, B->A). Non-matching: do nothing                     |
| IRP_SelectAction_A4 | 4 | Matching tags: de-assert SL or inventoried session flag -> B. Non-matching: assert SL or inventoried session flag -> A |
| IRP_SelectAction_A5 | 5 | Matching tags: de-assert SL or inventoried session flag -> B. Non-matching: assert SL or inventoried session flag -> A |
| IRP_SelectAction_A6 | 6 | Matching tags: do nothing. Non-matching: assert SL or inventoried session flag -> A                                    |
| IRP_SelectAction_A7 | 7 | Matching tags: do nothing. Non-matching: negate SL or invert inventoried session flag (A->B, B->A)                     |

#### IRP\_SelectTarget

| Name                | Value | Comment |
|---------------------|-------|---------|
| IRP_SelectTarget_S0 | 0     |         |
| IRP_SelectTarget_S1 | 1     |         |
| IRP_SelectTarget_S2 | 2     |         |
| IRP_SelectTarget_S3 | 3     |         |
| IRP_SelectTarget_SL | 4     |         |

#### IRP\_StopFlags

| Name                           | Value | Comment                                   |
|--------------------------------|-------|-------------------------------------------|
| IRP_StopFlags_ClearEventBuffer | 0x01  | Remove queued events from the send buffer |

#### IRP\_TagBank

| Name                 | Value | Comment |
|----------------------|-------|---------|
| IRP_TagBank_Password | 0     |         |
| IRP_TagBank_EPC      | 1     |         |
| IRP_TagBank_TID      | 2     |         |

|                  |   |  |
|------------------|---|--|
| IRP_TagBank_User | 3 |  |
|------------------|---|--|

#### IRP\_PN\_StatusMask

| Name                            | Value  | Comment                               |
|---------------------------------|--------|---------------------------------------|
| IRP_PN_StatusMask_Busy          | 0x0001 |                                       |
| IRP_PN_StatusMask_CriticalError | 0x0002 |                                       |
| IRP_PN_StatusMask_SendBufFull   | 0x0004 |                                       |
| IRP_PN_StatusMask_DeviceBooted  | 0x0008 |                                       |
| IRP_PN_StatusMask_HeartBeat     | 0x0010 |                                       |
| IRP_PN_StatusMask_TempWarning   | 0x0020 | Throttling RF as temp mitigation      |
| IRP_PN_StatusMask_TempError     | 0x0040 | Pausing RF due to over temp condition |

## 8.4. DATA STRUCTURES

From below tables you will find the data structures used within IRP.

#### IRP\_AntennaConfig

| Offset | Size | Type     | Item             | Comment                                                                                 |
|--------|------|----------|------------------|-----------------------------------------------------------------------------------------|
| 0      | 4    | uint32_t | antennaMask      | Defines the antenna(s) which this configuration is applied to                           |
| 4      | 2    | uint16_t | transitTime      | Defines the maximum time the reading is performed                                       |
| 6      | 1    | uint8_t  | defaultPowerDbm  | Power used during inventory. Default power for tag operations if not defined separately |
| 7      | 1    | uint8_t  | tagReadPowerDbm  | Power used during tag read operations. Set 255 to use defaultPowerDbm                   |
| 8      | 1    | uint8_t  | tagWritePowerDbm | Power used during tag write/lock/kill operations. Set 255 to use defaultPowerDbm        |
| 9      | 1    | uint8_t  | inventoryQ       | Defines the Q-value used in the inventory. Set 0 for automatic                          |
| 10     | 1    | uint8_t  | inventoryTarget  | Defines the target parameter used in the inventory                                      |
| 11     | 1    | uint8_t  | inventorySession | Defines the session parameter used in the inventory                                     |
| 12     | 1    | uint8_t  | inventoryRounds  | Defines the number of rounds done within one inventory cycle. Set 0 for automatic       |
| 13     | 1    | uint8_t  | reserved0        |                                                                                         |
| 14     | 1    | uint8_t  | reserved1        |                                                                                         |

|       |    |         |           |  |
|-------|----|---------|-----------|--|
| 15    | 1  | uint8_t | reserved2 |  |
| Total | 16 | bytes   |           |  |

### IRP\_InventoryEventTag

| Offset | Size | Type                         | Name          | Comment                                         |
|--------|------|------------------------------|---------------|-------------------------------------------------|
| 0      | 8    | uint64_t                     | firstSeenTime | Indicates the time when the tag was first seen  |
| 8      | 8    | uint64_t                     | lastSeenTime  | Indicates the time when the tag was last seen   |
| 16     | 4    | uint32_t                     | seenCount     | Indicates how many times the tag has been read  |
| 20     | 4    | uint32_t                     | freq          | Tag's reply frequency                           |
| 24     | 2    | int16_t                      | phaseDiff     | Tag's reply phase difference value              |
| 26     | 1    | uint8_t                      | antennaId     | ID of the antenna which read the tag            |
| 27     | 1    | int8_t                       | rssI          | Tag's reply received signal strength indication |
| 28     | 2    | uint16_t                     | pc            | Protocol control bits                           |
| 30     | 1    | uint8_t                      | dataLen       | Length of the data content                      |
| 31     | 1    | uint8_t                      | epcLen        | Length of the EPC content                       |
| 32     | 128  | uint8_t[IRP_MAX_EPCDATA_LEN] | epcData       | Tag's EPC (+data if InventoryRead is enabled)   |
| Total  | 160  | bytes                        |               |                                                 |

### IRP\_InventorySelectMask

| Offset | Size | Type                         | Name        | Comment                              |
|--------|------|------------------------------|-------------|--------------------------------------|
| 0      | 1    | uint8_t                      | target      | One of IRP_SelectTarget              |
| 1      | 1    | uint8_t                      | action      | One of IRP_SelectAction              |
| 2      | 1    | uint8_t                      | bank        | One of IRP_TagBank                   |
| 3      | 1    | uint8_t                      | maskLenBits | Length of the inventory select mask  |
| 4      | 4    | uint32_t                     | addressBits | Address of the inventory select mask |
| 8      | 32   | uint8_t[IRP_MAX_SELMASK_LEN] | maskData    | Inventory select mask                |
| Total  | 40   | bytes                        |             |                                      |



### IRP\_MsgHeader

| Offset | Size | Type       | Name     | Item                                                        |
|--------|------|------------|----------|-------------------------------------------------------------|
| 0      | 2    | uint16_t   | type     | One of IRP_MsgType                                          |
| 2      | 2    | uint16_t   | id       | request ID: responses use matching ID, events use zero here |
| 4      | 2    | uint16_t   | revision | IRP protocol revision                                       |
| 6      | 2    | uint16_t   | reserved |                                                             |
| 8      | 0    | uint8_t[0] | payload  |                                                             |
| Total  | 8    | bytes      |          |                                                             |

### IRP\_MsgPayload\_ReaderCapabilities

| Offset | Size | Type                     | Name             | Comment                                                                                   |
|--------|------|--------------------------|------------------|-------------------------------------------------------------------------------------------|
| 0      | 4    | int32_t                  | minPowerDbm      | Minimum power level setting supported by the reader                                       |
| 4      | 4    | int32_t                  | maxPowerDbm      | Maximum power level setting supported by the reader                                       |
| 8      | 4    | int32_t                  | stepDb           | Size of the power adjustment step                                                         |
| 12     | 1    | uint8_t                  | region           | Defines the operating region. Note that this is pre-configured based on SKU and is locked |
| 13     | 1    | uint8_t                  | maxNumAntennas   | Maximum number of antennas supported by the reader                                        |
| 14     | 1    | uint8_t                  | readerIdLen      | Length of the reader ID variable                                                          |
| 15     | 1    | uint8_t                  | icssFwVersionLen | Length of the ICSS firmware variable                                                      |
| 16     | 1    | uint8_t                  | rfidFwVersionLen | Length of the RFID firmware variable                                                      |
| 17     | 1    | uint8_t                  | modelNameLen     | Length of the reader model variable                                                       |
| 18     | 16   | char[IRP_MAX_CAPSTR_LEN] | readerId         | Reader ID                                                                                 |
| 34     | 16   | char[IRP_MAX_CAPSTR_LEN] | icssFwVersion    | ICSS firmware version                                                                     |
| 50     | 16   | char[IRP_MAX_CAPSTR_LEN] | rfidFwVersion    | RFID firmware version                                                                     |
| 66     | 16   | char[IRP_MAX_CAPSTR_LEN] | modelName        | Reader model                                                                              |
| 82     | 1    | uint8_t                  | irpRevision      |                                                                                           |
| 83     | 1    | uint8_t                  | reserved1        |                                                                                           |
| Total  | 84   | bytes                    |                  |                                                                                           |

### IRP\_MsgPayload\_ReaderConfig

| Offset | Size | Type                             | Name              | Comment                                                                                       |
|--------|------|----------------------------------|-------------------|-----------------------------------------------------------------------------------------------|
| 0      | 2    | uint16_t                         | tagReadTimeoutMs  | Set 0 to use IRP_DEFAULT_TIMEOUT_MS                                                           |
| 2      | 2    | uint16_t                         | tagWriteTimeoutMs | Set 0 to use IRP_DEFAULT_TIMEOUT_MS                                                           |
| 4      | 2    | uint16_t                         | tagLockTimeoutMs  | Set 0 to use IRP_DEFAULT_TIMEOUT_MS                                                           |
| 6      | 2    | uint16_t                         | tagKillTimeoutMs  | Set 0 to use IRP_DEFAULT_TIMEOUT_MS                                                           |
| 8      | 1    | uint8_t                          | rfProfile         | Defines the used RF-profile.                                                                  |
| 9      | 1    | int8_t                           | inventoryMinRssi  | Defines minimum signal strength for tag's inclusion in inventory operation. Set 0 to disable. |
| 10     | 1    | int8_t                           | tagOpMinRssi      | Defines minimum signal strength for tag's inclusion in tag operation. Set 0 to disable.       |
| 11     | 1    | uint8_t                          | numAntennaConfigs | The number of unique antenna configurations which will be set to the reader                   |
| 12     | 512  | IRP_AntennaConfig[IRP_MAX_ANNAS] | antennaConfigs    |                                                                                               |
| Total  | 524  | bytes                            |                   |                                                                                               |

### IRP\_MsgPayload\_ErrorCode

| Offset | Size | Type     | Name      | Comment              |
|--------|------|----------|-----------|----------------------|
| 0      | 4    | UInt32_t | errorCode | One of IRP_ErrorCode |
| Total  | 4    | Bytes    |           |                      |

### IRP\_MsgPayload\_InventoryReq

| Offset | Size | Type     | Name                | Comment                                                                   |
|--------|------|----------|---------------------|---------------------------------------------------------------------------|
| 0      | 4    | uint32_t | antennaMask         | Defines which antenna(s) to be used for reading                           |
| 4      | 4    | uint32_t | cycleTimeMs         | Defines the minimum time between sequential inventory operations          |
| 8      | 4    | uint32_t | stopTagLimit        | If set, reading will stop when set number of tags is found                |
| 12     | 4    | uint32_t | stopCycleLimit      | If set, reading will stop when set number of inventory operations is done |
| 16     | 4    | uint32_t | stopStaticTimeoutMs | If set, reading will stop when set time is expired                        |

|       |    |          |                        |                                                                                                     |
|-------|----|----------|------------------------|-----------------------------------------------------------------------------------------------------|
| 20    | 4  | uint32_t | stopNoNewTagsTimeoutMs | If set, reading will stop when no new tags found within the defined timeout                         |
| 24    | 4  | uint32_t | eventMask              | Multiple of IRP_EventMask                                                                           |
| 28    | 4  | uint32_t | eventTagTimeoutMs      | Defines the timeout used by the visibility and antenna events                                       |
| 32    | 4  | uint32_t | eventAntennaMask       | Defines the antenna(s) which will be considered when generating inventory antenna visibility events |
| 36    | 1  | uint8_t  | eventRssiThreshold     | Defines the RSSI threshold which triggers the RSSI threshold event                                  |
| 37    | 1  | uint8_t  | eventSeenCountInterval | Defines the seen count interval which triggers the seen count event                                 |
| 38    | 1  | uint8_t  | reserved0              |                                                                                                     |
| 39    | 1  | uint8_t  | reserved1              |                                                                                                     |
| Total | 40 | bytes    |                        |                                                                                                     |

#### IRP\_MsgPayload\_InventorySelectMaskConfig

| Offset | Size | Type                                          | Name              | Comment                                                |
|--------|------|-----------------------------------------------|-------------------|--------------------------------------------------------|
| 0      | 1    | uint8_t                                       | enabled           | SelectMask is active if enabled and numSelectMasks > 0 |
| 1      | 1    | uint8_t                                       | inventorySelState | One of IRP_InventorySelState                           |
| 2      | 1    | uint8_t                                       | reserved0         |                                                        |
| 3      | 1    | uint8_t                                       | numSelectMasks    | range: 0..IRP_MAX_INV_SELMASKS                         |
| 4      | 320  | IRP_InventorySelectMask[IRP_MAX_INV_SELMASKS] | selectMasks       |                                                        |
| Total  | 324  | bytes                                         |                   |                                                        |

#### IRP\_MsgPayload\_InventoryReadConfig

| Offset | Size | Type    | Name         | Comment                                                         |
|--------|------|---------|--------------|-----------------------------------------------------------------|
| 0      | 1    | uint8_t | enabled      | Enable or disable the inventory read functionality              |
| 1      | 1    | uint8_t | bank         | One of IRP_TagBank                                              |
| 2      | 1    | uint8_t | reserved0    |                                                                 |
| 3      | 1    | uint8_t | readLenBytes | Defines the read length. Must be even. Set 0 to read whole bank |

|       |   |          |              |                                        |
|-------|---|----------|--------------|----------------------------------------|
| 4     | 4 | uint32_t | addressBytes | Defines the read address. Must be even |
| Total | 8 | bytes    |              |                                        |

#### IRP\_MsgPayload\_TagOpSelectMaskReq

| Offset | Size | Type                         | name        | Comment                                                |
|--------|------|------------------------------|-------------|--------------------------------------------------------|
| 0      | 1    | uint8_t                      | enabled     | SelectMask is active if enabled and numSelectMasks > 0 |
| 1      | 1    | uint8_t                      | bank        | One of IRP_TagBank                                     |
| 2      | 2    | uint16_t                     | maskLenBits | Length of the tag operation select mask                |
| 4      | 4    | uint32_t                     | addressBits | Address of the tag operation select mask               |
| 8      | 32   | uint8_t[IRP_MAX_SELMASK_LEN] | maskData    | Tag operation select mask                              |
| Total  | 40   | bytes                        |             |                                                        |

#### IRP\_MsgPayload\_TagOpReadReq

| Offset | Size | Type     | Name         | Comment                                                                             |
|--------|------|----------|--------------|-------------------------------------------------------------------------------------|
| 0      | 4    | uint32_t | antennaMask  | Defines the used antenna(s) for the operation. Set 0 to use all configured antennas |
| 4      | 4    | uint32_t | accessPasswd | Tag access password. Set 0 to disable                                               |
| 8      | 1    | uint8_t  | bank         | One of IRP_TagBank                                                                  |
| 9      | 1    | uint8_t  | reserved0    |                                                                                     |
| 10     | 1    | uint8_t  | reserved1    |                                                                                     |
| 11     | 1    | uint8_t  | readLenBytes | Defines the read length. Must be even. Set 0 to read whole bank                     |
| 12     | 4    | uint32_t | addressBytes | Defines the read address. Must be even                                              |
| Total  | 16   | bytes    |              |                                                                                     |

### IRP\_MsgPayload\_TagOpReadResp

| Offset | Size | Type                        | Name         | Comment |
|--------|------|-----------------------------|--------------|---------|
| 0      | 2    | uint16_t                    | numReadBytes |         |
| 2      | 510  | uint8_t[IRP_MAX_READ_BYTES] | readBytes    |         |
| Total  | 512  | bytes                       |              |         |

### IRP\_MsgPayload\_TagOpWriteReq

| Offset | Size | Type                         | Name          | Comment                                                                             |
|--------|------|------------------------------|---------------|-------------------------------------------------------------------------------------|
| 0      | 4    | uint32_t                     | antennaMask   | Defines the used antenna(s) for the operation. Set 0 to use all configured antennas |
| 4      | 4    | uint32_t                     | accessPasswd  | Tag access password. Set 0 to disable                                               |
| 8      | 1    | uint8_t                      | bank          | One of IRP_TagBank                                                                  |
| 9      | 1    | uint8_t                      | reserved0     |                                                                                     |
| 10     | 2    | uint16_t                     | writeLenBytes | Defines the write length. Must be even                                              |
| 12     | 4    | uint32_t                     | addressBytes  | Defines the write address. Must be even                                             |
| 16     | 510  | uint8_t[IRP_MAX_WRITE_BYTES] | writeBytes    | Write data                                                                          |
| Total  | 526  | bytes                        |               |                                                                                     |

### IRP\_MsgPayload\_TagOpLockReq

| Offset | Size | Type     | Name         | Comment                                                                             |
|--------|------|----------|--------------|-------------------------------------------------------------------------------------|
| 0      | 4    | uint32_t | antennaMask  | Defines the used antenna(s) for the operation. Set 0 to use all configured antennas |
| 4      | 4    | uint32_t | accessPasswd | Tag access password                                                                 |
| 8      | 2    | uint16_t | mask         |                                                                                     |
| 10     | 2    | uint16_t | action       |                                                                                     |
| Total  | 12   | bytes    |              |                                                                                     |

### IRP\_MsgPayload\_TagOpKillReq

| Offset | Size | Type     | Name        | Comment                                                                             |
|--------|------|----------|-------------|-------------------------------------------------------------------------------------|
| 0      | 4    | uint32_t | antennaMask | Defines the used antenna(s) for the operation. Set 0 to use all configured antennas |
| 4      | 4    | uint32_t | killPasswd  | Tag kill password. Required                                                         |
| Total  | 8    | bytes    |             |                                                                                     |

### IRP\_MsgPayload\_StopAllReq

| Offset | Size | Type    | Name      | Comment                   |
|--------|------|---------|-----------|---------------------------|
| 0      | 1    | Uint8_t | stopFlags | Multiple of IRP_StopFlags |
| Total  | 1    | bytes   |           |                           |

### IRP\_MsgPayload\_InventoryAntennaVisibilityEvent

| Offset | Size | Type                  | Name      | Comment                          |
|--------|------|-----------------------|-----------|----------------------------------|
| 0      | 8    | uint64_t              | timestamp | MS resolution of UNIX time (UTC) |
| 8      | 160  | IRP_InventoryEventTag | tagInfo   |                                  |
| 168    | 1    | uint8_t               | antennaId | range: 1..IRP_MAX_ANTENNAS       |
| 169    | 1    | uint8_t               | direction | One of IRP_Direction             |
| 170    | 1    | uint8_t               | reserved0 |                                  |
| 171    | 1    | uint8_t               | reserved1 |                                  |
| 172    | 1    | uint8_t               | reserved2 |                                  |
| 173    | 1    | uint8_t               | reserved3 |                                  |
| 174    | 1    | uint8_t               | reserved4 |                                  |
| 175    | 1    | uint8_t               | reserved5 |                                  |
| Total  | 176  | bytes                 |           |                                  |

### IRP\_MsgPayload\_InventoryGlobalVisibilityEvent

| Offset | Size | Type                  | Name      | Comment                          |
|--------|------|-----------------------|-----------|----------------------------------|
| 0      | 8    | uint64_t              | timestamp | MS resolution of UNIX time (UTC) |
| 8      | 160  | IRP_InventoryEventTag | tagInfo   |                                  |

|       |     |         |           |                      |
|-------|-----|---------|-----------|----------------------|
| 168   | 1   | uint8_t | direction | One of IRP_Direction |
| 169   | 1   | uint8_t | reserved0 |                      |
| 170   | 1   | uint8_t | reserved1 |                      |
| 171   | 1   | uint8_t | reserved2 |                      |
| 172   | 1   | uint8_t | reserved3 |                      |
| 173   | 1   | uint8_t | reserved4 |                      |
| 174   | 1   | uint8_t | reserved5 |                      |
| 175   | 1   | uint8_t | reserved6 |                      |
| Total | 176 | bytes   |           |                      |

#### IRP\_MsgPayload\_InventoryRssiDeltaThresholdEvent

| Offset | Size | Type                  | Name      | Comment                          |
|--------|------|-----------------------|-----------|----------------------------------|
| 0      | 8    | uint64_t              | timestamp | MS resolution of UNIX time (UTC) |
| 8      | 160  | IRP_InventoryEventTag | tagInfo   |                                  |
| 168    | 1    | int8_t                | prevRssi  |                                  |
| 169    | 1    | uint8_t               | reserved0 |                                  |
| 170    | 1    | uint8_t               | reserved1 |                                  |
| 171    | 1    | uint8_t               | reserved2 |                                  |
| 172    | 1    | uint8_t               | reserved3 |                                  |
| 173    | 1    | uint8_t               | reserved4 |                                  |
| 174    | 1    | uint8_t               | reserved5 |                                  |
| 175    | 1    | uint8_t               | reserved6 |                                  |
| Total  | 176  | bytes                 |           |                                  |

#### IRP\_MsgPayload\_InventorySeenCountIntervalEvent

| Offset | Size | Type                  | Name          | Comment                          |
|--------|------|-----------------------|---------------|----------------------------------|
| 0      | 8    | uint64_t              | timestamp     | MS resolution of UNIX time (UTC) |
| 8      | 160  | IRP_InventoryEventTag | tagInfo       |                                  |
| 168    | 4    | uint32_t              | prevSeenCount |                                  |
| 172    | 1    | uint8_t               | reserved0     |                                  |

|       |     |         |           |  |
|-------|-----|---------|-----------|--|
| 173   | 1   | uint8_t | reserved1 |  |
| 174   | 1   | uint8_t | reserved2 |  |
| 175   | 1   | uint8_t | reserved3 |  |
| Total | 176 | bytes   |           |  |

#### IRP\_PN\_PropertiesStruct

| Offset | Size | Type    | Name      | Comment                                |
|--------|------|---------|-----------|----------------------------------------|
| 0      | 1    | uint8_t | enabled   | 0: false, 1: true. Must be first byte. |
| 1      | 1    | uint8_t | byteOrder | 0: native, 1: swap                     |
| Total  | 2    | bytes   |           |                                        |

#### IRP\_PN\_DeviceStatus

| Offset | Size | Type     | Name           | Comment                                         |
|--------|------|----------|----------------|-------------------------------------------------|
| 0      | 2    | uint16_t | statusMask     | Multiple of IRP_StatusMask                      |
| 2      | 2    | uint16_t | numMsgsWaiting | Number of messages on device waiting to be read |
| Total  | 4    | bytes    |                |                                                 |

#### IRP\_PN\_RecvStruct

| Offset | Size | Type          | Name      | Comment                                                           |
|--------|------|---------------|-----------|-------------------------------------------------------------------|
| 0      | 2    | uint16_t      | ackSendId | Device sets to match Send ID when ready to accept new requests    |
| 2      | 2    | uint16_t      | recvId    | Device changes value when new message available                   |
| 4      | 4    | uint32_t      | reserved  |                                                                   |
| 8      | 8    | IRP_MsgHeader | recvMsg   | Contents follow IRP_MsgHeader format in IRP message specification |
| Total  | 16   | bytes         |           |                                                                   |



#### IRP\_PN\_SendStruct

| Offset       | Size | Type          | Name      | Comment                                                           |
|--------------|------|---------------|-----------|-------------------------------------------------------------------|
| 0            | 2    | uint16_t      | ackRecvId | PLC sets to match Receive ID when ready to accept new messages    |
| 2            | 2    | uint16_t      | sendId    | PLC changes value to trigger request execution                    |
| 4            | 4    | uint32_t      | reserved  |                                                                   |
| 8            | 8    | IRP_MsgHeader | sendMsg   | Contents follow IRP_MsgHeader format in IRP message specification |
| <b>Total</b> | 16   | bytes         |           |                                                                   |

## 9. VERSION HISTORY

| Version | Date       | Modifications                                         |
|---------|------------|-------------------------------------------------------|
| 1.0     | 05.04.2024 | Initial release to match the Profinet firmware V1.0.0 |
|         |            |                                                       |
|         |            |                                                       |
|         |            |                                                       |
|         |            |                                                       |
|         |            |                                                       |
|         |            |                                                       |