

1 Purpose

This document is a guide to the use of Ethernet/IP[1] (EtherNet/Industrial Protocol or EIP) UDP/IP-based "implicit" messaging with Rockwell Automation ControlLogix systems. This document does not discuss EIP TCP/IP-based, "explicit" messaging beyond its use in the connection establishment process. See Ref #1 below for a discussion on explicit messaging with Rockwell Automation products.

EtherNet/IP "implicit" or I/O messaging is a multicast, producer/consumer protocol designed for "control" data transfer. In ControlLogix parlance, I/O messaging is the transfer of I/O data with an I/O device in the I/O configuration tree, or the transfer of Tag data via a Produced Tag connection.

In places, this document shows some steps to take using Rockwell Automation software products. This is not intended to be a complete tutorial on using these products. It is assumed that the reader has familiarity with the basic use of the controllers and software to program them.

2 CIP Background, Reference Information & Definitions

The *Control and Information Protocol* (CIP) is part of an architecture for industrial control devices that defines not only the message protocol for use on multiple networks (eg: DeviceNet, ControlNet & EtherNet/IP) but also defines the object model, I/O data, configuration data and behavior of common industrial control products. CIP uses object modeling techniques to convey this information. There is a library of standard objects in the CIP Common specification (Ref #2) as well as a library of device descriptions of common industrial control devices known as *Device Profiles*.

CIP is a connection oriented, producer/consumer based technology. It defines relationships between nodes called *Connections.* There are two types of connections, *Explicit Message* (a.k.a.: class 3) connections which are Client/Server type functions and *I/O Message* (a.k.a.: Implicit or class 1) connections which are producer/consumer data transfers.

CIP defines a connection as a communications relationship between two or moe nodes across a network. Connection must be set up and maintained, so there is some overhead associated with them. But it is minimal. Once established, resources in all participating nodes are reserved and message handling is streamlined.

2-1 References

The following items are necessary prerequisites to this document. References are available electronically at the noted web sites.

- **Ref#1.** "Communicating with Rockwell Automation Products Using EtherNet/IP Explicit Messaging", a document describing how to establish explicit communications with Rockwell Automation controllers, including the ControlLogix system. See www.rockwellautomation.com/enabled to download a copy of this document.
  Ref#2. "EtherNet/IP Specification" (Release 1.0, 05-June 2001) Eval copy at www.ethernet-ip.org.
  - **Volume 1:** CIP Common Specification
  - **Volume 2:** EtherNet/IP Adaptation of CIP

  Note: You can download a complete copy of the spec for evaluation on this web site at no cost. However, you must purchase a spec subscription from ODVA/CI and sign an EtherNet/IP Terms of Use Agreement in order to build an EtherNet/IP product for sale.

- **Ref#3.** "EtherNet/IP Terms and Definitions" document at http://www.odva.org/10_2/03_events/New-EtherNet/EthernetIP_Terms.pdf. Please see this document for definitions of terms used throughout this text.

- **Ref#4.** ControlLogix User Manual, publication 1756-PM001E-EN-P (http://www.ab.com/manuals/cl/1756-pm001e-en-p.pdf)

[1] EtherNet/IP is a registered trademark of ControlNet International

2-2 Definitions

In addition to the definitions given in Ref#3, there are other terms used in this document that may not be common knowledge.

2-2.1 RSLogix 5000

The software package used to configure the ControlLogix controller, the Rockwell Automation I/O devices connected to it and to program the logic used in the controller. This package is available from Rockwell Software, Inc.

2-2.2 RSNetWorx for EtherNet/IP

The network configuration tool from Rockwell Software that provides a) network "who" to find all EIP devices on the subnet, b) configuration of scanner devices on EtherNet/IP that support the Connection Configuration Object defined in CIP Common and c) configuration of simple EtherNet/IP-based devices that have an EDS. This tool is NOT used when configuring the ControlLogix system to talk to EtherNet/IP connected I/O devices.

2-2.3 RSLinx

RSLinx is the communications driver software used by many Rockwell Software products to provide network communications services to any of the Rockwell Automation supported networks. RSLogix 5000 and RSNetWorx for EtherNet/IP uses RSLinx to provide access to EtherNet/IP networks on a PC through standard Ethernet PC interface cards (i.e.; no special interface card is needed).

RSLinx makes use of EDS files to match an icon to the device's identity. If no EDS file is present, the device will show up on the network browse screen as an "Unknown Device".

2-2.4 Input, Output, Status

These terms are used throughout this to refer to data flowing between the controller and nodes in the I/O Configuration. The controller produces *Outputs* and consumes *Inputs* as well as *Status,* from these nodes.

2-2.5 Inhibited Connection

The user can Inhibit a connection by setting the *Inhibit Bit* on the appropriate Connection Properties tab. Inhibited connections will remain in the controller's memory, but are not opened. See Ref #4, User Manual - Inhibiting Connections - for more information on this.

2-2.6 "Heartbeat"

In this document, "heartbeat: is used to refer to the half of a connection relationship that produces a message in the Originator to Target (O-T) direction, which contains no data, just a sequence count. The purpose of the "heartbeat" is to "close the loop" between application objects in nodes that have a connection relationship, to provide "loss of partner" indication.

3 ControlLogix System Info

The information presented here is presented for the 1756-L55 logic controller; however it also applies to the other members of the ControlLogix family: CompactLogix, DriveLogix, SoftLogix and FlexLogix. The primary difference between controller family members is the name and catalog numbers of the network interface cards, and in the number of connections, amount of memory, etc. For brevity, the text will refer to the ControlLogix processor and L55 controller, and will use screen shots from RSLogix 5000 for these products. Any controller specific differences that impact the text will be noted where applicable. Please see product specific user manuals for details of the capabilities of each product.

3-1 ControlLogix Processor Module

The ControlLogix processor module (specifically, the 1756-L55) is the *Originator* of I/O Data Connections and Produced Tag connections. These connections will be "routed" through the communications module (1756-ENBT or 1756-ENET/B) to/from the controller.

The L55 controller is limited to 250 I/O connection “pairs”. This pool of connections is used for local (in chassis) I/O connections, networked I/O connections, Produced Tag connections. Please see the Product User Manuals (Ref#4) for more information on connection resource utilization. If you need further assistance with connection resource utilization problems, contact Rockwell Automation Technical Support at: (http://support.rockwellautomation.com).

3-2 Connection Support

ControlLogix supports both Explicit Message (aka: class 3) and I/O Message (aka: class 1) connections. For a discussion of ControlLogix Explicit Message support, see Ref#1. This document will focus on I/O Message connections. The two variations that are supported by ControlLogix are:

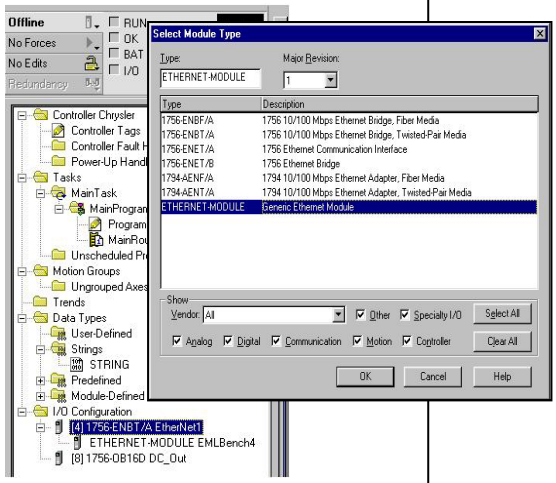
I/O Data Connections – these are described in Section 4 of this document. They are connections between a ControlLogix controller and I/O devices that exist in the controller’s I/O tree. The data sent/received is predetermined by the device manufacturer.

Produced Tag Connections –these are described in Section 5 of this document. They are direct connections to tags in the ControlLogix database. The user determines what tags they wish to be produced based on their application needs.

4 I/O Data Connections

I/O Data connections are typically used to transfer data between a controller and a field I/O device such as a rack a block of I/O, a drive, etc. These connections utilize transport class 1 with a cyclic trigger to transmit data at a timed rate. The ControlLogix controller always assumes the role of the connection *Originator* in this connection relationship and the field device being connected to is the *Target* of the connection request. The typical I/O Data target is an *Adapter* device, as defined by the ODVA/CI document in Ref#3.

Figure 2: RSLogix 5000 view of the Controller's I/O Tree and the selection box for a Generic Ethernet device



I/O Data connections are established as an O-T and T-O pair. Output data is sent in the O-T direction and Input data or a Heartbeat (no data) is sent in the T-O direction.

Note: Throughout this document the term “connection” refers to this pair of connections, unless otherwise noted.

Input data (consumed by the controller) can be set up as multicast (i.e.: 1 producer, n consumers). Multicast of output data (produced by the controller) is not supported by ControlLogix.

The *Target* device **must** be present in the controller’s I/O tree (See Figure 1). To adding a target to the controller’s I/O Configuration is accomplished offline by adding a new EIP interface "module" into the I/O tree in the Controller Properties pane in RSLogix 5000. Point to I/O Configuration, right click and select “add module”, select the interface module (eg: 1756-ENBT/A, 1756-ENET/B) and apply the changes. Please see user manual for more details of this process.

Once the EIP interface module is in the I/O Configuration list, point to the interface module, right click, select “Insert Module” and choose “ETHERNET-MODULE Generic Ethernet Module”, as shown in Figure 2.

Note: The ControlLogix system and RSLogix 5000 categorizes all non-Rockwell devices as Generic EtherNet/IP devices. This does not relate to the actual device type of the product.

There are several properties tabs that must be filled in so the controller knows how to talk to the target device. These will be covered in the next section.

When the steps to insert and configure a module are completed, the associated tags for this module will be created in the Controller's Tag database. It will also lock down the data sizes associated with the data transfers that will occur when the connections are opened.

The final step in the process is to go online and download the program to the controller. At that time the controller will begin to open the connections unless the connection is *Inhibited*.

4-1 I/O Data Connection Communications Formats

During the process of inserting an EIP module into the Controller I/O Configuration, the user will be presented with a Module Properties tab where the Communications Format and Connection Parameters are chosen (See figure 3). One of the items to be configured by the user is the data format (called Comm Format on this screen). There are a number of data formats to choose from, not all of which will be supported by every device. The product vendor **must** identify in their product documentation, which one(s) to use with their product.

Some of the choices presented are:

Data –DINT,	Data –DINT with Status,	
Data –INT,	Data –INT with Status,	
Data –REAL,	Data –REAL with Status,	
Data –SINT,	Data –SINT with Status,	
Input Data –DINT,	Input Data –DINT with Run/Program,	Input Data –DINT with Status
Input Data –INT,	Input Data –INT with Run/Program,	Input Data –INT with Status
Input Data –SINT,	Input Data –SINT with Run/Program,	Input Data –SINT with Status
Input Data –REAL,	Input Data –REAL with Status	

The following terms used above are defined as:

Data	'I' and 'O' data flows between devices (e.g.: bidirectional data flow)
Input Data	'I' data flows to the controller & a "heartbeat" flows to the Target (heartbeat means only a Sequence Count with no application data)
SINT, INT, DINT, REAL	CIP data types; sized: SINT= 1 byte, INT = 2 bytes, DINT & REAL = 4 bytes
"With Run/Program"	Special format for a legacy Rockwell product that should NOT be used by any other devices.
"With Status"	If selected, a **second connection is made** with the target to transfer additional device specific information in the T-O direction and a heartbeat in the O-T direction.

The selections made here determine the number of connections opened with the target, the amount of data, the content of the data and the size and structure of the tags that will be created in the controller's tag database in association with this device.

4-1.1 Determining communications packet sizes

The Comm Format selections made above, along with the application data sizes chosen (See figure 2, Connection Parameters box) will eventually determine the sizes used in the ForwardOpen request, and ultimately, the size of the I/O packet data. The size in the ForwardOpen request sent by the controller to the target can be determined as follows:

For "Data" connections:

T-O connection size = number of bytes for the data type chosen (eg: 1 for SINT, 2 for INT, 4 for DINT, 4 for REAL) * Input "Size" entered + 2 bytes for the sequence count.

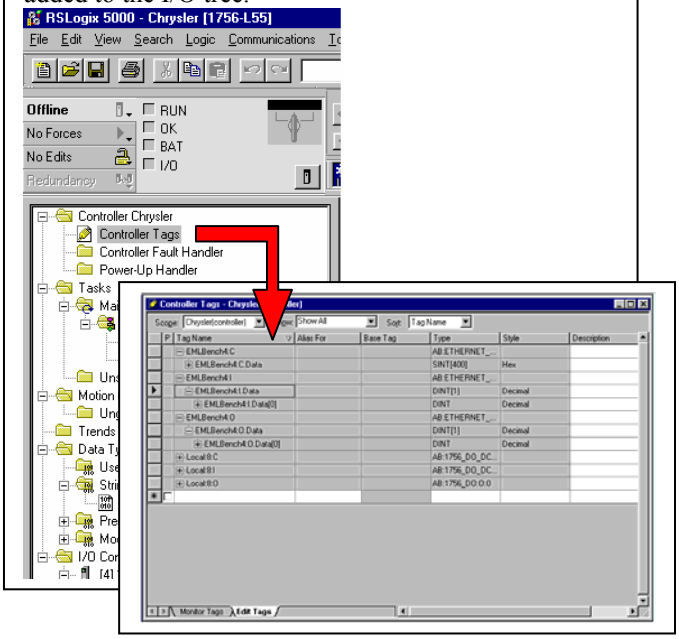
O-T connection size = number of bytes for the data type chosen * Output "Size" entered + 2 bytes for the sequence count + 4 bytes for the Run/Idle Header.

For "Input Data" connections:

T-O connection size = number of bytes for the data type chosen * Input "Size" entered + 2 bytes for the sequence count.

O-T connection size = 2 bytes for the sequence count. There is no data since this is a "heartbeat" only.

Figure 4: Controller Tags created when a module is added to the I/O tree.



For "with Status" connections:

T-O connection size = number of bytes for the data type chosen * Status Input "Size" entered + 2 bytes for the sequence count.

O-T connection size = number of bytes for the data type chosen * Output "Size" entered + 2 bytes for the sequence count + 4 bytes for the Run/Idle Header.

An example is appropriate here. Assume the Comm Format of **Data –DINT** is chosen and the Input and Output "Size" fields are both set to 1. The O->T (Input) data size specified in the ForwardOpen request will be 1 * 4 bytes (1 DINT) + 2 bytes (Sequence Count), for a total of 6 bytes. The O->T (Output) data size specified in the ForwardOpen request will be 1 * 4 bytes (1 DINT) + 2 bytes (Sequence Count) + 4 bytes (Run/Idle Header), for a total of 10 bytes. See the example traffic at the end of this section for an example.

The Comm Format selection also has options that are followed by "-with Status". This status information is device specific information transferred in a separate connection made with the device. Selecting "-with Status" Comm formats will result in two connections being opened with the target device, one for the status data and one for the I/O data.

4-1.2 Controller Tags Created for EIP-connected Modules

Each ETHERNET – MODULE inserted into the controller's I/O tree will result in the creation of associated Tags in the Controller Tags database. This occurs when the module selection and configuration process is completed. To view these tags select the "Controller Tags" item on the Controller resources tree. Refer to figure 4.

The tag name is the same name given to the module in the Module Properties screen, Name field (see Figure 2), which in this example is "EMLBench4".

There are 4 possible Tags created with each I/O device:

The ':O' tag is the Output data. This tag is not created when the "Input Data" Comm Formats are chosen.

The ':I' tag is the Input data.

The ':S' tag is the Input Status data. This tag is only created when the "-with Status" Comm Formats are chosen. See section 4-3, Status Connection for more information.

The ':C' tag is the Configuration data. This data is transferred to the device in a Data Segment at the end of the ForwardOpen. The format of this data is provided by the device manufacturer.

4-1.2.1 Tag Sizes

The sizes of the O, I and S tags include the exact data sizes chosen in the Module Properties screen. They do not include the Run/Idle Header or the Sequence count. The Configuration Tag size is always fixed at 400 SINTs however only the number of bytes that correspond to the size entered on the Module Properties screen will be sent to the device in the ForwardOpen. See section 4-2.2, Data Segment for further discussion.

4-2 Connection Paths for I/O Data Connections

The Connection Path is part of the ForwardOpen command parameters sent by the originator. It consists of several elements, which include: Electronic Key Segment, Application Path and optional Data Segments and Port Segments. The use and interpretation of these items follows.

ControlLogix expects the order of the encoded segments in the Connection Path to be: Electronic Key, Port Segment, Configuration Path, Consumed Path, Produced Path and optional Data Segment. The Data segment is optional depending on whether a non-zero Configuration Data size was entered or not.

Note: Produce Path and Consume Path terms are from the perspective of the target device.

4-2.1 Application Path

The Application Path consists of a Configuration path, Consumption path and Production path. Together, these represent the items in the target node that the controller wants to access. The values used here come from the entries made by the user in the Connection Properties screen (see figure 2). The user populates these values from the target node's product literature.

Note: It is critical that the product developer documents the appropriate values in their product literature so the user will know what to enter.

A typical Application path contains 4 items. An example is shown here: 20 04 24 03 2C 01 2C 02

- 20 04 This is the Class Identifier. ControlLogix Generic EtherNet/IP Module selection requires the use of the assembly object, class 04.
- 24 03 This specifies the Configuration Data instance. It comes from the value entered into the "Configuration" field on the Module Properties screen. See below for more information on this item.
- 2C 01 This is the Consumed Data Connection Point. It comes from the value entered into the "Output" field on the Module Properties screen, which represents an instance of the Assembly class in the target device.
- 2C 02 This is the Produced Data Connection Point form. It comes from the value entered into the "Input" field on the Module Properties screen, which represents an instance of the Assembly class in the target device.

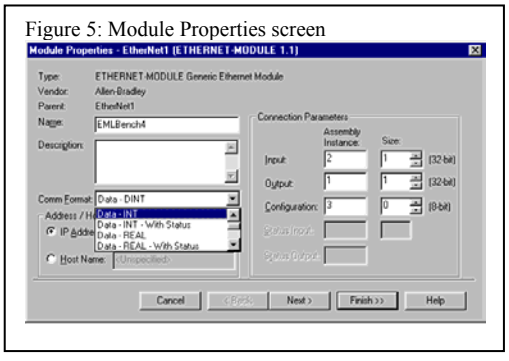
Note: The user interface in the Connection Parameters box on the Module Properties dialog (See figure 4) asks for an Assembly Instance number for Input, Output and Status Input, but the controller encodes these as connection points in the ForwardOpen.

4-2.1.1 Configuration Path

All Comm Formats include an entry for Configuration Data Instance. The instance number selected on this screen goes in the Configuration Path. If the user enters a non-zero size in this field a Data Segment will be included in the ForwardOpen.

The Configuration Path will always be included in the ForwardOpen even if the Configuration Data size is set to zero in the Connection Parameters box. (See Figure 5) In this situation, the ForwardOpen will NOT contain a Data Segment. The Configuration Path is irrelevant and should be ignored. So for the selections made in figure 5, the Configuration Path will be 24 03.

Figure 5: Module Properties screen



This method of sending configuration data to the target node is useful for devices that do not have non-volatile storage for their configuration data. The device's configuration is stored in the controller (and backed up by RSLogix 5000) and is delivered to the target device when the connection is opened.

The process for establishing the connection requires the target to validate the configuration data **before** the ForwardOpen response is sent. Device configuration data that results in an invalid configuration setup will result in a ForwardOpen reply that includes the appropriate error indication.

Note: See the description of the Status Connection in section 4-3 for discussion of the Comm Formats "with Status" and how it may affect configuration data delivery.

4-2.2 Data Segment

A Data Segment is included in the ForwardOpen when a non-zero configuration size is specified in the Module Properties screen. The purpose for the Data Segment is to carry the data present in the Configuration tag (see section 4-1.2, Controller Tags Created for EIP-connected Modules) to device when the connection is opened.

The Data Segment is defined in the CIP Common spec, Appendix C: Data Management as a Segment Type byte, followed by a size, followed by an array of bytes. The array contents are defined by the device manufacturer. The details of this should be described in the product literature, and also in an [Assem] entry in an Electronic Data Sheet. The following example is taken from the traffic captures at the end of this section and shows a simple data segment with 2 bytes of configuration data.

80 01 aa 55 Data Segment (80 = Segment Type; 01 = # words; aa 55 is configuration data[0] & [1]

4-2.3 Electronic Key Segment

The Electronic Key that is part of the ForwardOpen's Connection Path identifies the Vendor ID, Device type, product Code and Major/Minor Revision of the target device. It is used to validate that the connection target is the intended target. The level of keying (i.e.: how many of these fields are significant) is typically user selectable, however ControlLogix does not provide keying for generic EIP devices (i.e.: ETHERNET-MODULE). Therefore, this segment is sent as all zeros indicating that no keying is applied to this connection. This can be seen in any of the examples that follow at the end of this section.

4-2.4 Port Segment

The Port Segment describes the network hops required to get from the originator to the target. The format for this field is defined in the CIP Common spec, Appendix C: Data Management and will not be described here. Note that the Port Segment will immediately precede the Configuration Path.

4-3 Status Connection

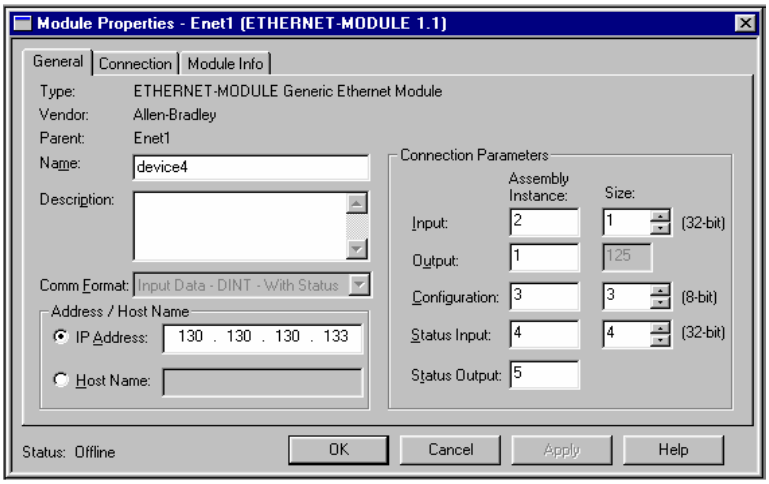
Most of the time, simple devices do not have any additional status information to convey to the control program, or the status is already part of the I/O data. When a user selects a Comm Format that is "with Status" a second connection is opened between the controller and target to transfer status information. Therefore, two ForwardOpen requests will be sent by the controller to the target device, one for the data and the other for status.

Note: There is no guarantee which ForwardOpen will be sent first by the controller. If a Configuration size greater than zero is entered, the Data Segment will be in the FIRST ForwardOpen, regardless of which one it is.

Figure 6 shows the Module Properties screen for a connection with Status. The Instance numbers entered represent the instance numbers of the assembly class that participate in this connection. The Status Input instance in the Target will produce the target device's status data. It is reflected in the Produced Data Path item in the ForwardOpen. The Consumed Data Path item is always a size of zero. It is like the O-T data in an "Input Data" connection – it is essentially a "heartbeat".

The example traffic in the section 4-4, ForwardOpen examples for I/O Data Connections, shows an example of an I/O device "with Status."

Figure 6: Module Properties for Input Connection with Status



4-4 ForwardOpen examples for I/O Data Connections

The following examples should help to clarify some of the points described above. These examples are given:

4-4.1 I/O Data Connection: No Status, No Configuration Data

4-4.2 I/O Data Connection: Configuration Data, No Status

4-4.3 I/O Data Connection: Status, No Configuration Data
4-4.3.5 I/O and Status Data Packets

Input Data Connection: No Status, No Configuration Data

4-4.1 I/O Data Connection: No Status, No Configuration Data

The example that follows shows the typical connection opening request/response for an I/O Data connection with a device in the ControlLogix controller's I/O tree. This example correlates to the previous examples, where both the input size and output size is 1 DINT, the configuration size is 0 and the connection Format chosen is Data –DINT. (See figure #2)

4-4.1.1 ForwardOpen Request

The following is the ForwardOpen Request generated by the ControlLogix system for this example.

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 8c 04 0e 00 00 40 06  67 93 82 97 84 d2 82 97 |
0020  84 ca 04 03 af 12 95 17  11 f2 00 82 c9 3f 50 18 |
0030  40 00 79 24 00 00 6f 00  4c 00 cc c8 82 00 00 00 | Encapsulation Header
0040  00 00 82 97 84 d2 00 00  00 15 00 00 00 00       |
                                                00 00 | Start of Encaps Data
0050  00 00 00 00 02 00 00 00  00 00 b2 00 3c 00       |
                                                54 02 | Start of FwdOpen Rqst
0060  20 06 24 01 05 9b 00 00  00 00 00 00 00 00 09 00 |
0070  01 00 e4 13 12 00 00 00  00 00 a0 86 01 00 0a 48 |
0080  a0 86 01 00 06 28 01 09  34 04 00 00 00 00 00 00 |
0090  00 00 20 04 24 03 2c 01  2c 02                   | End of FwdOpen Rqst
```


The ForwardOpen service breaks down as follows:

```
54                  Service Code
02 20 06 24 01      Path to service destination (size of path, class
                    segment, instance segment)
05 9b               Service priority/time per tick, Timeout (# of ticks)
00 00 00 00         O->T Connection Id
00 00 00 00         T->O Connection Id
09 00               Originator's Connection Serial Number
01 00               Originator's Vendor ID
ef 13 12 00         Originator's Serial Number
00                  Connection Timeout Multiplier (00 = 4 x RPI)
00 00 00            Reserved
a0 86 01 00         O->T RPI (in microseconds) (0x000186a0 * 1us = 100ms)
0a 48               O->T Conn. Parameters
a0 86 01 00         T->O RPI (in microseconds) (0x000186a0 * 1us = 100ms)
06 28               T->O Conn. Parameters
01                  Transport type/Trigger
09                  Connection Path Size (in words)
34 04 00 … 00       Electronic Key    (none)
20 04 24 03 2c 01 2c 02 Application Path:
      Class= 4, Configuration Inst= 3, Consume Inst= 1, Produce Inst= 2
```

4-4.1.2 ForwardOpen Response

This is the reply to the request shown above.

```
0000  00 00 bc 05 0e fd 00 60  08 1c 48 e2 08 00 45 00 |
0010  00 96 31 10 40 00 80 06  ba 86 82 97 84 ca 82 97 |
0020  84 d2 af 12 04 03 00 82  c9 3f 95 17 12 56 50 18 |
0030  21 a0 60 4e 00 00 6f 00  56 00 cc c8 82 00 00 00 |
0040  00 00 82 97 84 d2 00 00  00 15 00 00 00 00 00 00 |
0050  00 00 00 04 04 00 00 00  00 00 b2 00 1e 00 d4 00 | FwdOpen Response
0060  00 00 d6 c8 82 00 d7 c8  82 00 09 00 01 00 e4 13 |
0070  12 00 a0 86 01 00 a0 86  01 00 00 00 00 80 10 00 |
0080  02 00 08 ae 00 00 00 00  00 00 00 00 00 00 00 00 |
0090  01 80 10 00 02 00 08 ae  ef c0 1a 40 00 00 00 00 |
00a0  00 00 00 00                                     |
```

The ForwardOpen reply breaks down as follows:

```
d4                  Service Reply
00                  Pad
00                  Status (00 = success)
00                  Extended Status Size
d6 c8 82 00         Final O->T Connection ID
d7 c8 82 00         Final T->O Connection ID
09 00               Originator's Connection Serial Number
01 00               Originator's Vendor ID
e4 13 12 00         Originator's Serial Number
a0 86 01 00         O->T API
a0 86 01 00         T->O API
00                  Application reply size (none in this example)
00                  Pad
```

4-4.2 I/O Data Connection: Configuration Data, No Status

The example that follows shows the typical connection opening request/response for Controller I/O Data connections with devices in the ControlLogix controller's I/O tree. This example correlates to the previous examples

given where the input size and output size are both 1 DINT, but this specifies a configuration size of 2 bytes. The connection Format chosen is Data –DINT.

In this example note the differences in the Connection Path, specifically, the addition of the Data Segment to carry the configuration data to the target node. The Data Segment is defined in the CIP Common Specification in Appendix C, Data Management. (See Ref #2)

4-4.2.1 ForwardOpen Request

The following is the ForwardOpen Request generated by the ControlLogix in this example.

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 90 dd a1 00 00 40 06  8d fb 82 97 84 d2 82 97 |
0020  84 ca 04 05 af 12 f3 b1  35 b2 05 bd db 41 50 18 |
0030  40 00 dc c6 00 00 6f 00  50 00 fa da bd 05 00 00 | Encapsulation Header
0040  00 00 82 97 84 d2 00 00  88 23 00 00 00 00
                                                 00 00 | Start of Encaps Data
0050  00 00 00 00 02 00 00 00  00 00 b2 00 40 00
                                                 54 02 | Start of FwdOpen Rqst
0060  20 06 24 01 05 9b 00 00  00 00 00 00 00 00 09 00 |
0070  01 00 e4 13 12 00 02 00  00 00 10 27 00 00 0a 48 |
0080  10 27 00 00 06 28 01 0b  34 04 00 00 00 00 00 00 |
0090  00 00 20 04 24 03 2c 01  2c 02 80 01 aa 55       | End of FwdOpen Rqst
```

The ForwardOpen service breaks down as follows:

```
54                  Service Code
02 20 06 24 01      Path to service destination (size of path, class
                    segment, instance segment)
05 9b               Service priority/time per tick, Timeout (# of ticks)
00 00 00 00         O->T Connection Id
00 00 00 00         T->O Connection Id
09 00               Originator's Connection Serial Number
01 00               Originator's Vendor ID
ef 13 12 00         Originator's Serial Number
02                  Connection Timeout Multiplier (00 = 16 x RPI)
00 00 00            Reserved
10 27 00 00         O->T RPI (in microseconds) (0x00002710 * 1us = 10ms)
0a 48               O->T Conn. Parameters
10 27 00 00         T->O RPI (in microseconds) (0x00002710 * 1us = 10ms)
06 28               T->O Conn. Parameters
01                  Transport type/Trigger
0b                  Connection Path Size (in words)
34 04 00 … 00       Electronic Key    (none)
20 04 24 03 2c 01 2c 02 Application Path:
      Class= 4, Configuration Inst= 3, Consume Inst= 1, Produce Inst= 2
80 01 aa 55         Data Segment (80 = Segment Type; 01 = # words; aa 55
                    is configuration data[0] & [1]
```

4-4.2.2 ForwardOpen Reply

This is not significantly different than the previous reply.

4-4.3 I/O Data Connection: Status, No Configuration Data

The example that follows shows the typical connection opening request/response for Controller I/O Data connections with devices in the ControlLogix controller's I/O tree. This example correlates to the previous examples given where the input size and output size are both 1 DINT, but this specifies zero Configuration size and specifies a status connection with the device that is 2 bytes long. The connection Format chosen is Data –DINT- With Status.

In this example two connections are opened: One for I/O data and the other for Status data.

4-4.3.1 ForwardOpen Request – I/O Data, No Configuration Data

The following is the first ForwardOpen Request generated by the ControlLogix in this example.

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 8c ed 78 00 00 40 06  7e 28 82 97 84 d2 82 97 |
0020  84 ca 04 09 af 12 e6 9e  e8 a2 24 55 ef 19 50 18 |
0030  40 00 b1 be 00 00 6f 00  4c 00 e3 eb 55 24 00 00 | Encapsulation Header
0040  00 00 82 97 84 d2 00 00  8c 07 00 00 00 00
                                                 00 00 | Start of Encaps Data
0050  00 00 00 00 02 00 00 00  00 00 b2 00 3c 00
                                                 54 02 | Start of FwdOpen Rqst
0060  20 06 24 01 05 9b 00 00  00 00 00 00 00 00 09 00 |
0070  01 00 e4 13 12 00 02 00  00 00 10 27 00 00 0a 48 |
0080  10 27 00 00 06 28 01 09  34 04 00 00 00 00 00 00 |
0090  00 00 20 04 24 03 2c 01  2c 02                   | End of FwdOpen Rqst
```

This ForwardOpen service breaks down as follows:

```
54                  Service Code
02 20 06 24 01      Path to service destination (size of path, class
                    segment, instance segment)
05 9b               Service priority/time per tick, Timeout (# of ticks)
00 00 00 00         O->T Connection Id
00 00 00 00         T->O Connection Id
09 00               Originator's Connection Serial Number
01 00               Originator's Vendor ID
e4 13 12 00         Originator's Serial Number
02                  Connection Timeout Multiplier (00 = 16 x RPI)
00 00 00            Reserved
10 27 00 00         O->T RPI (in microseconds) (0x00002710 * 1us = 10ms)
0a 48               O->T Conn. Parameters
10 27 00 00         T->O RPI (in microseconds) (0x00002710 * 1us = 10ms)
06 28               T->O Conn. Parameters
01                  Transport type/Trigger
09                  Connection Path Size (in words)
34 04 00 … 00       Electronic Key    (none)
20 04 24 03 2c 01 2c 02 Application Path:
      Class= 4, Configuration Inst= 3, Consume Inst= 1, Produce Inst= 2
```

4-4.3.2 ForwardOpen Reply - I/O Data

This is not significantly different than the previous replies.

4-4.3.3 ForwardOpen Request – Status Data

The following is the second ForwardOpen Request generated by the ControlLogix in this example.

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 8c ed 7a 00 00 40 06  7e 26 82 97 84 d2 82 97 |
0020  84 ca 04 09 af 12 e6 9e  e9 06 24 55 ef 87 50 18 |
0030  40 00 b3 e7 00 00 6f 00  4c 00 e3 eb 55 24 00 00 | Encapsulation Header
0040  00 00 82 97 84 d2 00 00  8c 06 00 00 00 00
                                                 00 00 | Start of Encaps Data
0050  00 00 00 00 02 00 00 00  00 00 b2 00 3c 00
                                                 54 02 | Start of FwdOpen Rqst
0060  20 06 24 01 05 9b 00 00  00 00 00 00 00 00 0a 00 |
0070  01 00 e4 13 12 00 02 00  00 00 10 27 00 00 02 48 |
0080  10 27 00 00 0a 28 01 09  34 04 00 00 00 00 00 00 |
0090  00 00 20 04 24 03 2c 05  2c 04                   | End of FwdOpen Rqst
```

This ForwardOpen service breaks down as follows:

```
54                Service Code
02 20 06 24 01    Path to service destination (size of path, class
                  segment, instance segment)
05 9b             Service priority/time per tick, Timeout (# of ticks)
00 00 00 00       O->T Connection Id
00 00 00 00       T->O Connection Id
0a 00             Originator's Connection Serial Number
01 00             Originator's Vendor ID
e4 13 12 00       Originator's Serial Number
02                Connection Timeout Multiplier (00 = 16 x RPI)
00 00 00          Reserved
10 27 00 00       O->T RPI (in microseconds) (0x00002710 * 1us = 10ms)
02 48             O->T Conn. Parameters
10 27 00 00       T->O RPI (in microseconds) (0x00002710 * 1us = 10ms)
0a 28             T->O Conn. Parameters
01                Transport type/Trigger
09                Connection Path Size (in words)
34 04 00 … 00     Electronic Key    (none)
20 04 24 03 2c 05 2c 04 Application Path:
      Class= 4, Configuration Inst= 3, Consume Inst= 5, Produce Inst= 4
```

4-4.3.4 ForwardOpen Reply – Status data

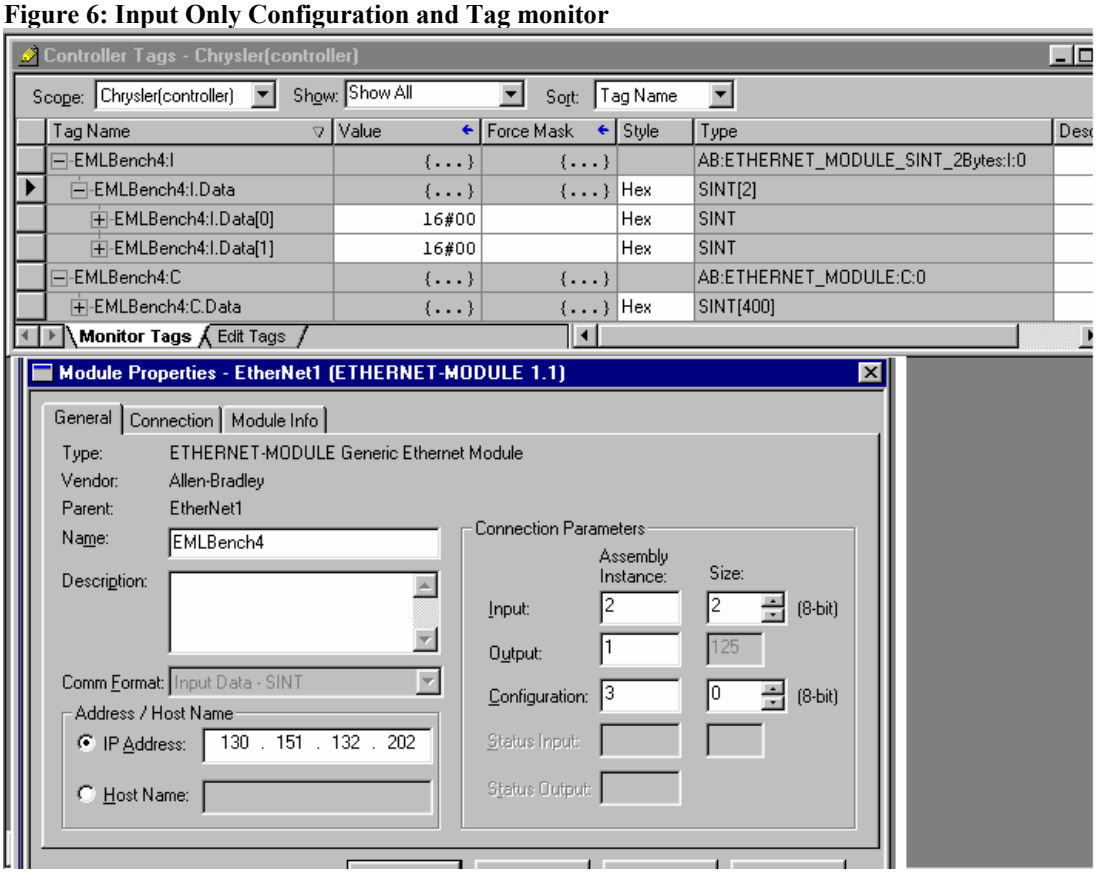
This is not significantly different than the previous replies.

4-4.3.5 I/O and Status Data Packets

4-4.4 Input Data Connection: No Status, No Configuration Data

The example that follows shows the typical connection opening request/response for Controller "**Input Data – SINT**" Comm Format Selection with a device in the ControlLogix controller's I/O Configuration tree. Please refer to Figure 6 for configuration options chosen and the resulting Controller Tag created for this connection.

Figure 6: Input Only Configuration and Tag monitor



4-4.4.1 ForwardOpen Request

The following is the ForwardOpen Request generated by the ControlLogix in this example.

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 8c ef b5 00 00 40 06  7b eb 82 97 84 d2 82 97 |
0020  84 ca 04 09 af 12 e6 9e  e9 ee 24 55 f0 61 50 18 |
0030  40 00 b9 2e 00 00 6f 00  4c 00 e3 eb 55 24 00 00 | Encapsulation Header
0040  00 00 82 97 84 d2 00 00  8c 03 00 00 00 00
                                                 00 00 | Start of Encaps Data
0050  00 00 00 00 02 00 00 00  00 00 b2 00 3c 00
                                                 54 02 | Start of FwdOpen Rqst
0060  20 06 24 01 05 9b 00 00  00 00 00 00 00 00 09 00 |
0070  01 00 e4 13 12 00 02 00  00 00 10 27 00 00 02 48 |
0080  10 27 00 00 04 28 01 09  34 04 00 00 00 00 00 00 |
0090  00 00 20 04 24 03 2c 01  2c 02                   | End of FwdOpen Rqst
```


This ForwardOpen service breaks down as follows:

```
54                  Service Code
02 20 06 24 01      Path to service destination (size of path, class
                    segment, instance segment)
05 9b               Service priority/time per tick, Timeout (# of ticks)
00 00 00 00         O->T Connection Id
00 00 00 00         T->O Connection Id
09 00               Originator's Connection Serial Number
01 00               Originator's Vendor ID
e4 13 12 00         Originator's Serial Number
02                  Connection Timeout Multiplier (00 = 16 x RPI)
00 00 00            Reserved
10 27 00 00         O->T RPI (in microseconds) (0x2710 * 1us = 10ms)
02 48               O->T Conn. Parameters
10 27 00 00         T->O RPI (in microseconds) (0x2710 * 1us = 10ms)
04 28               T->O Conn. Parameters
01                  Transport type/Trigger
09                  Connection Path Size (in words)
34 04 00 … 00       Electronic Key    (none)
20 04 24 03 2c 01 2c 02 Application Path
```

Sizes are: O->T 2 bytes for sequence count. This is the "heartbeat" message
T->O 4 bytes for sequence count and data (this example chose data size of 2 SINTs)

4-4.4.2 ForwardOpen Reply

This is not significantly different than the previous replies.

4-4.4.3 Input Only I/O Data Packets

4-4.4.3.1 T-O UDP Packet

```
0000  01 00 5e 40 1a 40 00 60  08 1c 48 e2 08 00 45 00 |
0010  00 32 cd 31 00 00 01 11  db 27 82 97 84 ca ef c0 |
0020  1a 40 08 ae 08 ae 00 1e  89 f6 02 00 02 80 08 00 |
0030  09 58 86 24 00 00 00 00  b1 00 04 00 02 00 00 00 |
```

4-4.4.3.2 O-T UDP Packet

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 30 00 01 00 00 40 11  6b f1 82 97 84 d2 82 97 |
0020  84 ca 08 ae 08 ae 00 1c  90 91 02 00 02 80 08 00 |
0030  08 58 86 24 00 00 00 00  b1 00 02 00 01 00       |
```

5 Produced Tags

The other type of I/O Message connection found in ControlLogix is the Produced Tag connection. This allows another device to make a direct connection to a tag in the controller's database. These connections are class 1, cyclic triggered, multicast transfers much like I/O Data connections are. The main difference is that rather than connecting to a predetermined I/O assembly, any tag in the controller that the user designates as a Produced Tag can be connected to.

Note: The user must select the Produce Tag option when creating the tag in the controller. Doing this makes the tag known to the Connection Manager as a potential connection target.

As previous paragraphs have alluded to, the ControlLogix controller assumes the role of connection *Target* in Produced Tag connections. Therefore a ForwardOpen must be received, requesting a specific Produced Tag in order for the controller to actually begin sending it on the network. The Originating controller can be any other node on the network as long as it follows the guidelines set forth in the remainder of this section.

Note: ControlLogix also has something called Consumed Tags. This is the ControlLogix mate for the Produced Tag. Since Consumed Tags can only be configured to exchange data with another ControlLogix controller, they will not be explained here.

5-1 Produced Tag Connections

The Produced Tag connection is unidirectional data flow, with a "heartbeat" in the return direction. A Produced tag is transmitted by the ControlLogix controller when another node initiates the ForwardOpen request for it. If no ForwardOpen is received by the controller, the tag will not be produced.

Many of the terms used in this description have been defined in the previous section. Therefore, this section will only cover what is unique about this type of connection as compared to connections to EIP I/O devices in the controller's I/O tree.

Note: a Produced Tag connection can be set up with any node on the network that is capable of originating connections that specify a symbol segment. The partner node (originator) does not need to be present in the Controller's I/O Configuration, although this is certainly allowed if the product functionality supports multiple connections and can simultaneously be a connection originator and a connection target.

5-1.1 Communications Formats and Packet Data Size for Produced Tags

There is no format selection for Produced Tags. They are transport class 1, cyclic triggered connections. The Tag data is an appropriate size for the Tag data type, plus 2 bytes for the Sequence Count. This flows in the T-O direction. In the O-T direction is a UDP packet with no application data, only a Sequence Count.

Note: Produced tags must have a data type that is 4 bytes or larger. Therefore, 8-bit (SINT) and 16-bit (INT) tags cannot be configured as produced tags. See the ControlLogix user manual Ref#4 for details of creating produced tags.

5-1.2 Connection Paths for Produced Tag Connections

This ForwardOpen is basically the same as all the previous ones in this document with one exception. The Connection Path uses an Extended Symbolic Segment instead of the more familiar Application Path consisting of a series of Logical segments. See the description of the ANSI Extended Symbol Segment in the CIP Common specification, Appendix C.

As mentioned earlier, the Produced Tag connection has a bi-directional aspect to it. The originator must produce a "heartbeat"packet at the API rate, using the Connection IDs agreed to in the ForwardOpen exchange. If this is not done, the controller will time out the Producing Tag connection and stop producing data.

The necessary connection parameters are described in the ControlLogix EDS file.

5-2 Traffic Capture of a Produced Tag Connection

The following traffic capture shows the opening sequence of messages from the ControlLogix controller to a generic EIP device. The tag to be requested in this example contains 11 characters, and is named "ProduceTag1".

5-2.1 ForwardOpen Request

```
0000  00 00 bc 05 0e fd 00 60  08 1c 48 e2 08 00 45 00
0010  00 94 d8 04 40 00 80 06  13 94 82 97 84 ca 82 97
0020  84 d2 04 b7 af 12 09 d0  85 be 96 35 35 02 50 18
0030  20 f4 df a8 00 00 6f 00  54 00 00 02 02 0e 00 00 Encapsulation Header
0040  00 00 82 97 84 ca 06 01  00 00 00 00 00 00
                                                  00 00 Start of Encaps Data
0050  00 00 00 04 02 00 00 00  00 00 b2 00 44 00
                                                  54 02 Start of FwdOpen Rqst
0060  20 06 24 01 0a f0 5b 95  d5 09 5a 95 d5 09 5c 95
0070  01 00 53 41 4d 50 02 00  00 00 a0 86 01 00 06 48
0080  a0 86 01 00 06 28 81 0d  01 00 34 04 00 00 00 00 Electronic Key
0090  00 00 00 00 91 0b 50 72  6f 64 75 63 65 54 61 67 Connection Path
00a0  31 00
```


This ForwardOpen service breaks down as follows:

```
54                  Service Code
02 20 06 24 01      Path to service destination (size of path, class
                    segment, instance segment)
0a f0               Service priority/time per tick, Timeout (# of ticks)
5b 95 d5 09         O->T Connection Id
5a 95 d5 09         T->O Connection Id
5c 95               Originator's Connection Serial Number
01 00               Originator's Vendor ID
53 41 4d 50         Originator's Serial Number
02                  Connection Timeout Multiplier (00 = 16 x RPI)
00 00 00            Reserved
a0 86 01 00         O->T RPI (in microseconds) (0x00002710 * 1us = 10ms)
06 48               O->T Conn. Parameters
a0 86 01 00         T->O RPI (in microseconds) (0x00002710 * 1us = 10ms)
06 28               T->O Conn. Parameters
01                  Transport type/Trigger
00                  Connection Path Size (in words)
34 04 00 … 00       Electronic Key    (none)
91 0b 50 72 6f 64 75 63 65 54 61 67 31 00
```

5-2.2 ForwardOpen Response

```
0000  00 60 08 1c 48 e2 00 00  bc 05 0e fd 08 00 45 00 |
0010  00 96 02 2a 00 00 40 06  69 6d 82 97 84 d2 82 97 |
0020  84 ca af 12 04 b7 96 35  35 02 09 d0 86 2a 50 18 |
0030  10 00 c9 7b 00 00 6f 00  56 00 00 02 02 0e 00 00 |
0040  00 00 82 97 84 ca 06 01  00 00 00 00 00 00 00 00 |
0050  00 00 00 04 04 00 00 00  00 00 b2 00 1e 00
                                                d4 00 | FwdOpen Response
0060  00 00 01 06 1d 00 81 06  1d 00 5c 95 01 00 53 41 |
0070  4d 50 a0 86 01 00 a0 86  01 00 00 00 00 80 10 00 |
0080  00 02 08 ae 00 00 00 00  00 00 00 00 00 00 00 00 |
0090  01 80 10 00 00 02 08 ae  ef c0 1b 40 00 00 00 00 |
00a0  00 00 00 00
```

The ForwardOpen reply breaks down as follows:

```
d4                  Service Reply
00                  Pad
00                  Status (00 = success)
00                  Extended Status Size
01 06 1d 00         Final O->T Connection ID
81 06 1d 00         Final T->O Connection ID
5c 95               Originator's Connection Serial Number
01 00               Originator's Vendor ID
53 41 4d 50         Originator's Serial Number
a0 86 01 00         O->T API
a0 86 01 00         T->O API
00                  Application reply size (none in this example)
00                  Pad
```

