Communicating with RA Products

Using

EtherNet/IP Explicit Messaging

Revision 1.2

Purpose

This document is a guide to the use of Ethernet/IP (EtherNet/Industrial Protocol[1] or EIP) TCP/IP-based, "explicit" messaging with AB products (primarily PLC5E, SLC5/05, and Logix5550 controllers). This document does not discuss EIP UDP/IP-based, "implicit" or "I/O" messaging. The former is a point-to-point, request/response (client/server) protocol typically used for unscheduled "information" messaging, while the latter is a multicast, producer/consumer protocol designed for scheduled "control" data transfer.

The term "explicit" refers to an address which explicitly details the path to a specific target object using an Path or Internal Object Identifier (EPATH or IOI). An "implicit" address is a Connection Identifier (CID) for a specific object between a client and server (or producer and consumer). A connected, explicit message actually contains both, a CID to the target device's Message Router object, and an EPATH (IOI) to the specific target object within the device. An unconnected, explicit message contains only an EPATH (IOI).

Note: This document only discusses explicit "messaging", not implicit "IO data transfer".

Where possible this document references existing sources of information rather than duplicating them.

Contents

[1] EtherNet/IP and EtherNet/Industrial Protocol are registered trademarks of ControlNet International

1. REFERENCES

The following references are available electronically for free at the RA websites: "www.ab.com" and "www.theautomationbookstore.com"), except for Ref#7 and 10 which are available from the EtherNet/IP website (www.ethernet-ip.org) hosted by CI (ControlNet International) and ODVA, and Ref.8 and 9, which are available from the CI website (www.controlnet.org).

- **Ref#1.** "DF1 Protocol and Command Set Reference Manual", Pub. 1770-6.5.16, Oct. 1996.
- **Ref#2.** "Logix5000 General Instructions Reference Manual", (Pub. 1756-RM003C-EN-P - November 2000)
- **Ref#3.** "Logix5000 Controllers Common Procedures Programming Manual", publication 1756-PM001B-EN-P).
- **Ref#4.** "Logix5000 Data Access Reference Manual", Publication 1756-RM005A-EN-E
- **Ref#5.** "ControlLogix System User Manual", publication 1756-UM001B-EN-P - November 2000)..
- **Ref#6.** "Integrated Architecture Logix Platforms", catalog number 1756-CD-DEC00 (part no. 957173-04)
- **Ref#7.** "EtherNet/IP Example Code Reference Guide" (guide to example source code)
- **Ref#8.** "ControlNet Specification", from ControlNet International
- **Ref#9.** "Real-Time Control on Ethernet" , a ControlNet International white-paper.
- **Ref#10.** "EtherNet/IP Specification" (Release 1.0, 05-June 2001, see above)
  - **Volume 1:** CIP Common Specification
  - **Volume 2:** EtherNet/IP Adaptation of CIP
- **Ref#11.** "1761-NET-ENI User Manual", (Pub.1761-un006a-en-p)
- **Ref#12.** "RSLinx 2.1 Service Pack Notes: ControlLogix Optimization", http://www.software.rockwell.com/download/comms/rslinx/clx_perf.zip
- **Ref#13**, "EtherNet/IP Performance and Application Guide: Application Solutions" (Pub. ENET-AP001A-EN-P - November 2000).

2. OVERVIEW

PLC5 and SLC500 processors communicate using PCCC (Programmable Controller Communication Commands) application-level commands (Ref#1). Ethernet TCP/IP connectivity was originally added to the SLC5/05 and PLC5E using the proprietary Client Server Protocol ("CSP") protocol, with embedded PCCC commands.

Later, the ControlLogix system was developed with a new communications architecture based on the **"Control and Information Protocol" (CIP)**. CIP is the application-level, object-oriented, connection-based communication architecture which is **shared by ControlNet, DeviceNet, and EtherNet/IP** to achieve interoperability and interchangeability among products from different vendors. EtherNet/IP is also referred to as "CIP over Ethernet". For a brief overview of the protocol, please refer to the White Paper, "Real-time Control on Ethernet" (Ref#9) especially the sections on "ControlNet Services over Ethernet" and "The Nuts and Bolts Version".

ControlLogix processors support direct, symbolic addressing of Tag data in "native" mode (i.e. without PCCC commands) using EIP explicit messages. For compatibility with existing serial devices, the ControlLogix DF1 serial channel supports the use of PLC2, PLC3, PLC5, and SLC PCCC commands to access Tag data that has been mapped to "compatibility files" (note: limited direct access to "simple" Tags is possible with PCCC Typed R/W using symbolic addressing; see section 11). And to allow peer-to-peer Ethernet communication between different families of Allen-Bradley processors, the ControlLogix, PLC5E and SLC5/05 added support for EIP with embedded PCCC commands ("EIP/PCCC").

The present PLC5E and SLC5/05 support both CSP and EIP protocols (“dual-stack”) concurrently on the same Ethernet cable at the same time. The RSLinx communication software normally communicates to PLC5E and SLC5/05 using the legacy CSP/PCCC protocol, but uses EIP/native for accessing ControlLogix data. RSLinx version 2.3 can be configured to communicate with PLC5 and SLC5/05 using EIP as well. (???). ControlLogix user programs use EIP/PCCC messaging to access RSLinx as a server.

EIP messaging is described in "Introduction to EtherNet/IP" in the EtherNet/IP Specification (Ref.#10, chapter 2, Volume 2), available for free download at the CI and ODVA websites. The TCP/IP client/server, explicit messaging portion of EIP is typically used to collect information. The chart below describes which Ethernet protocols are used between Allen-Bradley Processors.

Explicit Messaging Protocols Used Between RA Ethernet Devices

SERVER >	MicroLogix	SLC 5/05	PLC5E	ControlLogix	RSLinx
CLIENT					
MicroLogix via NET-ENI	EIP/PCCC	EIP/PCCC	EIP/PCCC	EIP/PCCC	EIP/PCCC
SLC 5/05	EIP/PCCC	CSP/PCCC	CSP/PCCC	EIP/PCCC	CSP/PCCC or EIP/PCCC
PLC5E	EIP/PCCC	CSP/PCCC	CSP/PCCC	EIP/PCCC	CSP/PCCC or EIP/PCCC
ControlLogix	EIP/PCCC	EIP/PCCC	EIP/PCCC	EIP/native	EIP/PCCC
RSLinx	EIP/PCCC	CSP/PCCC EIP/PCCC	CSP/PCCC EIP/PCCC	EIP/native	CSP/PCCC or EIP/PCCC

Refer to Appendix A for a list of the MSG instructions available in the ControlLogix ladder program instruction set, along with information about unconnected and connected messaging. EIP messages can be either **unconnected** (with explicit address only) or **T3 connected** (Transport Class 3, with both CID address for the connection to the Message Router (MR) and an explicit address (EPATH or IOI) interpreted by the MR to locate the target object. A single T3 connection to the MR can be used to access more than one object, but only one message at a time is permitted.

The EtherNet/IP protocol also includes implicit messaging using UDP/IP data transfer. This is the "EtherNet for Control" protocol for producer/consumer, scheduled control-messaging using implicit addresses (connections), which is also documented in the new EtherNet/IP Spec. EIP "scheduled" transfers differ from ControlNet "scheduled" transfers in that EIP only schedules the cyclic transfer time, without reserving network bandwidth to ensure delivery, where ControlNet does both. To that extent, EIP implicit tranfers are "non-deterministic", although a lightly-loaded ethernet should have sufficient bandwidth available.

The current PLC5E, SLC5/05 and MicroLogix (via NET-ENI) processors support EIP messaging, but not IO transfer. ControlLogix now supports EIP scheduled IO data transfer with the 1756-ENET/B module, but there are no plans at this time to add this to PLC5E, SLC5/05 and MicroLogix. EIP IO data transfer uses T1 connected (Transport Class 1, with implicit address, i.e. connection ID). T1 (Transport Class 1) connections for scheduled IO data transfer are made directly to a "connection point" on the target object itself.

The following chart categorizes the message Transport Classes supported by EtherNet/IP.

	Explicit Messaging Unscheduled TCP/IP	Implicit Data Transfer Scheduled UDP/IP
Unconnected	UCMM	n/a
Connected	Class 3 (T3)	Class 1 (T1)

The chart below indicates the RA Controller and IO products that will be capable of EIP(implicit).

I/O Control with EtherNet/IP Implicit Messaging

I/O Family	1771	1746	1794	1756	1769
Controllers					
SLC 5/05	No	No	No	No	No
PLC5E	No	No	No	No	No
ControlLogix	No	No	Yes	Yes	Future
SoftLogix5000	No	No	Future	Future	Future
IOLinx	No	No	No	No	No
MicroLogix via NET-ENI	No	No	No	No	No

3. USING THE REFERENCES

For information on EtherNet/IP protocol, developers should check the EtherNet/IP website (www.ethernet-ip.org) which contains the following useful information:

- Schedule for the Developer Training Class, “How To Put Your Product on Ethernet”,
- EtherNet/IP Specification (Ref.#10)
- EtherNet/IP Developer Tools, example source code, "EtherNet/IP Example Code Reference Guide",
- General Ethernet information

To understand EtherNet/IP and the CIP concepts, developers should obtain the EtherNet/IP Specification. Developers also need to know how data is organized and accessed in the Rockwell Automation PLC's, and the details of PCCC commands. The purpose of this document is to present or locate that RA-specific information.

A developer needs to be familiar with RA products. Many ControlLogix manuals, tutorials, and a demo RSLogix5000 software are contained on the free CD (Ref#6 above) available from "The Automation

Bookstore". The "EtherNet/IP Example Code Reference Guide" is a user manual for the EtherNet/IP example source code.

4. WHICH RA PRODUCTS SUPPORT EIP?

Refer to Section 2 above, "Overview". All devices which communicate with the ControlLogix controller (via 1756-ENET or 1756-ENB) over Ethernet must use "EtherNet/IP". All future standard RA Ethernet products should support EtherNet/IP. The PLC5 and SLC5/05 have had this capability for some time, as well as the **RSLinx and Interchange communication products from RSI**. The PLC5E added EIP support in early 1999. There are several hardware platforms being supported (Ser. C, D, and E) with firmware upgrades for each. Although not currently the latest revision, the first **PLC5E** revisions to support IP were:

Series C, Revision N.1
Series D, Revision E.1
Series E, Revision D.1

The MicroLogix controllers support EtherNet/IP via the 1761-NET-ENI (Ref#11). The ENI does not utilize the same services as the SLC-5/05 does. At this time we do not have information available that discusses how to access the MicroLogix via the NET-ENI, other than what is described in the manual, Ref#11.
The **PLC5 "sidecar"** module first supported EIP at **Series B, Revision A**. The **SLC5/05** was updated with EIP support in May 1999, with **Series A** firmware revision **OS501, FRN5.** If the SLC5/05 is an earlier revision, it can be Flash updated with new firmware. **.** All **Series B and C** SLC5/05's support EIP.
At this time, other RA EIP products include the new PowerMonitor and PanelView products, and IO Adapters (1756-ENET, 1756-ENB, 1794-AEN, 1734-AEN, 1769-AEN)

5. WHERE IS THE EIP PROTOCOL DOCUMENTED?

The EIP protocol is now documented in the EtherNet/IP Specification which can be downloaded for free at the EtherNet/IP website (see Ref#10). Volume 1 is the CIP Common Spec will apply to ControlNet and DeviceNet as well, in the near future. The EIP specific parts are contained in Volume 2, "EtherNet/IP Adaptation of CIP". The EIP (explicit) messaging protocol has been in the ControlNet spec for some time, but it is called the "TCP/IP Encapsulation Protocol". It details how CIP messages are to be encapsulated within TCP/IP. The encapsulation protocol was added to the ControlNet Spec in Ver. 2., Errata 2, Nov.1999.

6. WHAT ARE THE MINIMUM REQUIREMENTS FOR AN EIP PRODUCT?

The complete set of requirements are described in the EtherNet/IP Specification (Ref. 10, Vol. 1 and 2).

A distinction is made between **two types of EtherNet/IP products**:

a) COTS - "commercial-off-the-shelf" products; must meet all the EtherNet/IP requirements as noted in Ref #10, Vol 2, Chapter 8-5 .
b) Industrial EtherNet/IP products - must meet **ALL** EtherNet/IP requirements as noted in Ref#10, Vol 2, Chapter 8-6.

COTS products requirements are not as strict as Industrial Rated products in the following areas:

1. Media and Physical Layer (Ref.10, Vol.1, Chapter 8-5 vs Chapter 8-6)

2. LED Indicators (Ref.10, Vol.1, Chapter 9-5).
3. Environmental Specifications (Ref.10, Vol.1, Chapter 8-4.1only applies to Industrial Rated products)

However, the basic requirements for all EIP devices are:

a) Obtain a license and vendor ID,
b) Implement the Ethernet/IP "Encapsulation Protocol" (Ref#10, Vol.2. Ch.2)
c) Implement the minimum set of CIP objects (attributes/services, behavior, data types, etc.)
d) Test to ensure compliance

Not including the physical requirements listed above, the following are **minimum requirements**:

- **Must obtain the EtherNet/IP Specification (Ref#10).**
  The spec can be downloaded for free from the CI or ODVA websites, for evaluation. To build a product for sale requires a license.

- **Must obtain a License and Vendor ID from CI or ODVA.**
  A license is required to use the technology in a product.
  For CIP networks, the Vendor ID can be accessed over the network via the ID Object, and is used in various message services, and for configuration and diagnostic purposes. The same Vendor ID can be used for EIP, ControlNet, and DeviceNet.

- **Must implement the EtherNet/IP "Encapsulation Protocol"**
  This protocol is used to embed CIP messaging within TCP/IP and UDP/IP, and is described in Ref#10, Vol.2, Ch.2.

- **Must implement at least a "Level 1 Message Server", including the minimum set of objects and their "required" attributes/services**

  Refer to Appendix A for a typical Object Model of a CIP device. More detail is in the EtherNet/IP Spec. (Vol.1, Ch.4 & 6; Vol.2, Ch.4). Detailed descriptions of objects are contained in the EtherNet/IP Spec. Object Libraries (Vol.1, Ch.5; Vol.2, Ch5.), and the sections on the CIP Communication Objects (Vol.1, Ch3). Note that most of the attributes/services are optional. Refer to the ControlNet Statement of Conformance example (Ref#8) for the 1771-ACN15 communications adapter. Even unsupported attributes/services may require a specific response (e.g. error response) to a request.

  Refer also to the EtherNet/IP Example Code Reference Guide (Ref#7) for additional description of a "Level 1 Message Server" and "Level 3 Message Client". Although it may appear that a device may only need "client" operation for its intended application, the minimum "server" functionality must be implemented to support the interoperability required by the Open standard.

  Unconnected Message Manager (Ref.10, Vol.1, Chapter 2.2)
  - not really an object; UCMM;
  - at least “low-end server” functionality required

  Message Router (Ref.10, Vol.1, Chapter 5-3 and 2-4)
  - including a Class 3 connection to MR

  Connection Manager (Ref.10, Vol.1, Chapter 3-5)
  - to support the connection to MR

  Identity Object (Ref.10, Vol.1, Chapter 5-2)

- can be accessed via unconnected message or via connection to MR

EtherNet Link Object (Ref.10, Vol.2, Chapter 5-4)
- ethernet specific counters and status

In addition, some method of configuring the network connection is needed, so almost all devices will implement the following object:

TCP/IP Interface Object (Ref.10, Vol.2, Chapter 5-3)
- for configuring a device's network interface

If the EIP server is more than "minimum", it will provides access to application data as well. Application objects provide this data. Refer to Ref.10, Vol.1, Chapter 4-8 for methods of accessing data. For "real-time" repetitive data, the EIP IO data transfer protocol should be considered rather than explicit messaging. However, if a device serves data to other devices **using explicit messaging**, it will need an application object similar to the following:

Assembly Object (Ref.10, Vol.1, Chapter 5-5)
- general data
- serve data to ControlLogix (unconnected messages)

PCCC Object (refer to section 9, 11, 12 of this document)
- a Rockwell vendor-specific object
- serve data to ControlLogix (unconnected messages)
- serve data to PLC5 or SLC5/05 (connected messages)

Data Table Object (refer to section 9 of this document)
- a Rockwell vendor-specific object
- serve data to ControlLogix (connected messages)

- **Must create a valid Device Profile** (Refer to Ref. 10, Vol.1, Ch. 6)

  A Device Profile is a description of the objects implemented by the product. A device conforming to a standard Device Profile adheres to a set of guaranteed device behavior. It contains an object model, data formats, configuration, and public interfaces. If a device supports one of the standard Device Profiles listed in the CIP Common Spec. (Ref.10, Vol.1, Chapter 6), the vendor must document which "optional" attributes/services are supported. A Generic Device Profile (Ref.10, Vol.1, Chapter 6-8) is available for devices which do not fit one of the other profiles. A vendor can also create a custom Vendor Specific profile as long as it meets profile requirements (Ref.10, Vol.1, Chapter 6-6).

- **Must create an EDS sheet** (Refer to Ref. 10, Vol.1, Ch. 7)

  An EDS, Electronic Data Sheet, will be used by EtherNet/IP Network Configuration tools.

- **Must comply with the Industry Standard protocols which are included in the EIP protocol by reference; such as TCP and IP, IEEE802.3, etc.**

  Refer to the list of "**Normative References**" in Ref.10, Vol.2, Chapter 1-3.1. Other references are provided in Chapter 1.4, which are not Normative.

  Refer to the list provided in Ref.10, Vol.2, Chapter 9-3.

- **Must pass the EIP Conformance Test and submit an EtherNet/IP "Statement of Compliance".**

  An Ethernet version of the conformance test tool is available from ODVA or CI, for use with the Statement of Compliance. This tool tests for presence of required objects and implementation and behavior of standard objects.

7. WHERE SHOULD YOU START WITH THE SPECIFICATIONS?

A good place to start is with the **CIP Common Specification** (Ref.10, Vol.1) and device models. Refer to Chapter 4 to understand how Objects are modeled. Then refer to Chapter 6 (Device Profiles), to understand CIP devices. Chapter 6-2 shows the Object Model of a CIP device, and Chapter 6-2.1, "All Object Present in a Device", discusses the **minimum set of required objects**. The Table in Chapter 6-7 lists all the **standard device profiles**. Chapter 6-8 details the requirements of a "Generic Device". Chapter 6-13 details the requirements of a "Communications Adapter". Then refer to Chapter 5, "Object Library", for the detailed specifications of the required objects, as well as other optional **standard objects**.

Once you understand device objects, refer to Chapter 3, "CIP Communications Object Classes", to understand **connected and unconnected messaging**, and the functions performed by the UCMM, MR, CM. Review Chapter 3-4 for descriptions of **Transport Class 3 (T3)** which is the transport used for connected explicit messaging. Transports 0 and 1 only apply to IO data exchange. Class 4 and above are used for special functions.

Finally, go to the the **EtherNet/IP Adaptation of CIP** Spec (Ref. 10, Vol.2) Ch.2, for a description of the "**Encapsulation Protocol**" which is used to perform CIP messaging over Ethernet. Chapter 2-3 mentions the TCP and UDP port number (0xAF12) assigned to the encapsulation protocol. Chapter 2-4 describes the encapsulated portion of the message, which starts with a 24 byte "encapsulation header". Table 2-4.2 lists the encapsulation commands that can be specified in the header. “SendRRData” (Chapter 2-5.7) is used for **unconnected** CIP messages, e.g. Fwd_Open message, and “SendUnitData” (Chapter 2-5.8) is used for **connected** CIP messages, such as the Class 3 message to the MR. Session management and timeouts are described in Chapter 2-6.5.

8. WHAT'S THE EASIEST WAY TO IMPLEMENT EIP?

Standard off-the-shelf interfaces include the 1761-NET-ENI (Ref#11), which provides connectivity for RS232/485 devices that support the DF1 protocol and PCCC command set (Ref#1), or gateway devices from Encompass partners (http://www.automation.rockwell.com/encompass/index.html) such as the ProSoft "ProLinx" family (http://www.prosoft-technology.com). or standard communication drivers such as **RSLinx (Win/NT) or Interchange (UNIX,VMS, etc.) from Rockwell Software** (http://www.software.rockwell.com). Interchange provides limited EtherNet/IP support; i.e. access to ControlLogix tags that have been "mapped" to PLC5 style messages, or native CIP access to simple tags.

If a standard driver or interface module is not an option, a custom development is required. Custom developments can be done from scratch, or use a licensed code stack, or be contracted to a third party developer. For a custom development, the first step is to obtain the Specification and Example Code from the EtherNet/IP website (www.ethernet-ip.com) and then attend a training course,. Or, for contracting the development to a third party, the EtherNet/IP, CI, and ODVA websites contain a list of experienced VADP's (Value Added Development Partner). Or, to obtain a licensed code stack, many VADPs support the Rockwell Automation EtherNet/IP Master Library (EML) "C" source code, including the "Scanner Stack" for IO scanning.

9. WHAT TYPES OF MESSAGES DOES A SERVER NEED TO HANDLE?

There are some differences in the types of messages or application objects various RA clients use, so a server needs to accept the types of messages its intended client device can send and support the types of application objects the client can access. However, in general the minimum "Level 1 Message Server" described in Section 6 above implies the need to support:

- unconnected **Fwd_Open** requests to establish a T3 connection to the MR

- read access to the required objects via unconnected messages and via T3 connected messages. Typical services would be **GetAttributeSingle** and **GetAttributeAll** (Ref.10, Vol.1, Appendix A).

- read access to application data either through an application object (e.g. Assembly or PCCC) or through native tag access using symbolic segments (Ref#10, Vol.1, Appendix C).

Refer to the table in Section 2, Overview, for the general types of messages RA client devices will use with RA server devices. For examples of sending messages from the Logix5550 using MSG instructions, refer to Ref#2 and Ref#5. Ref#2, page 3-25, discusses "Setting up a Path" for the message. Refer also to Appendix A of this document for a review of ControlLogix MSG types and their characteristics.

Where T3 connected messages are desired, PLC5E and SLC5/05 can generate "**Typed R/W**" messages using embedded PCCC commands. However, these products also support an older RA ethernet protocol in addition to EIP. To force use of EIP, choose "multi-hop" in RSLogix5 or 500 MSG instruction. This requires use of a "port" and "slot" number in the path to the target, since that is required for ControlLogix, so **any server device directly attached to ethernet must accept or ignore this part of the path**, which is how PLC5E and SLC5/05 are able to support EIP for peer-to-peer messages between themselves. However, between PLC5E and SLC5/05 the older RA ethernet protocol is more efficient, especially for large messages. Newer RA products such as ControlLogix only support the EIP protocol.

The only **connected** explicit messages that Logix5550 generates are "**Data Table R/W**" messages. Although the Data Table object itself is not public, a server can emulate its behavior by associating its application data items with "tags" or "names", and implementing a default object which is the target of all messages that use "symbolic addressing". The services, service codes, packet formats, and error codes for "Data Table R/W" messages are described in Ref#4 ("CIP Read Data" and "CIP Write Data" services). The Data Table object class code is not required with these messages.

Where **unconnected** messages are desired, the Logix5550 "**Generic CIP**" message can be used, but there is no equivalent EIP message with PLC5E and SLC5/05 An alternative for PLC5/SLC500 is to use ControlNet processors (PLC5C and SLC500 with Scanner) with a CIO instruction "Generic CIP" message option, and then bridge to Ethernet through the ControlLogix Gateway. If a combination of connected and unconnected messaging is acceptable, a server's PCCC object could be accessed by PLC5E and SLC5/05 connected messages, and by Logix5550 "PLC5 Typed R/W" unconnected messages. A server of PCCC commands should implement the "Diagnostic Status" PCCC command (CMD=06, FNC=03). Non-RA devices should use the "third party" Type codes (Extended Interface = 2F, and Extended Processor = 2E), using the format of the 1747-L20 Diagnostic Status on page 10-3 of Ref#1 as a guide.

Where highly-repetitive data transfers are required, implicit **IO data transfers** (Class 1) should be considered rather than explicit messaging. However, PLC5E and SLC5/05 will not support EIP IO data transfers.

10. WHAT ARE THE STEPS IN ACCESSING DATA FROM A SERVER?

The steps in establishing the TCP session are detailed in Ref.10, Vol.1, Chapter 2-6, "Session Management". This Chapter discusses establishing, maintaining, and terminating a session. Timeouts are discussed in Chapter 2-6.5, but is only "informative". A note in Chapter 3-3.3 mentions that the EtherNet/IP spec does not require specific TCP session timeouts, and does not require that a TCP session be closed if all CIP connections on that session are closed.

After a session is established, unconnected messages can be sent. In general, devices can handle multiple unconnected messages at the same time. Refer to Ref.10, Vol.1, Chapter 3-5.5.4 for packet formats and parameters of general unconnected messages. For connected explicit messages, a T3 connection to the MR must be established first, using an unconnected Fwd_Open Request (Chapter 3-5.5.2). Refer to Chapter 3-4.6.3 for the "behavior" of an explicit messaging connection. A MR connection can be used to access more than one object (e.g. Identity Object, PCCC object, Assembly Object, etc.), although only one can be accessed at any one time on a single connection.

EIP messages to a PLC5E or SLC5/05 must access the PCCC Object. Using connected messaging, a T3 message is sent to the MR with explicit identification of the internal PCCC Object and its "execute PCCC" service. Refer to Ref.10, Vol.1, Chapter 3-5.5.1.9 for information on "Connection Path". The EPATH (or IOI) is composed of segments defining the route to the PCCC object. Refer to Ref.10, Vol.1, Appendix C for path Segment Types and the encoding syntax.

.

11. WHERE ARE PCCC COMMANDS DESCRIBED?

The embedded PCCC commands which are required to read/write data from/to the PLC5, SLC5/05, and the ControlLogix are listed in the Ref#1, chapters 6 and 7. The manual identifies which commands are supported by the PLC5 and SLC5/05.

The PCCC commands supported by ControlLogix are listed in the Logix5550 Instruction Set Manual, Ref#2, Section 3, "Input/Output Instructions", under the "Selecting the Message Type" subsection. In summary, the following are supported by the Logix5550:

PLC5/PLC3/SLC505	Typed Read/Write	(FNC 67, 68)
PLC5/PLC3	Word Range Read/Write	(FNC 00, 01)
SLC	Typed Read/Write	(FNC A2, AA, AB)
PLC2	Unprotected Read/Write	(CMD 01, CMD 08)

In order to access tag data in the Logix5550 using PCCC commands, it must be "mapped" to "PLC5 style Integer Files". Refer to Appendix A of the Logix5550 System User Manual, Ref#5, for instructions on mapping addresses. Any tag data can be mapped. Also, system data that is retrieved with a GSV instruction can be mapped.

Although "mapping" Tag data to PLC5 style integer files is the more general approach, ControlLogix does support direct access (non-mapped) to a few types of Tag data using PCCC commands with symbolic addresses. Use of a Symbolic system address is limited to Tag data with the following characteristics:

a. only PLC5 style PCCC commands
b. only the following symbolic form of system address is supported
"tag_name"
"tag_name[x]"
"tag_name[x,y]" or "tag_name[x][y]"
"tag_name[x,y,z]" or "tag_name[x][y][z]"
c. only the following data types, or arrays of these types:
SINT, INT, DINT, REAL

This limited symbolic addressing is also supported by the SLC5/05 but not by PLC5E, and both support Logical ASCII addressing which is similar. The most general approach for accessing ControlLogix data is to use native CIP messages, which do not contain PCCC commands. The "CIP Data Table Read/Write" and "Generic CIP" commands referenced in this section are "native" CIP messages. Refer to Ref#2, Ref#4, Ref#5 and Appendix B of this document for more information on ControlLogix messaging. An additional "native" CIP message, Read-Modify-Write, is documented here in Appendix C.

12. HOW IS A PCCC COMMAND EMBEDDED IN AN EIP MESSAGE?

The PLC5, SLC5/05, and Logix5550 support a Rockwell Automation vendor-specific PCCC Object that can execute PCCC commands. The PCCC object does not support a connection itself, but its "Execute PCCC" service can be accessed using a Class 3 connection to the Message Router.

The following Allen-Bradley publications contain a brief description of the **PCCC Object (Class Code = 67h)** and the message structure for **"Execute_PCCC" service (Service Code - 4Bh)** :

1. "Enhanced DeviceNet Communications Module User Manual", pub. 1203-5.12 - May 2000, page C-25 (http://www.ab.com/manuals/dr/1203/1203-5.13.pdf)
2. "ControlNet Communications Module User Manual", pub. 1203-5.13 - July 1998, page D-30. (http://www.ab.com/manuals/dr/1203/1203-5.13.pdf)

Refer to Appendix C at the end of this document for a description of CIP packet formats with embedded PCCC commands, and to Appendix D for example packets.

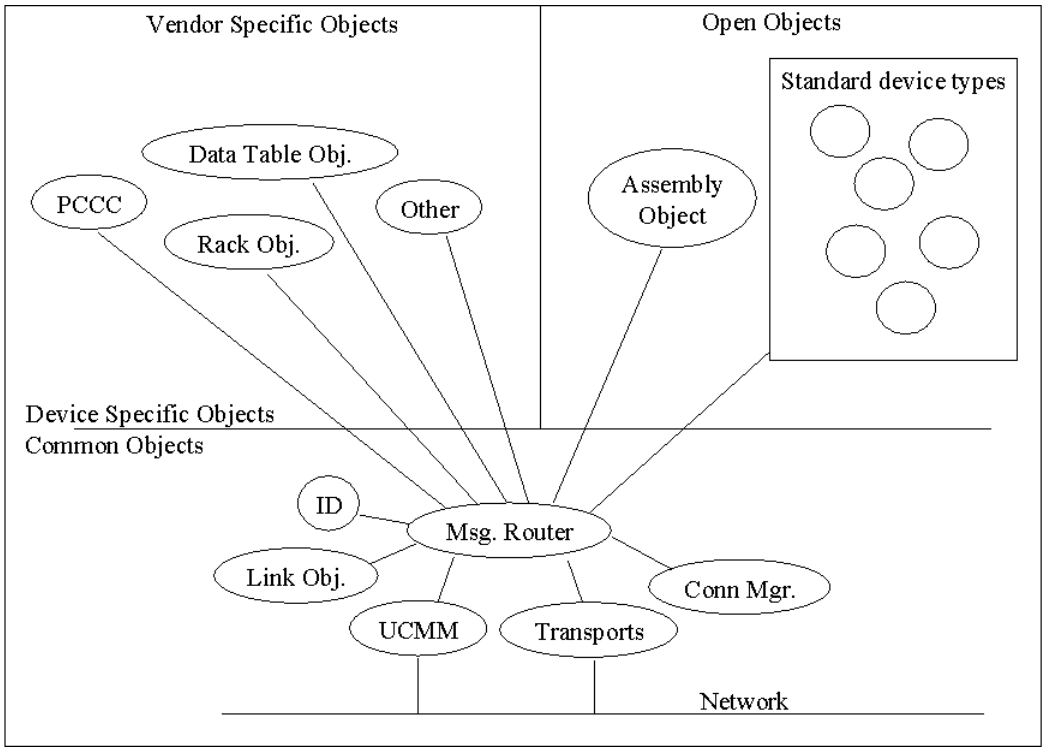
13. HOW IS A MESSAGE SENT THROUGH CONTROLLOGIX GATEWAY?

Explicit messages can be sent through the ControlLogix Gateway from EtherNet/IP to a device on any CIP network (EtherNet/IP, ControlNet, DeviceNet) just by constructing the proper path. Refer to Ref#2, page 3-25, for examples of message paths. Refer also to Appendix C of Ref#10, Vol.1, for CIP path and segment information.

14. WHAT PERFORMANCE DATA EXISTS FOR EIP EXPLICIT MESSAGING?

Refer to Ref#12 and Ref#13 for performance considerations.

Appendix A: CIP Device Object Model



Note: The "Common Objects" are required; the others are optional.

Appendix B: Logix5550 Client MSG Instructions

Target Server	Message Type (RSLogix5000 MSG Instruction)	MSG Communication Method	Target Object	EPATH (IOI segments; logical or symbolic)	Msg Class	Notes
Any CIP Device	Generic CIP Read/Write		any	Logical or symbolic	unc	2
Logix5550	Data Table R/W, RMW		Data Table	symbolic	T3/mr	3, 7
PLC5/SLC on CIP network	PLC5, SLC Typed R/W PLC5 WRR/WRW	CIP or DH+ (local/remote)	PCCC	logical	unc	4
RSLinx workstation	PLC5 Typed R/W	CIP with Source ID	PCCC	logical	unc	
PLC5/3/2/SLC on DH+ (via 1756-DHRIO)	PLC5 Typed R/W PLC5 WRR/WRW PLC3 Typed R/W PLC3 WRR/WRR PLC2 Unprotect R/W		DH+ Interface		T3/obj	5
1771 Module over RIO (via 1756-DHRIO)	Block-Transfer R/W		Block Transfer		T3/obj	6

Notes

1. General

- Unconnected Messages (unc).

  An error is returned if target is too busy to handle the message. If the message is not successful for other reasons, there may be no acknowledgement. In this case, a timeout can be used to initiate retries.

  Unconnected messages may not be successful if the server (or any routing device in the path) is too busy to service the message when it is received. However, the client can implement application-level retries. Connected messages are generally considered more reliable since resources to handle the message have been reserved ahead of time. For devices on the same network link (no routing), where the server is known to have sufficient resources for the planned message load, unconnected messaging may be sufficient.

  Communication with embedded PCCC messages may be either unconnected or T3/mr connected. A ControlLogix MSG to a PLC5 is unconnected, but a PLC5 MSG to the ControlLogix is connected. For a device that must serve data to both a ControlLogix and a PLC5, the PCCC message provides a common method, although both unconnected and connected messages must be supported. Refer to Section 12 in this document for PCCC object class code and "Execute PCCC" service code.

- T3 Connected Messages and "caching".

  T3 messages must be preceded by a connection establishment message, which associates some reserved resources with a specific CID (connection ID). Target of the unconnected "FWD_OPEN" MSG is explicitly identified with an IOI containing Class (20=CM), Instance (24), and Device Segments. The Logix5000 will open the connection when the first MSG to that

device is attempted. The connection will be closed when the message completes, unless the connection is “cached”. Connections are closed when the Logix5000 goes to Program or Remote Program Mode.

The Logix5550 can cache 16 messages concurrently. If the cache is full, the oldest connection that is not active (e.g. not in progress) will be replaced in the cache with the new connection. If messages are active on all connections, the new message will be sent as a non-cashed connected message; i.e. the connection will open, the message sent, and then the connection closed. Infrequent messages would normally not be cached, to avoid tying up resources. Non-cached connected messages can be more efficient than unconnected messages when message lengths are long and the message is fragmented into multiple packets.

- T3/mr MSG.

  A single T3/mr connection (to MR, Message Router) can be used for MSG to any object known by that MR, although only one MSG can be active at a time on that connection. If a second MSG is tried concurrently to the same target device, another connection will be opened if resources are available.

  Technically a T3/mr MSG is a combination of explicit and implicit messaging, with the connection to the MR being implicit (CID), and the target object being explicitly indentified (IOI).

- T3/obj MSG.

  A message using a connection directly to an application object. A separate connection is required for each target. However, very few objects actually support "connection points". A common application object that does support a connection is the Assembly object. However, the ControlLogix does not provide a MSG instruction which supports direct connection to an Assembly Object.

  A T3/obj MSG is "explicit" but not "explicit", since the connection ID (CID) is used to address the object.

- IOI Segments (EPATH)

  An IOI (Internal Object Identifier) is an “explicit” address or path to an object, which is decoded by the Message Router in “unc” and “T3/mr” messages. The IOI is composed of a series of path "segments".

- Error Codes

  CIP message error codes are listed in Chapter 3 of Ref#2. PCCC command error codes are listed in Ref#1.

- Controller Modes

  MSG instructions stop executing when Logix5550 is in Program or Remote Program modes. T1 (Class 1) messaging continues. Logix5550 also continues to respond to messages as server.

- Objects

  CIP specs by group functionality together as "objects", where each object provides a set of services. This is different from the PCCC model, where there is a long list of services, some of which worked on some items (such as data tables) and others worked on other items (such as

ladder programs). Refer to EtherNet/IP Spec, Vol. 1, Chapter 5, Object Library for detailed object specs.

2. "Generic CIP" MSG type

   - An explicit, unconnected message which can be used to read/write the attributes of any object by using an explicit address (IOI); e.g.
     - Assembly Object
       Service= 0x0E (GetAttributeSingle):
       Class=0x04 (Assembly Object),
       Instance=0x01 for first instance,
       Attribute=0x03 for the Data,
     - Identity Object
       Service=0x01 (GetAttribAll),
       Class=0x01 (Identity)
       Instance=0x01 for first instance

3. "Data Table R/W"

   - Used for Peer-to-Peer between CIP based PLC's. The "Data Table" object implements the "Reading and Writing to named tags" function, and it is not a public object. However, it is the only option for a server that needs connected, explicit messaging with a Logix5550. Refer to Ref#4 for more information.

4. PCCC Object

   - PLC5 and SLC only understand PCCC messages, so these are embedded in a CIP message to a PCCC Object, which is not public. It is also possible to access Logix5550 tag data that has been "mapped" to a PLC5 type file using PLC5 style messaging.

5. DH+ Interface Object

   - Messages to devices on DH+ are handled via a proxy object (DH+ Interface Object, which is not public) in the 1756-DHRIO module.

6. Block Transfer Object

   - Messages to devices on Block Transfer modules are handled via a proxy object (Block Transfer Object, which is not public) in the 1756-DHRIO module.

7. Logix5000 Data Table Read-Modify-Write (RMW) Message

Note:This message will be documented in the next revision of Ref#4.

The Read Modify Write (RMW) service provides a means to modify Tag data with individual bit resolution. Within ControlLogix, the Tag data is read, the **OR modification masks** are applied, then the **AND modification masks** are applied, and finally the data is written back to the Tag. It can be used to modify a single bit within a Tag without disturbing other data. Its purpose is similar to the PLC5 style "Read Modify Write" PCCC command described on page 25 of Ref#4.

RMW Service Request Parameters

Name	Type	Description of Response Data	Semantics of Values
Size of masks	UINT	Size in bytes of modify masks	Only 1,2,4,8,12 accepte
OR mask(s)	DWORD[]	Array of OR modify masks	"1" mask sets bit to "1"
AND mask(s)	DWORD[]	Array of AND modify masks	"0" mask resets bit to "0"

Note: size of masks must be the same or smaller than the size of the data type being accessed. For complete "data integrity" (i.e. to avoid possibility of a mix of old and new data when modifying dynamic data) the size of the mask should match the size of the data type.

service code (4E)	IOI string	size of masks (1,2,4,8,12)	OR mask(s)	AND mask(s)

RMW Service Response Parameters

Status/Error Code	Extended Error	Description of Status/Error
00	n/a	success
03	n/a	Bad parameter, size > 12 or size greater than size of element
10	2101	Device state Conflict: keyswitch position. The requestor is attempting to change FORCE data in HARD RUN mode.
xx		For other error codes, please refer to page 13 of Ref#4.

service code (CE)	00	status

For the format of the "IOI string" and "status" field, please refer to Ref#4.

Example:

A "DINT" Tag is identified in IOI. To set the third bit to a "1", and reset sixth bit to "0", and leave other bits unchanged. size=00000004 (DINT = 4 bytes), OR mask=00000004, AND mask=FFFFFFDF.

Appendix C: CIP Encapsulation of PCCC Commands

Note: In the following document the terms 'CIP" refers to the communcation architecture which is shared between ControlNet, DeviceNet, and EtherNet/IP. The EtherNet/IP TCP/IP Encapsulation Protocol is contained in Ref. 10, Vol.2, Chapter 2.

CIP Communication Architecture

The Open System Interconnect (OSI) reference model from the International Standards Organization (ISO) is used to help structure this document. The table below shows how the layers from the model are discussed in this document.

Table 1: Open System Interconnect (OSI) Layers

OSI layer:	OSI layer name:	Function	Description
7	Application	Messaging	Message format and meaning
6	Presentation		
5	Session		not used in CIP
4	Transport	Transport	End to end data integrity
3	Network	Routing	Determining the path from originator to destination
2	Data Link		Rules for accessing a single link
1	Physical		Rules for encoding bits on the media

The CIP networks share a common understanding of the layers above the Data Link Layer.
.

Messaging

CIP "native" messaging application layer is the preferred messaging system for CIP products. However, through "encapsulation", it can support other application protocols, including PCCC. This is the Programmable Controller Communication Commands protocol of Rockwell Automation. PCCC is used on DH, DH+, RS–232/DF1 and DH–485. It is provided in the CIP system primarily to ease the communication to these "legacy" networks and the new CIP networks. These are both request/response protocols, with exactly one response expected for each request.

Transport

CIP defines a set of transport protocol classes. Each of these classes defines a header for the PDU and appropriate state machines. Each class provides a different capability. The application chooses which class to use for that application. Refer to Ref.10, Vol.1, Chapter 3-4.3.3 for information on the standard transport protocols.

Routing

CIP messages can be sent "**unconnected**" or "**connected**". Connected messages use a **Connection ID** (implicit address of, and path to, object), where unconnected messages use only an EPATH (or IOI, Internal Object Identifier, refer to ControlNet Spec) to explicitly identify the route and target object. These connections are opened by one end–point of the connection (connection originator) by sending an unconnected Fwd_Open message to the target. Connected messaging is more reliable than unconnected because it reserves buffer space for the message, and is therefore less likely to be blocked by other message traffic.

.

Interface modules, which connect to non-CIP networks, must know how to route messages from other network types. This information is normally contained in a “routing table”. The Allen-Bradley ControlLogix chassis can hold communication interfaces to each network type. Each network interface module is a “half–gateway” which translates between the native network and the CIP supported protocols.

EtherNet/IP Messaging - TCP/IP Encapsulation of CIP and PCCC

The PCCC messaging protocol is delivered across the CIP system using either unconnected or connected messaging. For example, a PLC5 sends a PCCC message to ControlLogix as a connected message, but the ControlLogix sends a PCCC message to a PLC5 as an unconnected message.

Unconnected messages and T3 connected messages are sent to the device's Message Router, which routes them to the target of the IOI (e.g. PCCC Object). Before sending a connected message, the client must request a Connection ID (CID) for that message from the Message Router using an unconnected Fwd_Open message. The connected message then uses the CID in combination with an explicit IOI for the target object such as the PCCC Object. The embedded PCCC command is then processed by the "Execute PCCC" service of the PCCC object.

Table 2: TCP/IP Encapsulation Packet Structure

Structure (bytes)	Field	Bytes	Type	Description
TCP/IP Header (54)	Ethernet Header	14		Source/target Ethernet address, protocol type
	IP Header	20		Source/taget IP address, protocol type
	TCP Header	20		Source/target TCP ports, sequence no., protocol type; Encapsulation target port = 0xAF12
Encapsulation Header (24) Msg#1	Command Code	2	UINT	**0x006F SendRRData - unconnected msg** **0x0070 SendUnitData - T3 connected msg**
	Data Length	2	UINT	Length of data section
	Session Handle	4	UDINT	Unique number set by target
	Status	4	UDINT	0x00000000 in request
	Sender Context	8	Array of 8 USHORT	Transaction ID set by requestor, allows multiple req.
	Options	4	UDINT	0x00000000 -Future use
Encapsulation Data for SendRRData SendUnitData Msg#1	Handle	4	UDINT	0x00000000 for CIP encapsulated messages
	Timeout	2	UINT	For router, in seconds; 0x0000 for SendUnitData
	CPF Common Packet Format	See table 3 and 4	See below	Variable no. of address and data items **0x00B2 Unc. Data Item - unconnected message** **0x00A1 Conn. Address Item - connected message**
Encapsulated Msg#2				May not be present
TCP/IP Trailer	CRC code	4		Check bytes

Table 3: CPF for an Unconnected Message (SendRRData, including Fwd_Open)

Structure	Field	Bytes	Type	Description
No. of Items	Item Count	2	UINT	0x0002
Address Item	Item Type	2	UINT	0x0000 NULL Address Item
	Item Data Length	2	UNIT	0x0000
	Item Data			Not present
Data Item	Item Type	2	UINT	0x00B2 Unconnected Data Item
	Item Data Length	2	UINT	0x00NN variable (bytes)
	Item Data **T-PDU**	var		Variable - **Transport PDU**

Table 4: CPF for an T3 Connected Message (SendUnitData)

Structure	Field	Bytes	Type	Description
No. of Items	Item Count	2	UINT	0x0002
Address Item	Item Type	2	UINT	0x00A1 Connected Address Item
	Item Data Length	2	UINT	
	Item Data	4	UDINT	**Connection ID (CID)**
Data Item	Item Type	2	UINT	0x00B1 Connected Data Item
	Item Data Length	2	UINT	0x00NN variable (bytes)
	Item Data **T-PDU**	var	variable	variable - **Transport PDU**

Table 5: Requestwith Embedded PCCC

(Refer to Ref.10, Vol.1, Chapter 2-4, and Vol.2, Ch.2)

Unconnected Msg	Encapsulation Header	SendRRData (6F)
	CPF	Null Addr. Item (00), Unconn. Data Item (B2)
	Unconnected Request (note: Unconnected Send only present when routing through a bridge node; not seen by target device) (Ref#10,Vol.1, Chapter 3-5.5.4)	CM Service Request 52 Unconnected Send
		MR Service Request 4B Object 67 (PCCC) With PCCC cmd
		Unconnected Send Route Path

Connected Msg Class 3	Encapsulation Header	SendUnitData (70)
	CPF	SendUnitData (A1,B1) With connection CID
	Connected Request (below)	MR Service Request 4B Object 67 (PCCC) with PCCC cmd

Structure	Field	Bytes	Type	Description
CM Service Request **Unconnected Send**	Service Code	1	USINT	**0x52** Unconnected Send service request code
	Size of Req_Path	1	USINT	0x02 Path Size in words
	Request_Path	size	Array byte	EPATH 20,06 (class, CM); 24,01 (Instance 1)
Service Request Data (NOTE: Unconnected Send portion only present when indirect path through a bridge node is needed; **final target device only sees MR Service Request**)	Prior/TimeTick	1	BYTE	Used to calculate request timeout
	Time-out _ticks	1	USINT	Used to calculate request timeout
	MR Service Size	2	UINT	Size of MR Service in bytes
	MR Service Request and Data (see below)			
	Pad	1	USINT	Present if MR Service Size is odd value
	Route_Path_Size	1	USINT	Size of path in words
	Reserved	1	USINT	0x00 reserved
	Route_Path	size	EPATH padded	Route to the target device e.g. 01 00 port 1 slot 0

Structure	Field	Bytes	Type	Description
Packet Number	Sequence Count	2	UINT	**NOT IN UNCONNECTED MSG**; requestor
Message Router Service Request	Service Code	1	USINT	**0x4B** Execute PCCC service request code
	Size of Req_Path	1	USINT	0x02 Path Size in words
	Request_Path	size	Array byte	EPATH 20,67 (class, PCCC); 24,01 (Instance 1)
MR Service Request Data	Execute_PCCC Requestor ID	1	USINT	Lenght of Requestor ID (in bytes) (vendor + s/n + other + 1)
		2	UINT	CIP Vendor ID of requestor
		4	UDINT	CIP serial number
		var	Array byte	"Other" - may not be present
	Execute_PCCC PCCC Command	1	USINT	CMD - Command byte; typically 0x0F or 0x06
		1	USINT	STS - 0x00 in request
		2	UINT	TNSW - Same value in request and response
		1	USINT	**FNC - not used for all CMDs**
		var	Array byte	PCCC CMD/FNC specific data 244 max

Table 6: Response with Embedded PCCC

(Refer to Ref.10, Vol.1, Chapter 2-4, and Vol.2, Ch.2)

Unconnected Msg	Encapsulation Header	SendRRData (6F)
	CPF	Null Addr. Item (00), Unconn. Data Item (B2)
	Unconn. Response (below)	MR Service Reply CB **OR** Unconnected Send Reply D2 Object 67 (PCCC)

Connected Msg Class 3	Encapsulation Header	SendUnitData (70)
	CPF	Conn. Addr. Item (A1), Conn. Data Item (B1) With connection CID
	Connected Response(below)	MR Service Reply CB Object 67 (PCCC)

Structure	Field	Bytes	Type	Description
Packet Number	Sequence Count	2	UINT	**NOT IN UNCONNECTED MSG;** requestor
Message Router Service Reply **OR** Unconnected Send Service Reply (REF#10, Vol.1, Chapter 3-5.5.4)	Reply Service	1	USINT	**0xCB** Execute_PCCC reply **(0xCB OR 0xD2 for Unconnected Send)**
	Reserved	1	USINT	0x00 reserved
	General Status	1	USINT	Status Codes (Appendix B of CIP Common) 0x00 for success
	Size of Additional Status	1	USINT	Number of words (16 bit) of additional status 0x00 for success
	Additional Status	var	Array of UINT	May not be present
	Remaining Path Size	1	USINT	**ONLY PRESENT IN REPLY TO UNCONNECTED SEND WITH ROUTE ERROR**
MR ServiceData	Execute_PCCC Requestor ID	1 2 4 var	USINT UINT UDINT Array byte	Length of requestor ID (vendor + s/n + other) CIP Vendor ID of requestor CIP serial number Other; May not be present
	Execute_PCCC PCCC Command	1 1 2 1 var	USINT USINT UINT USINT Array byte	CMD - Command byte; typically 0x4F or 0x6F STS - **0xF0 implies EXT_STS present below** TNSW - Same value in request and response **EXT_STS** Not always present (see STS above) PCCC CMD/FNC specific data; 244 max

Table 7: T-PDU of Unconnected Fwd_Open Request for Class 3 Connection
(Refer to Ref#10, Vol.1, Chapter 3-5.5.2)

Encapsulation Header	SendRRData (6F)
CPF	Null Addr. Item (00), Unconn. Data Item (B2)
Unconnected Request (below)	MR Service Request 54 Object 06 (CM)

Structure	Field	Bytes	Type	Description
Message Router Header for Request	Service Code	1	USINT	**0x54** Fwd_Open request code
	Size of Request_Path	1	USINT	0x02 Path Size in words
	Request_Path	size	Array of bytes	EPATH (or IOI) 20,06 (class, CM object); 24,01 (Instance 1)
Connection Parameters	Connection Priority/tick Time	1	BYTE	1-second ticks (priority ignored); e.g. 0x0A
	Connection Timeout Ticks	1	USINT	e.g. 0x0F Timeout is 245 seconds (ticks*tick time)
	O-T Conn ID	4	UDINT	Typ 0x0000 in request; Returned by target in reply
	T-O Conn ID	4	UDINT	CID chosen by originator
	Conn S/N	2	UINT	Chosen by originator
	Originator Vendor ID	2	UINT	From ID object (CIP vendor ID)
	Originator S/N	4	UDINT	From ID object
	Conn Timeout Multiplier	1	USINT	e.g. 0x07 "512" (mult*RPI) "inactivity" timeout
	Reserved	3	Array of 3 octet	0x00 each octet
	O-T RPI	4	UDINT	Requested RPI, originator to target, microseconds
	O-T connection parameters	2	WORD	
	T-O RPI	4	UDINT	Requested RPI, target to originator, microseconds
	T-O connection parameters	2	WORD	
	Transport Class/ Trigger	1	BYTE	**0xA3**, Server transport, class 3, application trigger
	Size of Connection Path	1	USINT	Number of 16 bit words in Connection Path 0x03 CLX via backplane 0x02 direct network device
	Connection Path	var	Padded EPATH	(from PLC5E MSG instruction) e.g. 0x010020002401 01= Backplane port of 1756-ENET 00= Logix5550 in slot 0 20 02 = Class segment, 02 is MR 24 01 = Instance segment, no. 1

Table 8: T-PDU of Unconnected Fwd_Open Reply for Class 3 Connection
(Refer to Ref#10, Vol.1, Chapter 3-5.5.2)

Encapsulation Header	SendRRData (6F)
CPF	Null Addr. Item (00), Unconn. Data Item (B2)
Unconnected Response (below)	MR Service Reply D4 Object 06 (CM)

Structure	Field	Bytes	Type	Description
Message Router Header for Response	Response Service	1	USINT	**0xD4**; Fwd_Open service response code
	Reserved	1	USINT	0x00 reserved
	General Status	1	USINT	Status Codes (Appendix B of CIP Common) 0x00 for success
	Size of Additional Status	1	USINT	Number of words (16 bit) of additional status 0x00 for success
	Additional Status	var	Array of UINT	May not be present
Connection Parameters	O-T Conn ID	4	UDINT	CID chosen by target
	T-O Conn ID	4	UDINT	From request
	Conn S/N	2	UINT	From request
	Originator Vendor ID	2	UINT	From request
	Originator S/N	4	UDINT	From request
	O-T API	4	UDINT	Actual PI, originator to target, microseconds
	T-O API	4	UDINT	Actual PI, target to originator, microseconds
	Size of Application Reply	1	USINT	Number of words (16 bit) in Application Reply
	Reserved	1	USINT	Reserved
	Application Reply	var	Array of BYTE	Application Specific Data

References in EtherNet/IP Spec (Release 1.0, 05-Jun-2001)

The following references are useful for understanding specific fields in the above tables, and the example packets in Appendix D of this document.

Volume 1,"CIP Common Spec"

1. Explicit Messaging and the UCMM, Chapter 2-2.1
2. Message Router Request/Response Formats, Chapter 2-4.
3. Connection Manager Object, Ch.3
   - Packet Sequence Count, Chapter 3-4.3.3
   - Object Specific Services, Chapter 3-5.5
   - Object Specific Service Parameters, Chapter 3-5.5.1
     - Connection Timeout Multiplier definition is not included in the CIP Common Spec and is povided here for convenience only. Please see the current version of the ControlNet Specification, Part 4, clause 4.2.4 for the official description of this parameter.

       … from the ControlNet Specification:
       The Connection Time-out Multiplier specifies the multiplier applied to the RPI to obtain the connection time-out value. Device shall stop transmitting on a connection whenever the connection times out, even if the pending close has been sent. The multiplier shall be as represented by the following:

       Value 0 = RPI x 4 (default)
       Value 1 = RPI x 8
       Value 2 = RPI x 16
       Value 3 = RPI x 32
       Value 4 = RPI x 64
       Value 5 = RPI x 128
       Value 6 = RPI x 256
       Value 7 = RPI x 512
       Values 8-255 = Reserved

   - Forward Open, Chapter 3-5.5.2
   - Forward Close, Chapter 3-5.5.3
   - Unconnected Send, Chapter 3-5.5.4
4. Connection Manager Object Instance Error Codes, Chapter 3.5.6
5. Paths and CIP Segments, Appendix C.

Volume 2, "EtherNet/IP Adaptation of CIP"

1. Encapsulation, Chapter 2
2. Encapsulation Messages, Chapter 2-4

   Note: Little-Endian vs. Big Endian.
   Standard internet packet byte-ordering is "big-endian", but "multi-byte integer fields in the encapsulation messages " use "little-endian" ordering (Chapter 2-4.1)
3. SendRRData, Chapter 2-5.7
4. SendUnitData, Chapter 2-5.8
5. Session Management (establishing, maintaining, terminating), Chapter 2-6
6. Common Packet Format Spec, Chapter 2-7
7. Unconnected Messages, Chapter 3-3.1

Appendix D: Example Explicit Message Packets

CIP with Embedded PCCC Commands

Example 1: **Connected "Typed Read**" from PLC5E to Logix5550

- Packet 1: Request CIP Class 3 Connection Open from PLC5E to Logix5550 in ControlLogix Slot 1
- Packet 2: Response to CIP Class 3 Connection Open from Logix5550 to PLC5E
- Packet 3: Request containing PCCC Command from PLC5E to Logix5550
- Packet 4: Response containing PCCC Reply from Logix5550 to PLC5E

Example 2: **Unconnected "Typed Read"** from Logix5550 to PLC5E

- Packet 5: Request containing PCCC Command from Logix5550 to PLC5E
- Packet 6: Response containing PCCC Reply from PLC5E to Logix5550

Native CIP Messages

Example 3: **Unconnected "Generic CIP"** "GetAttributesAll" of ID Object in SLC5/05 from ControlLogix, routed through ControlLogix Gateway to demonstrate "**Unconnected Send**" protocol.

Message Request route: Logix5550 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > 1756-ENET in slot 7 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > SLC5/05
Message Reply route: reverse of above request route

- Packet 7: Request ENET in slot 4 > ENET in slot 7 (includes Unconnected Send)
- Packet 8: Request ENET in slot 4 > SLC5/05
- Packet 9: Response SLC5/05 > ENET in slot 4
- Packet 10: Response ENET in slot 7 > ENET in slot 4

Example 4: **Connected "Data Table Read"** from Logix5550 to Logix5550

- Packet 11: Request CIP Class 3 Connection Open
- Packet 12: Response to CIP Class 3 Connection Open
- Packet 13: Request "Data Table Read"
- Packet 14: Response to "Data Table Read"
- Packet 15: Request CIP Connection Close
- Packet 16: Response to CIP Connection Close

Notes:
In the examples above, the ethernet devices have the following IP and Ethernet addresses, and the Logix5550 (L1) is in slot 1 of the ControlLogix chassis.

```
130.151.132.121   0000bc034d06      1756-ENET in slot 4
130.151.132.122   0000bc034d0c      1756-ENET in slot 7
130.151.132.123                     computer
130.151.132.124   0000bc1c0042      PLC5E
130.151.132.125   0000bc1d3200      SLC5/05
```

Packet 1: Fwd Open Request from PLC5 to ControLogix

Example 1: Class 3 connection from PLC5 to MR in CLX processor in slot 1

```
1589    0.0010  0000bc034d06  0000bc1c0042  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = 4ed7d4e8  Ack Number = dda83450
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertized window = 4380
    TCP:  Checksum = 0x868d
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 88 (0x0058 hex) bytes.

    0000  00 00 bc 03 4d 06 00 00  bc 1c 00 42 08 00 45 00 | ..¼.M...¼..B..E.
    0010  00 80 ae 49 00 00 40 06  be 0a 82 97 84 7c 82 97 | .€®I..@.¾.,-„|,-
    0020  84 79 af 13 af 12 4e d7  d4 e8 dd a8 34 50 50 18 | „y¯.¯.N×ÔèÝ¨4PP.
    0030  11 1c 86 8d 00 00 6f 00  40 00 00 04 02 02 00 00 | ..†_..o.@.......
    0040  00 00 00 00 00 01 00 01  00 08 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 30 00 54 02 | ..........².0.T.
    0060  20 06 24 01 0a 0e 07 01  00 42 00 01 00 0b 0b 00 |  .$......B......
    0070  01 00 42 00 1c bc 01 00  00 00 40 82 1f 00 12 43 | ..B..¼....@,...C
    0080  40 82 1f 00 12 43 a3 03  01 01 20 02 24 01       | @,...C£... .$...

==========================================================================
(tcp/ip header)
    0000  00 00 bc 03 4d 06 00 00  bc 1c 00 42 08 00 45 00 | ..¼.M...¼..B..E.
    0010  00 80 ae 49 00 00 40 06  be 0a 82 97 84 7c 82 97 | .€®I..@.¾.,-„|,-
    0020  84 79 af 13 af 12 4e d7  d4 e8 dd a8 34 50 50 18 | „y¯.¯.N×ÔèÝ¨4PP.
    0030  11 1c 86 8d 00 00
(encapsulation header)      6f 00  40 00 00 04 02 02 00 00 | ..†_..o.@.......
    0040  00 00 00 00 00 01 00 01  00 08 00 00 00 00
(encapsulation data)                                  00 00 | ................
    0050  00 00 00
(CPF)               00 02 00 00 00  00 00 b2 00 30 00
(Unc. Rqst)                                     54 02 | ..........².0.T.
    0060  20 06 24 01
(conn parameters)     0a 0e 07 01  00 42 00 01 00 0b 0b 00 |  .$......B......
    0070  01 00 42 00 1c bc 01 00  00 00 40 82 1f 00 12 43 | ..B..¼....@,...C
    0080  40 82 1f 00 12 43
(transport/path)            a3 03  01 01 20 02 24 01       | @,...C£... .$...

==========================================================================
54    Fwd_Open Request
07 01 00 42 proposed CID connection O>T (0x7a006080)
00 01 00 0b CID connection T>O (0x0b000100)
0b 00       Connection s/n
a3          class 3 transport
03 01 01 20 02 24 01 path (3word,port1(backplane),slot1,MR(2002),inst1(2401))
```

Packet 2: Fwd Open Response from ControLogix to PLC5

Example 1: Class 3 connection from PLC5 to MR in CLX processor in slot 1

```
1591    0.0010  0000bc1c0042  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = dda83450  Ack Number = 4ed7d540
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertized window = 3196
    TCP:  Checksum = 0x10f5
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 70 (0x0046 hex) bytes.

    0000  00 00 bc 1c 00 42 00 00  bc 03 4d 06 08 00 45 00 | ..¼..B..¼.M...E.
    0010  00 6e 69 4b 00 00 3c 06  07 1b 82 97 84 79 82 97 | .niK..<...,-„y,-
    0020  84 7c af 12 af 13 dd a8  34 50 4e d7 d5 40 50 18 | „|¯.¯.Ý¨4PN×Õ@P.
    0030  0c 7c 10 f5 00 00 6f 00  2e 00 00 04 02 02 00 00 | .|.õ..o.........
    0040  00 00 00 00 00 01 00 01  00 08 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 1e 00 d4 00 | ..........²...Ô.
    0060  00 00 80 60 00 7a 00 01  00 0b 0b 00 01 00 42 00 | ..€`.z........B.
    0070  1c bc 40 82 1f 00 40 82  1f 00 00 00             | .¼@,..@,.....µ ¡

================================================================================
(tcp/ip header)
    0000  00 00 bc 1c 00 42 00 00  bc 03 4d 06 08 00 45 00 | ..¼..B..¼.M...E.
    0010  00 6e 69 4b 00 00 3c 06  07 1b 82 97 84 79 82 97 | .niK..<...,-„y,-
    0020  84 7c af 12 af 13 dd a8  34 50 4e d7 d5 40 50 18 | „|¯.¯.Ý¨4PN×Õ@P.
    0030  0c 7c 10 f5 00 00
(encapsulation header)      6f 00  2e 00 00 04 02 02 00 00 | .|.õ..o.........
    0040  00 00 00 00 00 01 00 01  00 08 00 00 00 00
(encapsulation data)                                  00 00 | ................
    0050  00 00 00
(CPF)             00 02 00 00 00  00 00 b2 00 1e 00
(Unc. Response)                                       d4 00 | ..........²...Ô.
    0060  00 00
(conn params)   80 60 00 7a 00 01  00 0b 0b 00 01 00 42 00 | ..€`.z........B.
    0070  1c bc 40 82 1f 00 40 82  1f 00 00 00             | .¼@,..@,.....µ ¡
================================================================================
d4    Fwd_Open Reply
80 60 00 7a final CID connection O>T (0x7a006080)
00 01 00 0b CID connection T>O (0x0b000100)
0b 00       Connection s/n
```

Packet 3: T3 Connected Typed Read Request from PLC5 to ControlLogix

Example 1: read request for 8 words from mapped address N7:0 of CLX in slot 1

```
1594    0.0019  0000bc034d06  0000bc1c0042  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = 4ed7d540  Ack Number = dda83496
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 4380
    TCP:  Checksum = 0xb7c3
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 74 (0x004a hex) bytes.

    0000  00 00 bc 03 4d 06 00 00  bc 1c 00 42 08 00 45 00 | ..¼.M...¼..B..E.
    0010  00 72 ae 4a 00 00 40 06  be 17 82 97 84 7c 82 97 | .r®J..@.¾.,-„|,-
    0020  84 79 af 13 af 12 4e d7  d5 40 dd a8 34 96 50 18 | „y¯.¯.N×Õ@Ý¨4–P.
    0030  11 1c b7 c3 00 00 70 00  32 00 00 04 02 02 00 00 | ..·Ã..p.2.......
    0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 a1 00  04 00 80 60 00 7a b1 00 | ......¡...€`.z±.
    0060  1e 00 01 00 4b 02 20 67  24 01 07 01 00 42 00 1c | ....K. g$....B..
    0070  bc 0f 00 dd 36 68 00 00  08 00 07 00 07 00 08 00 | ¼..Ý6h..........

================================================================================
(tcp/ip header)
    0000  00 00 bc 03 4d 06 00 00  bc 1c 00 42 08 00 45 00 | ..¼.M...¼..B..E.
    0010  00 72 ae 4a 00 00 40 06  be 17 82 97 84 7c 82 97 | .r®J..@.¾.,-„|,-
    0020  84 79 af 13 af 12 4e d7  d5 40 dd a8 34 96 50 18 | „y¯.¯.N×Õ@Ý¨4–P.
    0030  11 1c b7 c3 00 00
(encapsulation header)      70 00  32 00 00 04 02 02 00 00 | ..·Ã..p.2.......
    0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
(encapsulation data)                                00 00 | ................
    0050  00 00 00 00
(CPF)                 02 00 a1 00  04 00 80 60 00 7a b1 00 | ......¡...€`.z±.
    0060  1e 00
(Conn Rqst)     01 00 4b 02 20 67  24 01
(Requestor ID)                           07 01 00 42 00 1c | ....K. g$....B..
    0070  bc
(PCCC)        0f 00 dd 36 68 00 00  08 00 07 00 07 00 08 00 | ¼..Ý6h..........

70 00 SendUnitData - connected message (0x0070)
a1 00 cpf - Connected Address Item; CID to follow (0x00a1)
80 60 00 7a CID (0x7a006080)
b1 00 cpf - Connected Data Item (0x00b1)
01 00 sequence count (0x0001)
4b    "Execute_PCCC" service request
0f 00 dd 36 68    Typed Read (cmd sts tns1 tns2 fnc)
07 00 07 00       PLC5 system address (mask, section, file(7), element(0))
```

Packet 4: T3 Connected Typed Read Reply from ControlLogix to PLC5E

Example 1: read reply for 8 words from mapped address N7:0 of CLX in slot 1

```
 1596    0.0237  0000bc1c0042  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = dda83496  Ack Number = 4ed7d58a
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertized window = 3196
    TCP:  Checksum = 0x63be
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 81 (0x0051 hex) bytes.

    0000  00 00 bc 1c 00 42 00 00  bc 03 4d 06 08 00 45 00 | ..¼..B..¼.M...E.
    0010  00 79 69 4d 00 00 3c 06  07 0e 82 97 84 79 82 97 | .yiM..<...,-„y,-
    0020  84 7c af 12 af 13 dd a8  34 96 4e d7 d5 8a 50 18 | „|¯.¯.Ý¨4–N×ÕŠP.
    0030  0c 7c 63 be 00 00 70 00  39 00 00 04 02 02 00 00 | .|c¾..p.9.......
    0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 a1 00  04 00 00 01 00 0b b1 00 | ......¡.......±.
    0060  25 00 01 00 cb 00 00 00  07 01 00 42 00 1c bc 4f | %...Ë......B..¼O
    0070  00 dd 36 99 09 11 42 41  44 41 54 41 44 41 54 41 | .Ý6™..BADATADATA
    0080  44 41 54 41 44 41 54                            | DATADAT./¤¡].£¡]
==============================================================================
(tcp/ip header)
    0000  00 00 bc 1c 00 42 00 00  bc 03 4d 06 08 00 45 00 | ..¼..B..¼.M...E.
    0010  00 79 69 4d 00 00 3c 06  07 0e 82 97 84 79 82 97 | .yiM..<...,-„y,-
    0020  84 7c af 12 af 13 dd a8  34 96 4e d7 d5 8a 50 18 | „|¯.¯.Ý¨4–N×ÕŠP.
    0030  0c 7c 63 be 00 00
(encapsulation header)      70 00  39 00 00 04 02 02 00 00 | .|c¾..p.9.......
    4000 00 00 00 00 00 00 00 00  00 00 00 00 00 00
(encapsulation data)                                00 00 | ................
    0050  00 00 00 00
(CPF)                 02 00 a1 00  04 00 00 01 00 0b b1 00 | ......¡.......±.
    0060  25 00
(Conn Reply)                01 00 cb 00 00 00  07 01 00 42 00 1c bc
(PCCC)                                                 4f | %...Ë......B..¼O
    0070  00 dd 36 99 09 11 42 41  44 41 54 41 44 41 54 41 | .Ý6™..BADATADATA
    0080  44 41 54 41 44 41 54                            | DATADAT./¤¡].£¡]

70 00       SendUnitData - connected message (0x0070)
a1 00       cpf - Connected Address Item; CID to follow (0x00a1)
00 01 00 0b       CID (0x0b000100)
b1 00       cpf - Connected Data Item (0x00b1)
01 00       sequence count (0x0001)
cb          "Execute_PCCC" service reply
4f 00 dd 36       Typed Read (cmd,sts,tns1,tns2)
99 09 11 42       flag,type(array),size(17byte),descriptor(integer,2byte)
```

Packet 5: Request containing PCCC Command from Logix5550 to PLC5E

Example 2: Unconnected message containing Typed Read Cmd

```
5    0.0000  0000bc1c0042  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = f6a19421  Ack Number = 443219a4
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0x7e06
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 71 (0x0047 hex) bytes.


    0000  00 00 bc 1c 00 42 00 00  bc 03 4d 06 08 00 45 00 | ..¼..B..¼.M...E.
    0010  00 6f b2 0a 00 00 3c 06  be 5a 82 97 84 79 82 97 | .o².....<.¾Z,—„y,—
    0020  84 7c af 13 af 12 f6 a1  94 21 44 32 19 a4 50 18 | „|¯.¯.ö¡”!D2.¤P.
    0030  0b 68 7e 06 00 00 6f 00  2f 00 00 03 02 00 00 00 | .h~...o./.......
    0040  00 00 00 00 00 01 00 28  1e 4d 00 00 00 00 00 00 | .......(.M......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 1f 00 4b 02 | ..........².....K.
    0060  20 67 24 01 07 01 00 f2  0c 02 00 0f 00 4c 09 68 |  g$....ò.....L.h
    0070  00 00 0a 00 00 24 4e 37  3a 30 00 0a 00          | .....$N7:0...G4.
```

Notes:
unconnected "Typed Read" from ControlLogix to PLC5

```
4b                "Execute_PCCC" service of PCCC object
20 67 24 01       PCCC object (67), instance 1
0f 00 4c 09 68    typed read PCCC command (cmd sts tns1 tns2 fnc) (Ref#1)
00 24 4e 37 3a 30 00    system address "$N7:0"
```

Packet 6: Response containing PCCC Reply from PLC5E to Logix5550

Example 2: Unconnected message containing Typed Read Reply

```
    6    0.0010  0000bc034d06  0000bc1c0042  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = 443219a4  Ack Number = f6a19468
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertized window = 3196
    TCP:  Checksum = 0x781c
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 79 (0x004f hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 1c 00 42 08 00 45 00 | ..¼.M...¼..B..E.
    0010  00 77 7b 2b 00 00 40 06  f1 31 82 97 84 7c 82 97 | .w{+..@.ñ1,—„|,—
    0020  84 79 af 12 af 13 44 32  19 a4 f6 a1 94 68 50 18 | „y¯.¯.D2.¤ö¡”hP.
    0030  0c 7c 78 1c 00 00 6f 00  37 00 00 03 02 00 00 00 | .|x...o.7.......
    0040  00 00 00 00 00 01 00 28  1e 4d 00 00 00 00 00 00 | .......(.M......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 27 00 cb 00 | ..........².'.Ë.
    0060  00 00 07 01 00 f2 0c 02  00 4f 00 4c 09 99 09 15 | .....ò...O.L.™..
    0070  42 55 55 aa aa 00 00 00  00 00 00 00 00 00 00 00 | BUUªª...........
    0080  00 00 00 00 00 00                               | ......Ä.¡_œ.¡_..
```

Notes:
unconnected "Typed Read" from ControlLogix to PLC5

cb "Execute_PCCC" service reply
4f 00 4c 09 typed read PCCC reply (cmd sts tns1 tns2) (Ref#1)
99 09 15 42 flag,type(array),size(21bytes),descriptor(integer,2byte)

Packet 7: CIP Request ENET slot 4 > ENET slot 7 (includes Unconnected Send)

Example 3: "Generic CIP" message (unconnected) from ControlLogix with "GetAttributeAll" to ID object in SLC5/05 at address 130.151.132.125. routed through ControlLogix Gateway to demonstrate "Unconnected Send" protocol.

Message Request route: Logix5550 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > 1756-ENET in slot 7 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > SLC5/05
Message Reply route: reverse of above request route

```
    7    0.0067  0000bc034d0c  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = 3eb75874  Ack Number = 3ccab25b
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0x9cda
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 78 (0x004e hex) bytes.


    0000  00 00 bc 03 4d 0c 00 00  bc 03 4d 06 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 76 7f 5d 00 00 3c 06  f1 02 82 97 84 79 82 97 | .v ].. <.ñ.,—„y,—
    0020  84 7a af 13 af 12 3e b7  58 74 3c ca b2 5b 50 18 | „z ¯.¯.>·Xt<Ê²[P.
    0030  0b 68 9c da 00 00 6f 00  36 00 00 01 02 00 00 00 | .hœÚ..o.6.......
    0040  00 00 00 00 00 01 00 05  00 37 00 00 00 00 00 00 | .........7......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 26 00 52 02 | ..........².&.R.
    0060  20 06 24 01 07 e9 06 00  01 02 20 01 24 01 0a 00 |  .$..é.... .$...
    0070  01 04 12 0f 31 33 30 2e  31 35 31 2e 31 33 32 2e | ....130.151.132.
    0080  31 32 35 00                                     | 125.\É.¡ ».¡ ...
```

Notes:
Message is generated by ENET4 in slot 4 to ENET7 in slot 7
ENET7 address is at TCP/IP level
CPF SendRRData for 38 bytes of data 00 02 00 00 00 00 00 b2 00 26 00
52 Unconnected Send (**in bold**, Ref.#10, vol.1, Chapter 3-5.5.4)
MR Service Request **01** 02 20 01 24 01 (GetAttributeAll,2 word,ID object,inst1)
CIP path to target SLC5/05 (130.151.132.125)
- 0a 00 size of path = 10 words
- 01 backplane
- 04 slot 4
- 12 port 2 with extended address bit (Ref.#10, vol.1, Chapter C-1.4.1)
- 0f extended address of 15 bytes

Packet 8: CIP Request ENET in slot 4 > SLC5/05

Example 3: "Generic CIP" message (unconnected) from ControlLogix with "GetAttributeAll" to ID object in SLC5/05 at address 130.151.132.125. routed through ControlLogix Gateway to demonstrate "Unconnected Send" protocol.

Message Request route: Logix5550 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > 1756-ENET in slot 7 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > SLC5/05
Message Reply route: reverse of above request route

```
    8    0.0128  0000bc1d3200  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = 372ef314  Ack Number = bc1e5b5b
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0x6fea
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 46 (0x002e hex) bytes.


    0000  00 00 bc 1d 32 00 00 00  bc 03 4d 06 08 00 45 00 | ..¼.2...¼.M...E.
    0010  00 56 7f 5e 00 00 3c 06  f1 1e 82 97 84 79 82 97 | .V ^..<.ñ.,—„y,—
    0020  84 7d af 13 af 12 37 2e  f3 14 bc 1e 5b 5b 50 18 | „}¯.¯.7.ó.¼.[[P.
    0030  0b 68 6f ea 00 00 6f 00  16 00 00 01 02 00 00 00 | .hoê..o.........
    0040  00 00 00 00 00 01 00 06  00 36 00 00 00 00 00 00 | .........6......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 06 00 01 02 | ..........².....
    0060  20 01 24 01                                     |  .$. e.¡ R.¡ ..Ÿ
```

Notes:
message is generated by ENET4 in slot 4 to SLC5/05
SLC5/05 target address is at TCP/IP level
CPF SendRRData for 6 bytes of data 00 02 00 00 00 00 00 b2 00 06 00
MR Service Request **01** 02 20 01 24 01 (GetAttributeAll,2 word,ID object,inst 1)

Packet 9: Response SLC5/05 > ENET in slot 4

Example 3: "Generic CIP" message (unconnected) from ControlLogix with "GetAttributeAll" to ID object in SLC5/05 at address 130.151.132.125. routed through ControlLogix Gateway to demonstrate "Unconnected Send" protocol.

Message Request route: Logix5550 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > 1756-ENET in slot 7 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > SLC5/05
Message Reply route: reverse of above request route

```
    9    0.0152  0000bc034d06  0000bc1d3200  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = bc1e5b5b  Ack Number = 372ef342
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 3196
    TCP:  Checksum = 0xe918
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 83 (0x0053 hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 1d 32 00 08 00 45 00 | ..¼.M...¼.2...E.
    0010  00 7b 0d ec 00 00 3c 06  62 6c 82 97 84 7d 82 97 | .{.ì..<.bl,—„},—
    0020  84 79 af 12 af 13 bc 1e  5b 5b 37 2e f3 42 50 18 | „y¯.¯.¼.[[7.óBP.
    0030  0c 7c e9 18 00 00 6f 00  3b 00 00 01 02 00 00 00 | .|é...o.;.......
    0040  00 00 00 00 00 01 00 06  00 36 00 00 00 00 00 00 | .........6......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 2b 00 81 00 | ..........².+.?
    0060  00 00 01 00 0e 00 13 00  01 06 60 00 00 32 1d bc | ..........`..2.¼
    0070  18 31 37 34 37 2d 4c 35  35 31 20 41 2f 46 20 2d | .1747-L551 A/F -
    0080  20 44 43 20 32 2e 31 39  20                     |  DC 2.19 S..¡ É.
```

Notes:
message is generated by SLC5/95 to ENET4
ENET4 address is at TCP/IP level
CPF SendRRData 43 bytes of data 02 00 00 00 00 00 b2 00 2b 00
MR Service Response
81 00 00 00 (GetAttribAll reply 81, 00, status 00, additional status 00)
01 00 vendor ID (1 = RA)
0e 00 (product type = 0eh)
13 00 (product code - 13h)
01 06 (major rev 1, minor rev 6)
60 00 (status 60h)
00 32 1d bc (serial number)
18 31 37 34 37 2d 4c 35 35 31 20 41 2f 46 20 2d 20 44 43 20 32 2e 31 39 20
(18 byte product name "1747-L551 A/F - DC 2.19 ")

Packet 10: Response ENET in slot 7 > ENET in slot 4

Example 3: "Generic CIP" message (unconnected) from ControlLogix with "GetAttributeAll" to ID object in SLC5/05 at address 130.151.132.125. routed through ControlLogix Gateway to demonstrate "Unconnected Send" protocol.

Message Request route: Logix5550 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > 1756-ENET in slot 7 > 1756 backplane > 1756-ENET in Slot 4 > (ethernet) > SLC5/05
Message Reply route: reverse of above request route

```
    10    0.0142  0000bc034d06  0000bc034d0c  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = 3ccab25b  Ack Number = 3eb758c2
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 3196
    TCP:  Checksum = 0xa467
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 83 (0x0053 hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 03 4d 0c 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 7b 81 9f 00 00 3c 06  ee bb 82 97 84 7a 82 97 | .{?B..<.î»,—„z,—
    0020  84 79 af 12 af 13 3c ca  b2 5b 3e b7 58 c2 50 18 | „y¯.¯.<Ê²[>·XÂP.
    0030  0c 7c a4 67 00 00 6f 00  3b 00 00 01 02 00 00 00 | .|¤g..o.;.......
    0040  00 00 00 00 00 01 00 05  00 37 00 00 00 00 00 00 | .........7......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 2b 00 81 00 | ..........².+.?
    0060  00 00 01 00 0e 00 13 00  01 06 60 00 00 32 1d bc | ..........`..2.¼
    0070  18 31 37 34 37 2d 4c 35  35 31 20 41 2f 46 20 2d | .1747-L551 A/F -
    0080  20 44 43 20 32 2e 31 39  20                     |  DC 2.19 .~.¡ e.
```

Notes:
message is generated by ENET7 to ENET4
ENET4 address is at TCP/IP level
81 GetAttribAll Reply
encapsulated data is exactly same as packet #9 (starts with 6f 00)

Packet 11: Request CIP Class 3 Connection Open

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    11    0.0239  0000bc034d0c  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = e9beb81e  Ack Number = e9ab8d0a
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0xd1b2
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 88 (0x0058 hex) bytes.


    0000  00 00 bc 03 4d 0c 00 00  bc 03 4d 06 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 80 aa de 00 00 3c 06  c5 77 82 97 84 79 82 97 | .€ªÞ..<.Åw,—„y,—
    0020  84 7a af 13 af 12 e9 be  b8 1e e9 ab 8d 0a 50 18 | „z¯.¯.é¾¸.é« .P.
    0030  0b 68 d1 b2 00 00 6f 00  40 00 00 02 02 00 00 00 | .hÑ²..o.@.......
    0040  00 00 00 00 00 01 00 18  1d ce 00 00 00 00 00 00 | .........Î......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 30 00 54 02 | ..........².0.T.
    0060  20 06 24 01 07 e8 00 00  04 80 80 68 00 17 07 22 |  .$..è...€€h..."
    0070  01 00 f2 0c 02 00 00 00  00 00 e0 70 72 00 f6 43 | ..ò.......àpr.öC
    0080  e0 70 72 00 f6 43 a3 03  01 01 20 02 24 01       | àpr.öC£... .$..»
```

Notes:

54 "Fwd_Open" request for T3 connection to MR (Ref#10, vol.1, sec. 3-5.5.2)

00 00 04 80 proposed O>T connection ID (CID)

80 68 00 17 T>O connection ID (CID)

07 22 connection s/n

path 01 01 20 02 24 01 (backplane (ENET port 01), 1756-L1 in slot 01, MR 02, instance 01)

Packet 12: Response to CIP Class 3 Connection Open

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    12    0.0362  0000bc034d06  0000bc034d0c  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = e9ab8d0a  Ack Number = e9beb876
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 3196
    TCP:  Checksum = 0x18ab
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 70 (0x0046 hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 03 4d 0c 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 6e 01 b6 00 00 3c 06  6e b2 82 97 84 7a 82 97 | .n.¶..<.n²,—„z,—
    0020  84 79 af 12 af 13 e9 ab  8d 0a e9 be b8 76 50 18 | „y¯.¯.é« .é¾¸vP.
    0030  0c 7c 18 ab 00 00 6f 00  2e 00 00 02 02 00 00 00 | .|.«..o.........
    0040  00 00 00 00 00 01 00 18  1d ce 00 00 00 00 00 00 | .........Î......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 1e 00 d4 00 | ..........²...Ô.
    0060  00 00 80 67 00 47 80 68  00 17 07 22 01 00 f2 0c | ..€g.G€h..."..ò.
    0070  02 00 e0 70 72 00 e0 70  72 00 00 00             | ..àpr.àpr......¡
```

Notes:

d4 "Fwd_Open" reply for success on opening T3 connection to MR. (Ref#10, vol.1, sec. 3-5.5.2)

80 67 00 47 selected O>T connection ID (CID)

80 68 00 17 T>O connection ID (CID)

07 22 connection s/n

Packet 13: Request "Data Table Read"

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    13    0.0010  0000bc034d0c  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = e9beb876  Ack Number = e9ab8d50
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0x3bd2
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 64 (0x0040 hex) bytes.


    0000  00 00 bc 03 4d 0c 00 00  bc 03 4d 06 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 68 aa e0 00 00 3c 06  c5 8d 82 97 84 79 82 97 | .hªà..<.Å ,—„y,—
    0020  84 7a af 13 af 12 e9 be  b8 76 e9 ab 8d 50 50 18 | „z¯.¯.é¾¸vé« PP.
    0030  0b 68 3b d2 00 00 70 00  28 00 00 02 02 00 00 00 | .h;Ò..p.(.......
    0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 a1 00  04 00 80 67 00 47 b1 00 | ......¡...€g.G±.
    0060  14 00 01 00 4c 07 91 0b  72 65 61 64 5f 76 61 6c | ....L.‘.read_val
    0070  75 65 73 00 01 00                               ues...¯..¡ ..¡ .
```

Notes:

4c T3 connected "Data Table Read" Request (Ref#4)

symbolic segment (**91**) of tag "**read_values**"

80 67 00 47 connection ID (CID O>T)

01 00 packet number (connected messages only)

Packet 14: Response to "Data Table Read"

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    14    0.0006  0000bc034d06  0000bc034d0c  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = e9ab8d50  Ack Number = e9beb8b6
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 3196
    TCP:  Checksum = 0x8bf3
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 54 (0x0036 hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 03 4d 0c 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 5e 01 b7 00 00 3c 06  6e c1 82 97 84 7a 82 97 | .^.·..<.nÁ,—„z,—
    0020  84 79 af 12 af 13 e9 ab  8d 50 e9 be b8 b6 50 18 | „y¯.¯.é« Pé¾¸¶P.
    0030  0c 7c 8b f3 00 00 70 00  1e 00 00 02 02 00 00 00 | .|‹ó..p.........
    0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
    0050  00 00 00 00 02 00 a1 00  04 00 80 68 00 17 b1 00 | ......¡...€h..±.
    0060  0a 00 01 00 cc 00 00 00  c3 00 8e e7             | ....Ì...Ã.•ç->.¡
```

Notes:

cc T3 Connected "Data Table Read" Reply (Ref#4)

80 68 00 17 connection ID (CID T>O)

01 00 packet number (connected messages only)

00 00 00 c3 00 8e e7 (00, status 00 00, integer type c3 00, value 0xe78e)

Packet 15: Request CIP Connection Close

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    15    0.0259  0000bc034d0c  0000bc034d06  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf13  (Data)
    TCP:  Destination port = 0xaf12  (Data)
    TCP:  Sequence Number = e9beb8b6  Ack Number = e9ab8d86
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 2920
    TCP:  Checksum = 0xbd23
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 64 (0x0040 hex) bytes.


    0000  00 00 bc 03 4d 0c 00 00  bc 03 4d 06 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 68 aa e1 00 00 3c 06  c5 8c 82 97 84 79 82 97 | .hªá..<.ÅŒ,—„y,—
    0020  84 7a af 13 af 12 e9 be  b8 b6 e9 ab 8d 86 50 18 | „z¯.¯.é¾¸¶é« †P.
    0030  0b 68 bd 23 00 00 6f 00  28 00 00 02 02 00 00 00 | .h½#..o.(.......
    0040  00 00 00 00 00 01 00 18  1d cf 00 00 00 00 00 00 | .........Ï......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 18 00 4e 02 | ..........².....N.
    0060  20 06 24 01 07 e8 07 22  01 00 f2 0c 02 00 03 00 |  .$..è."..ò.....
    0070  01 01 20 02 24 01                               | .. .$.M..¡ ..¡ .
```

Notes:

4e "Fwd_Close" request () to close the connection (Ref#10, Vol.1, Ch 3-5.5.3)

07 22 connection s/n

01 01 20 02 24 01 (backplane, slot 1, MR, instance 1)

Packet 16: Response to CIP Connection Close

Example 4: Connected "Data Table Read" from Logix5550 to Logix5550

```
    16    0.0010  0000bc034d06  0000bc034d0c  TCP Data

    TCP:  ----- Transmission Control Protocol header -----
    TCP:
    TCP:  Source port = 0xaf12  (Data)
    TCP:  Destination port = 0xaf13  (Data)
    TCP:  Sequence Number = e9ab8d86  Ack Number = e9beb8f6
    TCP:  Offset to data in this datagram = 20
    TCP:  Code = 18 ACK PSH
    TCP:  Advertised window = 3196
    TCP:  Checksum = 0xe3ce
    TCP:  Urgent data pointer = 0
    TCP:  Encapsulated Data data, size = 54 (0x0036 hex) bytes.


    0000  00 00 bc 03 4d 06 00 00  bc 03 4d 0c 08 00 45 00 | ..¼.M...¼.M...E.
    0010  00 5e 01 b8 00 00 3c 06  6e c0 82 97 84 7a 82 97 | .^.¸..<.nÀ,—„z,—
    0020  84 79 af 12 af 13 e9 ab  8d 86 e9 be b8 f6 50 18 | „y¯.¯.é« †é¾¸öP.
    0030  0c 7c e3 ce 00 00 6f 00  1e 00 00 02 02 00 00 00 | .|ãÎ..o.........
    0040  00 00 00 00 00 01 00 18  1d cf 00 00 00 00 00 00 | .........Ï......
    0050  00 00 00 00 02 00 00 00  00 00 b2 00 0e 00 ce 00 | ..........².ÎÎ.
    0060  00 00 07 22 01 00 f2 0c  02 00 00 00             | ..."..ò.....».¡
```

Notes:

ce "Fwd_Close" reply, success closing connection (Ref#10,vol.1,sec.3-5.5.3)

07 22 connection s/n