IPA Drive Controllers
88-032525-01 A

EtherNet/IP Programmer's Guide

Effective: February 2015

ENGINEERING YOUR SUCCESS.

User Information

Warning — IPA Drive Controllers are used to control electrical and mechanical components of motion control systems. Test all motion systems for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

IPA Drive Controllers and the information in this guide, including any apparatus, methods, techniques, and concepts described herein, are the proprietary property of Parker Hannifin Corporation or its licensers and may not be reproduced, disclosed, or used for any purpose not expressly authorized by the owner thereof.

Since Parker Hannifin constantly strives to improve all of its products, we reserve the right to change this user guide, software, and hardware mentioned therein at any time without notice.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of the equipment, firmware, or this user guide.

Technical Assistance

Contact your local automation technology center (ATC) or distributor.

North America and Asia
Parker Hannifin Corporation
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9068 or (707) 584-7558
Fax: (707) 584-3793
Email: emn_support@parker.com
Internet: http://www.parkermotion.com

Parker Hannifin Technical Support E-mail:	emn_support@parker.com

Table of Contents

Table of Tables

Table of Figures

Introduction

This guide describes how to use a Parker-Hannifin IPA controller in an EtherNet/IP™ (Ethernet industrial protocol) network. It provides an overview of EtherNet/IP, a discussion of IPA adapter functions, and specifics of pertinent commands, parameters, and CLASS 1 and Class 3 connections.

Compatible Parker Hannifin Products

EtherNet/IP functionality is available on ACR9xxx and IPA Controller products. The minimum operating system (firmware version) of the products are listed in Table 1.

Product	Adapter	Scanner
ACR9000, 9030, 9040	1.26	TBD
ACR9600, 9630, 9640	2.27	TBD
IPA Controller	4.40	TBD

Table 1: Product Compatibility Specifications

Assumptions of Technical Experience

Before setting up an EtherNet/IP network, it is essential to have a fundamental understanding of the following:

- CIP (Control and Information Protocol) object models for devices
- CIP object classes for connected and unconnected messaging
- EtherNet/IP adaptation of CIP

To install and troubleshoot an IPA controller, a fundamental knowledge of the following is necessary:

- Electronic concepts such as voltage, current, and switches
- Mechanical motion control concepts such as inertia, torque, velocity, distance, and force.

Overview of EtherNet/IP

EtherNet/IP is the TCP/IP encapsulation of CIP, which is also shared by ControlNet™ and DeviceNet™. (For a detailed description of EtherNet/IP and CIP, refer to the ODVA publication *The Specification for EtherNet/IP™.)* The following exploration of EtherNet/IP provides context for the terms used in this guide.

A **device** is a product that supports EtherNet/IP. Some devices conform to industry standard profiles, but there is no such profile for the IPA controllers.

A **connection** is a logical link between two devices. Different types of connections are described below. Two devices may share more than one connection.

A **scanner** is a device that initiates a connection or a request. It may be thought of as a master or a controlling device.

An **adapter** is a device that receives a connection request or an individual service request. Typically one scanner on a network may be connected to several adapters.

An **assembly** is a pre-defined collection of data residing in an adapter. Each assembly is identified by a unique **instance number**. The assemblies are further characterized by their size and type. Three types of assemblies are *producing* (data to be sent), *consuming* (data to be received), and *configuration* (a data area reserved for information about how consumed and produced data is to be interpreted).

A **Class 3 connection** is used for individual request/response transactions. A request from a scanner always results in a response from the adapter indicating the success or failure of the request. The response may also include a data payload if it was part of the request. Class 3 connections are handled in EtherNet/IP via TCP.

A request from a scanner is called a **Service Request** and the meaning of the request is identified by a one-byte **service code** inside the request packet. Most service codes have meanings pre-defined by the CIP specification, but codes 0x4B through 0x63 have meanings that are specific to the destination object of the service request.

The destination of the service request is defined by a portion of the service request packet called the **path**. The path is either a literal ASCII character string or an object description. The adapter receiving the service request can distinguish between an ASCII character string path and an object description path by header bytes inside the path.

A request to an **object** is identified inside the path by its **class** number, **instance** number, and **attribute** number. *Class* identifies which type of object is being referenced, and *instance* defines the particular object of that type. For example, a carton of eggs contains twelve objects. These objects are instances 1 through 12 of class Egg. And each object may have one or more attributes. In our egg example, attributes 1 and

2 could be size and color. So, a service request might ask for the color of egg number 6.

A **CLASS 1 connection** establishes a periodic exchange of data between the scanner and the adapter. The connection request from the scanner establishes the repetition interval, or RPI, in both directions. The acronym **RPI** stands for Requested Packet Interval and is generally expressed in milliseconds. This connection request also establishes the instance numbers of the producing, consuming, and configuration assemblies, and the size of each assembly. It also may contain data destined for the adapter's configuration assembly, which allows the adapter to interpret subsequent data exchange. In EtherNet/IP, the CLASS 1 connection is established via TCP, but the subsequent data exchange uses UDP.

A CLASS 1 connection request also indicates whether the adapter should send its data point-to-point or multicast. **Point-to-point** data is addressed only to the scanner. **Multicast** data is sent to a multicast address group that includes the scanner. This enables other devices on the network to receive that adapter's data. If a CLASS 1 connection request indicates multicast, but the adapter does not support multicast, the connection request fails.

Individual **Class 3 messages** may be sent as **Connected Messages** or **Unconnected Messages**. These messages are commands or data requests from the scanner to individual target nodes. Connected Messages establish a formal CIP connection between devices, allowing either device to detect and report the presence or loss of connection. Unconnected Messages result in no periodic Class 3 connection being established. They are managed by the internal stack's **Unconnected Message Manager** (**UCMM**).

Adapter

IPA Adapter Functions

The IPA adapter functions can be divided into two groups, i.e., Class 3 service requests and CLASS 1 connections. Both are intended to allow a scanner access to the IPA P parameters, but in different ways. The scanner is typically a PLC or HMI software such as Parker-Hannifin's InteractX. PLCs will usually make both a Class 3 and a CLASS 1 connection. InteractX software makes only a Class 3 connection. The IPA controllers allow only one CLASS 1 connection at a time, but unlimited Class 3 connections.

The IPA adapter plays a passive role in both CLASS 1 and Class 3 connections because it simply responds to connection requests and service requests from the scanner. No IPA commands or parameters are required to allow these connections, but certain IPA commands and parameters allow an IPA user or program to monitor the status and description of these connections.

A CLASS 1 connection simply sets up a periodic exchange of data between the IPA P parameters and data tags in the scanner memory. The exact configuration of which P parameters, how many, and in which direction is set up at the scanner. This is usually part of a PLC configuration step, separate from the PLC ladder programming.

The Class 3 service requests may result from software driver implementation, such as Parker-Hannifin's InteractX, or they may be part of a message box inside a ladder rung of a PLC program. All service requests contain a service code and a path. The service code specifies what is being requested and the path specifies the destination object of the request. For some of the service codes supported by IPA products, the path may take the form of an ASCII character string called a tag. Others require specification of class, an instance, and an attribute. The section Service Codes Using Tags on page 18 contains a discussion of IPA service codes that accept a tag and IPA object classes.

IPA Commands

The CIP status report command shows the IPA parameters being accessed if a CLASS 1 connection is active.

In the following example, a CLASS 1 connection is present. The CLASS 1 connection is producing and consuming every 10 milliseconds, accessing sixteen blocks of parameters.

```
SYS>CIP
Class3 Message Stream  = 0
Class1 Received  = 21619317
Class3  Received  = 64321
Total number of connections = 1
Class1 connection instance 1, client IP "192.168.100.20"
Producing every 10msec, consuming every 10msec.
Configured for 16 parameter blocks:
  Sending 8 long(s) starting at P12288
  Sending 1 long(s) starting at P4120
  Sending 1 long(s) starting at P4360
  Sending 1 long(s) starting at P4600
  Sending 8 long(s) starting at P4096
  Sending 1 long(s) starting at P4392
  Sending 1 long(s) starting at P4408
  Sending 5 float(s) starting at P12315
  Sending 6 float(s) starting at P12370
  Sending 8 long(s) starting at P4128
  Sending 8 long(s) starting at P38912
  Reading 4 long(s) starting at P4156
  Reading 4 float(s) starting at P12348
  Reading 8 long(s) starting at P39000
  Reading 8 float(s) starting at P39200
  Reading 8 float(s) starting at P39208
```

The CIP status report command also reports a CLASS 1 configuration error. There are seven possible configuration errors, listed in the next section. In the example that follows, a Class 3 connection is present, but a CLASS 1 connection has configuration error 7. Group 1 tries to access P10, but the IPA controller has not created P10 with the DIM statement, so it does not exist. Group numbering starts from zero; so in the following example, the four groups are groups 0, 1, 2, and 3.

```
SYS>cip
Class3 Message Stream  = 2
Class1 Received  = 0
Class3  Received  = 108
Total number of connections = 1
Class1 connection instance 2, client IP "192.168.10.20"
Configuration error 7, parameter error
Config entries 4, group # 1, parameter # 10, length 8
```

IPA Parameters

The data required by the CIP status report is available in the form of parameters, starting at P37424. These parameters are shown in Table 2 and Table 3. The terms stemming from *produce* and *consume* used in the tables are from the perspective of the IPA controller, as are the respective terms *send* and *read* in the CIP status report. That is, data coming to the controller is *read* and *consumed*, and data going to the scanner is *produced* and *sent* by the controller.

CIP Status Parameters	P Number
Class 3 message stream	P37424
Number of I/O messages sent	P37425
Number of Class 3 messages sent	P37426
Number of Class 3 messages queued	P37427
Total connections	P37428
Client IP address	P37429
CLASS 1 connection instance	P37430
CLASS 1 producing interval (ms)	P37431
CLASS 1 consuming interval (ms)	P37432
CLASS 1 connection status	P37433
Number of CLASS 1 configuration entries	P37434
Last configuration entry group number	P37435
Last configuration entry parameter number	P37436
Last configuration entry length	P37437
Reserved	P37438
Reserved	P37439

Table 2: CIP Status Report Parameters

The values of P37433 (CLASS 1 connection status) range from 0 to 7. Values 1 and 2 indicate the basic state of the connection. Values 2 through 7 represent configuration errors. Each value is defined in Table 3.

Value	Connection Status Meaning
0	No connection attempted, or connection closed
1	Connection active
2	Parameter groups specified greater than 16
3	Parameters in a group greater than 8
4	Sum of long specified for production greater than 100
5	Sum of long specified for consumption greater than 100
6	Length of FSTAT group not 80
7	Invalid or non-existent parameter number

Table 3: Connection Status Error Values and Meanings

CLASS 1 I/O

To establish a CLASS 1 connection, the scanner sends a **ForwardOpen** request to the adapter. The **ForwardOpen** request contains device-specific configuration information along with the type and number of parameters to access. The connection may be multicast or point-to-point.

Specifications

This CLASS 1 I/O is cyclic, and the update rate is user defined. Update interval limits and I/O size limits appear in Table 4.

Value	IPA Controller
Min RPI	1 ms
Max Total Parameters	100 (each direction)
Max Total I/O Bits	3200 (each direction)
Max Group	16
Max Parameters per Group	8

Table 4: CLASS 1 I/O Limits

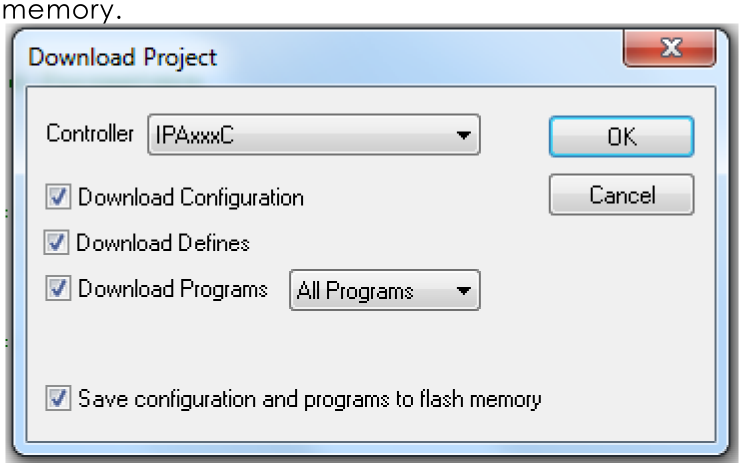
IPA with Logix Controllers

The IPA is compatible for use with CompactLogix and ControlLogix PLCs and can utilize both Class 1 I/O messaging and Class 3 MSG instructions. Add-On instructions are available to enhance the development of IPA applications within the RSLogix environment. The following sections detail the set-up of the IPA and the Logix controller when using the pre-defined AOI's.

ACR-View Set-up for AOIs

Basic settings for the IPA controller are established using ACR-View software.

1. Using ACR-View software, open the project called IPA_EIP_AOI. The project includes AcroBasic code in Programs0 and 1 that execute the functions called by Add-On Instructions from the Logix Controllers.
2. Step thru the configuration wizard, select the correct motor and download motor parameters.
3. Complete the configuration wizard, including axis scaling, fault settings and tuning gains.
4. Download the entire project to the controller and save to flash memory.



5. Either cycle power to the controller, or issue the REBOOT command in the terminal emulator.
6. When the controller re-starts, programs will execute automatically (PBOOT). The programs will create the I/O configuration for the Logix AOIs.

I/O Configuration for AOI's

This CLASS 1 I/O is settings can be set either in the IPA or from the Logix PLC. The I/O can include up to 16 groups of parameters, with up to 8 consecutive parameters in a group. Groups are created by designated a starting parameter, the number of parameters, data direction and data type.

Data Direction	Parameter Value
IPA sends to PLC	0
IPA reads from PLC	1

Data Type	Parameter Value
DINT	1
REAL	2

IPA parameters for I/O configuration.

	Starting Parameter	Number of Parameters	Data Direction	Data Type
Group0	P37440	P37441	P37442	P37443
Group1	P37444	P37445	P37446	P37447
Group2	P37448	P37449	P37450	P37451
Group3	P37452	P37453	P37454	P37455
Group4	P37456	P37457	P37458	P37459
Group5	P37460	P37461	P37462	P37463
Group6	P37464	P37465	P37466	P37467
Group7	P37468	P37469	P37470	P37471
Group8	P37472	P37473	P37474	P37475
Group9	P37476	P37477	P37478	P37479
Group10	P37480	P37481	P37482	P37483
Group11	P37484	P37485	P37486	P37487
Group12	P37488	P37489	P37490	P37491
Group13	P37492	P37493	P37494	P37495
Group14	P37496	P37497	P37498	P37499
Group15	P37500	P37501	P37502	P37503

The AOIs are based on defined values for the I/O groups as follows:

	Starting Parameter	Number of Parameters	Data Direction	Data Type
Group0	12288	8	0	1
Group1	4120	1	0	1
Group2	4360	1	0	1
Group3	4600	1	0	1
Group4	4096	8	0	1
Group5	4392	1	0	1
Group6	4408	1	0	1
Group7	12315	5	0	2
Group8	12370	6	0	2
Group9	4128	8	0	1
Group10	38912	8	0	1
Group11	4156	4	1	1
Group12	12348	4	1	2
Group13	39000	8	1	1
Group14	39200	8	1	2
Group15	39208	8	1	2

Program1 of the project IPA_EIP_AOI loads these values on each power cycle.

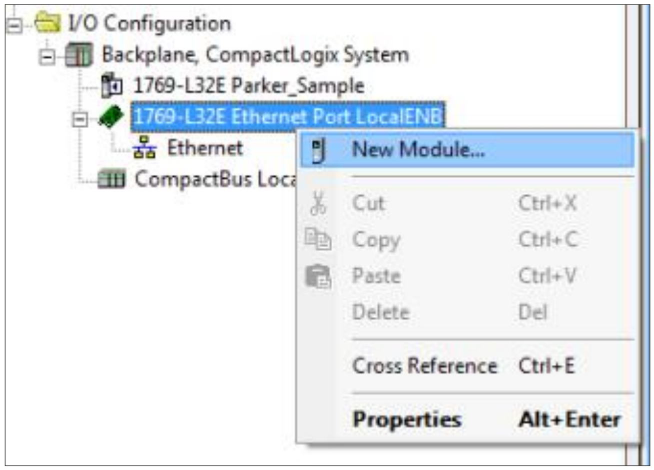
```
_ConfigureEIP
   'Set the Total Number of Groups being used
  P37434=16
    'Group0 Current Position
P37440=12288 : P37441=8 : P37442=0 : P37443=1
    'Group1  Primary Axis Flags
P37444=4120 : P37445=1 : P37446=0 : P37447=1
    'Group2  QuanAxisFlags
P37448=4360 : P37449=1 : P37450=0 : P37451=1
    'Group3 QuinAxisFlags
P37452=4600 : P37453=1 : P37454=0 : P37455=1
    'Group4  InputsOutpsMisc
P37456=4096 : P37457=8 : P37458=0 : P37459=1
    'Group5 Drive Status1 Flags
P37460=4392 : P37461=1 : P37462=0 : P37463=1
    'Group6 Drive Status2 Flags
P37464=4408 : P37465=1 : P37466=0 : P37467=1
    'Group7 Monitor Parameters
P37468=12315 : P37469=5 : P37470=0 : P37471=2
    'Group8 Scaling
P37472=12370 : P37473=6 : P37474=0 : P37475=2
    'Group9 Program Flags
```

```
P37476=4128 : P37477=8 : P37478=0 : P37479=1
   'Group10 User DINTs
P37480=38912 : P37481=8 : P37482=0 : P37483=1
   'Group11 User Flags-Control Words
P37484=4156 : P37485=4 : P37486=1 : P37487=1
   'Group12 Jog Profile Parameters
P37488=12348 : P37489=4 : P37490=1 : P37491=2
   'Group13 User DINTs
P37492=39000 : P37493=8 : P37494=1 : P37495=1
   'Group14 User Reals
P37496=39200 : P37497=8 : P37498=1 : P37499=2
   'Group15 User Reals
P37500=39208 : P37501=8 : P37502=1 : P37503=2     P37501=8
    P37502=1
    P37503=2
  RETURN
```

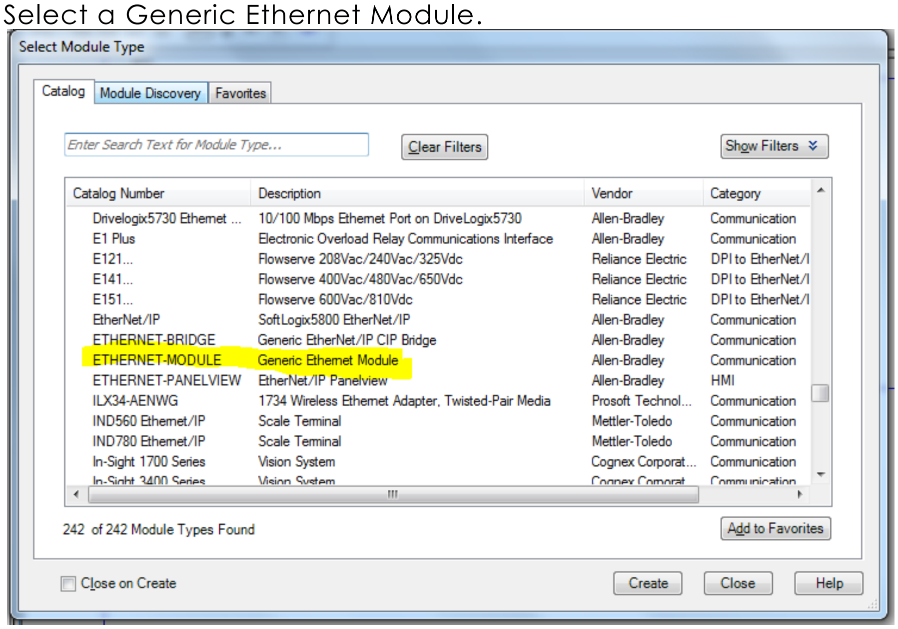
Adding IPA to RSLogix

Now that the IPA has been configured, it is ready to be added to the PLC project.

1. Add a new module to I/O configuration in RSLogix.



2. Select a Generic Ethernet Module.



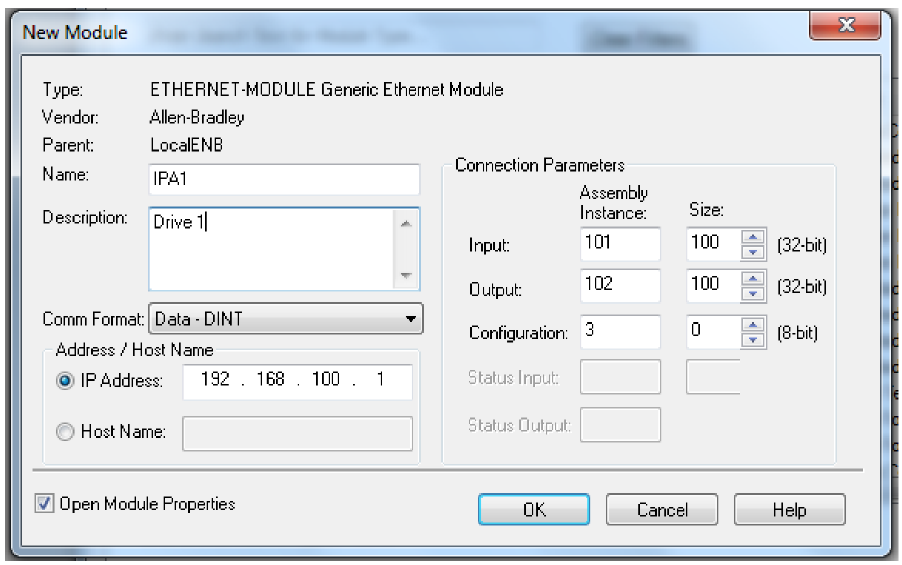
3. Enter the following values in the Module Properties dialog box.

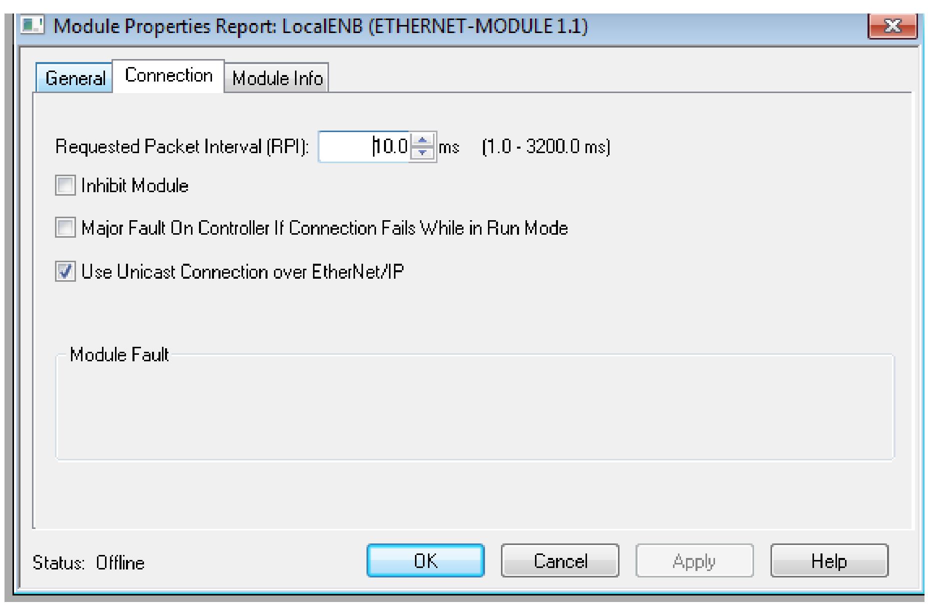
Name:	IPA (create a unique name for each drive used)
IP Address:	Enter IP address setting for the IPA; 192.168.100.1 is the default.

Connection Parameters

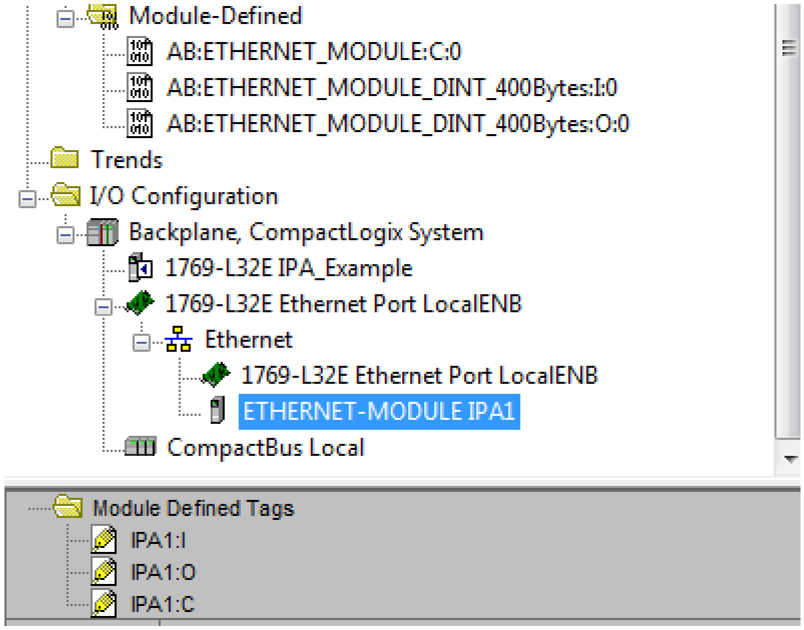
These values are specific to an IPA.

	Assembly Instance:	Size:
Input:	101	100
Output:	102	100
Configuration:	3	0





Adding the module will also create Module Define Tags, which are used as inputs to the AOI's



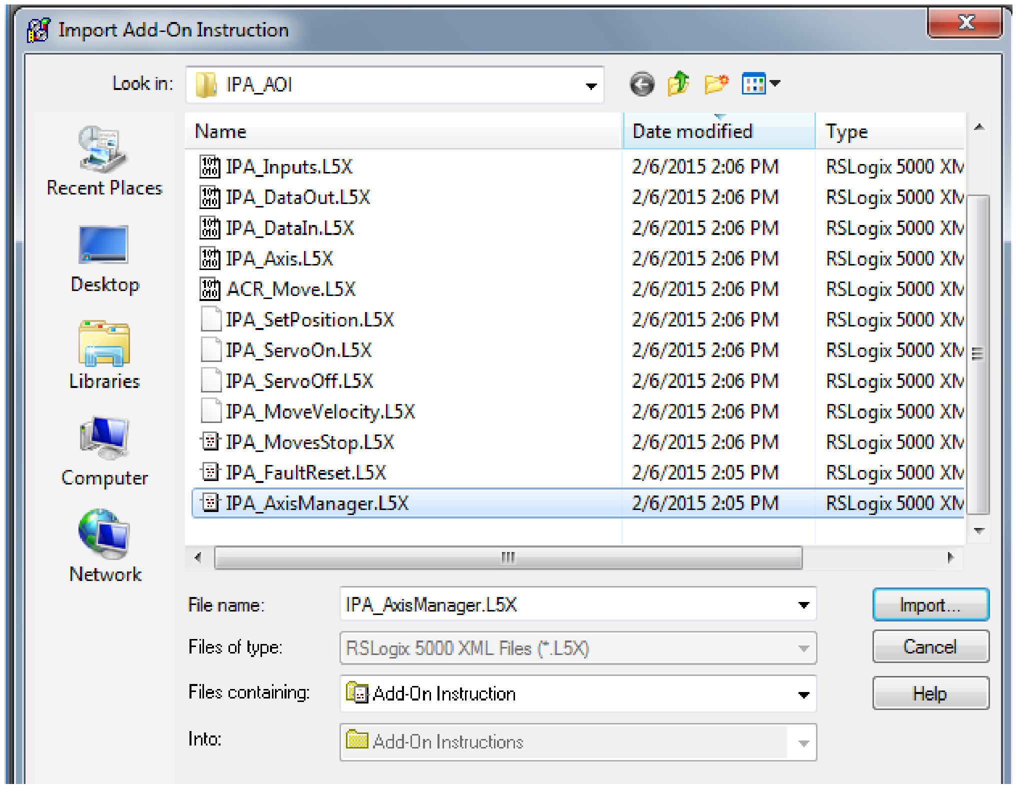
Using IPA Add-On Instructions

Now that the IPA has been configured, it is ready to be added to the PLC project.

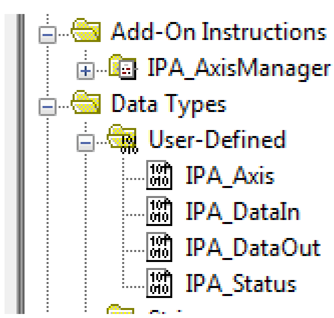
1. Add a new module to I/O configuration in RSLogix.

AOI Name	Function
IPA_AxisManager	Manages the data transfer and AOIs
IPA_ServoOn	Energizes the motor
IPA_ServoOff	De-energizes the motor
IPA_FaultReset	Recovers drive from fault conditions
IPA_SetPosition	Pre-loads position value
IPA_Home	Initiates homing routine
IPA_Move	Initiates absolute and relative position moves
IPA_MoveStop	Stops moves in progress
IPA_MoveVelocity	Initiates a constant velocity move

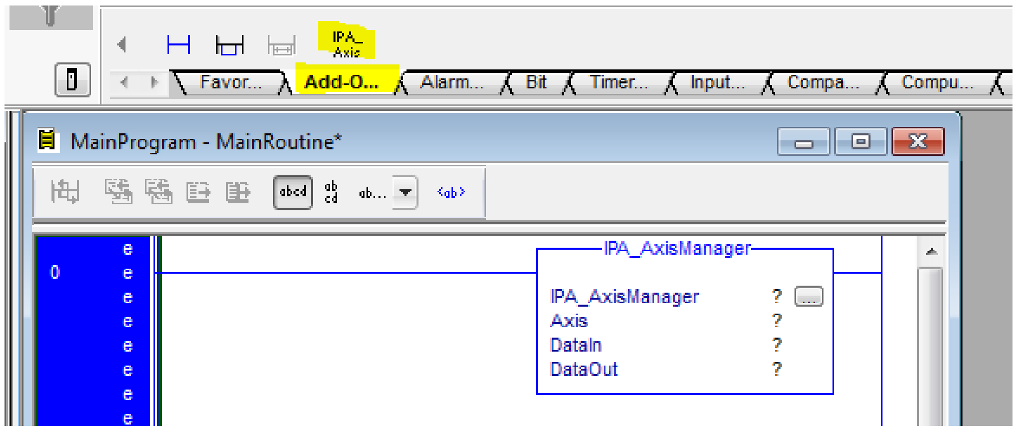
Right click the Add-On Instructions folder and select "Import Add-On Instructions". Navigate to where the IPA_AOIs were downloaded, select IPA_AxisManager and click Import.



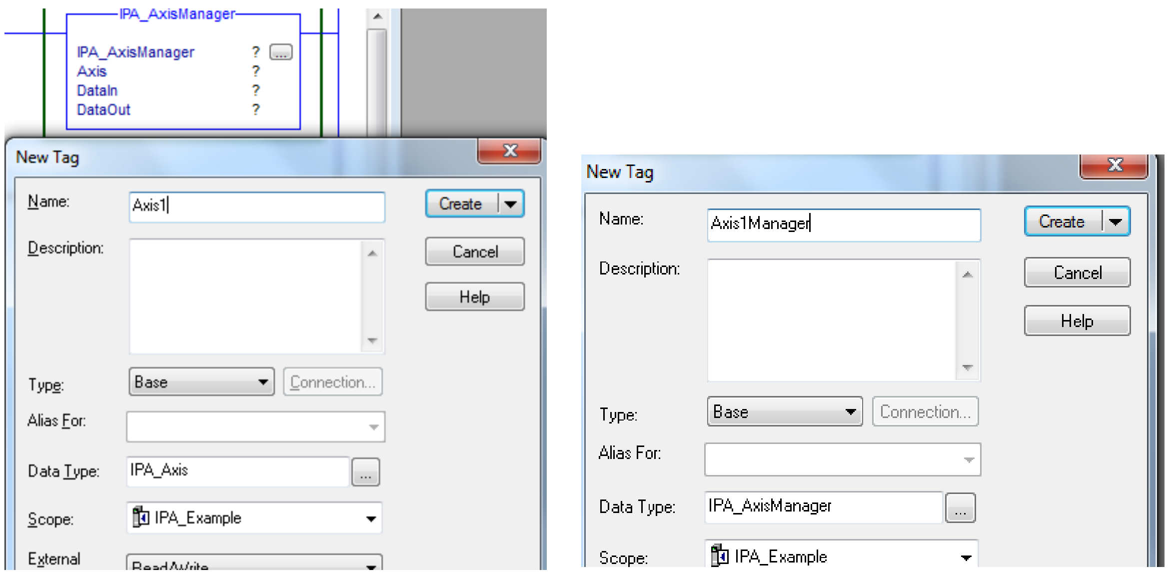
User Defined Data Types that support the AOI's are also imported.



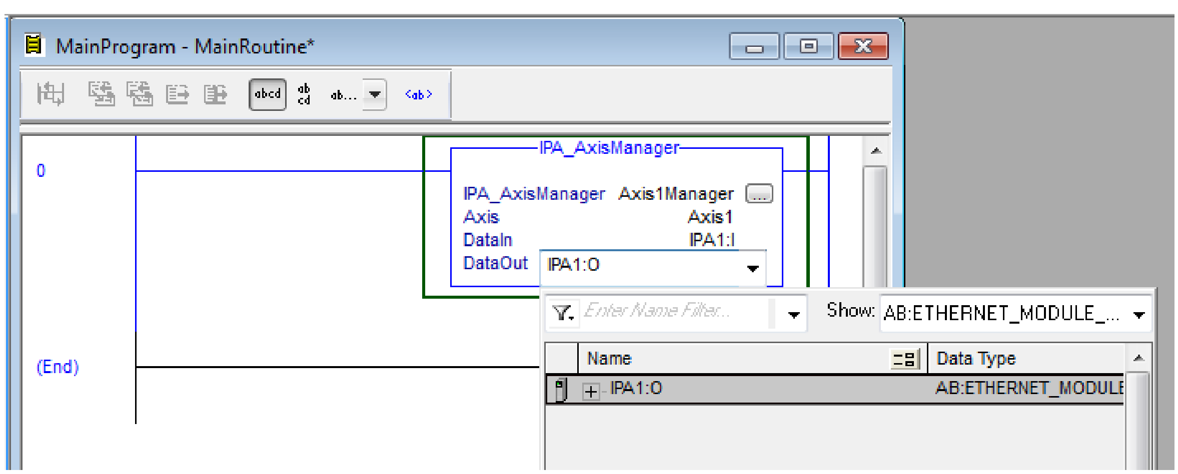
In a program, add the IPA_AxisManager



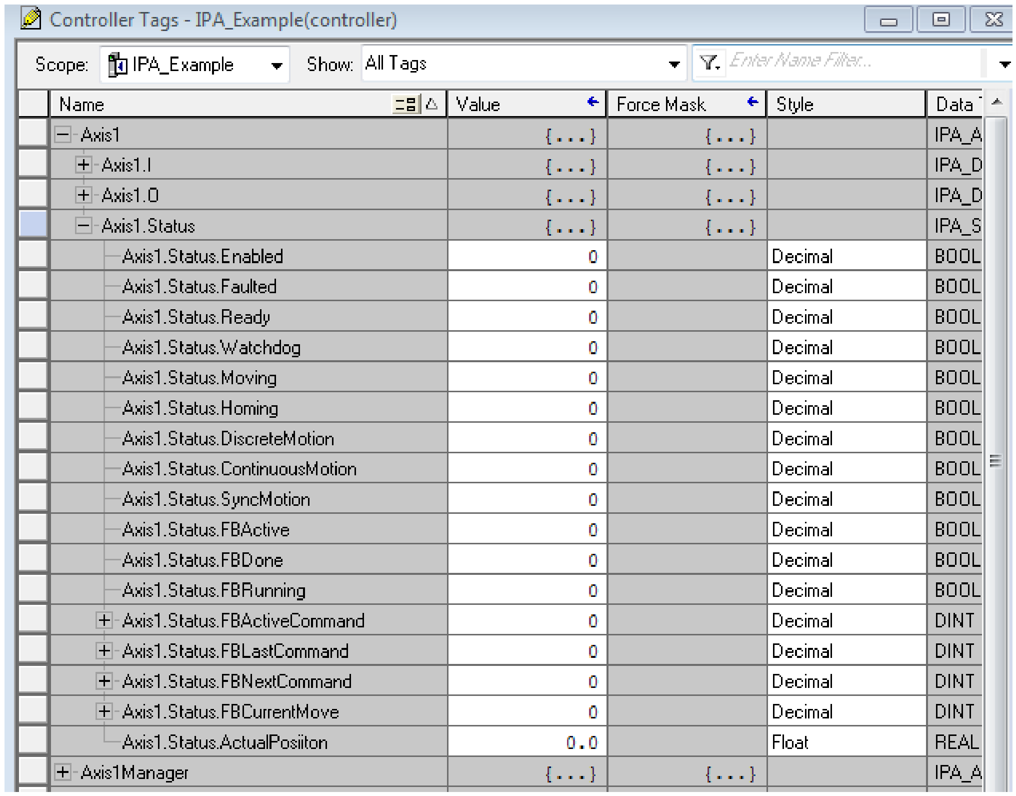
Create new Tags for the AOI with DataT ype IPA_AxisManager and for the Axis with the Data Type IPA_Axis



Next, assign the Module Defined Tags IPA1:I and IPA1:O to the DataIn and DataOut Inputs.



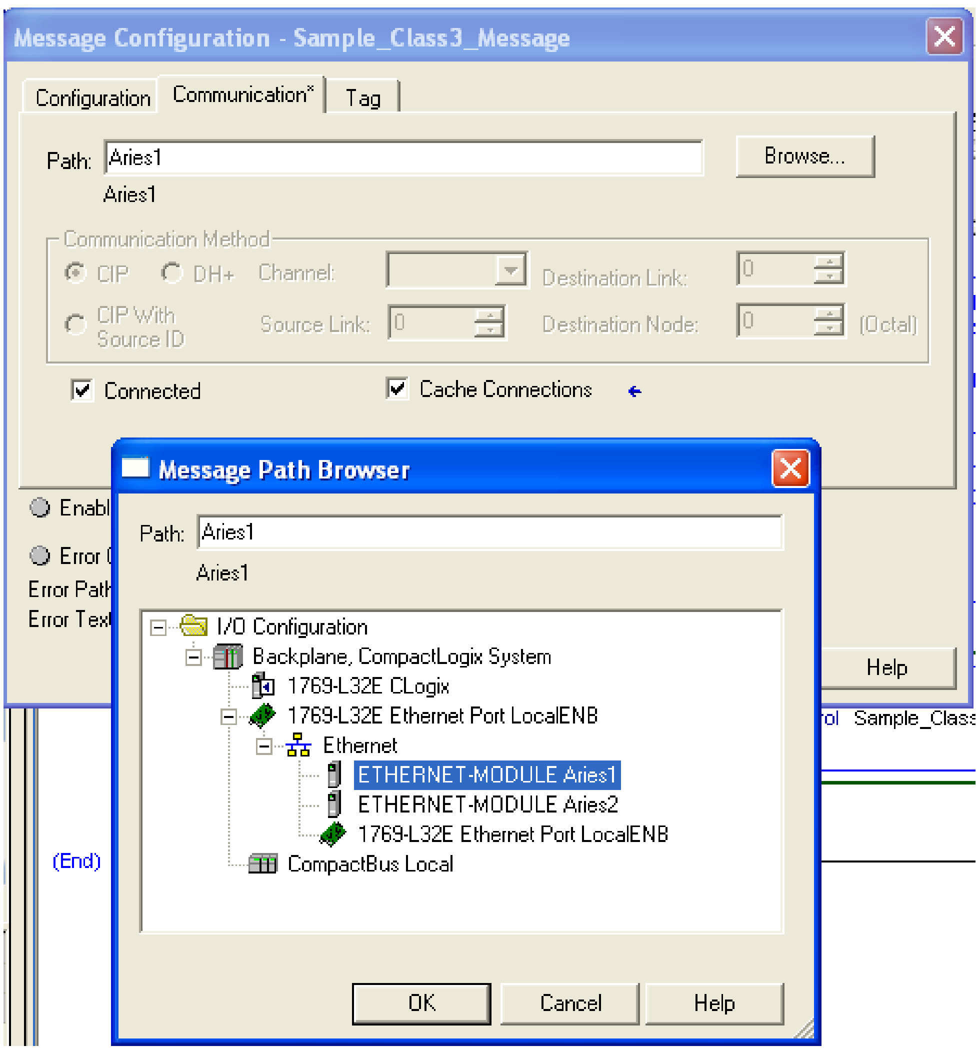
The tag Axis1 (data type IPA_Axis) contains all of the input and output data exchanged between the IPA and the PLC application. Additional Status tags are created that manage the interaction between the other AOI's.



Class 3 CIP Messages

With Class 3 communication, the scanner initiates the connection. If data needs repeated transmission, the connection should be cached. This reduces the overhead related to establishing and closing connections.

2. Once the controller has been added as an Ethernet Module, it can be selected as a Path for any CIP Messages. On the Communication tab of a Message Configuration dialog, select the controller. Check "Connected" to allow the PLC to create a connection to the controller. If the message will be repeated often, check "Cache Connections" to reduce the overhead associated with each connection attempt.



Service Codes Using Tags

For ACR service codes that accept a tag, the tag always takes the form of a parameter number specification, for example, "P4012." Table 5 lists service codes using tags and their descriptions.

Code	Title	Description
0x4C	CIP Data Table Read	Read a block of consecutive DINT data
0x4D	CIP Data Table Write	Write a block of consecutive DINT or Float Data
0x48	Vendor Float Read	Read a block of consecutive Float Data
0x4E	Vendor AND and OR Mask	Clear and set bits of destination P parameter

Table 5: ACR Service Codes Using Tags

PLCs provide a message box that must be programmed by the user. In particular, Allen Bradley PLCs use the 0x4C and 0x4D service codes with tag paths. The following examples show how to formulate these two message types using ControlLogix.

Data Table Read Example

The CIP Data Table Read service may be used to read up to 16 consecutive longs from the ACR controller into a DINT block on the ControlLogix. In the Message Configuration dialog box, the field "Source Element" names the starting P parameter to read from the ACR controller (see Figure 1). In this example, three parameters are read into the ControlLogix DINT array tag "READ_IN_MOTION".

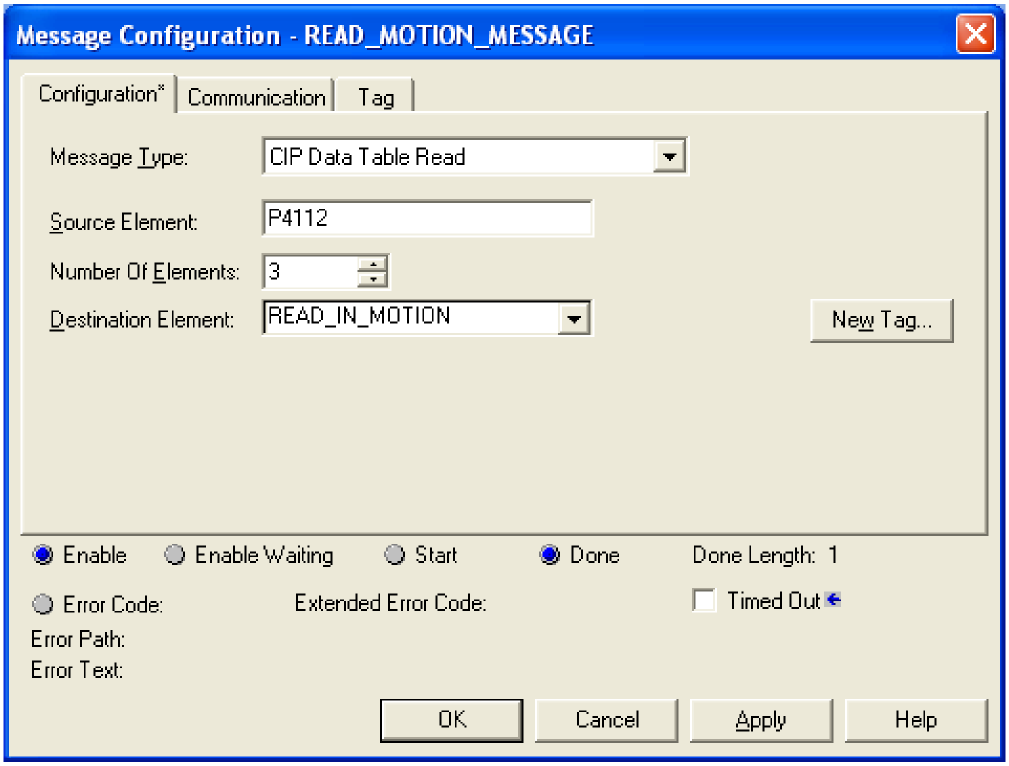


Figure 1: CIP Data Table Read Message Configuration

Data Table Write Example

The CIP Data Table Write service may be used to write up to 16 consecutive longs or floats from the ControlLogix to an ACR controller. In the Message Configuration dialog box in Figure 2, the Source Element field names the ControlLogix DINT array tag "WRITE_LONG" as the start of the data to write to ACR controller parameters starting with P4100. In this example, only one parameter is written into the controller. The CIP Data Table Write service also sends a header to the controller to indicate whether the accompanying data is of type REAL or DINT. If the data type of the destination P parameter does not match the data type of the incoming data, the ACR controller will perform an automatic type conversion before storing the data in the parameter.

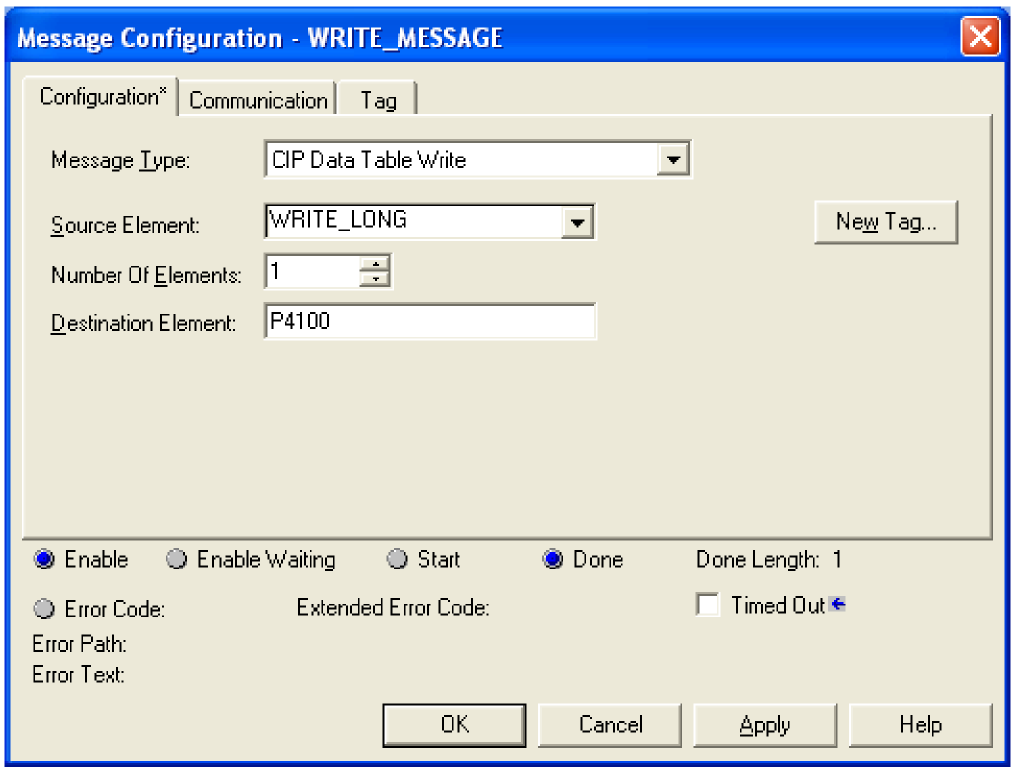


Figure 2: CIP Data Table Write Message Configuration

Classes and Service Codes

Other service codes accepted by ACR controllers require the path to be specified by class, instance, and attribute. Allen Bradley PLCs use the message type CIP Generic to send these service requests. In the Message Configuration dialog box, the Service Type field should always have the value "Custom," and the Attribute field should always have a value of 1. The attribute is actually ignored by these services, but a value of 1 ensures the ACR recognizes it as a valid value. Figure 3 shows a dialog box with the correct field entries. The fields for Service Code, Class, and Instance should be filled in according to the descriptions of the individual classes and service codes presented later in this section.

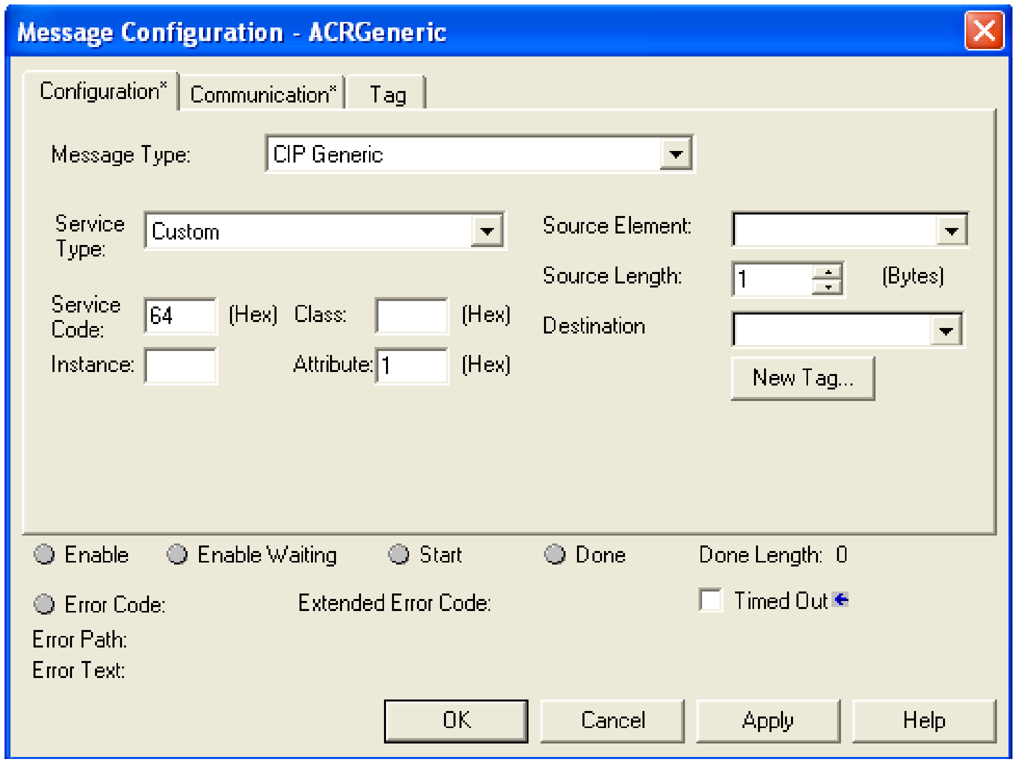


Figure 3: Service Request Message Configuration

All service requests may also carry data to the ACR controller, and accept data from the controller response. The meaning and size of this data depends on the service request.

The data going to the ACR controller is specified by the fields Source Element and Source Length. The Source Element is the name of the data tag in the PLC that holds the source data, and the Source Length is the length of that data in bytes. All data accepted by the controller will either be type DINT or REAL, which are both four bytes in length, so the value of Source Length should always be four times the number of data elements in the Source Element tag.

The Destination Element is the name of the data tag in the PLC that will hold the data coming from the ACR controller. The size is determined by the controller, but if the destination data tag in the PLC is not large enough to hold the incoming data, an error will result.

For service codes that read from an ACR controller (0x51 and 0x53), the data going to the controller is the number of parameters to read, and the data coming from the controller is the block of parameter values. The Source Element tag should be a DINT that contains the number of parameters requested from the ACR. Source Length is the size of Source Element: 4 bytes.

For service codes that write to an ACR controller (0x50, 0x52, 0x54, and 0x34), the data that is being written must put in the source array named by Source Element, and the value of Source Length must be four times the size of the array being written. Since no data is returned from the controller for these writes, the Destination field may be left blank.

Table 6 contains the vendor-specific classes and service codes used with the ACR adapter. Additional information about these classes is provided in the next sections, as well as an example of the application of each service code.

Class	Service Code	Description
Class 100, 0x64 (ACR Parameter)	0x50	Write Float(s)
	0x51	Read Float(s)
	0x52	Write Long(s)
	0x53	Read Long(s)
	0x54	AND and OR(s)
Class 101, 0x65 (ACR Group)	0x34	Write Binary Command(s)
	0x53	Read Groups(s)

Table 6: Vendor-Specific Classes and Corresponding Service Codes

ACR Parameter Class (100)

Class 100, (0x64) is the vendor-specific class that allows access to ACR P parameters. The class range is defined by the CIP common specification. Each P parameter is a separate instance of this class, so the P number defines the instance. The attribute specification is not used and is ignored. There are five Class 100 service codes for getting, setting, and modifying parameters, as shown in Table 6. Specifics of command message configuration for these five service codes follow, along with application examples.

Write Float(s) - Service Code 0x50 (Class 0x64)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x64
Service Code:	0x50
Instance:	Source P parameter
Attribute:	1

Read Float(s) - Service Code 0x51 (Class 0x64)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x64
Service Code:	0x51
Instance:	Source P parameter
Attribute:	1

Write Long(s) - Service Code 0x52 (Class 0x64)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x64
Service Code:	0x52
Instance:	Source P parameter
Attribute:	1

Read Long(s) - Service Code 0x53 (Class 0x64)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x64
Service Code:	0x53
Instance:	Source P parameter
Attribute:	1

AND and OR(s) - Service Code 0x54 (Class 0x64)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x64
Service Code:	0x54
Instance:	Source P parameter
Attribute:	1

Service codes 0x50, 0x51, 0x52, and 0x53 could be applied to any P parameter. These codes take advantage of the ACR controller's ability to perform type conversions as required internally. For example, if service code 0x53 (read long) were applied to P12304, a float, the controller would convert the value of P12304 to a long, and then return the long. Service code 0x54 is meant to set and clear bits, and therefore used with ACR LONG parameters, such as ACR flags.

Writing Example

A ControlLogix PLC writes values to the ACR controller's jog profile parameters for Axis 0. The axis jog parameters (velocity, acceleration, deceleration, and jerk) are consecutive P parameter numbers, allowing one **Write** operation to set all four values. In this example, the message is configured as follows and is illustrated in Figure 4 and Figure 5.

Message Type:	CIP Generic
Service Code:	50 (floating point data)
Instance:	12348 (first ACR parameter to write to) P12348: JOG VEL Setting P12349: JOG ACC Setting P12350: JOG DEC Setting P12351: JOG JRK Setting
Attribute:	1 (Always select 1; this value is not used by the ACR controller.)
Source Element:	PLC Tag (variable) that contains the data to send from the PLC to the ACR controller. The tag, Axis0JogData, is a user-defined data type containing four REAL numbers.
Source Length:	16 (bytes) (The number of parameters to be written to should be multiplied by four. In this example, four parameters are written to; Source Length is 4 x 4 = 16.)

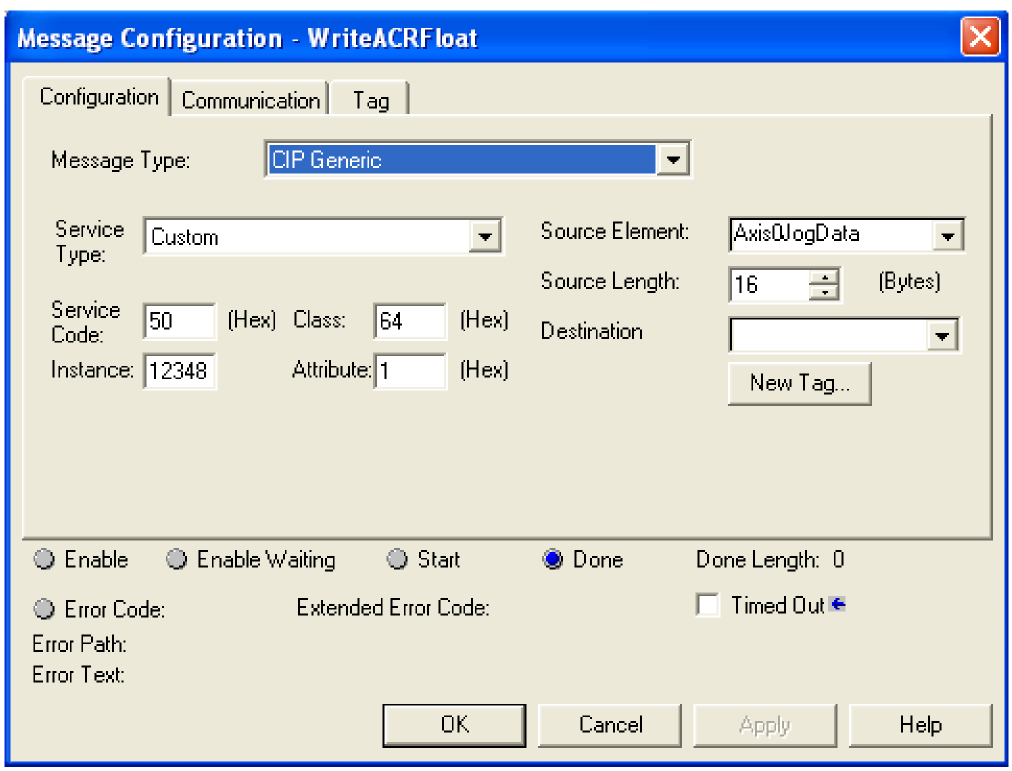


Figure 4: WriteACRFloat Message Configuration Example

Tag Name	Value	Forc	Style	Type
AxisQJogData	{...}	{...}		ACRJogProfile
AxisQJogData.Vel	15.0		Float	REAL
AxisQJogData.Acc	150.0		Float	REAL
AxisQJogData.Dec	125.0		Float	REAL
AxisQJogData.Jrk	625.0		Float	REAL

Figure 5: ControlLogix Tags for WriteACRFloat

Reading Example

A ControlLogix PLC reads values to the ACR controller's Quaternary Axis Flags parameters for Axes 0 to 2. Quaternary Axis Flags contain 32 flags that describe the settings and status of an axis. In this example, the message is configured as follows and shown in Figure 6 and Figure 7.

Message Type:	CIP Generic
Service Code:	53 (long integer data)
Instance:	4360 (first ACR parameter to write to)
Attribute:	1 (Always select 1; this value is not used by the ACR controller.)
Source Element:	PLC Tag (variable) that contains the number of parameters to read from the ACR controller. The PLC program will need to set the tag GetLongs equal to 3.
Source Length:	4 (size of the source element tag/variable in bytes)
Destination:	PLC tag/variable to store the data read from the ACR controller. QuantAxisFlags is a tag based on user-defined data type ACRLongGroup.

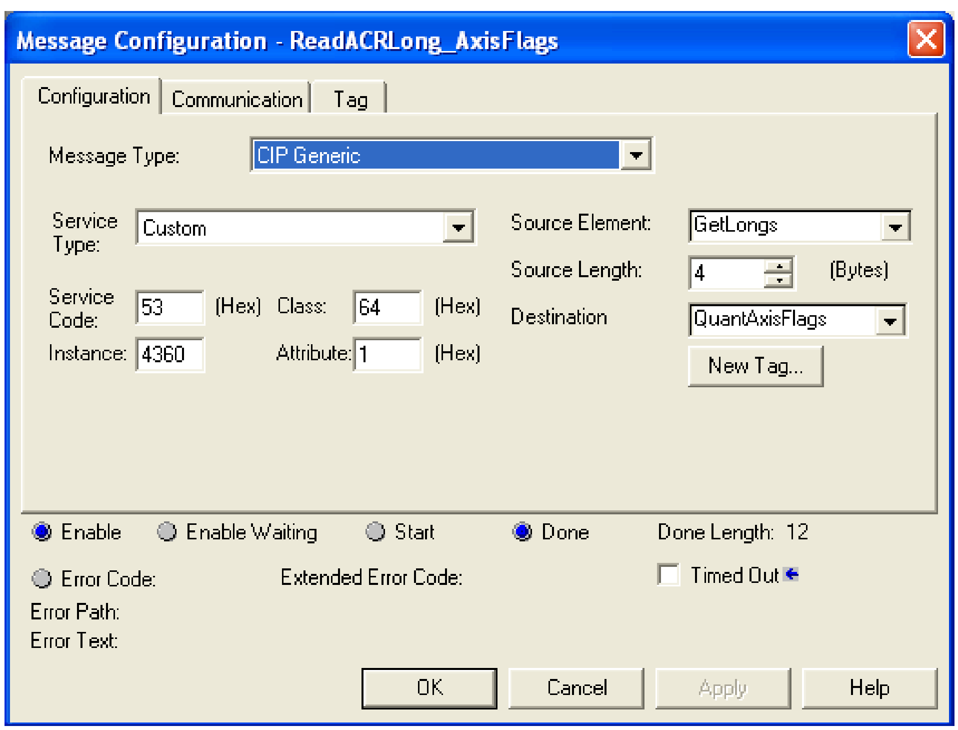
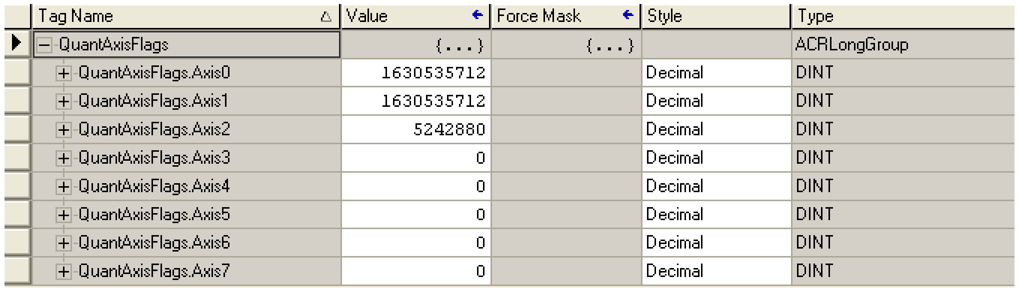


Figure 6: ReadACRLong_AxisFlags Message Configuration Example



Tag Name	Value	Force Mask	Style	Type
QuantAxisFlags	{...}	{...}		ACRLongGroup
QuantAxisFlags.Axis0	1630535712		Decimal	DINT
QuantAxisFlags.Axis1	1630535712		Decimal	DINT
QuantAxisFlags.Axis2	5242880		Decimal	DINT
QuantAxisFlags.Axis3	0		Decimal	DINT
QuantAxisFlags.Axis4	0		Decimal	DINT
QuantAxisFlags.Axis5	0		Decimal	DINT
QuantAxisFlags.Axis6	0		Decimal	DINT
QuantAxisFlags.Axis7	0		Decimal	DINT

Figure 7: ControlLogix Tags for ReadACRLong_AxisFlags

AND and OR Example

The service code 0x54 performs an AND OR operation on the P parameter named as the instance of class 0x64. Service code 0x54 requires only two pieces of data: the P parameter named as the instance of class 0x64 is first AND'ed with the first element in the Source Element data array, then OR'ed with the second element in the source data array. The Source Length field of the CIP Generic screen must be at least 8 in order to fully pass both masks as data to the 0x54 service request. A value less than 8 results in a vendor-specific error. The message configuration dialog box shown in Figure 8 uses class 0x64, service 0x54 to apply the two masks in the DINT array "And_Or_Masks" to P4111. P4111 is AND'ed with 0xffffff00, then OR'ed with 0x55, as shown in Figure 9.

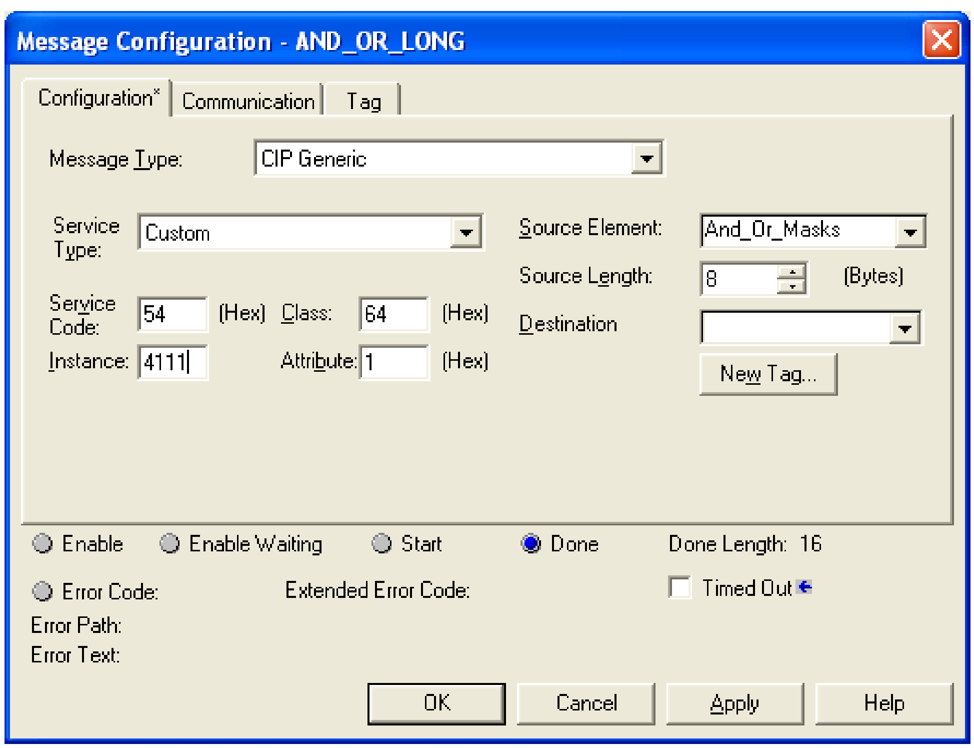


Figure 8: AND_OR_LONG Message Configuration Example

⊟ And_Or_Masks	{...}	{...}	Decimal	DINT[2]
⊞ And_Or_Ma...	16#ffff_ff00		Hex	DINT
⊞ And_Or_Ma...	16#0000_0055		Hex	DINT

Figure 9: Specifying Masks in a DINT Array

ACR Group Class (101)

Class 101 (0x65) is a vendor-specific class called "ACR Group" that takes advantage of the internal organization of ACR parameters as groups of similar parameters. Class 101 can be accessed via the CIP Generic function. Command message configuration specifics are shown here, followed by a discussion and an example.

Read Group – Service Code 0x53 (Class 0x65)

Message Type:	CIP Generic
Service Type:	Custom
Class:	0x65
Service Code:	0x53
Instance:	(group code x 256) + index
Attribute:	1

The service codes and the ACR Parameter class access ACR parameters as sequential P parameter numbers, for example, P12288, P12289, etc. Though, internally, ACR parameters are also organized as groups of similar parameters. As an example, the actual positions of axes 0 to 7 are accessed as group 0x30, index 2. Each pair of group and index is a separate instance of this class, so the group code and index combined defines the instance. The exact formula is:

Instance = (group code x 256) + index

It's important to remember that the group and index number are specified in hex in ACR user guides, but the instance is specified in decimal in the CIP Generic function. Internally, the ACR controller decodes the instance back into group and index. If the group code does not fall into a valid range, the service request results in a vendor-specific error. Valid ranges are:

Code < 0x80 or code ≥ 0xC0

For many kinds of parameters, such as axis, master, and object, this formula yields an instance number that is the same as the P parameter number of the first element in the group. For example, the actual positions of axes 0 to7 are accessed as group 0x30, index 2. Using the formula given previously:

Instance = (0x30 * 256) + 2 = 12290, axis 0 actual position P number

The service code used by the ACR Group class is 0x53. This service does not allow an exact mask specification like the corresponding binary ACR command, but instead allows a block size of between 1 and 8 parameters. A block size outside this range results in a vendor-specific error.

The Read Group service code is 0x53, which is also the service code for Class 100 service Read Long(s), but the meaning of the code changes when applied to Class 101. It will either return a group of REALs or a group of DINTs, depending on the data type of the ACR parameters specified by the group and index in the instance value. The data type of the destination array must match the data type of the ACR group and index.

Read Group Example

This example reads a group of parameters from the ACR controller. The block size to transfer is specified by the value of the controller tag named in the Source Element field of the CIP Generic screen. The actual data is placed in the controller tag named in the Destination field. The message configuration dialog box in Figure 10 uses class 0x65, service 0x53 to read a block of axis PGAINS starting with Axis0, P12304. The number of PGAINS to read is specified by the value of "block_size", which is a DINT. The length of a DINT is four bytes. The destination is the float array "Floats_l_read". If the value of "block_size" were 4, for example, the service would read PGAIN for axes 0, 1, 2, and 3.

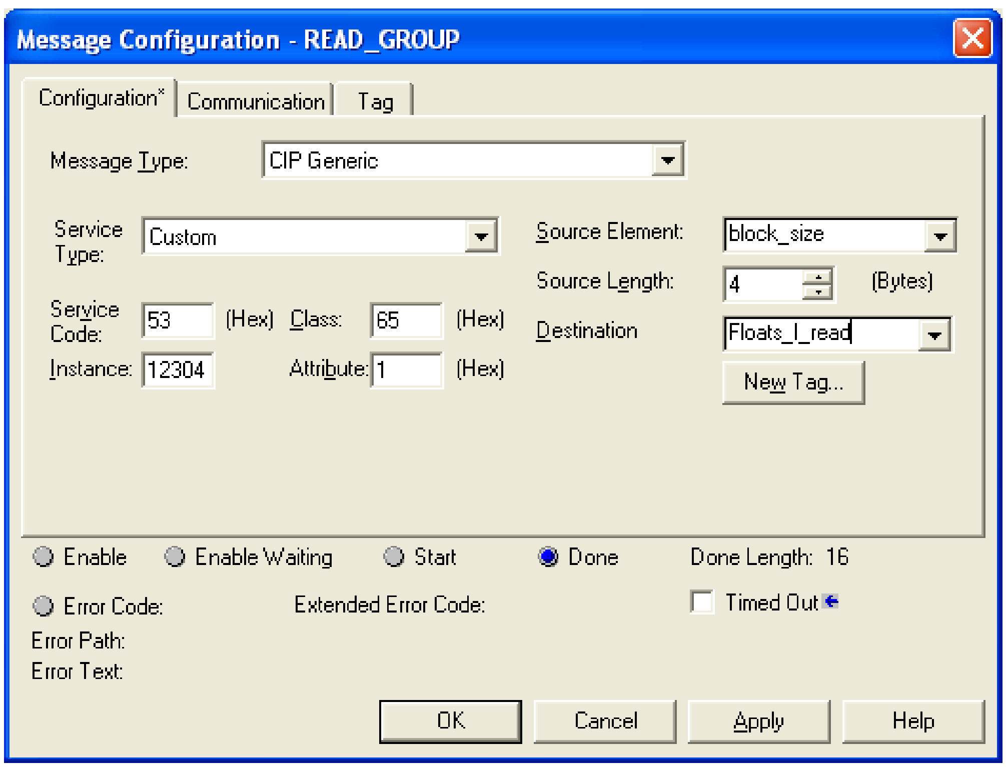


Figure 10: : READ_GROUP Message Configuration Example

Write Binary Command

The Write Binary command requires the user to properly form a valid binary packet. The ACR controller inspects the first byte of the incoming command to determine the expected length of the command. If the actual length does not match, the controller will not respond with an error and could either be left waiting for additional data or incorrectly interpret additional data as a new command.

Details of the available binary commands can be found in the *ACR Command Language Reference* and in the *ACR Programmer's Guide* chapter "Binary Host Interface." ACR binary commands are listed in Table 7 with an indication of their compatibility with the Write Binary message. If a service code is available, that method should be used instead of the raw binary commands.

Command	**Write Binary Compatible**	**Preferred Service Code Method**
Binary Data Packets	No	Class 101, Service Code 0x53
Binary Get Long	No	Class 100, Service Code 0x53
Binary Set Long	Yes	Class 100, Service Code 0x52
Binary Get IEEE	No	Class 100, Service Code 0x51
Binary Set IEEE	Yes	Class 100, Service Code 0x50
Binary Peek	No	n/a
Binary Poke	No	n/a
Binary Address	No	n/a
Binary Parameter Address	No	n/a
Binary Mask	No	n/a
Binary Parameter Mask	Yes	AND and OR, Service Code 0x54
Binary Move	Yes	n/a
Binary SET and CLR	Yes	n/a
Binary FOV	Yes	n/a
Binary ROV	Yes	n/a

Table 7: ACR Binary Commands

CIP Service Error Codes

All CIP service requests always return a "general status" and an "extended status." If all is well, the values of both are zero. If a problem exists, the general status indicates the basic error and the extended status offers detail specific to that general status. If no further detail is required, the extended status value is zero.

Table 8 provides the possible general status errors returned when a message to the ACR controller is rejected. It also shows the corresponding error case codes. Meanings for the error cases are listed after the table.

General Status Code (in hex)	**Status Name**	**Description of Status**	**Error Case**
05	Path destination unknown	The path is referencing an object class, instance, or structure element that is not known or is not contained in the processing	7

		node.	
08	Service not supported	The requested service was not implemented or was not defined for this Object Class/Instance.	1
10	Device state conflict	The device's current mode/state prohibits the execution of the requested service.	2
13	Not enough data	The service did not supply enough data to perform the specified operation.	5
15	Too much data	The service supplied more data than was expected.	4
20	Invalid parameter	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.	3, 6

Table 8: CIP General Status Error Codes

The Error Case codes shown in the last column of Table 8 are defined as follows:

1. The requested service is not implemented for the specified class.
2. No Class 3 connection exists.
3. A 0x4C, 0x51, or 0x53 service request has requested to read more than 16 blocks.
4. A 0x4D, 0x50, or 0x52 service request has requested to write more than 16 blocks.
5. Service request 0x54 (AND OR MASK) has provided less than 8 bytes of data, which is not enough to specify the two masks.
6. The number of parameters requested in the group request (class 0x65, service request 0x53) is less than one or more than eight.
7. This group code (instance) in the group request (class 0x65, service request 0x53) is not valid.

Glossary

Adapter	Device that receives a connection request or individual service request; on a network, one scanner may be connected to several adapters
Assembly	Pre-defined collection of data residing in an adapter characterized by size and type; three types are producing (data to be sent), consuming (data to be received), and configuration (a data area reserved for information about how consumed and produced data is to be interpreted)
Attribute Number	Identifies characteristics of an object in a service request
CLASS 1 Connection	Establishes a periodic exchange of data between a scanner and an adapter; indicates whether an adapter should send data point-to-point or multicast; connected messages
Class 3 Connection	Used for individual request/response transactions; handled in EtherNet/IP via TCP; unconnected messages
Class 3 Messages	Commands or data requests sent from the scanner to individual target nodes
Class Number	Identifies which type of object is being referenced in a service request
Configuration Data	Data area in an adapter reserved for information about how consumed and produced data is to be interpreted
Connection	Logical link between two devices; two devices may share more than one connection
Consuming Data	Type of data residing in an adapter that is to be received
Device	Product that supports EtherNet/IP, which may or may not have an industry standard profile or conform to one
Instance Number	Unique number that identifies an assembly; in a service request, defines which particular object of a class
Multicast	Request that sends data to a multicast address group, which can include a scanner

Object	Self-contained module of data and its associated processing
Path	Portion of a service request packet that defines the destination of the service request, which is either a literal ASCII character string or an object description
Point-to-Point	Request that transmits data to one point only (the scanner)
Producing Data	Type of data residing in an adapter that is to be sent
RPI	Requested Packet Interval, generally expressed in milliseconds; the interval of periodic exchange of data between the scanner and the adapter. Connection request from scanner establishes the repetition interval, or RPI, in both directions
Scanner	Device that initiates a connection or a request; master device
Service Code	One-byte identifier inside a service request packet; most have meanings pre-defined by the CIP specification, but some have meanings specific to the destination object of a service request
Service Request	Request (packet) sent from a scanner to an adapter
UCMM	Unconnected Message Manager - Manager in the internal stack that controls unconnected messages
Unconnected Messages	Messages for which no periodic Class 3 connection has been established; managed by the internal stack's Unconnected Message Manager (UCMM)