

## Driver Manual

# FS-8704-14 EtherNet/IP

### **APPLICABILITY & EFFECTIVITY**

Effective for all systems manufactured after February 2023.



Driver Revision: 1.12  
Document Revision: 5.C



**fieldserver**

MSA Safety  
1000 Cranberry Woods Drive  
Cranberry Township, PA 16066 USA  
Website: [www.MSAsafety.com](http://www.MSAsafety.com)

U.S. Support Information:  
+1 408 964-4443  
+1 800 727-4377  
Email: [smc-support@msasafety.com](mailto:smc-support@msasafety.com)

EMEA Support Information:  
+31 33 808 0590  
Email: [smc-support.emea@msasafety.com](mailto:smc-support.emea@msasafety.com)

## Contents

<b>1</b>	<b>Description</b> .....	<b>5</b>
<b>2</b>	<b>Driver Scope of Supply</b> .....	<b>5</b>
	2.1 Supplied by MSA Safety.....	5
<b>3</b>	<b>Hardware Connections</b> .....	<b>6</b>
<b>4</b>	<b>Data Array Parameters</b> .....	<b>7</b>
<b>5</b>	<b>Client Side Configuration</b> .....	<b>8</b>
	5.1 Client Side Connection Parameters.....	8
	5.2 Client Side Node Parameters.....	9
	5.3 Client Side Map Descriptor Parameters.....	9
	5.3.1 FieldServer Specific Map Descriptor Parameters.....	9
	5.3.2 Driver Related Map Descriptor Parameters.....	10
	5.3.3 Timing Parameters.....	12
	5.4 Map Descriptor Examples.....	13
	5.4.1 Unconnected Messages.....	13
	5.4.2 Data Table Messages.....	13
	5.4.3 PCCC Messages.....	13
	5.4.4 Implicit IO Messages.....	13
<b>6</b>	<b>Server Side Configuration</b> .....	<b>14</b>
	6.1 Server Side Connection Parameters.....	14
	6.2 Server Side Node Parameters.....	14
	6.3 Server Side Map Descriptor Parameters.....	15
	6.3.1 FieldServer Specific Map Descriptor Parameters.....	15
	6.3.2 Driver Related Map Descriptor Parameters.....	15
	6.4 Map Descriptor Examples.....	18
	6.4.1 Unconnected Messages.....	18
	6.4.2 Data Table Messages.....	18
	6.4.3 PCCC Messages.....	18
	6.4.4 Implicit IO Messages.....	18
<b>7</b>	<b>Useful Features</b> .....	<b>19</b>
	7.1 General Notes.....	19
	7.2 FieldServer as an Adapter and Scanner.....	19
	7.3 Common Paths.....	19
	7.4 Setting the Data Type for Stored Data.....	19
	7.4.1 Transfer Binary Values Using EIP.....	19
	7.5 Configuring a PLC to Read/Write Data to/from the FieldServer with Explicit Messaging.....	19
	7.5.1 FieldServer Configuration File.....	20
	7.5.2 The PLC Program.....	21
	7.6 Configuring a FieldServer to Read/Write Data to/from a PLC with Explicit Messaging.....	23
	7.6.1 FieldServer Configuration File.....	23
	7.6.2 The PLC Program.....	24
	7.7 Read/Write Structures and Value of EIP_Structure_Handle.....	24
	7.8 Implicit Messaging.....	25
	7.8.1 Configuring a PLC to Read and Write Data to and from the FieldServer.....	25
	7.8.2 FieldServer Configuration File.....	26
	7.8.3 The PLC Setup.....	27
	7.8.4 EDS AOP.....	28
	7.9 Configuring the PLC with a FieldServer EDS File.....	31
<b>8</b>	<b>Troubleshooting</b> .....	<b>36</b>
	8.1 Firmware Update Downloading.....	36

<b>9</b>	<b>Vendor Information .....</b>	<b>37</b>
9.1	ControlLogix.....	37
9.2	Connection Information – Allen Bradley Message Blocks.....	37
9.3	FieldServer Not Recognized by RSlinx .....	37
9.4	Using EIP Data_Types with RSLogix .....	37
<b>10</b>	<b>Reference .....</b>	<b>38</b>
10.1	Error Messages .....	38
10.2	Classes and Attributes Supported.....	39
10.3	Error Codes .....	41

## 1 Description

The EtherNet/IP (EIP) driver allows the FieldServer to transfer data to and from devices over Ethernet using the EtherNet/IP protocol. The FieldServer can emulate either a Server or Client.

EtherNet/IP uses CIP (Control and Information Protocol), the common network, transport and application layers. EtherNet/IP then makes use of standard Ethernet and TCP/IP technology to transport CIP communications packets. The result is a common, open application layer on top of open and highly popular Ethernet and TCP/IP protocols.

The Driver is able to read/write using the Data Table structure employed by all Logix Series PLC's.

PCCC support is also provided for legacy devices that do not fully support CIP encapsulation. EIP PCCC Encapsulation was tested at the MSA Safety factory using a PLC5 I785 ENET card. The following data types were tested:

- N
- F
- S

The Driver also supports PCCC communication on SLC and MicroLogix (Tested on MicroLogix 1400 Device).

Fragmented Services (0x52) is supported for data\_table read and write operations.

The driver supports only Logical Binary Addressing. The remote device should be configured to use Logical Binary Addressing.

The default EtherNet/IP TCP Port is 44818.

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

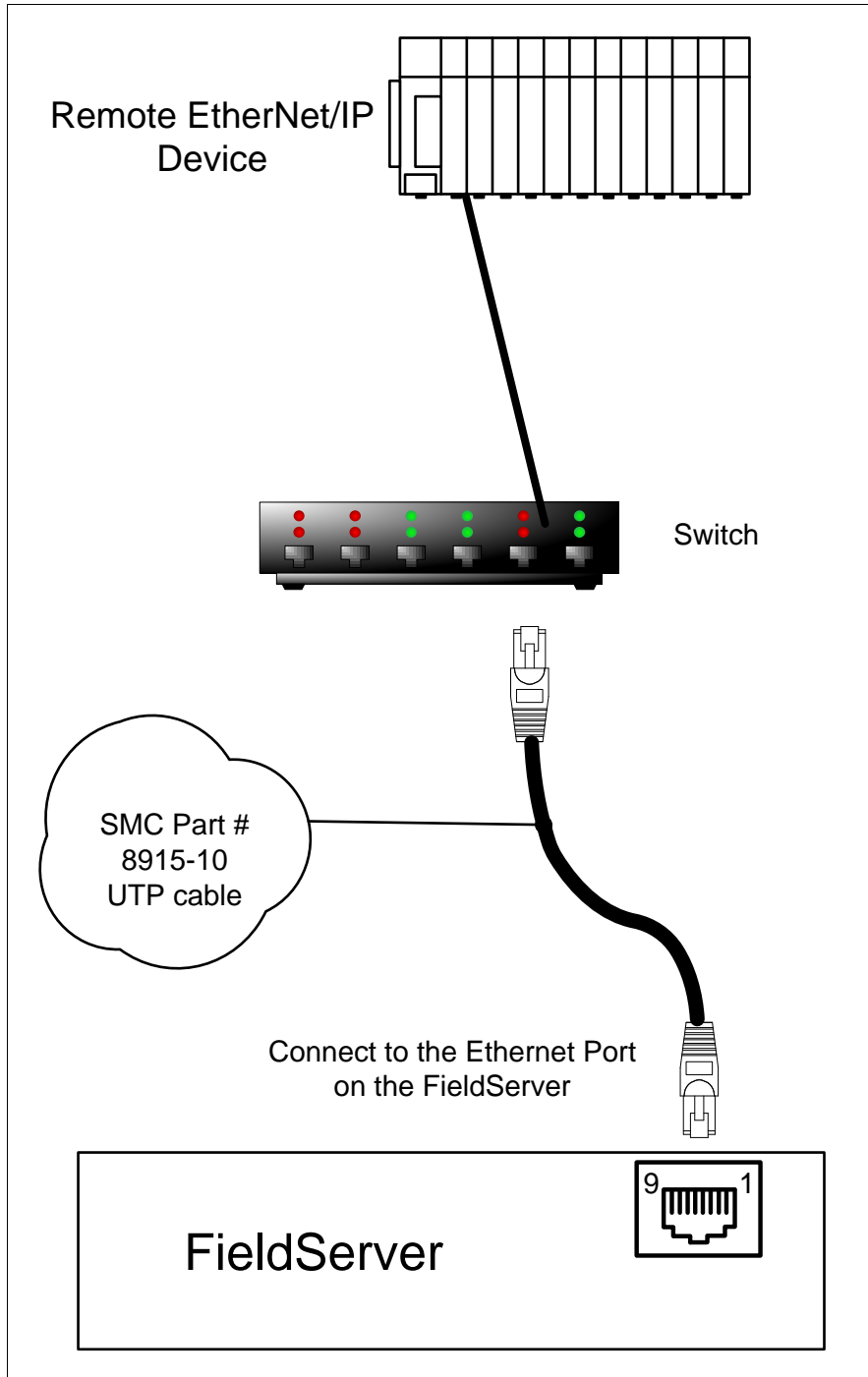
## 2 Driver Scope of Supply

### 2.1 Supplied by MSA Safety

Part #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection

## 3 Hardware Connections

The FieldServer is connected to the network as shown in connection drawing.



## 4 Data Array Parameters

Data Arrays are “protocol neutral” data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

Section Title		
Data_Arrays		
Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array.	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, Byte, UInt16, UInt32, Sint16, Sint32
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10000

### Example

```
// Data Arrays
Data_Arrays
Data_Array_Name , Data_Array_Format , Data_Array_Length
DA_AI_01 , UInt16 , 200
DA_AO_01 , UInt16 , 200
DA_DI_01 , Bit , 200
DA_DO_01 , Bit , 200
DA_OUTPUTS , UInt16 , 3
DA_INPUTS , UInt16 , 8
DA_Config , UInt16 , 1
```

## 5 Client Side Configuration

For detailed information on FieldServer configuration, refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (see “.csv” sample files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EtherNet/IP Server.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for EtherNet/IP communications, the driver independent FieldServer buffers need to be declared in the “Data Arrays” section, the destination device addresses need to be declared in the “Client Side Nodes” section, and the data required from the servers needs to be mapped in the “Client Side Map Descriptors” section. Details on how to do this can be found below.

**NOTE:** In the tables below, \* indicates an optional parameter and bold legal values are default.

### 5.1 Client Side Connection Parameters

Section Title		
Adapter		
Column Title	Function	Legal Values
Adapter	Adapter Name.	N1, N2, WLAN <sup>1</sup>
Protocol	Specify protocol used.	EtherNet/IP
UDP_Local_IO_Port	Specify if it is required to receive implicit IO messages on the non-default port.	1-65534, <b>2222</b>

#### Example

```
// Client Side Connections
Connections
Adapter , Protocol , UDP_Local_IO_Port
N1 , EtherNet/IP , 2222
```

<sup>1</sup> Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.



## 5.2 Client Side Node Parameters

Section Title		
Nodes <sup>2</sup>		
Column Title	Function	Legal Values
Node_Name	Provide name for node.	Up to 32 alphanumeric characters.
IP_Address	Address of Server.	Any valid address on subnet.
Protocol	Specify protocol used.	EtherNet/IP
Adapter	Specify port Adapter used.	N1, N2, WLAN <sup>3</sup>
Plc_Type*	Specify the type of remote PLC. Only required for PCCC Services.	MicroLogix, SLC5, <b>PLC5</b>
EIP_Cache_Connection*	Specify if connection to remote server should be reused or closed after every data transfer transaction.	<b>YES, NO</b>
EIP_Skip_Cmd*	Some devices do not support all 'under the hood' commands specified by the EIP driver. Specify the command that should not be issued by the FieldServer.	-, LIST_IDENTIFY

### Example

```
// Client Side Nodes
Nodes
Node_Name , IP_Address , Adapter , Protocol
PLC 1 , 192.168.1.174 , N1 , EtherNet/IP
```

## 5.3 Client Side Map Descriptor Parameters

### 5.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor.	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer.	One of the Data Array names from <b>Section 4</b>
Data_Array_Offset	Starting location in Data Array.	0 to (Data_Array_Length -1) as specified in <b>Section 4</b>
Function	Function of Client Map Descriptor.	Rdbc, Wrbc, Wrbx <b>NOTE:</b> It is possible with Data Table Read on the Client side to read and write to the same tag by using the "Write Thru" property of the Rdbc function.

<sup>2</sup> Only one explicit connection is created per node. All explicit Map Descriptors for that node will use the same explicit connection.

<sup>3</sup> Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

## 5.3.2 Driver Related Map Descriptor Parameters

### Unconnected Messages

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Get_Attrib, Set_Attrib
EIP_Class	Class to be polled.	0-65535, default 0 (refer to <b>Section 10.2</b> for commonly used classes)
EIP_Attribute	Attribute associated with the class given.	0-255, default 0 (refer to <b>Section 10.2</b> for commonly used attributes)
EIP_Con_Typ	The type of data transfer required. Also referred to as the "Transport Method".	Unconnected, Explicit
EIP_Path*	Used to stipulate the path to the CPU in certain PLC's. Paths vary and are dependent on the structure of the network.	Any space delimited numerical value. Refer to vendor's device documentation. ( <b>Section 7.3</b> )
EIP_Data_Type	This parameter should be used to force the data type of the attribute to match the data type used in the remote device.	Value Alias (as used in PLC) Float REAL Uint32 DINT Uint16 INT BYTE SINT default INT
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to <b>Section 10</b> for further information.	For any given Map Descriptor there can be 200 Floats, 400 Integers or 800 Bytes.
Address	Instance of the class to be polled.	0-65535, default 0

### Data Table Read/Write

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Data_Table_Read, Data_Table_Write
EIP_Con_Type	The type of data transfer required.	Explicit, UnConnected
EIP_Path*	Used to stipulate the path to the CPU in certain PLC's. Paths vary and are dependent on the structure of the network.	Any space delimited numerical value. Refer to vendor's device documentation. ( <b>Section 7.3</b> )
EIP_Tag_Name	Tag name expressed in PLC program. The data type of this parameter is used to set the data format of the Data Array if the EIP_DATA_TYPE parameter is not specified.	Maximum length 48 characters
EIP_Data_Type*	This parameter can be used to force the data type of the tag to match the data type used in the remote device. If this parameter is not specified the Data Type of the Data Array will be used. Refer to <b>Section 7.4</b> for more information. Data Types can be specified in either FieldServer or Rockwell Data Type. Refer to <b>Section 9.4</b> for more information.	Value Alias (as used in PLC) Float REAL Uint32 DINT Uint16 INT BYTE SINT BIT BOOL & BOOLEAN
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. See <b>Section 10</b> for further information.	0 to (Data_Array_Length -1) as specified in <b>Section 4</b>
EIP_Structure_Handle*	This parameter is required to read/write structures. The driver supports read/write structures having members of same type, i.e. all members are of type Byte, UINT16, UINT32 or Float etc. When this parameter is defined, the number of structure members must be specified as the length of the Map Descriptor. Refer to <b>Section 7.7</b> for more information.	Any 16bit Integer number (e.g. 59592), <b>0</b>

## PCCC

Column Title	Function	Legal Values
EIP_Service	Action to be performed.	Exec_PCCC (encapsulation using Allen Bradley PCCC)
EIP_Con_Typ	The type of data transfer required.	Explicit
EIP_Path*	Used to stipulate the path to the CPU in certain PLC's. Paths vary and are dependent on the structure of the network.	Any space delimited numerical value. Refer to vendor's device documentation. Also see <b>Section 7.3, 0 0</b> .
File_Type	Allen Bradley file type.	N Integer F Float O Output B Boolean I Input S Status
File_Number	Allen Bradley file number.	Any valid numerical value
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to <b>Section 10</b> for further information.	For any given Map Descriptor there can be 200 floats, 400 integers or 800 bytes
Address	Address in the file.	Any valid numerical value between 0 to 255

## Implicit IO Messages

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required. Also referred to as the "Transport Method".	Implicit
EIP_Class	EIP class to be polled.	Integer value. In most cases it will be the assembly class (i.e. value will be 4)
Address	Production/Consumption connection point or Instance of the class.	Integer value depending upon server configuration
Length	Number of data items to be polled.	For any given Map Descriptor there can be 125 floats/32bit integers, 250 integers or 500 bytes
Parent_Map_Descriptor	Specify the name of previously created Map Descriptor to which this Map Descriptor should be linked.	Use – for no setting, or specify name of other Map Descriptor
EIP_Real_Time_Format	Specify if real time format is 'HeartBeat'.	Heartbeat, -

### 5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval*	Rate at which data is polled.	≥0.001s

## 5.4 Map Descriptor Examples

### 5.4.1 Unconnected Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Scan_Interval , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Class , Address , EIP_Attribute , EIP_Service , Length
CMD_PRO_03 , 0s , DA_AI_01 , 0 , Rdbc , Unconnected , EIP_01 , 10 , 1 , 3 , Get_Attrib , 1
CMD_PRO_02 , 0s , DA_AI_01 , 1 , Rdbc , Unconnected , EIP_01 , 10 , 2 , 3 , Get_Attrib , 1
```

### 5.4.2 Data Table Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Scan_Interval , Data_Array_Name , Data_Array_Offset , EIP_Con_Typ , Node_Name , Function , EIP_Service , EIP_Path , EIP_Tag_Name , Length
Cmd_Pro_09 , 0s , DA_AI_05 , 0 , Explicit , EIP_01 , Rdbc , Data_Table_Read , 1 1 , analog_in_3 , 2
Cmd_Pro_10 , 0s , DA_AI_06 , 0 , Explicit , EIP_01 , Rdbc , Data_Table_Read , 1 1 , analog_in_4 , 2
```

### 5.4.3 PCCC Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Service , EIP_Path , File_Type , File_Number , Address , Length
CMD_01 , DA_F_01 , 0 , Rdbc , Explicit , EIP_01 , Exec_PCCC , 1 0 , F , 8 , 30 , 10
```

### 5.4.4 Implicit IO Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Scan_Interval , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Class , Address , Length , Parent_Map_Descriptor , EIP_Real_Time_Format
CMD_Producer , 1s , DA_OUTPUTS , 0 , Wrbc , Implicit , EIP_01 , 4 , 101 , 3 , - , -
CMD_Consumer , 1s , DA_INPUTS , 0 , Passive , Implicit , EIP_01 , 4 , 102 , 8 , CMD_Producer , -
CMD_Config , - , DA_Config , 0 , Passive , Implicit , EIP_01 , 4 , 103 , 1 , CMD_Producer , -
```

#### Example comments:

- **Map\_Descriptor\_Name** – The config Map Descriptor (Rockwell calls it Configuration) is optional for FieldServer (as per EIP specs), but it is used by Controllogix and technically it is an instance of Assembly Class.
- **Address** – The address numbers are random but should be according to Server information; at what numbers server will consume and produce data.
- **Parent\_Map\_Descriptor** – To open an implicit connection, both Producer and Consumer connection information is required. Producer is an active Map Descriptor, so consumer should be linked to producer.
- **EIP\_Real\_Time\_Format** – **If not used or set to “-“:** The FieldServer and remote server both act as producer/ consumer endpoints. **If set to 'Heartbeat' on wrbc MD:** The FieldServer acts as a consumer only endpoint that produces the heartbeat; the remote server acts as a producer only end point. **If set to 'HeartBeat' on passive MD:** The FieldServer acts as a producer only endpoint and remote server acts as a consumer only endpoint that generates a heartbeat.

## 6 Server Side Configuration

For detailed information on FieldServer configuration, refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (see “.csv” files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EtherNet/IP Client.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for EtherNet/IP communications, the driver independent FieldServer buffers need need to be declared in the “Data Arrays” section, the FieldServer virtual node(s) needs to be declared in the “Server Side Nodes” section, and the data to be provided to the client needs to be mapped in the “Server Side Map Descriptors” section. Details on how to do this can be found below.

**NOTE: In the tables below, \* indicates an optional parameter, with the bold legal value as default.**

### 6.1 Server Side Connection Parameters

Section Title		
Connections		
Column Title	Function	Legal Values
Adapter	Adapter Name	N1, N2, WLAN <sup>4</sup>
Protocol	Specify protocol used.	EtherNet/IP
UDP_Local_IO_Port	Specify if it is required to receive implicit IO messages on the non-default port.	1-65534, <b>2222</b>

#### Example

```
// Server Side Connections
Connections
Adapter , Protocol , UDP_Local_IO_Port
N1 , EtherNet/IP , 2222
```

### 6.2 Server Side Node Parameters

Section Title		
Nodes <sup>5</sup>		
Column Title	Function	Legal Values
Node_Name	Provide name for Node.	Up to 32 alphanumeric characters
Protocol	Specify protocol used.	EtherNet/IP
Server_Hold_Timeout*	Specifies time FieldServer will reserve server side connection while waiting for the Client side to update data.	>1.0s
IP_Address	Specify multicast IP Address to which the production data should be sent. It will be used only if client requested multicast connection. This parameter is only required for Implicit IO Messages.	IP Address

#### Example

```
// Server Side Nodes
Nodes
Node_Name , Protocol
EIP_01 , EtherNet/IP
```

<sup>4</sup> Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

<sup>5</sup> Only one explicit connection is created per node. All explicit Map Descriptors for the node use the same explicit connection.

## 6.3 Server Side Map Descriptor Parameters

### 6.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor.	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer.	One of the Data Array names from <b>Section 4</b>
Data_Array_Offset	Starting location in Data Array.	0 to maximum as specified in <b>Section 4</b>
Function	Function of Server Map Descriptor.	Passive
Server_Hold_Timeout*	Specifies the length of time that the FieldServer will reserve the Server side connection while waiting for the Client side to update data in Data Array (if necessary).	>1.0s

### 6.3.2 Driver Related Map Descriptor Parameters

#### Unconnected Messages

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Get_Attrib, Set_Attrib
EIP_Class	Class to be served.	0-65535, default 0 (refer to <b>Section 10.2</b> for commonly used classes)
EIP_Attribute	Attribute associated with the class served.	0-255, default 0 (refer to <b>Section 10.2</b> for commonly used attributes)
EIP_Data_Type	This parameter should be used to force the data type of the attribute to match the data type used in the remote device.	Value Alias (as used in PLC) Float REAL Uint32 DINT Uint16 INT BYTE SINT default INT
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to <b>Section 10</b> for further information.	For any given Map Descriptor there can be 200 floats, 400 integers or 800 bytes
Address	Instance of the class to be served.	0-65535, default 0

### Data Table Read/Write

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Data_Table_Read, Data_Table_Write <b>NOTE: Separate Map Descriptors need to be configured for each service.</b>
EIP_Tag_Name	Tag name expressed in PLC program. The data type of this parameter is used to set the data format of the Data Array if the EIP_Data_Type parameter is not specified.	Maximum length 48 characters
EIP_Data_Type*	If set, the data will be stored in the specified format which may be different to the format of the tag being polled. If the parameter is not set, the data type of the Data Array will be used. This is only applicable to Data Table Read when FieldServer is the Server. Refer to <b>Section 7.4</b> for further information.	Float, Uint16, Uint32, Bit, Byte, Boolean, -
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to <b>Section 10</b> for further information.	0 to (Data_Array_Length -1) as specified in <b>Section 4</b>
EIP_Structure_Handle*	This parameter is required only for read structures i.e. where EIP_Service is Data_Table_Read. The driver supports read structures having members of same type, i.e. all members are of type Byte, Uint16, Uint32 or Float, etc. When this parameter is defined, the number of structure members must be specified as the length of the Map Descriptor.	Any 16bit Integer number e.g. 59592, <b>0</b>



### PCCC

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required.	Explicit
EIP_Service	Action to be performed.	EXEC_PCCC (encapsulation using Allen Bradley PCCC)
File_Type	Allen Bradley file type.	N Integer F Float O Output B Boolean I Input S Status
File_Number	Allen Bradley file number.	Any valid numerical value
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to <b>Section 10</b> for further information.	For any given Map Descriptor there can be 61 floats, 122 integers or 244 bytes
Address	Address in the file.	Any valid numerical value between 0 to 255

### Implicit IO Messages

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required. Also referred to as the "Transport Method".	Implicit
EIP_Class	Class to be served.	Integer value (in most cases it will be the assembly class i.e. 4)
EIP_Attribute	Attribute associated with the class served.	Integer value. In most cases it will be data attribute 3. Refer to <b>Section 10.2</b> for legal attribute values
Length	Number of data elements to be mapped.	1 to Data Array length
Address	Production or Consumption connection point address.	Any integer number
EIP_IO_Type	Input means the FieldServer consumes data and Output means the FieldServer produces data.	Input, <b>Output</b>

## 6.4 Map Descriptor Examples

### 6.4.1 Unconnected Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , EIP_Class , Address , EIP_Attribute , EIP_Service , Length
SMD_PRO_01      , DA_AI_01      , 0      , Passive , EIP_01      , 10      , 1      , 3      , Get_Attrib , 1
SMD_PRO_02      , DA_AI_01      , 1      , Passive , EIP_01      , 10      , 2      , 3      , Get_Attrib , 1
```

### 6.4.2 Data Table Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , Function , EIP_Service , EIP_Tag_Name , Length
SMD_PRO_09      , DA_AI_05      , 0      , EIP_01      , Passive , Data_Table_Read , Analog_in_3 , 2
SMD_PRO_10      , DA_AI_06      , 0      , EIP_01      , Passive , Data_Table_Read , Analog_in_4 , 2
```

### 6.4.3 PCCC Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Service , File_Type , File_Number , Address , Length
SRV_AI_01      , DA_F_01      , 0      , Passive , Explicit , EIP_01      , Exec_PCCC , F      , 8      , 30      , 20
```

### 6.4.4 Implicit IO Messages

```
// Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , EIP_CON_TYP , Function , EIP_CLASS , Address , EIP_ATTRIBUTE , Length , EIP_IO_Type
MD_CONSUMER      , DA_INPUTS      , 0      , EIP_01      , IMPLICIT , Server , 4      , 101      , 3      , 100      , Input
MD_PRODUCER      , DA_OUTPUTS      , 0      , EIP_01      , IMPLICIT , Server , 4      , 102      , 3      , 100      , Output
```

## 7 Useful Features

### 7.1 General Notes

- The connection type does not need to be specified in the Server side Map Descriptor, but must be stipulated on the Client side of the driver.
- Data\_Table\_Read as a service can only be used when creating an explicit connection.

### 7.2 FieldServer as an Adapter and Scanner

It is possible for the FieldServer to act as a scanner and an adapter at the same time so long as the scanner and adapter are configured on different ports. Consequently, this functionality is not possible on the QuickServer platform.

### 7.3 Common Paths

Device	Typical Path
Direct AB	1 0
AB ENI module	3 1
AB ControlLogix 1756-L55 (With network card 1756-ENBT/A)	1 1 or 1 0
CompactLogix ENI (1769-L31 using the 1761-NET-ENI)	3 1
CompactLogix Direct Connection (P/N 1769-L35E)	1 1 or 1 0
If not required by remote device (ex. Rockwell Micro820SD Controller)	Not_used

### 7.4 Setting the Data Type for Stored Data

The default Data Type of stored data is determined by the Data Type of the Data Array. It is possible to configure the driver to store the data as a different type. This can be achieved by specifying the data type under the parameter EIP\_Data\_Type.

**NOTE: The EIP\_DATA\_TYPE parameter has meaning only for DATA\_TABLE\_WRITE where the FieldServer is the Client and for DATA\_TABLE\_READ where the FieldServer is the Server.**

#### 7.4.1 Transfer Binary Values Using EIP

The standard FieldServer data types do not follow the same naming convention as Rockwell. However, it is possible to enter the data types in the EIP\_Data\_Type field in either the FieldServer or the Rockwell format.

EIP\_DATA\_TYPE should be specified as Bit, BOOL or BOOLEAN.

Set the Map Descriptor length to 1 for the driver to transfer 1 bit. If a length other than 1 is specified then bits will be transferred in 32 bit words.

### 7.5 Configuring a PLC to Read/Write Data to/from the FieldServer with Explicit Messaging

This example makes use of the Data Table Read/Write method for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all Rockwell PLC's that support EtherNet/IP communications and Data Table Read/Write operations in Message blocks should be able to communicate this way. The Map Descriptors create an explicit connection to the Server and then transfer data in the data table format. The EIP\_Tag\_Name field contains the tag name polled from the client. DATA\_TABLE\_READ and DATA\_TABLE\_WRITE are the only legal values for EIP\_SERVICE.

## 7.5.1 FieldServer Configuration File

The configuration file used for this example is configured with the following Connection, Node and Map Descriptor Parameters:

```
// Data Arrays
//
Data_Arrays
Data_Array_Name , Data_Format6 , Data_Array_Length
DA_Read , Float , 100
DA_Write , Float , 100
```

```
// Server Side Connections
//
Connections
Adapter , Protocol , Turnaround_delay
N1 , EtherNet/IP , 0.01s
```

```
// Server Side Nodes
//
Nodes
Node_Name , Protocol
EIP_01 , EtherNet/IP
```

```
// Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , Function , EIP_SERVICE , EIP_TAG_NAME , Length
FS_TO_PLC_DATA , DA_Read , 0 , EIP_01 , Passive , DATA_TABLE_READ , Read_Data , 20
PLC_TO_FS_DATA , DA_Write , 0 , EIP_01 , Passive , DATA_TABLE_WRITE , Write_Data , 20
```

### Example comments:

- EIP\_TAG\_NAME – FieldServer Tag names that will be called in the PLC Message Block. The names must match what is written in the Message block in the PLC exactly.

**NOTE: The corresponding PLC Tag Name can be different and probably will be. See Message Block Below.**

- Length – Number of data points made available for reading or writing within the Tag.

<sup>6</sup> The Data Format should match the Data Type used in the PLC program and tags to prevent a mismatch in the display of the Data Values.

## 7.5.2 The PLC Program

The PLC program example below shows the minimum steps necessary to program communications with the FieldServer. Depending on the real intended application, additional steps may be necessary for completeness.

**Step 1:** Configure tags in the PLC for storing FieldServer read and write data.

Name	Alias For	Base Tag	Data Type	Style	Descrip
FS_Status			REAL[76]	Float	
FS_Control			REAL[76]	Float	

Tag length must be equal to or greater than the number of points being written/read.

Avoid using UDT Types. The data will be read but the exact placement of the data in the tags and arrays will be hard to determine.

**Step 2:** Configure message tags for storage of message block data.

MSG_01	MESSAGE
MSG_02	MESSAGE

MESSAGE data type must be used.

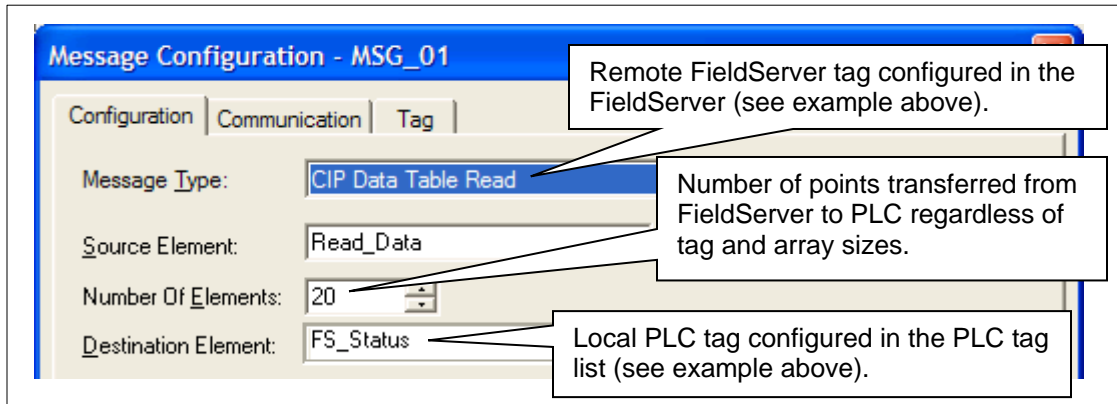
**Step 3:** Write ladder logic to exercise a read and write message block.

The logic shown will cause the PLC to poll the FieldServer at a very high speed. This may overload network traffic - logic that schedules the communication at a slower rate is generally more advisable.

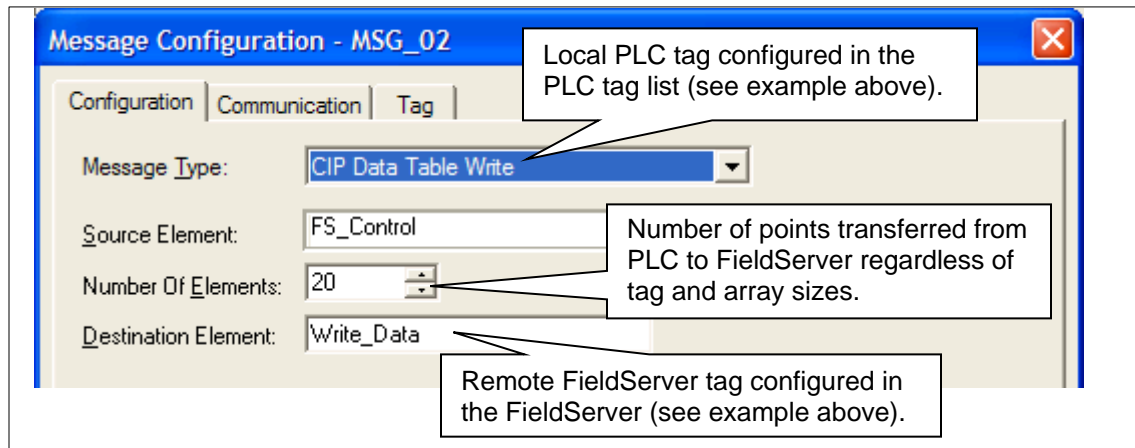
**NOTE:** It is important to use the ladder logic to ensure that only one message block can be executing at any time.

**Step 4:** Configure the properties for the two message blocks by clicking on the “...” button.

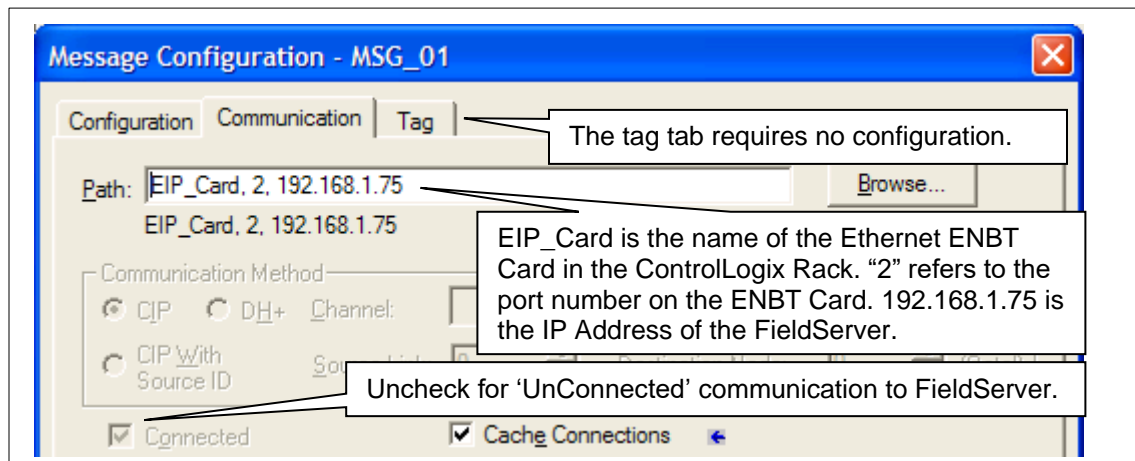
## Read Message Block



## Write Message Block



## Communication Tab (for both message blocks)



## NOTE:

- A message block only executes on the leading edge of a rung condition.
- No more than one message block executes at a time.

**Step 5:** Download the program and set the PLC to Run Mode.

## 7.6 Configuring a FieldServer to Read/Write Data to/from a PLC with Explicit Messaging

This example makes use of the Data Table Read/Write method for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all RockWell PLC's that support EtherNet/IP communications and Data Table Read/Write operations in Message blocks should be able to communicate this way. These map descriptors will create an explicit connection to the server and will then transfer data in the data table format. The EIP\_Tag\_Name field contains the tag name referenced in the server and the EIP\_Path field represents the path (through different ports) to the server. Each port jump is separated by a space. This field generally holds a backplane/cpu slot combination. DATA\_TABLE\_READ and DATA\_TABLE\_WRITE are the only legal values for EIP\_SERVICE.

Note that this is by far the preferred method for communicating with Allen Bradley PLC's due to its ease of configuration, quantity of data that can be transferred and speed of transfer.

When the FieldServer is the active component as shown below (i.e: the FieldServer Polls the PLC and not the other way around), then very little programming is needed in the PLC, other than the Tag Creation and setting the PLC IP Address.

### 7.6.1 FieldServer Configuration File

```
// Data Arrays
//
Data_Arrays
Data_Array_Name , Data_Format , Data_Array_Length
DA_Read , Float , 50
DA_Write , Float , 50
```

```
// Client Side Connections
//
Connections
Adapter , Protocol
N1 , EtherNet/IP
```

```
// Client Side Nodes
//
Nodes
Node_Name , IP_Address , Protocol , Adapter
EIP_01 , 192.168.1.9 , EtherNet/IP , N1
```

```
// Client Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name , Scan_Interval , Data_Array_Name , Data_Array_Offset , EIP_CON_TYP , Node_Name , Function , EIP_SERVICE , EIP_PATH , EIP_TAG_NAME , Length
PLC_TO_FIELDSERVER , 0.1 , DA_Read , 0 , EXPLICIT , EIP_01 , Rdbc , DATA_TABLE_READ , 1 0 , FS_Status , 20
FIELDSERVER_TO_PLC , 0.1 , DA_Write , 0 , EXPLICIT , EIP_01 , Wrbc , DATA_TABLE_WRITE , 1 0 , FS_Control , 20
```

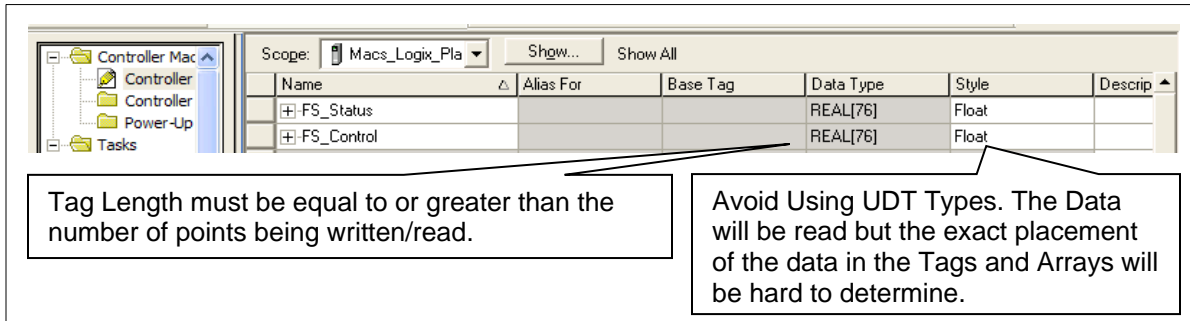
#### Example comments:

- EIP\_PATH – **Section 7.3** lists the paths for specific devices.
- EIP\_TAG\_NAME – These are the PLC Tag names that will be accessed in the PLC. The names must match the PLC tag name exactly.

## 7.6.2 The PLC Program

The PLC program example below shows the minimum steps necessary to program communications with the FieldServer. Depending on the real intended application, additional steps may be necessary for completeness.

When the FieldServer is polling the PLC, all that is needed is to configure the tags being accessed.



Name	Alias For	Base Tag	Data Type	Style	Descrip
FS_Status			REAL[76]	Float	
FS_Control			REAL[76]	Float	

Tag Length must be equal to or greater than the number of points being written/read.

Avoid Using UDT Types. The Data will be read but the exact placement of the data in the Tags and Arrays will be hard to determine.

**NOTE: Providing dedicated (long), flat structured (not UDT) tags for communications interface to the FieldServer is preferable to polling single length tags in the PLC as it allows for much more efficient communications and reduces complexity when mapping data in the FieldServer. It is better to think of these tags as a “I/O Buffer” Interface that the real tags in the PLC get mapped to.**

## 7.7 Read/Write Structures and Value of EIP\_Structure\_Handle

Some devices require that a specific value be used for the EIP\_Structure\_Handle field while writing the structure to them. This value may be specified in the vendor documentation. If any non-zero integer is used in a Read Map Descriptor for EIP\_Structure\_Handle, the value will be updated internally. The Map Descriptor can then be browsed to obtain this value. Other devices do not validate this field when the structure is written by the third-party device. A summary of the procedures to obtain this value is presented below.

### FieldServer as a Client:

- Read
  - Use a value of 1, the driver will automatically update the field when a response is received from the device; the Map Descriptor can then be browsed to obtain the value if required
- Write
  - Use the value supplied by the vendor OR
  - Use the value obtained in the Read Map Descriptor above OR
  - Use any non-zero value if the other device doesn't validate it

### FieldServer as a Server:

- Read
  - Use the value supplied by the vendor OR
  - Use the value obtained in the Read Map Descriptor above OR
  - Use any value if the other device doesn't validate it
- Write
  - Not required – use any non-zero value if the parameter is specified



**Example:** Consider a situation where a customer defines a type in RSlogix SSS\_SINT3 with 3 members of each type SINT.

```
SSS_SINT3
  SINT room1_temp
  SINT room2_temp
  SINT room3_temp
```

Now he has his own type SSS\_SINT3 and he can define tags: TAG\_3ROOM\_TEMPS of type SSS\_SINT3

```
// Read/write structures

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , EIP_CON_TYP , Node_Name , EIP_SERVICE
CMD_Struct_SINT3   , DA_STRUCT_R   , 0           , Rdbc   , EXPLICIT   , EIP_01   , DATA_TABLE_READ

, EIP_Path , EIP_TAG_NAME   , EIP_Structure_Handle , Length , Scan_Interval
, 1 0     , TAG_3ROOM_TEMPS , 59592           , 3     , 1.0s
```

## 7.8 Implicit Messaging

### 7.8.1 Configuring a PLC to Read and Write Data to and from the FieldServer

Implicit messaging uses the User Datagram Protocol (UDP/IP) to produce/consume data over the EtherNet/IP network. Implicit messages are not regular poll/response messages. Implicit connections are time critical, scheduled and use a requested packet interval (RPI) to specify the rate at which data updates.

This example makes use of the Implicit messaging (I/O) for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all Rockwell PLC's that support EtherNet/IP communications and implicit (I/O) messaging should be able to communicate this way. When the PLC is configured, it will open a connection to the FieldServer.

The Map Descriptors accept a connection from the PLC to consume and produce data at the rate requested by the PLC. The data produced by the PLC will be consumed by Map Descriptor 'MD\_CONSUMER' and data produced by Map Descriptor 'MD\_PRODUCER' will be consumed by the PLC.

Download the fserver.eds file from the FieldServer by entering the IP Address into the address bar of your web browser, followed by "/fserver.eds" (i.e. "192.168.2.101/fserver.eds"). Save the file as "fserver.eds". Use this file with the PLC setup instructions in **Section 7.8.3**.

## 7.8.2 FieldServer Configuration File

The configuration file used for this example is configured with the following Connection, Node and Map Descriptor Parameters.

```
// Data Arrays
//
Data_Arrays
Data_Array_Name , Data_Format , Data_Array_Length
DA_INPUTS      , Uint16      , 100
DA_OUTPUTS     , Uint16      , 100
```

```
// Server Side Connections
//
Connections
Adapter      , Protocol
N1          , EtherNet/IP
```

```
// Server Side Nodes
//
Nodes
Node_Name    , Node_ID    , Protocol    , IP_Address
EIP_01      , 1          , EtherNet/IP , 192.168.1.60
```

### Example comments:

- IP\_Address – IP Address is multicast IP Address and will be used only if EIP client requests FS to create multicast production connection.
- EIP\_TAG\_NAME – These are the PLC Tag names that will be accessed in the PLC. The names must match the PLC tag name exactly.

```
// Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , EIP_CON_TYP , Function , EIP_CLASS , Address , EIP_ATTRIBUTE , Length , EIP_IO_Type
MD_CONSUMER        , DA_INPUTS      , 0                  , EIP_01   , IMPLICIT    , Server   , 4          , 101     , 3         , 100      , Input
MD_PRODUCER        , DA_OUTPUTS     , 0                  , EIP_01   , IMPLICIT    , Server   , 4          , 102     , 3         , 100      , Output
```

### Example comments:

- EIP\_CON\_TYP – Connection type set to Implicit.
- Length – Number of data points to be transferred.

## 7.8.3 The PLC Setup

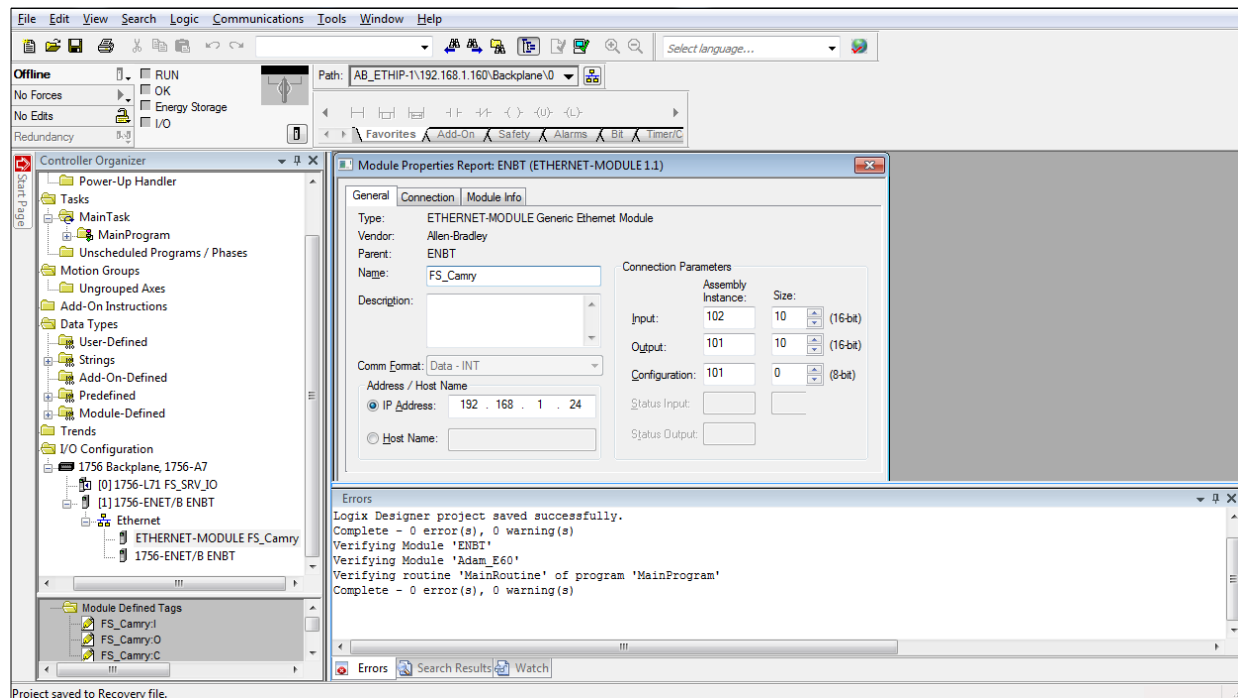
The PLC (192.168.1.60) should be connected to the FieldServer (192.168.1.24).

To start Implicit Communications, use the following 3 steps:

- Add a “Generic Ethernet Module”.
- Configure the Connection Parameters as per the screenshot below.
- Enter the FieldServer’s IP Address.

The PLC will produce and consume 10 16bit words, but the FieldServer is configured for 100 words in case more data is needed. The FieldServer will produce only as much data as requested by the PLC during session set up.

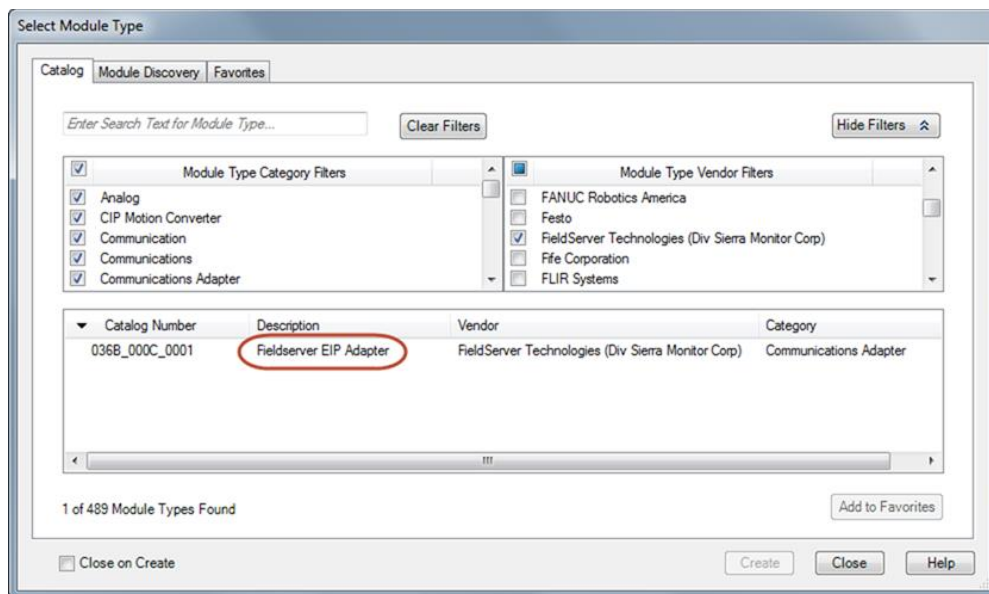
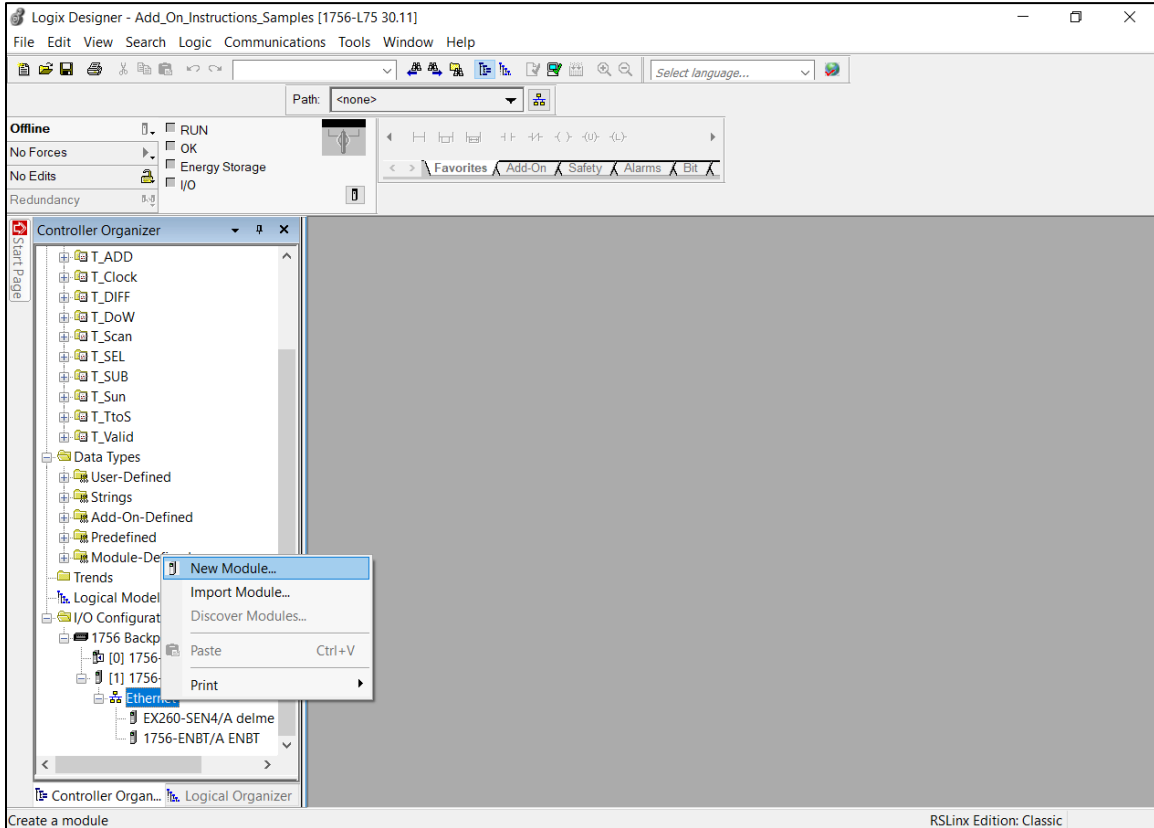
No further PLC programming is needed.



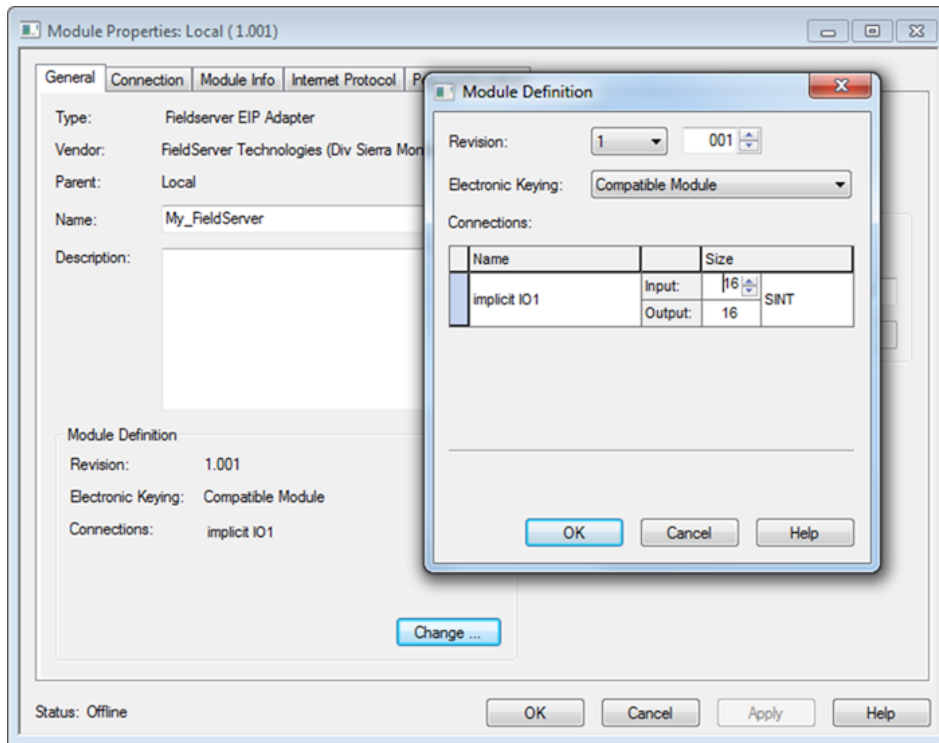
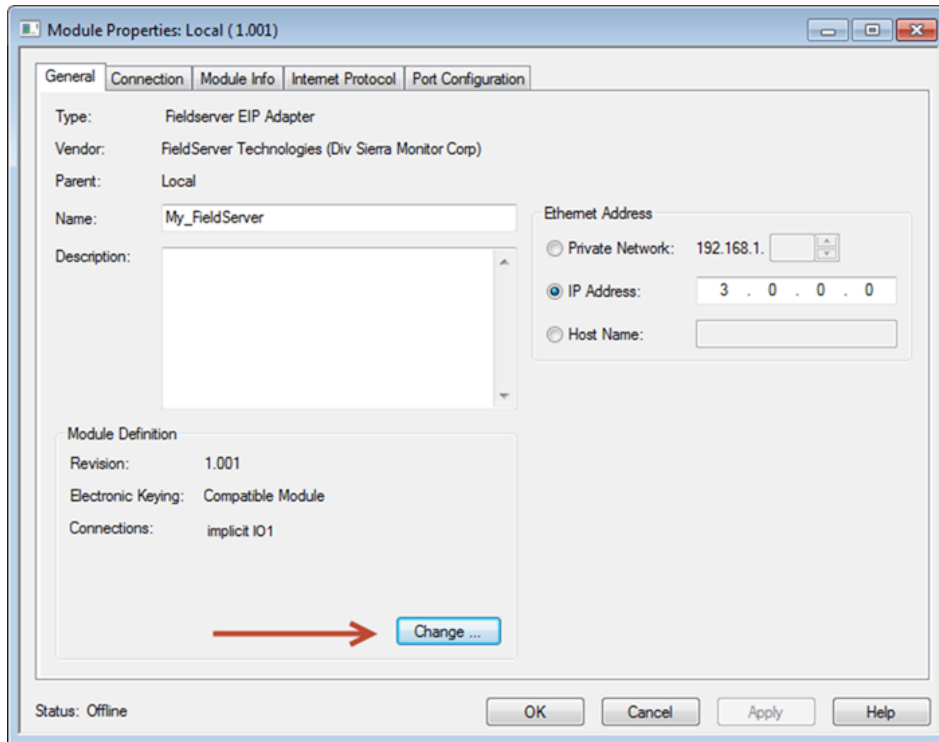
## 7.8.4 EDS AOP

Use Studio 5000 Logix Designer to do the following:

Choose the device to add to a project, the AOP, pick the implicit I/O sizes and the I/O SINT arrays.



# Additional Information



# Additional Information

Controller Tags - V28\_CompactLogix\_Temp(controller)

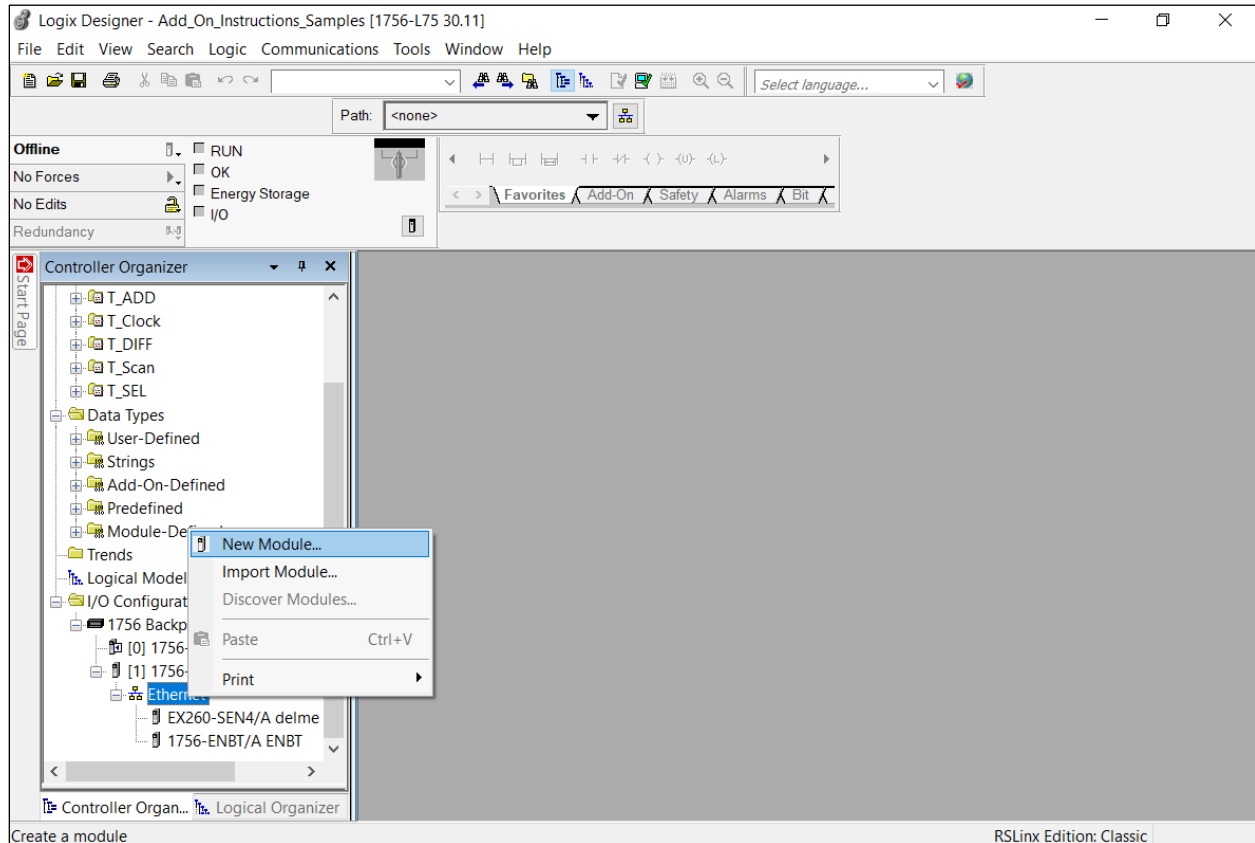
Scope: V28\_CompactLo Show: All Tags Enter Name Filter...

Name	Value	Force Mask	Style	Data Type	Description	Constant
+ Local:1.C	{...}	{...}		AB:Embedded_Di...		<input type="checkbox"/>
+ Local:1.I	{...}	{...}		AB:Embedded_Di...		<input type="checkbox"/>
+ Local:1.O	{...}	{...}		AB:Embedded_Di...		<input type="checkbox"/>
- My_FieldServer:I	{...}	{...}		_036B.000C_000...		<input type="checkbox"/>
- My_FieldServer:I.ConnectionFaulted	0		Decimal	BOOL		
- My_FieldServer:I.Data	{...}	{...}	Decimal	SINT[16]		
+ My_FieldServer:I.Data[0]	0		Decimal	SINT		
+ My_FieldServer:I.Data[1]	0		Decimal	SINT		
+ My_FieldServer:I.Data[2]	0		Decimal	SINT		
+ My_FieldServer:I.Data[3]	0		Decimal	SINT		
+ My_FieldServer:I.Data[4]	0		Decimal	SINT		
+ My_FieldServer:I.Data[5]	0		Decimal	SINT		
+ My_FieldServer:I.Data[6]	0		Decimal	SINT		
+ My_FieldServer:I.Data[7]	0		Decimal	SINT		
+ My_FieldServer:I.Data[8]	0		Decimal	SINT		
+ My_FieldServer:I.Data[9]	0		Decimal	SINT		
+ My_FieldServer:I.Data[10]	0		Decimal	SINT		
+ My_FieldServer:I.Data[11]	0		Decimal	SINT		
+ My_FieldServer:I.Data[12]	0		Decimal	SINT		
+ My_FieldServer:I.Data[13]	0		Decimal	SINT		
+ My_FieldServer:I.Data[14]	0		Decimal	SINT		
+ My_FieldServer:I.Data[15]	0		Decimal	SINT		
- My_FieldServer:O	{...}	{...}		_036B.000C_000...		<input type="checkbox"/>
- My_FieldServer:O.Data	{...}	{...}	Decimal	SINT[16]		
+ My_FieldServer:O.Data[0]	0		Decimal	SINT		
+ My_FieldServer:O.Data[1]	0		Decimal	SINT		
+ My_FieldServer:O.Data[2]	0		Decimal	SINT		
+ My_FieldServer:O.Data[3]	0		Decimal	SINT		
+ My_FieldServer:O.Data[4]	0		Decimal	SINT		
+ My_FieldServer:O.Data[5]	0		Decimal	SINT		
+ My_FieldServer:O.Data[6]	0		Decimal	SINT		
+ My_FieldServer:O.Data[7]	0		Decimal	SINT		
+ My_FieldServer:O.Data[8]	0		Decimal	SINT		
+ My_FieldServer:O.Data[9]	0		Decimal	SINT		
+ My_FieldServer:O.Data[10]	0		Decimal	SINT		
+ My_FieldServer:O.Data[11]	0		Decimal	SINT		
+ My_FieldServer:O.Data[12]	0		Decimal	SINT		
+ My_FieldServer:O.Data[13]	0		Decimal	SINT		
+ My_FieldServer:O.Data[14]	0		Decimal	SINT		
+ My_FieldServer:O.Data[15]	0		Decimal	SINT		

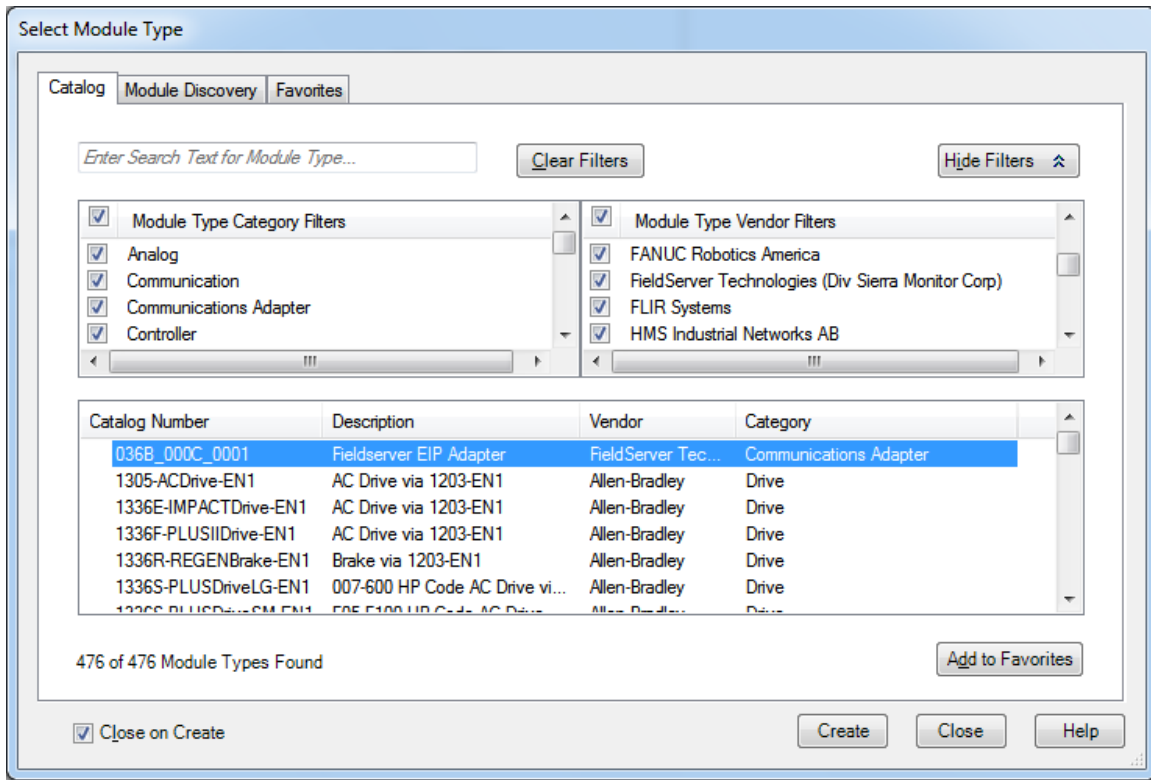
Monitor Tags / Edit Tags /

## 7.9 Configuring the PLC with a FieldServer EDS File

1. Save the fserver.eds file on the local PC.
2. Launch Studio5000.
3. Click on the Tools dropdown menu and select the EDS Hardware Installation Tool.
4. Follow the instructions and select the fserver.eds file.
5. In the Controller Organizer window, right click on “Ethernet” then click on “New Module”.

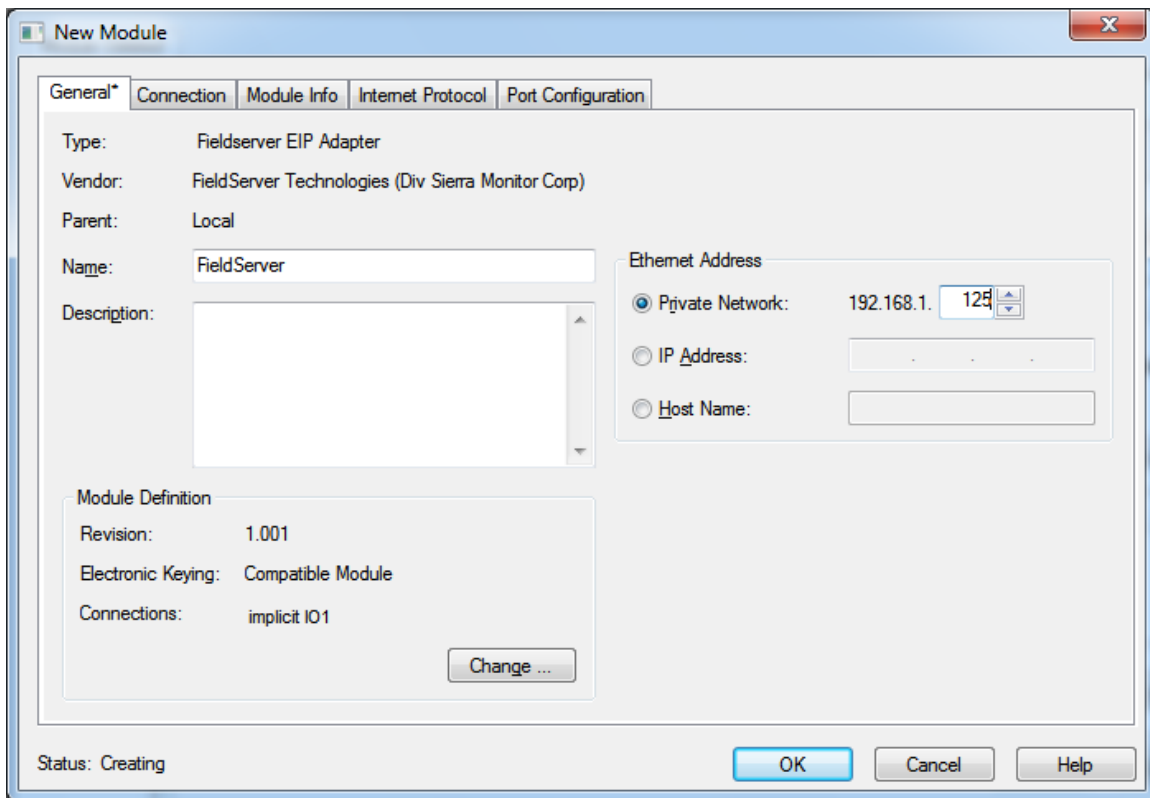


6. Select the “FieldServer EIP Adapter” entry and press the Create button.



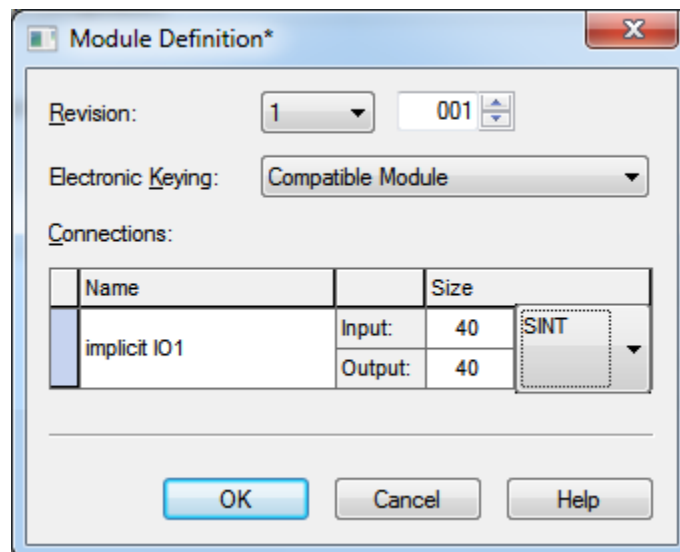


7. On the General tab, specify the module name (any string) and enter the IP Address of the FieldServer. Then press the Change button under the “Module Definition” heading.



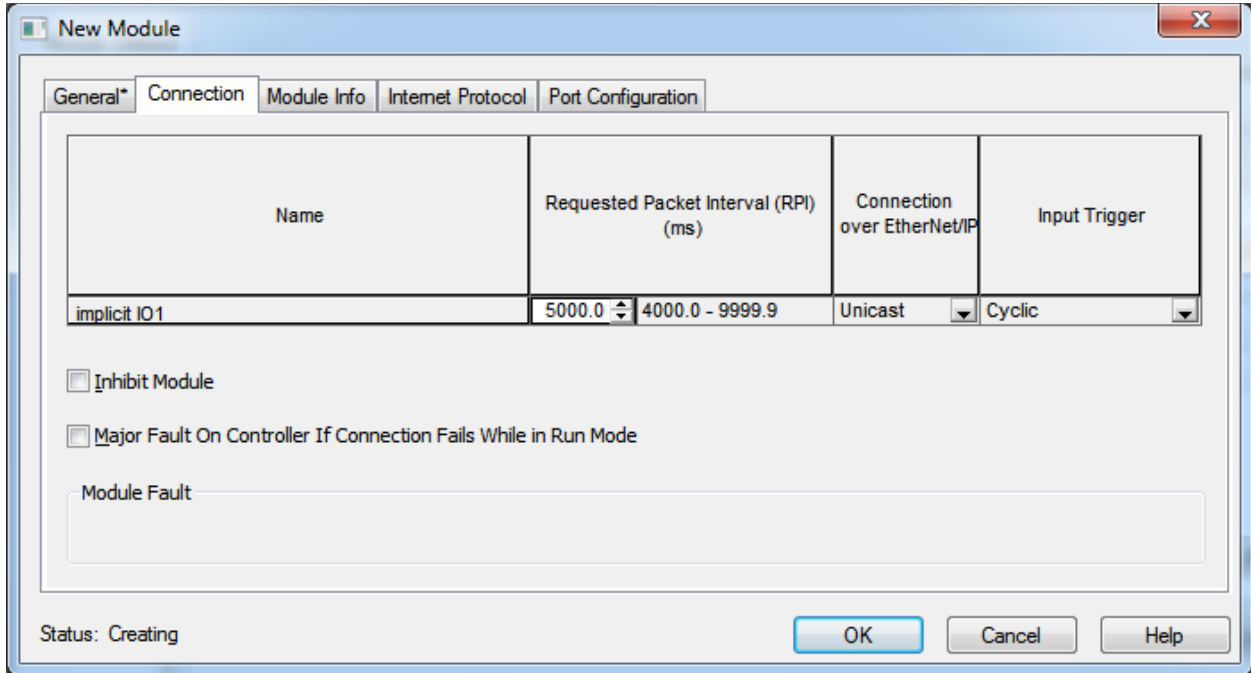
8. In the Module Definition window, click on the “SINT” dropdown menu and select the correct format to match the size/format of the FieldServer configuration.

**NOTE: By default, Studio5000 will create tags in SINT format.**

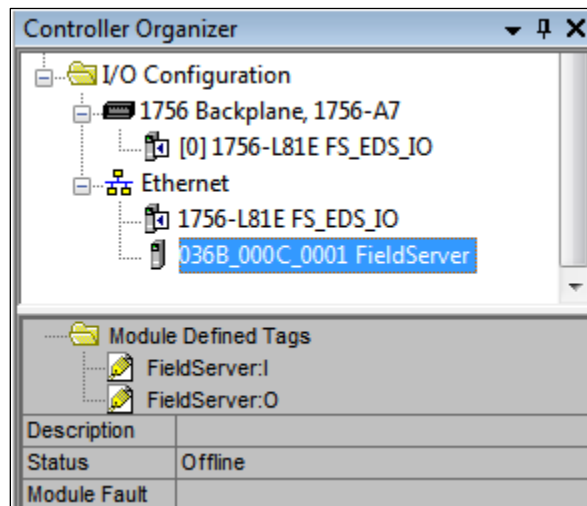


9. Click on the numbers under the Size header to change the size.
10. Then click OK when adjusted appropriately.

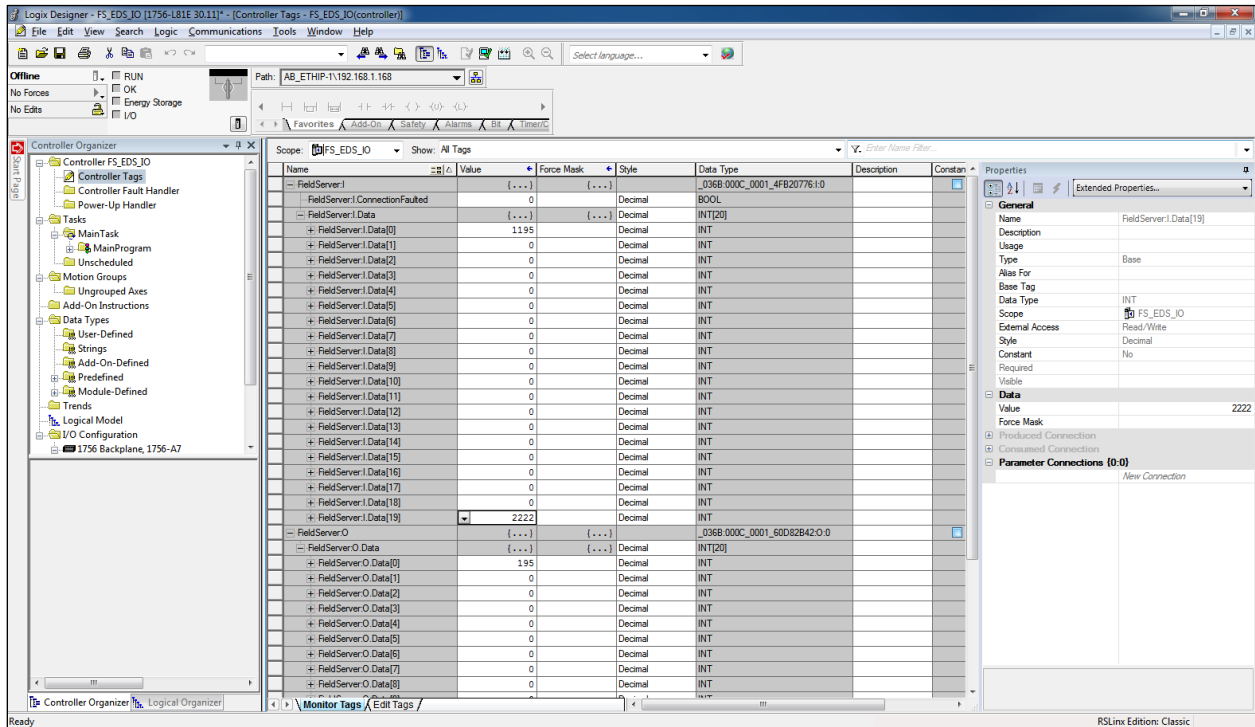
**NOTE: To adjust how often data will be created (optional), select the Connection tab in the New Module window and adjust the RPI setting as needed.**



11. Once all desired settings are configured in the New Module window, click OK. This will create the FieldServer module under the Ethernet tree.



And Studio5000 will create the corresponding tags.



12. Finally, download the project to the PLC.

### 8 Troubleshooting

#### 8.1 Firmware Update Downloading

If you are trying to update firmware and continuously get failed messages it might be due to the traffic on the N1 or N2 ports. EtherNet/IP is a high traffic protocol and once a connection is created continuous data transfer occurs. In this situation, the best way to download new firmware would be to manually disconnect the scanner or adapter connected to the FieldServer.

## 9 Vendor Information

### 9.1 ControlLogix

If EIP Error Code 100 is experienced when using a ControlLogix PLC Message block to read and write data to and from the FieldServer, it may be possible to resolve this error by choosing to cache the connection in the message block settings.

### 9.2 Connection Information – Allen Bradley Message Blocks

The Driver is not able to split data between 2 Data Arrays when writing, or to read a Server mapping that is discontinuous.

For example, on the Server Side, if:

Server map 1: N21: 0-31

Server map 2: N21: 32-100

This will panic the FieldServer and crash RS view as the DH+ will attempt to map N21: 0-100. If set up as Server Map 1: N21: 0-100, no problems are experienced.

Similarly, on the Client Side, it is not possible to read sections e.g. 32bits to one array and then 16bits from the same file type to another array etc. Read Blocks must be contiguous.

### 9.3 FieldServer Not Recognized by RSlinx

If RSlinx does not recognize the FieldServer (message “? Unrecognized Device”), load the EtherNet/IP EDS file into RSLinx. This file is available on the MSA Safety Website.

- Press Start|all programs|Rockwell Software|RSlinx tools|EDS Hardware Installation tool|add|register a single file and browse to the location of the EtherNet/IP EDS file.
- Run RSLinx, press communication|RSwho and all EIP devices on the network should be visible.

### 9.4 Using EIP Data\_Types with RSLogix

The EIP\_Data\_Types defined for the FieldServer are not all supported by the PLC when working with RSLogix. The following aliases have been created so that data can be transferred in the required format.

FieldServer Data Type	Rockwell Data Type	Data Format
BYTE	SINT	8 bit integers
UINT16	INT	16 bit integers
UINT32	DINT	32 bit integers
FLOAT	REAL	32bit floats
BIT	BOOL	Single 1 bit value
BIT_ARRAY, BOOLEAN	BOOL_ARRAY	Multiple 1 bit values

## 10 Reference

### 10.1 Error Messages

Message	Description
"EIP:#01 FYI. %d out of %d data elements will be stored" "MD=%s, data_type=0x%04X, raw bytes=%d"	If the number of data elements exceeds the Map Descriptor length only the number of data elements corresponding to the Map Descriptor will be stored. This message will print once per Map Descriptor.
"DRV->AB_PCCC : PLC5 logical ASCII addressing unsupported DRV->AB_PCCC : Unknown Typed Write data_type= 2"	The driver only supports Logical Binary Addressing, update the PLC messaging type.
"Compatibility - Device Type 0 or Mismatch"	DEVICE TYPE MISMATCH: The Device Type specified in the electronic key logical segment does not match the Device Type of the device. If the compatibility bit is set this error code is returned when the requested Device Type is 0 or the device cannot emulate the specified Device Type. EXCEPTION: If the compatibility bit is set. Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number.

## 10.2 Classes and Attributes Supported

EtherNet/IP is an object orientated protocol. The Object Oriented structure therefore allows for classes, instances, attributes and services. The 'data types' listed below are to be considered as the objects supported in the protocol. Each of these has attributes that have been supported to differing degrees.

FieldServer Data Type	Description (or Device Data Type)	
Identity – Class Code 0x01	<b>Attributes Supported</b> <i>One instance supported (0x01)</i> Attributes List: <ul style="list-style-type: none"> <li>• Vendor ID</li> <li>• Device Type</li> <li>• Product Code</li> <li>• Device Revision</li> <li>• Status</li> <li>• Serial Number</li> <li>• Device Description (text)</li> </ul>	<b>Services Supported</b> Get_Attribute_All; Get_Attribute_Single
Message Router – Class Code 0x02	<b>Attributes Supported</b> <i>One instance supported (0x01)</i> Attributes List: <ul style="list-style-type: none"> <li>• Max Connections</li> </ul>	<b>Services Supported</b> Get_Attribute_Single
Assembly – Class Code 0x04	<b>Attributes Supported</b> <i>Class Instance Support (0x00)</i> Class Attributes: 0x02 (Max Instance) <i>Two instances supported (0x0100 and 0x0101)</i> Attributes List: <ul style="list-style-type: none"> <li>• Member List</li> <li>• Data (0x03)</li> </ul>	<b>Services Supported</b> Get_Attribute_Single
Connection Manager – Class Code 0x06	Forward Open Service. Forward Close Service.	
Register – Class Code 0x07	<b>Attributes Supported</b> <i>Class Instance Support (0x00)</i> Class Attributes: 0x02 (Max Instance) <i>Two instances supported (0x01 and 0x02)</i> Attributes List: <ul style="list-style-type: none"> <li>• Status Flag</li> <li>• Direction (read/write)</li> <li>• Size of Data (bits)</li> </ul>	<b>Services Supported</b> Get_Attribute_Single
Discrete Input Point – Class Code 0x08	No visible interface currently.	
Discrete Output Point – Class Code 0x09	No visible interface currently.	
Analog Input Point – Class Code 0x0A	<b>Attributes Supported</b> <i>Class Instance Support (0x00)</i> Class Attributes: 0x02 (Max Instance) <i>Two instances supported (0x01 and 0x02)</i> Attributes List: <ul style="list-style-type: none"> <li>• Number of Attributes</li> <li>• Analog value (UINT16)</li> <li>• Vendor ID</li> </ul>	<b>Services Supported</b> Get_Attribute_Single

FieldServer Data Type	Description (or Device Data Type)	
Analog Output Point – Class Code 0x0B	<p><b>Attributes Supported</b>  <i>Class Instance Support (0x00)</i>            Class Attributes: 0x02 (Max Instance)  <i>Two instances supported (0x01 and 0x02)</i>            Attributes List:</p> <ul style="list-style-type: none"> <li>• Number of Attributes</li> <li>• Analog value (UINT16)</li> <li>• Vendor ID</li> </ul>	<p><b>Services Supported</b>            Set_Attribute_Single;            Get_Attribute_Single</p>
TCP/IP Interface Object – Class Code 0xF5	<p><b>Attributes Supported</b>  <i>One instance supported (0x01)</i>            Attributes List:</p> <ul style="list-style-type: none"> <li>• Status</li> <li>• Configuration Capability</li> <li>• Configuration Control</li> <li>• Physical Link Object</li> <li>• Interface Configuration</li> <li>• Host Name</li> </ul>	<p><b>Services Supported</b>            Get_Attribute_Single</p>
Ethernet Link Object – Class Code 0xF6	<p><b>Attributes Supported</b>  <i>One instance supported (0x01)</i>            Attributes List:</p> <ul style="list-style-type: none"> <li>• Interface Speed</li> <li>• Interface Flags</li> <li>• Physical Address</li> <li>• Interface Counters</li> <li>• Media Counters</li> </ul>	<p><b>Services Supported</b>            Get_Attribute_Single</p>
Data Table Object – Private Object	<p><b>Attributes Supported</b>            This object does not support instances or attributes but uses the data table structure, and associated tags, in Logix5000 PLC's.</p>	<p><b>Services Supported</b>            CIP Read Data</p>



### 10.3 Error Codes

General Status	Extended Status	Description
0x01	0x0100	Connection in Use or Duplicate Forward Open.
0x01	0x0103	Transport Class and Trigger combination not supported.
0x01	0x0106	Ownership conflict.
0x01	0x0107	Connection not found at target application.
0x01	0x0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection.
0x01	0x0109	Invalid connection size.
0x01	0110	Device not configured.
0x01	0x0111	RPI not supported. May indicate problem with connection time-out multiplier or production inhibit time.
0x01	0x0113	Connection Manager cannot support any more connections.
0x01	0x0114	Either the Vendor Id or the Product Code in the key segment didn't match the device.
0x01	0x0115	Product Type in the key segment did not match the device.
0x01	0x0116	Major or Minor Revision information in the key segment did not match the device.
0x01	0x0117	Invalid Connection Point.
0x01	0x0118	Invalid Configuration Format.
0x01	0x0119	Connection request fails since there is no controlling connection currently open.
0x01	0x011A	Target Application cannot support any more connections.
0x01	0x011B	RPI is smaller than the Production Inhibit Time.
0x01	0x0203	Connection cannot be closed since the connection has timed out.
0x01	0x0204	Unconnected Send timed out waiting for a response.
0x01	0x0205	Parameter Error in Unconnected Send Service.
0x01	0x0206	Message too large for Unconnected message service.
0x01	0x0207	Unconnected acknowledge without reply.
0x01	0x0301	No buffer memory available.
0x01	0x0302	Network Bandwidth not available for data.
0x01	0x0303	No Tag filters available.
0x01	0x0304	Not Configured to send real-time data.
0x01	0x0311	Specified Port Not Available – FieldServer attempts to open a connection but target port isn't available on remote device. Resolve using the correct EIP_Path parameter.
0x01	0x0312	Link Address specified in Port Segment Not Available.
0x01	0x0315	Invalid Segment Type or Segment Value in Path.
0x01	0x0316	Path and Connection not equal in close.
0x01	0x0317	Either Segment not present or Encoded Value in Network Segment is invalid.
0x01	0x0318	Link Address to Self Invalid.
0x01	0x0319	Resources on Secondary Unavailable.
0x01	0x031A	Connection already established.
0x01	0x031B	Direct connection already established.
0x01	0x031C	Miscellaneous.
0x01	0x031D	Redundant connection mismatch.
0x01	0x031D	No more consumer resources available in the producing module.
0x01	0x031E	No connection resources exist for target path.
0x01	0x0320-0x7FF	Vendor Specific.
0x02	n/a	Connection Manager resources are unavailable to handle service request.

## Additional Information

General Status	Extended Status	Description
0x03	n/a	Invalid connection number specified by the Get_Connection_Data service. This is also returned by the Search_Connection_Data service if the specified connection is not found.
0x04	Zero Based Word Offset	Segment Type in path is invalid. The Extended Status shall be the word offset (0 based) to the word in the path where the error occurred. The offset starts at the first word after the path size. This error shall not be returned if an error occurs when parsing the Connection Path. This error commonly occurs when tag names are incorrectly spelled, mismatches in case or do not exist. Verify Map Descriptor EIP_Tag_Name(s) spelling and case match PLC tag names.
0x05	Zero Based Word Offset	Destination in path is invalid. The Extended Status shall be the word offset (0 based) to the word in the path where the error occurred. The offset starts at the first word after the path size. This error shall not be returned if an error occurs when parsing the Connection Path.
0x07	n/a	Connection has been lost. This is used by the Get/Set Services when they are made through a connection.
0x08	n/a	Connection Manager does not support the requested Service.
0x09	Index to Element	Error in Data Segment. Extended Status shall be index to where the error was encountered in the Data Segment. The Configuration Revision Number if present in the Data Segment shall always be index 1. If the error occurs with the Get/Set Services, then the extended status indicates the attribute number that failed.
0x0C	Optional	Service cannot be performed while Object is in current state. The 1st word of Extended Status may optionally contain the object's current state.
0x10	Optional	Service cannot be performed while Device is in current state. The 1st word of Extended Status may optionally contain the device's current state.
0x11	n/a	Response data too large. This is used by the get services to indicate the amount of data requested was too large to fit into the response buffer.
0x13	n/a	Not enough data was received.
0x14	Attribute Id	Attribute specified in FIND service is not supported by Connection Manager.
0x15	n/a	Too much data was received.
0x25	0x0114	Either the Vendor Id or the Product Code in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x25	0x0115	Product Type in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x25	0x0116	Major or Minor Revision information in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x026	n/a	Invalid path size.

**NOTE:**

- The word “n/a” in the Extended Status Column is used to signify that there is no additional Extended Status which is required to be returned for the particular General Status Code.
- The word “optional” in the Extended Status Column is used to signify that if Extended Status information is used, then the first word of that extended status is already defined.