# INTER-ROLLER BAGGAGE HANDLING SYSTEM
# INTERFACE SPECIFICATION
# TRANSPORT PROTOCOL: EIP&CIP

| | |
|---|---|
| Document No.: | IR-102-08 |
| Project No.: | HLC-Standardization |
| Author: | XJ |
| Release Date: | 31-Jul-2008 |
| Revision: | 1.00 |
| Pages: | 62 |

**Responsible for the contents**

Inter-Roller Engineering Limited

Singapore

# Contents

## Table of Figure

# 1. REVISIONS

| Version | Release | Date | Init. | Description |
|---|---|---|---|---|
| 1 | 00 | 31-Jul-2008 | XJ | Initial version. |

# 2. INTRODUCTION

## 2.1 OVERVIEW

The Ethernet Industrial Protocol (EtherNet/IP, EIP) and Control & Information Protocol (CIP) is the protocol layer in the IR-BHS EIP&CIP protocol stack, which consists of a number of logical layers on top of Ethernet. The reader is referred to ODVA (Open DeviceNet Vendor Association) specifications (CIP Common Specification, and EtherNet/IP Adaptation of CIP specification) for further detail on the protocol stack.

The IR-BHS implementation of the EIP&CIP protocol does not a fully implement the EIP and CIP protocol as specified in the above 2 ODVA specifications. Rather the focus has been to create a simple and reliable communication protocol. This implementation is primarily developed for SAC developer to meet the requirements of communication between IR SAC and Rockwell ControlLogix PLC. The reader should be aware that the EIP and CIP protocols are implemented fully by the network card of a ControlLogix PLC, i.e. no user program necessary in PLC side to implement this protocol.

## 2.2 CIP CONNECTION

Though CIP protocol can be encapsulated by EIP protocol to support UDP and TCP transport layer protocols, in IR-BHS standard SAC to ControlLogix PLC interface design, there is only TCP transport layer protocol is required for exchange messages between SAC and ControlLogix PLCs.

As defined by the CIP protocol, all application layer messages (SAC application message) have to be encapsulated into the CIP messages before they are put on the CIP network. The CIP Transport Class 3 connection is required to be established prior to the exchanging of CIP messages.

In order to have the bi-direction communication between SAC and ControlLogix PLC, two CIP connections need to be opened, one for data sending and one for data receiving. The one who need actively send data is the originator (client) to be responsible for establishing a connection path to the target (server). Hence, SAC and PLCs have to support both originator and target roles.

For the CIP connection in which SAC is the originator (SAC actively send data to PLC), PLC acts as the TCP server to accept the TCP client (SAC) connection request to open the TCP connection. PLC is also the EIP server to accept the EIP client (SAC) RegisterSession request to open the EIP session. Similarly, PLC is also the CIP server to accept the CIP client (SAC) Forward_Open request to open the CIP connection.

For the CIP connection in which PLC is the originator (PLC actively send data to SAC), SAC acts as the TCP server to accept the TCP client (PLC) connection request to open the TCP connection. SAC is also the EIP server to accept the EIP client (PLC) RegisterSession request to open the EIP session. Similarly, SAC is also the CIP server to accept the CIP client (PLC) Forward_Open request to open the CIP connection.

Chapter below is the detail procedures of establishing the CIP connections.

### 2.2.1 Establishing a CIP Connection

In order to open a CIP connection for exchanging messages between SAC and RA ControlLogix PLC, the following connections must be established in advance according to following steps:

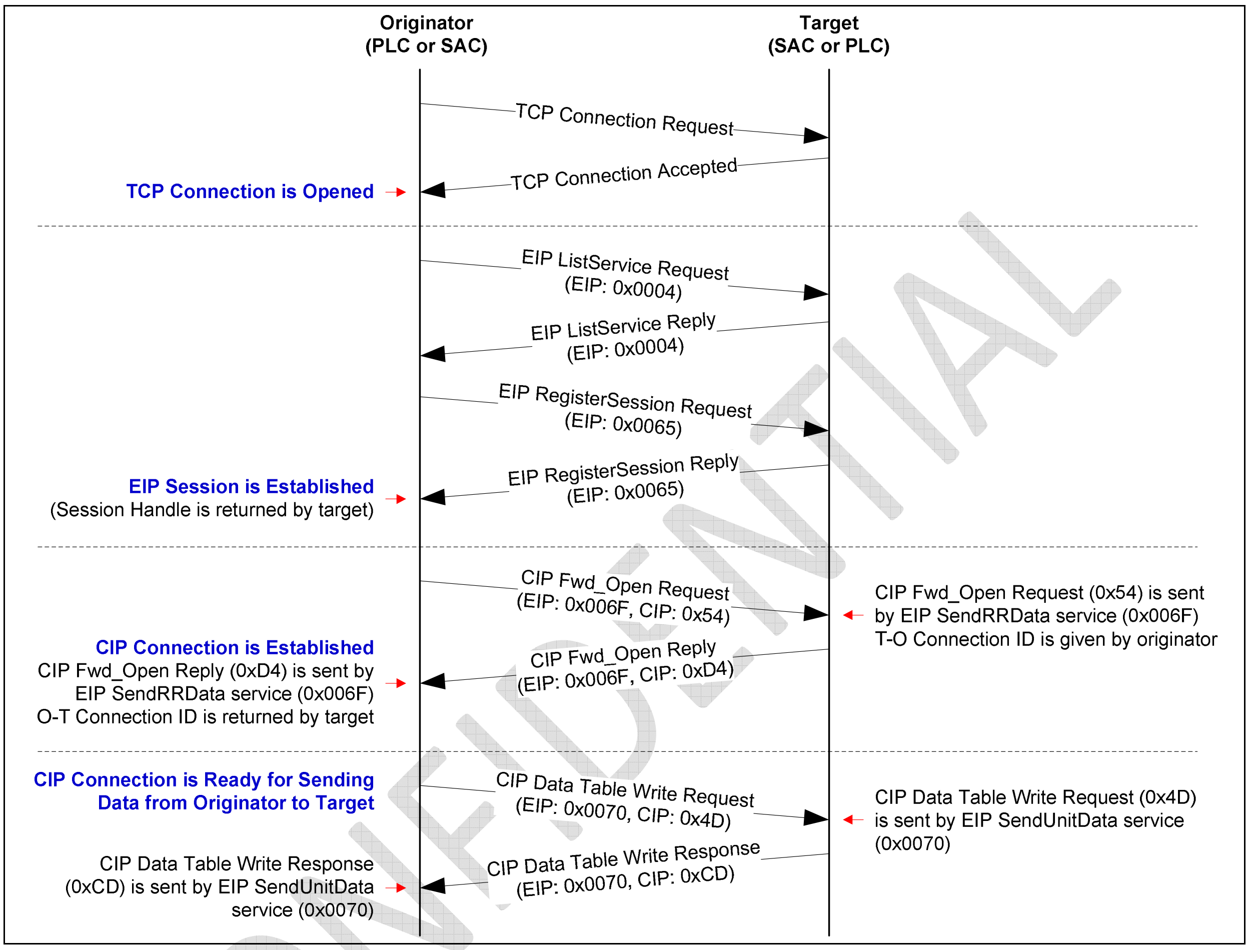

Originator (PLC or SAC)
Target (SAC or PLC)
TCP Connection Request
TCP Connection Accepted
TCP Connection is Opened
EIP ListService Request (EIP: 0x0004)
EIP ListService Reply (EIP: 0x0004)
EIP RegisterSession Request (EIP: 0x0065)
EIP RegisterSession Reply (EIP: 0x0065)
EIP Session is Established
(Session Handle is returned by target)
CIP Fwd_Open Request (EIP: 0x006F, CIP: 0x54)
CIP Fwd_Open Request (0x54) is sent by EIP SendRRData service (0x006F) T-O Connection ID is given by originator
CIP Fwd_Open Reply (EIP: 0x006F, CIP: 0xD4)
CIP Connection is Established
CIP Fwd_Open Reply (0xD4) is sent by EIP SendRRData service (0x006F) O-T Connection ID is returned by target
CIP Connection is Ready for Sending Data from Originator to Target
CIP Data Table Write Request (EIP: 0x0070, CIP: 0x4D)
CIP Data Table Write Request (0x4D) is sent by EIP SendUnitData service (0x0070)
CIP Data Table Write Response (EIP: 0x0070, CIP: 0xCD)
CIP Data Table Write Response (0xCD) is sent by EIP SendUnitData service (0x0070)

Figure 1: Establishing of CIP Connection

a. The originator shall open a TCP/IP connection to the target, using the reserved TCP port number **0xAF12** (**44818**);
b. The target will accept the TCP connection request to open the TCP connection;
c. If TCP connection is successfully opened, the originator may or may not send EIP ListService request to target to determine which encapsulation service classes the target device supports;
d. If target support encapsulation of CIP packets, then return EIP ListService reply to originator;
e. The originator shall send a RegisterSession Request (Service Code: 0x0065) to the target to open EIP session;
f. The target shall check the protocol version in the command message to verify it supports the same protocol version as the originator. If not, the target shall return a RegisterSession Reply (Service Code: 0x0065) with an appropriate Statue field along with the highest supported protocol version; If target is able to register the session, target shall assign a new (unique) Session Handle and shall return it to originator by sending a RegisterSession Reply (Service Code: 0x0065) to the originator.
g. If EIP session is successfully opened, the originator shall send a CIP Forward Open Request (Service Code: 0x54) message to target by using EIP SendRRData Request (Service Code:

0x006F) service. The CIP Fwd_Open Request message will be encapsulated into EIP SendRRData Request message and sent to target via opened TCP connection;

h. If target accept the CIP Forward Open Request, it will return the O-T network connection ID to original by sending CIP Fwd_Open Response message via EIP SendRRData Reply (Service Code: 0x006F) service. The CIP Fwd_Open Response message will be encapsulated into EIP SendRRData Reply message and sent to originator via opened TCP connection;

i. Once CIP connection is established, the interface is ready for originator to send data to target.

### 2.2.2 Terminating a CIP Connection

Either the originator or target may initiate the closing of CIP connections by sending CIP Forward Close Request (Service Code: 0x4E) message via EIP SendRRData (Service Code: 0x006F) service to opposite party.

Upon receives the CIP Fwd_Close Request message, the receiver shall close the CIP connection and release all resources that participating in the CIP connection. The CIP Forward Close Response (Service Code: 0xCE) message shall be returned to sender to inform it close the sender side CIP connection and release its resources.

As per the CIP protocol, the CIP Fwd_Close service will only close the CIP connection, but remain bottom EIP session and TCP connection opening.

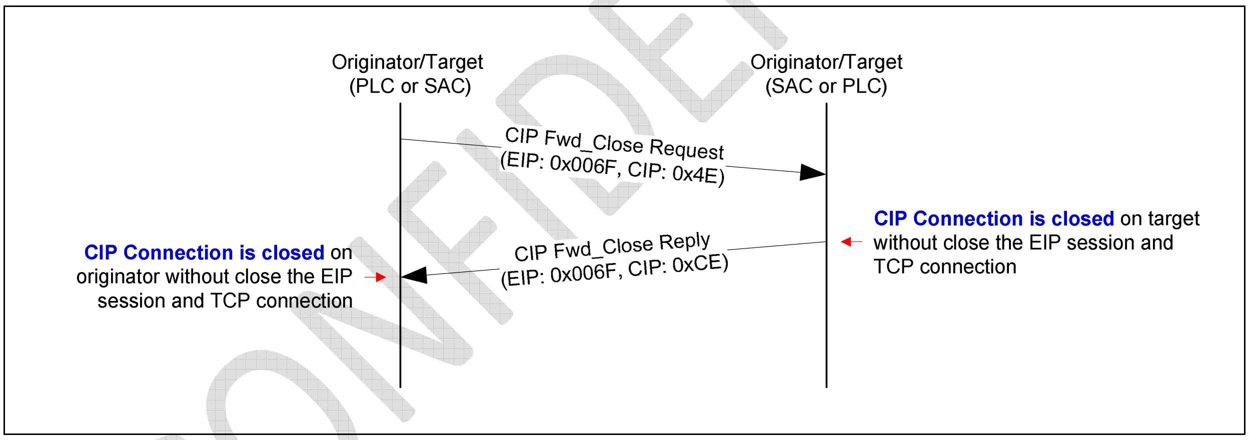

Originator/Target
(PLC or SAC)
Originator/Target
(SAC or PLC)
CIP Fwd_Close Request
(EIP: 0x006F, CIP: 0x4E)
CIP Connection is closed on target without close the EIP session and TCP connection
CIP Fwd_Close Reply
(EIP: 0x006F, CIP: 0xCE)
CIP Connection is closed on originator without close the EIP session and TCP connection

Figure 2: Terminating of CIP Connection

### 2.2.3 Terminating a EIP Session and TCP Connection

Either the originator or the target may terminate the session. Sessions shall be terminated in either of two ways:

- The originator or target shall close the underlying TCP connection. The corresponding target or originator shall detect the loss of the TCP connection, and shall close its side of the connection;
- The originator or target shall send an UnRegisterSession command and shall wait to detect the closing of the TCP connection. The corresponding target or originator shall then close its side of the TCP connection. The sender of the UnRegisterSession shall detect the loss of the TCP connection, and then it shall close its side of the connection.

The 2nd method is preferred since it results in more timely clean up of the TCP connection.

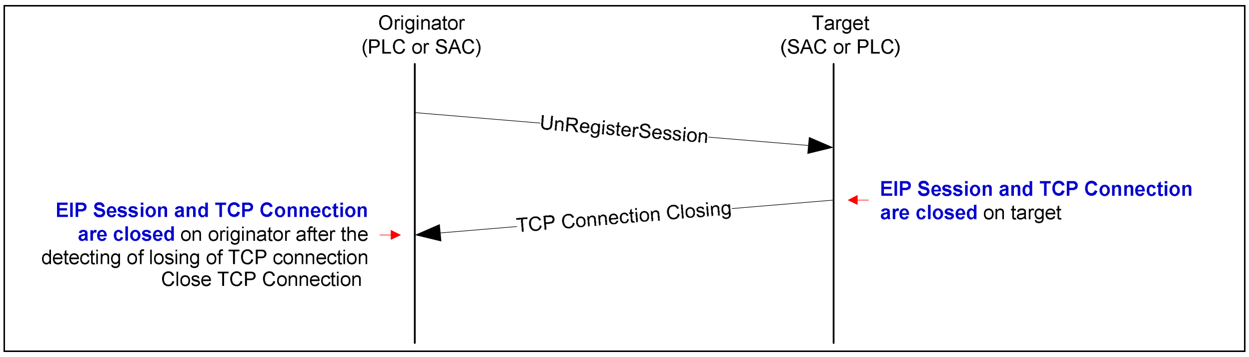

Originator
(PLC or SAC)
Target
(SAC or PLC)
UnRegisterSession
EIP Session and TCP Connection are closed on target
TCP Connection Closing
EIP Session and TCP Connection are closed on originator after the detecting of losing of TCP connection
Close TCP Connection

Figure 3: Terminating a EIP Session and TCP Connection

# 3. ETHERNET/IP (ETHERNER/INDUSTRIAL PROTOCOL)

## 3.1 INTRODUCTION

EtherNet/IP (Ethernet/Industrial Protocol) is a communication system suitable for use in industrial environment. EtherNet/IP allows industrial devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable logic controllers, welders, and process controllers.

EtherNet/IP encapsulates CIP (Control and Information Protocol), the common network, transport and application layers also shared by ControlNet and DeviceNet. EtherNet/IP then makes use of standard Ethernet and TCP/IP technology to transport CIP communication packets.

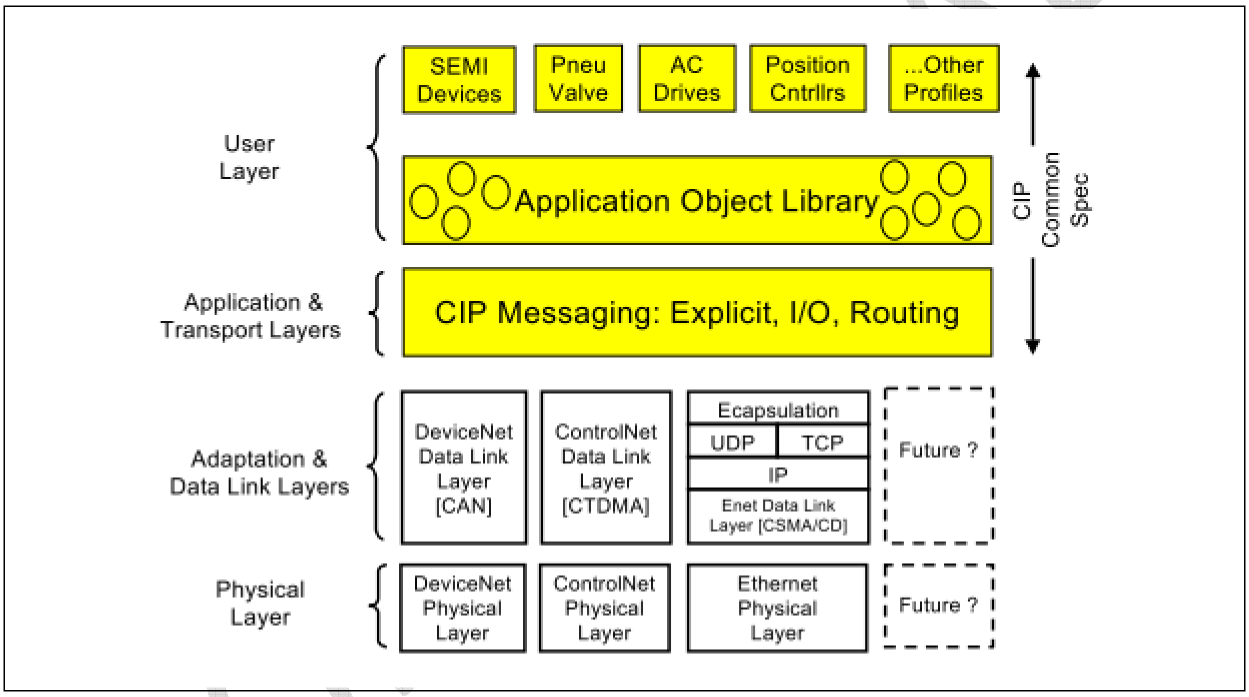

SEMI Devices
Pneu Valve
AC Drives
Position Cntrllrs
...Other Profiles
User Layer
Application Object Library
CIP Common Spec
Application & Transport Layers
CIP Messaging: Explicit, I/O, Routing
Adaptation & Data Link Layers
DeviceNet Data Link Layer [CAN]
ControlNet Data Link Layer [CTDMA]
Ecapsulation
UDP
TCP
IP
Enet Data Link Layer [CSMA/CD]
Future ?
Physical Layer
DeviceNet Physical Layer
ControlNet Physical Layer
Ethernet Physical Layer
Future ?

Figure 4: CIP Common Overview

The encapsulation protocol defines a reserved TCP port number that shall be supported by all EtherNet/IP devices. All EtherNet/IP devices shall accept as least **2 TCP connections** on TCP port number **0xAF12** (**44818**). Once the TCP connection to TCP port number 0xAF12 is established, all data sent through the TCP stream shall be in Encapsulation Message format.

The encapsulation protocol also defines a reserved UDP port number that shall be supported by all EtherNet/IP devices. All EtherNet/IP devices shall accept UDP packets on UDP port number **0xAF12** (**44818**). Since UDP, unlike TCP, does not have an ability to reorder packets, whenever UDP is used to send an encapsulated message, the entire message shall be sent in a single UDP packet. Only one encapsulated message shall be present in a single UDP packet destined to UDP port 0xAF12.

Some encapsulated messages shall only be sent via TCP. Other may be sent via either UDP or TCP.

## 3.2 ENCAPSULATION MESSAGE PACKET STRUCTURE

All encapsulation messages, sent via TCP or sent to UDP port 0xAF12, shall be composed of a fixed-length header of 24 bytes followed by an optional data portion. The total encapsulation message length (including header) shall be limited to 65535 bytes.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description |
|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code |
| | Length | 2 | UINT | Length, in bytes, of the data portion of the message, i.e., the number of bytes following the header. |
| | Session Handle | 4 | UDINT | Session identification (application dependent) |
| | Status | 4 | UDINT | Status code<br>(Refer to chapter: **3.2.4-Status Field**) |
| | Sender Context | 8 | Array of 8 Octet | Information pertinent only to the sender of an encapsulation command. Length of 8. |
| | Options | 4 | UDINT | Options flags. |
| Command Specific Data | Encapsulated Data | 2 | Array of 0 to 65511 USINT | The encapsulation data portion of the message is required only for certain commands. |

Note:

- The encapsulation message length shall not override length restrictions imposed by the encapsulated protocol. For example, a CIP UCMM message is still limited to 504 bytes even when encapsulated.
- Multi-byte interfere fields in the encapsulation message shall be transmitted in little-endian byte order.

### 3.2.1 Command Field

In order to communicate with RA ControlLogix PLC, following commands must be supported.

| No. | Command Code | Name | Comment |
|---|---|---|---|
| 1 | 0x0004 | ListServices | May be sent using either UDP or TCP |
| 2 | 0x0065 | RegisterSession | May be sent only using TCP |
| 3 | 0x0066 | UnRegisterSession | May be sent only using TCP |
| 4 | 0x006F | SendRRData | May be sent only using TCP |
| 5 | 0x0070 | SendUnitData | May be sent only using TCP |

Note:

- A device shall accept commands that it does not support without breaking the session or underlying TCP connection.
- A status code indicating that an unsupported command was received shall be returned to the sender of the message.

### 3.2.2 Length Field

The length field in the header shall specify the size in bytes of the data portion of the message. The field shall contain zero for messages that contain no data. The total length of a message shall be the sum of the number contained in the length field plus the 24-bytes size of the encapsulation header.

Note:

- The entire encapsulation message shall be read from the TCP/IP connection even if the length in invalid for a particular command of exceeds the target's internal buffers.
- Failure to read the entire message can result in losing track of the message boundaries in the TCP byte stream.

### 3.2.3 Session Handle Field

The Session Handle shall be generated by the target and returned to the originator in response to a RegisterSession Request. The originator shall insert it in all subsequent encapsulated packers to that particular target. In the case where the target initiates and sends a command to the originator, the target shall include this field in the request that it sends to the originator.

### 3.2.4 Status Field

The value in the Status field shall indicate whether or not the receiver was able to execute the requested encapsulation command. A value of zero in a reply shall indicate successful execution of the command. In all requests issued by the sender, the Status field shall contain zero. If the receiver receives a request with a non-zero Status field, the request shall be ignored and no reply shall be generated. The status codes shall be as follows:

| No. | Status Code | Comment |
|---|---|---|
| 1 | 0x0000 | Success |
| 2 | 0x0001 | The sender issued an invalid or unsupported encapsulation command. |
| 3 | 0x0002 | Insufficient memory resources in the receiver to handle the command. This is not an application error. Instead, it only results if the encapsulation layer cannot obtain memory resources that it needs. |
| 4 | 0x0003 | Poorly formed or incorrect data in the data portion of the encapsulation message. |
| 5 | 0x0004 – 0x0063 | Reserved for legacy (RA). |
| 6 | 0x0064 | An originator used an invalid session handle when sending an encapsulation message to the target. |
| 7 | 0x0065 | The target received a message of invalid length |
| 8 | 0x0066 – 0x0068 | Reserved for legacy (RA). |
| 9 | 0x0069 | Unsupported encapsulation protocol revision. |
| 10 | 0x006A – 0xFFFF | Reserved for future expansion |

### 3.2.5 Sender Context Field

The sender of the command shall assign the value in the Sender Context field of the header. The receiver shall return this value without modification in its reply. Commands with no expected reply may ignore this field. The sender of command may place any value in this field. It could be used to match requests with their associated reply.

### 3.2.6 Options Field

The sender of an encapsulated packet shall set the options field to zero. The receiver of an encapsulated packet shall verify that the option field is zero. The intent of this field is to provide bits that modify the meaning of the various encapsulation commands. No particular use for this field has not yet been specified.

Note:

- The receiver shall discard encapsulated packets with a non-zero option field.
- The intent of this field is to provide bits that modify the meaning of the various encapsulation commands. No particular use for this field has not yet been specified.

### 3.2.7 Command Specific Data Field

The structure of the command specific data field depends on the command code. To organize their command specific data field, most commands use either or both of the following two medhods:

1) Use a fixed structure;
2) Use the **common packet format**.

The common packet format allows commands to structure their command specific data field in an extensible way.

### 3.2.8 Common Packet Format

#### 3.2.8.1 General

The common packet format shall consist of an item count, followed by an address item, then a data item (in that order) as shown below. Additional optional items may follow.

| No. | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Item Count | UINT | Number of items to follow (shall be at least 2) |
| 2 | Address Item | Refer to Address Item Structure Format | Addressing information for encapsulated packet |
| 3 | Data Item | Refer to Data Item Structure Format | The encapsulated data packet |
| 4 | Optional additional items … | | |

Address and Data Item Structure Format:

| No. | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Item ID | UINT | Type of item encapsulated. Refer to Item ID Numbers below. |
| 2 | Item Length | UINT | Length in bytes of data to follow |
| 3 | Data | Variable | The data (if length > 0) |

Item ID Numbers:

| No. | Item ID Number | Item Type | Description |
|---|---|---|---|
| 1 | **0x0000** | **Address** | **Null (Used for UCMM messages)**<br>**Indicates that encapsulation routing is NOT needed. Target is either local (Ethernet) or routing info is in a data item.** |
| 2 | 0x0001 – 0x000B | | Reserved for legacy (RA) |

| | | | |
|---|---|---|---|
| 3 | 0x000C | | ListIdentity response |
| 4 | 0x000D – 0x0083 | | Reserved for legacy (RA) |
| 5 | 0x0084 – 0x0090 | | Reserved for future expansion |
| 6 | 0x0091 | | Reserved for legacy (RA) |
| 7 | 0x0092 – 0x00A0 | | Reserved for future expansion |
| 8 | **0x00A1** | **Address** | **Connection-based (used for connected message)** |
| 9 | 0x00A2 – 0x00A4 | | Reserved for legacy (RA) |
| 10 | 0x00A5 – 0x00B0 | | Reserved for future expansion |
| 11 | **0x00B1** | **Data** | **Connected Transport packet** |
| 12 | **0x00B2** | **Data** | **Unconnected message** |
| 13 | 0x00B3 – 0x00FF | | Reserved for future expansion |
| 14 | **0x0100** | | **ListServices response** |
| 15 | 0x0101 – 0x010F | | Reserved for legacy (RA) |
| 16 | 0x0110 – 0x7FFF | | Reserved for future expansion |
| 17 | 0x8000 | Data | Sockaddr Info, originator-to-target |
| 18 | 0x8001 | Data | Sockaddr Info, target-to-originator |
| 19 | 0x8002 | | Sequenced Address item. |
| 20 | 0x8003 – 0xFFFF | | Reserved for future expansion |

### 3.2.8.2 Address Items

#### 3.2.8.2.1 Null Address Item

The null address item shall contain only the type ID and the length as shown below. The length shall be zero. No data shall follow the length.

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x0000 |
| 2 | Item Length | UINT | 2 | 0x0000 |

Since the null address item contains no routing information, it shall be used when the protocol packet itself contains any necessary routing information.

The null address item shall be used for Unconnected Messages.

#### 3.2.8.2.2 Connected Address Item

The connected address item shall be used when the encapsulated protocol is connection-oriented. The data shall contain a connection identifier (Connection identifiers are exchanged in the Forward_Open service of the Connection Manager).

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|

|  |  |  |  |  |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x00A1 |
| 2 | Item Length | UINT | 2 | 0x0004 |
| 3 | Data | UDINT | 4 | Connection Identifier |

#### 3.2.8.2.3 Sequenced Address Item

The sequenced address item shall be used for CIP transport class 0 and class 1 (UDP) connected data. The data shall contain a connection identifier and a sequence number.

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x8002 |
| 2 | Item Length | UINT | 2 | 0x0008 |
| 3 | Data | UDINT | 4 | Connection Identifier |
|  |  | UDINT | 4 | Sequence Number |

### 3.2.8.3 Data Items

#### 3.2.8.3.1 Unconnected Data Item

The data item that encapsulates an unconnected message shall be as follows:

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x00B2 |
| 2 | Item Length | UINT | 2 | Length, in bytes, of the unconnected message. |
| 3 | Data | Variable | ? | The unconnected message |

The format of the “Data” field is dependent on the encapsulated protocol. When used to encapsulated CIP, the format of the data field is that of a Message Router request or Message Router reply.

The Sender Context field in the encapsulation header shall be used for unconnected request/reply matching.

#### 3.2.8.3.2 Connected Data Item

The data item that encapsulates a **connected transport packet** shall be as follows:

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x00B1 |
| 2 | Item Length | UINT | 2 | Length, in bytes, of the unconnected message. |
| 3 | Data | UINT | 2 | Sequence number |
|  |  | Variable | ? | The transport packet |

The format of the “Data” field is dependent on the encapsulated protocol.

#### 3.2.8.3.3 Sockaddr Info Item

The Sockaddr Info items shall be used to encapsulate socket address information necessary to send datagrams (the connected data) between the target and originator. There are separate items for originator-to-target and target-to-originator socket information.

| No. | Field Name | Data Type | Data Length (Bytes) | Value |
|---|---|---|---|---|
| 1 | Item ID | UINT | 2 | 0x8000 for O-T, 0x8001 for T-O |
| 2 | Item Length | UINT | 2 | 16 bytes |
| 3 | sin_family | INT | 2 | Shall be AF_INET=2. This field shall be sent in big endian order. |
| 4 | sin_port | UINT | 2 | Shall be set to the TCP or UDP port on which packets for this CIP connection will be sent. This field shall be sent in big endian order. |
| 5 | sin_addr | UDINT | 4 | Shall be set to the IP address to which packet for this CIP connection will be sent. This field shall be sent in big endian order. |
| 6 | sin_zero | Array of USINT | 8 | Shall be 0. This field shall be sent in big endian order. Length of 8. |

## 3.3 EIP ENCAPSULATION MESSAGE FORMAT

### 3.3.1 ListServices Message

The ListServices command shall determine which encapsulation service classes the target device supports.

The ListServices request message is the first message that will be sent by RA ControlLogix PLC, if PLC is the Originator and need to establish the CIP connection to SAC. Hence, the SAC application must support this message and return the ListServices reply to PLC.

**NOTE:**

- Each service class has a unique type code, and an optional ASCII name.

#### 3.3.1.1 ListServices Request Message

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code ListServices (**0x04**) | 04 00 |
| | Length | 2 | UINT | Length of command specific data portion, 0 bytes | 00 00 |
| | Session Handle | 4 | UDINT | ignored | 00 00 00 00 |
| | Status | 4 | UDINT | 0 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |

**Sample Message:**

```
040000000000000000000000005341433250 4C433100000000
```

#### 3.3.1.2 ListServices Reply Message

The receiver shall reply with a standard encapsulation message, consisting of the header and data, as shown below. The data portion of the message shall provide the information on the services supported.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code ListServices (**0x04**) | 04 00 |
| | Length | 2 | UINT | Length of command specific data portion, 26 bytes | 1A 00 |
| | Session Handle | 4 | UDINT | ignored | 00 00 00 00 |
| | Status | 4 | UDINT | 0 in request | 00 00 00 00 |

| | | | | | |
|---|---|---|---|---|---|
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Item Count | 2 | UINT | Number of items to follow, 1 | 01 00 |
| | Target Items (Interface Information) | 2 | UINT | Item Type Code, **0x0100** – ListService Response | 00 01 |
| | | 2 | UINT | Item Length, 20 | 14 00 |
| | | 2 | UINT | Version of encapsulated protocol shall be set to 1 | 01 00 |
| | | 2 | UINT | Capability flags | 20 01 |
| | | 16 | Array of 16 USINT | Name of Service, “Communications” | 43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 00 00 |

**Sample Message:**

```
04001A0000000000000000005341433250 4C4331000000000100001140001002001
436F6D6D756E69636174696F6E730000
```

The Type Code shall identify the service class as follows:
One service class is defined, with type code **0x0100** and name “**Communications**”. This service class shall indicate that the device supports encapsulation of CIP packets.

All devices that support encapsulating CIP shall support the ListServices request and Communications service class.

The Version field shall indicate the version of the service supported by the target to help maintain compatibility between applications.

Each service shall have a different set of capability flags. Unused flags shall be set to zero. The Capability Flags, defined for the Communications service, shall be as follows:

| Flag Value | Description |
|---|---|
| Bits 0 - 4 | Reserved for legacy (RA) |
| Bit 5 | 1 – Support CIP Encapsulation via TCP,<br>0 – Unsupport. |
| Bits 6 - 7 | Reserved for legacy (RA) |
| Bit 8 | 1 – Support CIP Encapsulation via UDP,<br>0 – Unsupport. |
| Bits 9 - 15 | Reserved for future expansion |

The Name field shall allow up to a 16-byte, NULL-terminated ASCII string for descriptive purpose only. The 16-byte limit shall include the NULL character.

## 3.3.2 RegisterSession Message

### 3.3.2.1 RegisterSession Request Message

An originator shall send a RegisterSession command to a target to initiate a session.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code RegisterSession (**0x65**) | 65 00 |
| | Length | 2 | UINT | Length of command specific data portion, 4 bytes | 04 00 |
| | Session Handle | 4 | UDINT | 0 in request | 00 00 00 00 |
| | Status | 4 | UDINT | 0 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Protocol Version | 2 | UINT | Request protocol version shall be set to 1. | 01 00 |
| | Options Flags | 2 | UINT | Session options shall be set to 0<br>Bits 0-7 are reserved for legacy (RA)<br>Bits 8-15 are reserved for future expansion<br>Note: This field is not the same as the option flags in the encapsulation header. | 00 00 |

**Sample Message:**

`650004000000000000000000534143325040C4331000000000001000000`

### 3.3.2.2 RegisterSession Reply Message

The target shall send a RegisterSession reply to indicate that it has registered the originator. The reply shall have the same format as the request as shown below.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code RegisterSession (**0x65**) | 65 00 |
| | Length | 2 | UINT | Length of command specific data portion, 4 bytes | 04 00 |
| | Session Handle | 4 | UDINT | Hander returned by RegisterSession reply. E.g. 0x15020A00 | (00 0A 02 15) |
| | Status | 4 | UDINT | 0 | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Context preserved from the corresponding RegisterSession request. e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Protocol Version | 2 | UINT | Request protocol version shall be set to 1. | 01 00 |
| | Options Flags | 2 | UINT | Session options shall be set to 0 | 00 00 |

| | | | | | |
|---|---|---|---|---|---|
| 🟩 | | | | Bits 0-7 are reserved for legacy (RA)<br>Bits 8-15 are reserved for future expansion<br>Note: This field is not the same as the option flags in the encapsulation header. | |

**Sample Message:**

```
65000400000a02150000000053414332504c43310000000001000000
```

The session Handle field of the header shall contain a target-generated identifier that the originator shall save and insert in the Session handle field of the header for all subsequent requests to that target. This field shall be valid only if the Status field is zero (0).

The Sender Context field of the header shall contain the same values present in the original sender request. If the originator has been registered with the target, the Status field shall be zero (0). If the target was unable to register, the Status field shall be set to 0x69 (unsupported encapsulation protocol revision).

The Protocol Version field shall equal the requested version if the originator was successfully registered. If the target does not support the requested version of the protocol,

- The session shall not be created;
- The Status field shall be set to unsupported encapsulation protocol 0x69;
- The target shall return the highest supported version in the Protocol Version field.

If all requested options are supported, the Options field shall return the originator's value. This value shall be zero.

### 3.3.3 UnregisterSession Message

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code UnRegisterSession (**0x66**) | 66 00 |
| | Length | 2 | UINT | Length of command specific data portion, 0 bytes | 00 00 |
| | Session Handle | 4 | UDINT | Hander returned by RegisterSession reply. Once the client has sent this command, it shall no longer use the handle. | (00 0A 02 15) |
| | Status | 4 | UDINT | 0 | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | 53 41 43 32 50 4C 43 31 |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |

**Sample Message:**

```
66000000000a021500000000534143325 04c433100000000
```

Either an originator or a target may send this command to terminate the session. The receiver shall initiate a close of the underlying TCP/IP connection when it receives this command. The session shall also be terminated when the transport connection between the originator and target is terminated. The receiver shall perform any other associated cleanup required on its end. There shall be no reply to this command.

## 3.3.4 SendRRData Message

### 3.3.4.1 SendRRData Request Message

The SendRRData command shall transfer an encapsulated request/reply packet between the originator and target, where the originator initiates the command. The actual request/reply packets shall be encapsulated in the data portion of the message and shall be the responsibility of the target and originator.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code SendRRData (**0x6F**) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion. | ? |
| | Session Handle | 4 | UDINT | Hander returned by RegisterSession reply. | ? |
| | Status | 4 | UDINT | 0 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | 53 41 43 32 50 4C 43 31 |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface Handle | 4 | UDINT | Shall be 0 for encapsulating CIP packets. | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation timeout | ? |
| | Encapsulated Packet | ? | Array of Octet | Refer to chapter **3.2.8:** Common Packet Format Specification. | ? |

**Sample Message (Sending CIP Fwd Open Request Message vis EIP SendRRData Request, originator send to target):**

```
6F004000001F02130000000053414332504C43310000000000000000000002000000
0000B20030005402200624010 7E80000000001000000000E00F00F1974600000000
80C3C901F64380C3C901F643A30301002002 2401
```

The target shall abort the requested operation after the timeout expires. When the "timeout" field is in the range 1 to 65535, the timeout shall be set to this number of seconds. When the "timeout" field is set to 0, the encapsulation protocol shall not have its own timeout. Instead, it shall rely on the timeout mechanism of the encapsulated protocol.

NOTE:

- When used to encapsulate the CIP packets, the SendRRData request and response are used to send encapsulated ***UCMM messages (unconnected messages)***.
- When used to encapsulate the CIP packets, the timeout field is usually set to 0 sine CIP provides its own timeout mechanism for connected messages..

### 3.3.4.2 SendRRData Reply Message

The SendRRData reply, as shown below, shall contain data **in response to** the SendRRData request. The reply to the original encapsulated protocol request shall be contained in the data portion of the SendRRData reply.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code SendRRData (**0x6F**) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion. | ? |
| | Session Handle | 4 | UDINT | Hander returned by RegisterSession reply. | ? |
| | Status | 4 | UDINT | 0 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Context preserved from the corresponding SendRRData request. Length of 8. | ? |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface Handle | 4 | UDINT | Shall be 0 for encapsulating CIP packets. | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation timeout (not used) | ? |
| | Encapsulated Packet | ? | Array of Octet | Refer to chapter **3.2.8:** Common Packet Format Specification. | ? |

**Sample Message (Sending CIP Fwd Open Response Message vis EIP SendRRData Reply, target send to originator):**

```
6F002E00000100213000000005341433250 4C43310000000000000000000000002000000
0000B2001E00D4000000241FD0001000000000E00F00F1974600E0707200E0707200
0000
```

The format of the data portion of the reply message shall be the same as that of the SendRRData request message.

NOTE:

- Since the request and reply share a common format, the reply message contains a timeout field; however, it is not used.

## 3.3.5 SendUnitData Message

The SendUnitData command shall send encapsulated connected messages. This command may be used when the encapsulated protocol has its own underlying end-to-end transport mechanism.

A reply shall not be returned. The SendUnitData command may be sent by either end of the TCP connection.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code SendUnitData (**0x70**) | 70 00 |
| | Length | 2 | UINT | Length of command specific data portion. | ? |
| | Session Handle | 4 | UDINT | Hander returned by RegisterSession reply. | ? |
| | Status | 4 | UDINT | 0 | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Any sender context. Length of 8. e.g. SAC2PLC1 | 53 41 43 32 50 4C 43 31 |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface Handle | 4 | UDINT | Shall be 0 for encapsulating CIP packets. | 00 00 00 00 |
| | Timeout | 2 | UINT | Shall be 0. The timeout field is not used since no reply is generated upon receipt of a SendUnitData command. | 00 00 |
| | Encapsulated Packet | ? | Array of Octet | Refer to chapter **3.2.8:** Common Packet Format Specification. | ? |

**Sample Message (Encapsulate Application Data "12345" into CIP Data Table Write Request message and send via EIP SendUnitData):**

```
70002D00001F02130000000053414332504C43310000000000000000000000200A100
04000299FD00B1001900010004D069108544C475F5265636528000C2000500031323334
35
```

**Sample Message (CIP Data Table Write Reply message via EIP SendUnitData):**

```
70001A00001F02130000000053414332504C43310000000000000000000000200A100
04000299FD00B10006000100CD000000
```

NOTE:

- When used to encapsulate the CIP packets, the SendUnitData command is used to send **CIP connected data** in both the O-T and T-O directions..

# 4. CIP (CONTROL & INFORMATION PROTOCOL)

## 4.1 INTRODUCTION

The Control and Information Protocol (CIP) is a peer to peer object oriented protocol that provides connection between industrial devices (e.g. sensors, actuators) and higher-level devices (controllers). CIP is physical media and data line layer independent.

CIP defines a connection-based scheme to facilitate all application communications. A CIP connection provides a communication path between multiple end-points. The end-points of a connection are applications that need to share data. Transmissions associated with a particular connection are assigned an identification value when a connection is established. This identification value is called the ***Connection ID (CID)***. If the connection involves a bi-directional exchange, then two Connection ID value are assigned. See figure below.

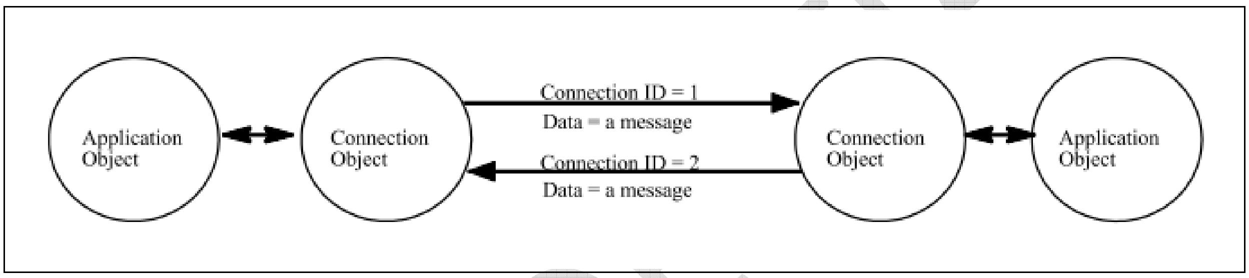


Figure 5: CIP Connections and Connection IDs

CIP's connection-based scheme defines a dynamic means by which the following two types of connections can be established:

- **I/O Connections** – Provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data moves through these ports and is often referred to implicit messaging.
- **Explicit Messaging Connections** – Provide generic, multi-purpose communication paths between two devices. These connections often are referred to as just Messaging Connections. Explicit Messages provide the typical request/response-oriented network communications.

The Unconnected Message Manager (UCMM) is responsible for processing Unconnected Explicit Requests and Responses. This includes establishing both Explicit Messaging and I/O connections by sending Forward Open Request via UCMM SendRRData servise.

When using the UCMM to establish an explicit messaging connection, the target application object is the Message Router object (Class Code 2).

IR-BHS standard SAC to ControlLogix PLC interface is using explicit messaging connections.

## 4.2 MESSAGE ROUTER REQUEST/RESPONSE FORMAT

CIP defines a standard data format for delivering data to and from the Message Router object. This data format is used in various places within CIP including the Unconnected Send service of the Connection Manager object and the UCMM data structures of most of the CIP networks.

The Message Router Request Format:

| Parameter Name | Data Length (Bytes) | Data Type | Description |
|---|---|---|---|
| Service Code | 1 | USINT | Service code of request |
| Request Path Size | 1 | USINT | The number of 16 bit words in the Request_Path field (next element). |
| Request Path | ? | Padded EPATH | This is an array of bytes whose contents convey the path of the request (Class ID, Instance ID, etc) for this transaction. |
| Request Data | ? | Array of octet | Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty. |

The Message Router Response Format:

| Parameter Name | Data Length (Bytes) | Data Type | Description |
|---|---|---|---|
| Reply Service Code | 1 | USINT | Reply Service code |
| Reserved | 1 | USINT | Shall be zero |
| General Status | 1 | USINT | One of the General Stauts codes listed in Appendix A (Status Codes) |
| Size of Additional Status | 1 | USINT | Number of 16 bit words in Additional Status array. |
| Additional Status | ? | Array of UINT | Additional status. |
| Response Data | ? | Array of octet | Response data from request or additional error data if General Status indicated an error. |

## 4.3 CIP MESSAGE FORMATS

The standard CIP protocol defines a series of service messages, but this document only describes following 3 CIP messages that are used by IR-BHS standard SAC-ControlLogix PLC interface.

| No. | Message Name | CIP Service Code | Message Type | Description |
|---|---|---|---|---|
| 1 | Forward Open Request | 0x54 | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 2 | Successful Forward Open Response | 0xD4 | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 3 | Unsuccessful Forward Open Response | 0xD4 | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 4 | Forward Close Request | 0x4E | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 5 | Successful Forward Close Response | 0xCE | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 6 | Unsuccessful Forward Close Response | 0xCE | Unconnected Message | To be sent by EIP UCMM SendRRData service (EIP Service Code: 0x006F) |
| 7 | CIP Data Table Write Request | 0x4D | Connected Message | To be sent by EIP SendUnitData service (EIP Service Code: 0x0070) |
| 8 | CIP Data Table Write Response | 0xCD | Connected Message | To be sent by EIP SendUnitData service (EIP Service Code: 0x0070) |

### 4.3.1 Forward Open Message

CIP Forward Open Service (0x54) is used to establish a CIP Connection with a Target device. The CIP connection need to be opened before exchange any (SAC) application layer data.

#### 4.3.1.1 Forward Open Request Message Format

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>64 bytes | 40 00 |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message | 00 00 |

| | | | | |
|---|---|---|---|---|
| | | | (Refer to chapter: **3.2.8 - Common Packet Format**) | |
| Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | 00 00 |
| Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | B2 00 |
| Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet) | (30 00) |
| Below Fields are the CIP Forward open Request Message | | | | |
| Below fields are Message Router Request Packet Header Fields | | | | |
| Service Code | 1 | USINT | **0x54** Fwd_Open request code | **54** |
| Request Path Size | 1 | USINT | 0x02 Path size in words | (02) |
| Request Path | 4 | Array of bytes | EPATH (or IOI) 20,06 (class, CM object);24,01 (Instance 1) | (20 06 24 01) |
| Below fields are Message Router Request Packet Data Fields | | | | |
| Connection Priority/Time_tick | 1 | BYTE | 1-second ticks (priority ignored), e.g. 0x07<br>(Refer to chapter: **4.3.1.1.1-Connection Priority / Time_Tick)** | (07) |
| Connection Timeout_Ticks | 1 | USINT | (ticks*tick_time)<br>e.g. 0xE8 (232), the Total Timeout Value = $2^7$ x 232 = 29696ms<br>(Refer to chapter: **4.3.1.1.2-Timeout_ticks)** | (E8) |
| O-T Connection ID | 4 | UDINT | Type 0x00000000 in request;<br>Returned by target in Fwd_Open Response<br>(Refer to chapter: **4.3.1.1.3-Network Connection ID (CID)**) | 00 00 00 00 |
| T-O Connection ID | 4 | UDINT | CID chosen by originator | (01 00 00 00) |
| Connection Serial# | 2 | UINT | Chosen by originator<br>(Refer to chapter: **4.3.1.1.4-Connection Serial Number**) | (00 F0) |
| Originator Vendor ID | 2 | UINT | From ID object (CIP vendor ID).<br>E.g. Rockwell Automation (0x0001)<br>Let's use **0x4952** to represent "**IR**"<br>(Refer to chapter: **4.3.1.1.5-Originator Vendor ID**) | (52 49) |
| Originator Serial# | 4 | UDINT | Unique# for all devices manufactured by the same vendor.<br>In IR-BHS SAC, Each SAC2PLC GW service should be assigned with a unique serial number. E.g.<br>• SAC2PLC1 GW – 0x00000001<br>• SAC2PLC2 GW – 0x00000002<br>• SAC2PLC3 GW – 0x00000003<br>(Refer to chapter: **4.3.1.1.6-Originator Serial Number**) | (01 00 00 00) |
| Connection Timeout Multiplier | 1 | USINT | Used to calculate the connection path (inactivity) timeout (= Multiplier x RPI). CIP connection will be closed if the connection path timeout elapsed.<br>(Refer to chapter: **4.3.1.1.8-Connection Timeout Multiplier**) | (00) |
| Reserved | 3 | Array of 3 octet | 0x00 each octet | 00 00 00 |

| | | | | | |
|---|---|---|---|---|---|
| | O-T RPI | 4 | UDINT | Requested RPI, originator to target, microseconds. 30000ms (0x01C9C380) (Refer to chapter: **4.3.1.1.9-Required Packet Interval (RPI)**) | (80 C3 C9 01) |
| | O-T Connection Parameters | 2 | WORD | O->T Network Connection Parameters: 0x43F6<br>0... .... .... .... = Owner: Exclusive (0)<br>.10. .... .... .... = Connection Type: Point to Point (2)<br>.... 00.. .... .... = Priority: Low Priority (0)<br>.... ..1. .... .... = Connection Size Type: Variable (1)<br>.... ...1 1111 0110 = Connection Size: 502<br>(Refer to chapter: **4.3.1.1.7-Network Connection Parameters**) | (F6 43) |
| | T-O RPI | 4 | UDINT | Requested RPI, target to originator, microseconds 30000ms (0x01C9C380) | (80 C3 C9 01) |
| | T-O Connection Parameters | 2 | WORD | T->O Network Connection Parameters: 0x43F6<br>0... .... .... .... = Owner: Exclusive (0)<br>.10. .... .... .... = Connection Type: Point to Point (2)<br>.... 00.. .... .... = Priority: Low Priority (0)<br>.... ..1. .... .... = Connection Size Type: Variable (1)<br>.... ...1 1111 0110 = Connection Size: 502 | (F6 43) |
| | Transport Class/Trigger | 1 | BYTE | **0xA3**, Server transport, class 3, application trigger (Refer to chapter: **4.3.1.1.11 - Transport Class and Trigger**) | **(A3)** |
| | Connection Path Size | 1 | USINT | Number of 16 bit words in Connection Path:<br>0x03 – ControlLogix via backplane<br>0x02 – direct network device | (03) |
| | Connection Path | ? | Padded EPATH | (from PLC5E MSG instruction)<br>e.g. 0x010020022401<br>01 – Backplane port of 1756-ENET<br>00 – Logix5550 in slot 0<br>20 02 – Class segment, 02 is MR<br>24 01 – Instance segment, No.1 | (01 00 20 02 24 01) |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

#### 4.3.1.1.1 Connection Priority / Time_Tick

The Priority/Time_tick parameter determines the priority of the unconnected message and the time duration of a "tick" (Tick Time) specified in the Timeout_ticks parameter. The bit fields are:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Description | Reserved | | | Priority<br>0 = Normal<br>1 = Reserved | Tick Time | | | |

The Reserved and Priority fields shall be set to zero (0). The tick time shall be used in conjunction with the Timeout_ticks parameter value to determine the total time out value. The following formula is used:

$$\text{Actual Time Out Value} = 2^{\text{time\_tick}} \times \text{Timeout\_tick}$$

If the tick_time is 0x0000 (1ms), a Timeout_ticks value of 5 would translate into Actual Time Out Value of 5ms. If the tick_time is 0x0010 (4ms), a Timeout_ticks value of 5 would translate into Actual Time Out Value of 20ms.

#### 4.3.1.1.2 Timeout_ticks

The Timeout_ticks parameter shall be used to specify the amount of time the originating application shall wait for the transaction to be completed. When used with the Tick Time portion of the Priority/Time_tick field, a total timeout value can be calculated.

If a timeout is imminent, then the intermediate device returns an error response rather than just letting the originator timeout. Also when an intermediate device times out, an error response is returned. In both cases all resources associated with that unconnected message are released.

#### 4.3.1.1.3 Network Connection ID (CID)

The network Connection ID shall be link specific and shall not be related to the connection serial number, which is connection specific and the same over all the links. The fields of the network CID shall be used to set the screening mechanism for the specific link. The Network CID is either a CIP Produced (O-T) Connection ID or CIP Consumed (T-O) Connection ID.

#### 4.3.1.1.4 Connection Serial Number

The connection serial number shall be a unique 16-bit value selected by the connection manager at the originator of the connection. The originator shall make sure that the 16-bit value is unique for the device. There shall be no other significance places on the number by any other nodes in the connection path. The connection serial numbers shall be unique but do not have to be sequential. For example, an operator interface may have a large number of connections open at the same time, each with a unique number. The same values could be repeated at other operator interface stations.

A possible implementation would be to have a connection list which points to the descriptor for each connection, and the connection serial number could be the index into the table.

#### 4.3.1.1.5 Originator Vendor ID

The vendor ID shall be a unique number assigned to the various vendors of products. Each vendor has a unique number assigned.

#### 4.3.1.1.6 Originator Serial Number

The originator serial number shall be a unique 32-bit value that is assigned to a device at the time of manufacture. This value shall be guaranteed to be unique for all devices manufactured by the same vendor. No significance shall be attached to the number. The combination of Vendor ID and Originator Serial Number shall be unique throughout the system.

In IR-BHS SAC, Each SAC2PLC GW service should be assigned with a unique serial number.

#### 4.3.1.1.7 Network Connection Parameters

Network connection parameters shall be provided as a single 16-bit word that contains the fields in the following figure:

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Description | Redundant Owner | Connection Type | | Reserved | Priority | | Fixed / Variable | Connection Size – in bytes | | | | | | | | |

- **Connection Size** – The maximum size, in bytes, of the data for each direction (where applicable) of the connection. For a variable sized connection, the size shall be the maximum size of the buffer for any transfer. The actual size of the transfer for a variable connection shall be equal to or less than the size specified for the network connection. The maximum buffer size shall be dependent on the links that the connection traverses.
- **Fixed/Variable** – With a fixed size connection, the amount of data on each transmission shall be the size specified in the Connection Size parameter. With a variable size connection, the amount of data on each transmission may be a variable size, up to the size specified in the Connection Size parameter.

  0 = Fixed

  1 = Variable
- **Priority** – The priority shall be one of :

  00 = Low Priority

  01 = High Priority

  10 = Scheduled

  11 = Urgent
- **Connection Type** –

  00 = Null (may be used to reconfigure a connection)

  01 = Multicast

  10 = Point to Point

  11 = Reserved
- **Redundant Owner** –

  The redundant owner bit in the O-T direction shall be set (=1) to indicate that more than one owner may be permitted to make a connection simultaneously. The bit shall be clear (=0) to indicate an exclusive-owner, input only or listen-only connection.
- **Reserved** field shall be set to zero.

#### 4.3.1.1.8 Connection Timeout Multiplier

Connection Timeout Multiplier is used to calculate the CIP Connection Path (inactivity) Timeout.
Path Timeout = Multiplier x RPI.

CIP connection will be closed if the connection path timeout elapsed.

Refer to chapter **4.3.1.1.10: Actual Packet Interval (API)** below.

#### 4.3.1.1.9 Required Packet Interval (RPI)

The requested packet interval shall be the requested time between packets in microseconds. The format of the RPI shall be a 32-bit integer in microseconds.

#### 4.3.1.1.10 Actual Packet Interval (API)

The actual packet interval shall be the actual time between packets in microseconds. The format of the API shall be a 32-bit integer in microseconds.

The RPI shall be the time between packets requested by the receiving device. The value shall be used to allocate bandwidth at each of the producing nodes. The allocation of bandwidth may have to be adjusted when the actual packet rate or actual packet interval is returned, since it is possible for the two values to differ. **The path time-out value at each of the intermediate and target nodes shall also be set to the Connection Timeout Multiplier times the API**. The RPI is therefore required for all connections.

**Scheduled Priority**

For scheduled priority, the RPI shall be the packet rate of the repetitive data. On links that support bandwidth allocation, bandwidth shall be reserved for this packet. For scheduled priority, the data shall also be restricted to the specified packet rate, which means that if data arrives at an intermediate node faster than the specified packet rate, the node shall filer the packets to the specified rate. Since each node's scheduled priority update rate is in discrete quanta, the Actual Packet Interval (API) may be smaller (more rapid) than the RPI. **The Connection Path Timeout value shall be set to the Connection Timeout Multiplier times the API**.

**High Priority**

For high priority, the RPI shall be used to set the path timeout in the intermediate and target nodes. **The RPI shall therefore be set to the slowest packet rate expected, which shall preclude having the connection close due to a path timeout**. The longer the path timeout, the longer the time required to reclaim resources in the intermediate nodes as a result of faults in the network. Since the high priority is not quantized at any of the nodes, the API shall equal to RPI. To maintain consistency, however, the path timeout value shall again be set to the Connection Timeout Multiplier times to API.

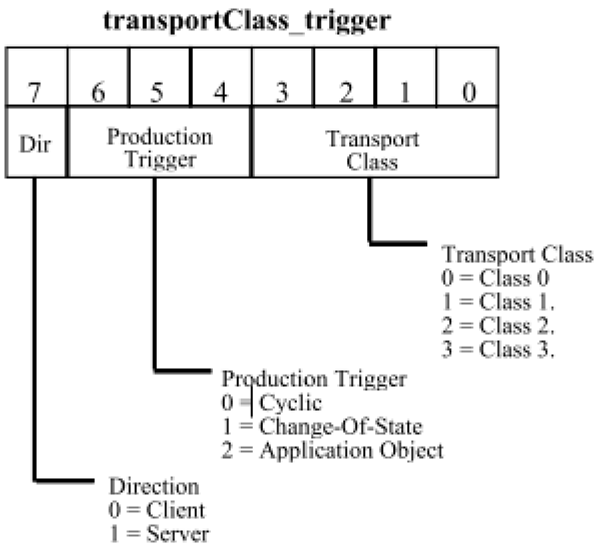
**Low Priority**

For low priority, the RPI shall be used to set the path timeout in the intermediate and target nodes. **The RPI shall therefore be set to the slowest packet rate expected, which shall preclude having the connection close due to a path timeout**. The longer the path timeout, the longer the time required to reclaim resources in the intermediate nodes as a result of faults in the network. Since the low priority is not quantized at any of the nodes, the API shall equal the RPI. To maintain consistency, however, the path timeout values shall again be set to the Connection Timeout Multiplier times the API.

#### 4.3.1.1.11 Transport Class and Trigger

The transport class and trigger specify the type of transport required for the connection. This information shall not be used by the connection manager bit passed on to the application. The application shall determine if the transport type is supported and if an instance of the required transport is available.

The transport class and trigger defines whether this is a producing only, consuming only, or both producing and consuming connection. If this end point is to perform a data production, this attribute also defines the event that triggers the production. The eight (8) bits are divided as follows:



The Direction bit of the transportClass_trigger byte indicates whether the end-point is to act as the Client or the Server on this connection. The following values are defined:

| Value | Meaning | |
|---|---|---|
| 0 | Client | This end-point provides the Client behavior associated with this Connection. Additionally, this value indicates that the Production Trigger bits within the transportClass_trigger byte contain the description of when the Client is to produce the message associated with this connection. Client connections with production trigger value of 0 or 1 (Cyclic or Change-of-State) shall produce immediately after transitioning to the Established state. |
| 1 | Server | This end-point provides the Server behavior associated with this Connection. In addition, this value indicates that the Production Trigger bits within the transportClass_trigger byte are to be **IGNORED**. The Production Trigger bits are ignored due to the fact that a Server end-point reacts to the transmission from the Client. The only means by which a Server end-point is triggered to transmit is when this reaction calls for the production of a message (Transport Classes 2 or 3). |

The following table lists the values that are possible within the Production Trigger bits of the transportClass_trigger attribute:

| If the value is: | Then the Production of a message is: | |
|---|---|---|
| 0 | Cyclic | The expiration of the Transmission Trigger Timer triggers the data production. |
| 1 | Change-of-State | Production occurs when a change-of-state is detected by the Application Object. Note that the consuming end-point may have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point. |
| 2 | Application Object Triggered | The Application Object decides when to trigger the production. Note that the consuming end-point may have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point. |
| 3 - 7 | Reserved by CIP | |

Table below lists possible values within the Transport Class nibble of the transportClass_trigger attribute.

| Value | Meaning | |
|---|---|---|
| 0 | Transport Class 0 | Based on the value within the Dir bit, this connection end-point will be a producing only or consuming only end-point. Upon application of this Connection instance, the module instantiates either a Link Producer (Dir bit = Client, producing only) or a Link Consumer (Dir bit = Server, consuming only) to be associated with this Connection. |
| 1 | Transport Class 1 | |
| 2 | Transport Class 2 | Indicates that the module will both produce AND consume across this connection. The Client end-point generates the first data production that is consumed by the Server, which causes the Server to return a production that is consumed by the Client. |
| 3 | Transport Class 3 | |
| 4 | Transport Class 4 | Non-blocking |
| 5 | Transport Class 5 | Non-blocking, fragmenting |
| 6 | Transport Class 6 | Multicast, fragmenting |
| 7 - F | Reserved | |

A 16-bit **Sequence** count value is prepended to all Calss 1, 2, and 3 transports. This value is used to detect delivery of duplicate data packets. Sequence count values are initialized on the first message production and incremented on each subsequent new data production. A resend of old data shall not cause the sequence count to change, and consumer shall ignore data when it is received with a duplicate sequence count. Consuming applications can use this mechanism to distinguish between new samples and old samples that were sent to maintain the connection.

#### 4.3.1.1.12 Connection Path Size

The connection path size shall be the length of the connection path in 16-bit words. The length of the connection path varies during the connection process, since each node in the connection path removes the current port segment and forwards only the remaining path segments to the next node.

### 4.3.1.2 Successful Forward Open Response Message Format

Success shall be returned when the connection requested has been established from this point forward in the path. This reply also shall indicate the connection serial number and the actual packet rate of the connection. Once the successful reply has been received, the CIP connection shall be open from this point forward in the path.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>40 bytes | (2E 00) |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (00 00) |
| | Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | (00 00) |
| | Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B2 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 30 bytes | (1E 00) |
| | Below Fields are the CIP Forward open Response Message | | | | |
| | Below fields are Message Router Response Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0xD4** Fwd_Open response code | **D4** |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | General Status | 1 | USINT | One of the General Status codes , 0x00 for success<br>(Refer to chapter:6.2: Appendix 2: CIP General Status Codes) | (00) |
| | Additional Status Size | 1 | USINT | Number of words (16bit) of additional status. | (00) |
| | Additional Status | 0 | Array of UINT | Additional status. Not be presented if size of additional status is 0. | |
| | Below fields are Message Router Response Packet Data Fields | | | | |
| | O-T Connection ID | 4 | UDINT | Network Connection ID to be used for the locak link, originator to target.<br>Value is generated by target and returned to originator via Fwd_Open Response message.<br>(Refer to chapter: **4.3.1.1.3-Network Connection ID** | (01 A9 FD 00) |

| | | | | | |
|---|---|---|---|---|---|
| | | | | **(CID)**) | |
| | T-O Connection ID | 4 | UDINT | Return same value received in the Fwd_Open request message. | (01 00 00 00) |
| | Connection Serial# | 2 | UINT | Return same value received in the Fwd_Open request message. | (00 F0) |
| | Originator Vendor ID | 2 | UINT | Return same value received in the Fwd_Open request message. | (52 49) |
| | Originator Serial# | 4 | UDINT | Return same value received in the Fwd_Open request message. | (01 00 00 00) |
| | O-T API | 4 | UDINT | Actual Packet Rate, originator to target, microseconds. A router shall use the lesser of this value and the T_to_O API for the expected_packet_rate of the connection.<br>e.g. 7500ms (0x007270E0)<br>(Refer to chapter: **4.3.1.1.9-Required Packet Interval (RPI)**) | (E0 70 72 00) |
| | T-O API | 4 | UDINT | Actual Packet Rate, target to originator, microseconds. A router shall use the lesser of this value and the T_to_O API for the expected_packet_rate of the connection.<br>e.g. 7500ms (0x007270E0) | (E0 70 72 00) |
| | Application Reply Size | 1 | USINT | Number of words (16 bit) in Application Reply<br>e.g. 0x00 – No application reply data | (00) |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | Application Reply | 0 | Array of Byte | Not be presented | |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

After CIP connection is opened, Targets will monitor the incoming packet **interval (O-T RPI, default is 30seconds)**. If there is no any incoming CIP message is received from Originator, target will close the CIP connection **without** sending any Fwd_Close Request message to originator. And then target will not return any CIP response message to originator. It will wait for originator to re-initiate the opening of connection.

When CIP connection is opened, after originator sent the first CIP message to target, it will resend the last outgoing CIP message (with the same CIP sequence number assigned) when O-T RPI timeout and no any CIP message need to be sent. This is to make sure the CIP connection will not be closed by the target due to the O-T RPI timeout.

#### 4.3.1.3 Unsuccessful Forward Open Response Message Format

The following format shall be used for all Forward Open failures. The requested connection shall not be established, and the object specific status words shall contain information about the reason for the failure. The remaining_path_size shall contain the length of the path at the point the connection request failed.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>32 bytes | (20 00) |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (00 00) |
| | Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | (00 00) |
| | Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B2 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 16 bytes | (10 00) |
| | Below Fields are the CIP Forward open Response Message | | | | |
| | Below fields are Message Router Response Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0xD4** Fwd_Open response code | **D4** |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | General Status | 1 | USINT | One of the General Status codes , 0x00 for success<br>(Refer to chapter:6.2: Appendix 2: CIP General Status Codes) | (01) |
| | Additional Status Size | 1 | USINT | Number of words (16bit) of additional status. | (01) |
| | Additional Status | 2 | Array of UINT | Additional status. Not be presented if size of additional status is 0. | (04 02) |
| | Below fields are Message Router Response Packet Data Fields | | | | |
| | Connection Serial# | 2 | UINT | Return same value received in the Fwd_Open request message. | (00 F0) |
| | Originator Vendor ID | 2 | UINT | Return same value received in the Fwd_Open request message. | (52 49) |

| | | | | | |
|---|---|---|---|---|---|
| | Originator Serial# | 4 | UDINT | Return same value received in the Fwd_Open request message. | (01 00 00 00) |
| | Remaining Path Size | 1 | USINT | This field is only present with routing type errors and indicates the number of words in the original route path (Connection_Path parameter of the Forward Open Request) as seen by the router that detects the error. | (03) |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

In the failure response, the remaining remaining_path_size shall be the “pre-stripped” size. This shall be the size of the path when the node first receives the request and has not yet started processing it. A target node may return either the “pre-stripped” size or 0 for the remaining remaining_path_size.

A duplicate Forward_Open service shall be defined as a Forward_Open service whose vendor_ID, connection_serial_number, and originator_serial_number match an existing connection’s parameters. If the duplicate Forward_Open service is a null Forward_Open service (defined as the connection type in both the O-T and T-O network connection parameter fields are NULL), the the Forward_Open service shall be forwarded to the application for further processing. Null Forward_Open request may be used to reconfigure the connection. The Connection Manager in the intermediate nodes need not allocate additional resources for a duplicate Forward_Open request since the resource have already been allocated. If the duplicate Forward_Open request is not NULL, the a general status = 0x01, extended status = 0x0100 shall be returned.

## 4.3.2 Forward Close Message

CIP Forward Close Service (0x4E) is used to close a CIP Connection with a Target device (and all other nodes in the connection path). The Forward_Close request shall remove a connection from all the nodes participating in the original connection. The Forward Close shall be sent between Connection Managers as specified in the connection_path. The Forward Close request shall cause all resources in all nodes participating in the CIP connection to be de-allocated, including connection Ids, bandwidth, and internal memory buffers.

If an intermediate node cannot find the connection that is to be closed (it may have timed out at the node), the Forward Close request shall still be forwarded to downstream nodes or the target application.

### 4.3.2.1 Forward Close Request Message Format

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>64 bytes | 40 00 |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (00 00) |
| | Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | (00 00) |
| | Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B2 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 24 bytes | (18 00) |
| | Below Fields are the CIP Forward Close Request Message | | | | |
| | Below fields are Message Router Request Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0x4E** Fwd_Close request code | **4E** |
| | Request Path Size | 1 | USINT | 0x02 Path size in words | (02) |
| | Request Path | 4 | Array of bytes | EPATH (or IOI) 20,06 (class, CM object);24,01 (Instance 1) | (20 06 24 01) |
| | Below fields are Message Router Request Packet Data Fields | | | | |
| | Connection | 1 | BYTE | 1-second ticks (priority ignored), e.g. 0x07 | (07) |

| | | | | | |
|---|---|---|---|---|---|
| | Priority/Time_tick | | | (Refer to chapter: **4.3.1.1.1-Connection Priority / Time_Tick)** | |
| | Connection Timeout_Ticks | 1 | USINT | (ticks*tick_time)<br>e.g. 0xE8 (232), the Total Timeout Value = $2^7$ x 232 = 29696ms<br>(Refer to chapter: **4.3.1.1.2-Timeout_ticks)** | (E8) |
| | Connection Serial# | 2 | UINT | Chosen by originator<br>(Refer to chapter: **4.3.1.1.4-Connection Serial Number**) | (00 F0) |
| | Originator Vendor ID | 2 | UINT | From ID object (CIP vendor ID).<br>E.g. Rockwell Automation (0x0001)<br>Let's use **0x4952** to represent "**IR**"<br>(Refer to chapter: **4.3.1.1.5-Originator Vendor ID**) | (52 49) |
| | Originator Serial# | 4 | UDINT | Unique# for all devices manufactured by the same vendor.<br>In IR-BHS SAC, Each SAC2PLC GW service should be assigned with a unique serial number. E.g.<br>• SAC2PLC1 GW – 0x00000001<br>• SAC2PLC2 GW – 0x00000002<br>• SAC2PLC3 GW – 0x00000003<br>(Refer to chapter: **4.3.1.1.6-Originator Serial Number**) | (01 00 00 00) |
| | Connection Path Size | 1 | USINT | Number of 16 bit words in Connection Path:<br>0x03 – ControlLogix via backplane<br>0x02 – direct network device | (03) |
| | Reserved | 1 | USINT | Shall to 0x00 | 00 |
| | Connection Path | ? | Padded EPATH | (from PLC5E MSG instruction)<br>e.g. 0x010020022401<br>01 – Backplane port of 1756-ENET<br>00 – Logix5550 in slot 0<br>20 02 – Class segment, 02 is MR<br>24 01 – Instance segment, No.1 | (01 00 20 02 24 01) |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

#### 4.3.2.2 Successful Forward Close Response Message Format

Forward Close service request shall be successful when a request is received whose Originator Vender ID, Connection Serial Number, and Originator Serial Number match an existing connection's parameters. Additional information provided by this service (ie, Connection Path) shall be ignored by the target.

Success shall be returned when the CIP connection has been deleted at the target. The originator, and each intermediate node along the path, closes the connection and releases resources associated with that CIP connection when the success response is received.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>64 bytes | 40 00 |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (00 00) |
| | Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | (00 00) |
| | Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B2 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 14 bytes | (0E 00) |
| | Below Fields are the CIP Forward Close Response Message | | | | |
| | Below fields are Message Router Response Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0xCE** Fwd_Close response code | **CE** |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | General Status | 1 | USINT | One of the General Status codes , 0x00 for success<br>(Refer to chapter:6.2: Appendix 2: CIP General Status Codes) | (00) |
| | Additional Status Size | 1 | USINT | Number of words (16bit) of additional status. | (00) |
| | Additional Status | 0 | Array of UINT | Additional status. Not be presented if size of additional status is 0. | |
| | Below fields are Message Router Response Packet Data Fields | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | Connection Serial# | 2 | UINT | Return same value received in the Fwd_Open request message. | (00 F0) |
| | Originator Vendor ID | 2 | UINT | Return same value received in the Fwd_Open request message. | (52 49) |
| | Originator Serial# | 4 | UDINT | Return same value received in the Fwd_Open request message. | (01 00 00 00) |
| | Application Reply Size | 1 | USINT | Number of words (16 bit) in Application Reply<br>e.g. 0x00 – No application reply data | (00) |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | Application Reply | 0 | Array of Byte | Not be presented | |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

### 4.3.2.3 Unsuccessful Forward Close Response Message Format

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendRRData (0x6F) | 6F 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>32 bytes | (20 00) |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be 0 to indicate a UCMM message<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (00 00) |
| | Address Length | 2 | UINT | This field shall be 0 since UCMM messages use the NULL address item. | (00 00) |
| | Data Item ID | 2 | UINT | This field shall be 0x00B2 to indicate that the following data field is the CIP Unconnected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B2 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 16 bytes | (10 00) |
| | Below Fields are the CIP Forward Close Response Message | | | | |
| | Below fields are Message Router Response Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0xD4** Fwd_Close response code | **D4** |
| | Reserved | 1 | USINT | Shall be 0x00 | (00) |
| | General Status | 1 | USINT | One of the General Status codes , 0x00 for success<br>(Refer to chapter:6.2: Appendix 2: CIP General Status Codes) | (01) |
| | Additional Status Size | 1 | USINT | Number of words (16bit) of additional status. | (01) |
| | Additional Status | 2 | Array of UINT | Additional status. Not be presented if size of additional status is 0. | (04 02) |
| | Below fields are Message Router Response Packet Data Fields | | | | |
| | Connection Serial# | 2 | UINT | Return same value received in the Fwd_Open request message. | (00 F0) |
| | Originator Vendor ID | 2 | UINT | Return same value received in the Fwd_Open request message. | (52 49) |
| | Originator Serial# | 4 | UDINT | Return same value received in the Fwd_Open request message. | (01 00 00 00) |
| | Remaining Path Size | 1 | USINT | This field is only present with routing type errors and indicates the number of words in the original route path (Connection_Path parameter of the Forward | (03) |

| | | | | | |
|---|---|---|---|---|---|
| 🟩 | | | | Open Request) as seen by the router that detects the error. | |
| 🟩 | Reserved | 1 | USINT | Shall be 0x00 | (00) |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

## 4.3.3 ControlLogix5000 CIP Data Table Write Message

### 4.3.3.1 Introduction

In the IR-BHS SAC-ControlLogx 5000 PLC interface, the application data is exchanged between SAC and PLCs by using CIP Data Table Write service (**0x4D**) to write a block of data to target. The data starts from the specified address at the IOI string. And the data type must match exactly for the write to occur.

The CIP Data Table Write message is CIP connected message. It needs to be sent by EIP SendUnitData service.

**CIP Data Table Write Request Format:**

| Parameter Name | Data Length (Bytes) | Data Type | Description |
|---|---|---|---|
| Service Code | 1 | USINT | CIP data table write request service: 4D |
| Request Path Size | 1 | USINT | The number of 16 bit words in the Request_Path field |
| Request Path (IOI) | ? | Padded EPATH | IOI Segments<br>(Refer to chapter: **4.3.3.1.1-IOI (Internal Object Identifier) Segments**) |
| Abbreviated Data Type | 2 | UINT | 0x00C1 = BOOL<br>0x00C2 = SINT<br>0x00C3 = INT<br>…<br>(Refer to chapter: **4.3.3.1.2-Abbreviated Data Type** Segments) |
| Data Size | 2 | UINT | Data length in specific Abbreviated Data Type |
| Data | ? | Array of Data Type values | Application data |

**CIP Data Table Write Reponse Format:**

| Parameter Name | Data Length (Bytes) | Data Type | Description |
|---|---|---|---|
| Reply Service Code | 1 | USINT | CIP data table write response service: CD |
| Reserved | 1 | USINT | Shall be zero (0x00) |
| General Status | 1 | USINT | One of the General Stauts codes listed in Appendix A (Status Codes) |
| Size of Additional Status | 1 | USINT | Number of 16 bit words in Additional Status array. |
| Additional Status | ? | Array of UINT | Additional status. Not be presented if size of additional status is 0. |

#### 4.3.3.1.1 IOI (Internal Object Identifier) Segments

CIP addressing is done through IOI segments. The controller used the following types of segments (adheres to the ControlNet International specification):

- Class segment versions
  - One byte version

| | |
|---|---|
| 20 | xx |

Where: xx is the class code value.

- Two byte version

| | | | |
|---|---|---|---|
| 21 | 00 | xx | xx |

Where: xxxx is the class code value, low byte first.

- Instance segment versions
  - One byte version

| | |
|---|---|
| 24 | xx |

Where: xx is the instance code value.

- Two byte version

| | | | |
|---|---|---|---|
| 25 | 00 | xx | xx |

Where: xxxx is the instance code value, low byte first.

- Element segment versions
  - One byte version

| | |
|---|---|
| 28 | xx |

Where: xx is the element number.

- Two byte version

| | | | |
|---|---|---|---|
| 29 | 00 | xx | xx |

Where: xxxx is the element number, low byte first.

- Four byte version

| | | | | | |
|---|---|---|---|---|---|
| 2A | 00 | xx (lowest) | xx | xx | xx (highest) |

Where: xx is the element number, low byte to highest byte.

- Symbolic segment version, one byte character version:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 91 | Len | 1st | 2nd | ... ... | last | 00 (Pad) | 28 | 00 |

Where:

**Len**, USINT type value, is the length of the symbolic string in bytes (not including header or trailing pad).

**1st, 2nd, 3rd** are the characters in order.

**Pad**, USINT value 0x00, is necessary when the number of characters is odd.

If SAC needs send data to a specific tag defined in one ControlLogx 5000 PLC, the tag name will be defined in Symbolic Segment format in the CIP connected message.

Example 1: if the tag name is ***TLG_Rec***, then the IOI segment will be:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 91 | 07 | 54 | 4C | 47 | 5F | 52 | 65 | 63 | 00 | 28 | 00 |

Example 2: if the tag name is ***TLG_Rece***, then the IOI segment will be:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 91 | 08 | 54 | 4C | 47 | 5F | 52 | 65 | 63 | 65 | 00 | 28 | 00 |

#### 4.3.3.1.2 Abbreviated Data Type Segments

CIP data table read and writes require an abbreviated type field.

- **BOOL**

| | | |
|---|---|---|
| C1 | 00 | xx |

Where: xx is 0x00 if the bit is 0, and 0xFF if the bit is 1.

- **BIT ARRAY**

| | | | | | |
|---|---|---|---|---|---|
| D3 | 00 | lowest | low | high | Highest |

The 4 bytes are packed lowest to highest.

- **SINT**

| | | |
|---|---|---|
| C2 | 00 | xx |

Where: xx is data value.

- **INT**

| | | | |
|---|---|---|---|
| C3 | 00 | low | high |

Data bytes packed low byte to high byte.

- **DINT**

| | | | | | |
|---|---|---|---|---|---|
| C4 | 00 | lowest | low | high | Highest |

The 4 bytes are packed lowest to highest.

- **REAL**

| | | | | | |
|---|---|---|---|---|---|
| CA | 00 | lowest | low | high | Highest |

The 4 bytes are packed lowest to highest.

0x00C2 (SINT) is prefered data type for exchange application data between SAC and PLCs.

### 4.3.3.2 CIP Data Table Write Request Message Format

Following message is for sending application data ***"12345"*** to tag ***"TLG_R"*** in the target node.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendUnitData (0x70) | 70 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>43 bytes | 2B 00 |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be **0x00A1** to indicate a connected address item<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (A1 00) |
| | Address Length | 2 | UINT | This field shall be 4 to indicate the O-T CID. | (04 00) |
| | Address | 4 | UDINT | O-T CIP Connection ID, returned by target via Fwd_Open response message. | (01 03 54 01) |
| | Data Item ID | 2 | UINT | This field shall be **0x00B1** to indicate that the following data field is the CIP Connected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B1 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 23bytes | (17 00) |
| | Sequence Count | 2 | UINT | A 16-bit **Sequence** count value is prepended to all Calss 1, 2, and 3 transports. This value is used to detect delivery of duplicate data packets. Sequence count values are initialized on the first message production and incremented on each subsequent new data production. A resend of old data shall not cause the sequence count to change, and consumer shall ignore data when it is received with a duplicate sequence count. Consuming applications can use this mechanism to distinguish between new samples and old samples that were sent to maintain the connection. | (01 00) |
| | Below Fields are the CIP Data Table Write Request Message | | | | |
| | Below are CIP Data Table Write Request Packet Header Fields | | | | |
| | Service Code | 1 | USINT | **0x4D,** CIP data table write request service | **4D** |
| | Request Path Size | 1 | USINT | Path size in words. 0x05 – 5 words<br>Including length of IOI Segment Type, Symbol Path Size, Symbol Path, Pad, Reserved 5 fields. | (05) |
| | IOI Segment Type | 1 | USINT | **0x91**, to indicate the path is Symbolic segment | (91) |
| | Symbol Path Size | 1 | USINT | Length of the symbolic string in bytes (not including header or trailing pad). 0x05 – 5 bytes | (05) |

| | | | | | |
|---|---|---|---|---|---|
| | Symbol Path | 5 | Array of USINT | The name of tag ***"TLG_R"***, 5 bytes | (54 4C 47 5F 52) |
| | Pad | 1 | USINT | 0x00<br>Need to be padded to the symbol path string if the number of characters of Symbol Path is odd. | (00) |
| | Reserved | 2 | UINT | Always be **0x0028**. | (28 00) |
| | Below are CIP Data Table Write Request Packet Data Fields | | | | |
| | Abbreviated Data Type | 2 | UINT | 0x00C2 (SINT) is prefered data type for exchange application data between SAC and PLCs.<br>(Refer to chapter: **4.3.3.1.2-Abbreviated Data Type** Segments) | (C2 00) |
| | Data Size | 2 | UINT | Data length in specific Abbreviated Data Type.<br>0x0005 – 5 8-bit integers | (05 00) |
| | Data | 5 | Array of SINT | Application data: ***"12345"*** | (31 32 33 34 35) |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

### 4.3.3.3 CIP Data Table Write Response Message Format

Following message is the CIP Data Table Write response for of above CIP Data Table Write request message.

| Structure | Field Name | Data Length (Bytes) | Data Type | Description | Sample Value |
|---|---|---|---|---|---|
| Encapsulation Header | Command | 2 | UINT | Encapsulation command code, SendUnitData (0x70) | 70 00 |
| | Length | 2 | UINT | Length of command specific data portion<br>26 bytes | 1A 00 |
| | Session Handle | 4 | UDINT | Unique number set by target and returned to originator by RegisterSession reply message | (00 0A 02 15) |
| | Status | 4 | UDINT | 0x00000000 in request | 00 00 00 00 |
| | Sender Context | 8 | Array of 8 USHORT | Transaction ID set by requestor, e.g. SAC2PLC1 | (53 41 43 32 50 4C 43 31) |
| | Options | 4 | UDINT | 0x00000000, Reserved for future use | 00 00 00 00 |
| Command Specific Data | Interface handle | 4 | UDINT | Shall be 0 for CIP | 00 00 00 00 |
| | Timeout | 2 | UINT | Operation Timeout | 00 00 |
| | Item Count | 2 | UINT | This field shall be 2 since one address item and one data item are used. | 02 00 |
| | Address Item ID | 2 | UINT | This field shall be **0x00A1** to indicate a connected address item<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (A1 00) |
| | Address Length | 2 | UINT | This field shall be 4 to indicate the O-T CID. | (04 00) |
| | Address | 4 | UDINT | T-O CIP Connection ID, Assigned by originator and sent to target via Fwd_Open request message. | (01 00 00 00) |
| | Data Item ID | 2 | UINT | This field shall be **0x00B1** to indicate that the following data field is the CIP Connected Message.<br>(Refer to chapter: **3.2.8 - Common Packet Format**) | (B1 00) |
| | Data Length | 2 | UINT | Length of the next data field in bytes (length of the MR request packet), 6bytes | (06 00) |
| | Sequence Count | 2 | UINT | Return the same sequence count value received in the CID Data Table Write Request message | (01 00) |
| | Below Fields are the CIP Data Table Write Response Message | | | | |
| | Service Code | 1 | USINT | **0xCD,** CIP data table write response service | **CD** |
| | Reserved | 1 | USINT | Shall be zero (0x00) | (00) |
| | General Status | 1 | USINT | One of the General Status codes , 0x00 for success<br>(Refer to chapter:6.2: Appendix 2: CIP General Status Codes) | (00) |
| | Size of Additional Status | 1 | USINT | Number of 16 bit words in Additional Status array. | (00) |
| | Additional Status | 0 | Array of UINT | Additional status. Not be presented if size of additional status is 0. | |

Note: The values inside the bracket in the Sample Data column are the data sample of particular fields. The value without brackets is the constant value of the field.

## 5. PROJECT RELATED PROTOCOL PARAMETERS

Due to the BHS layout and control system design varies in each BHS projects. The settings of protocol parameters are different too. Following are the list of project related parameters of TCP, EIP and CIP protocols.

### 5.1 TCP PROTOCOL LAYER

| No. | Parameter | Description | Value |
|---|---|---|---|
| 1 | IP Address | The IP address of both SAC and ControlLogix PLCs. Both SAC and PLC are TCP server and TCP clients. | Refer to project IP Address Assignment List. |
| 2 | TCP Port Number | The reserved TCP port number for EtherNet/IP protocol is **0xAF12** (**44818**).<br>Both SAC and PLC TCP server object will be listening on this port to accept the connection request from TCP clients.<br>There is no reserved port number on TCP client nodes. TCP client could use any available port number to connect to the TCP server port 44818. | 44818 |

### 5.2 EIP PROTOCOL LAYER

| No. | Parameter | Description | Value |
|---|---|---|---|
| 1 | SessionHandle | Target (client) is responsible for generating the SessionHandle and reply to Originator via RegisterSession Reply message. | 4 bytes unique number |
| 2 | SenderContext | Name of sender application<br>For SAC-PLC GW Service:<br>**SAC2PLC1** – SAC to PLC01 GW Svc<br>**SAC2PLC2** – SAC to PLC02 GW Svc<br><br>For ControlLogix PLC:<br>Sender Context will be generated by PLC H/W itself. | Refer to Application Layer Protocol – SAC2PLC of individual project |
| 3 | ProtocolVersion | Request protocol version shall be set to 0x0001. | 0100 |
| 4 | Capability Flags | Returned by ListServices Reply Message. 0x0120<br>Bits 0 – 4: Reserved for legacy (RA)<br>Bit 5: 1 – Support CIP Encapsulation via TCP,<br>0 – Unsupport.<br>Bits 6 – 7: Reserved for legacy (RA)<br>Bit 8: 1 – Support CIP Encapsulation via UDP,<br>0 – Unsupport.<br>Bits 9 – 15: Reserved for future expansion | 2010 |

## 5.3 CIP PROTOCOL LAYER

| No. | Parameter | Description | Value |
|---|---|---|---|
| 1 | PriorityTime_tick | (Refer to chapter: **4.3.1.1.1-Connection Priority / Time_Tick)**<br>0x07 | 07 |
| 2 | ConnectionTimeout_Ticks | (Refer to chapter: **4.3.1.1.2-Timeout_ticks)**<br>0xE8 | E8 |
| 3 | O-T_CID | (Refer to chapter: **4.3.1.1.3-Network Connection ID (CID)**) | 00000000 in request<br>And actual value is returned by target Fwd_Open Response |
| 4 | T-O_CID | (Refer to chapter: **4.3.1.1.3-Network Connection ID (CID)**) | Assigned by originator and sent to target by Fwd_Open Request |
| 5 | ConnectionSN | CIP Connection Serial Number<br>(Refer to chapter: **4.3.1.1.4-Connection Serial Number**) | Assigned by originator |
| 6 | OriginatorVendorID | Originator Vendor ID<br>IR-BHS SAC vendor ID is ***"IR"*** (0x4952)<br>(Refer to chapter: **4.3.1.1.5-Originator Vendor ID**) | 5249 |
| 7 | OriginatorSN | Unique# for all devices manufactured by the same vendor.<br>In IR-BHS SAC, Each SAC2PLC GW service should be assigned with a unique serial number. E.g.<br>SAC2PLC1 GW – **0x00000001**<br>SAC2PLC2 GW – **0x00000002**<br>SAC2PLC3 GW – **0x00000003**<br>(Refer to chapter: **4.3.1.1.6-Originator Serial Number**) | Refer to Application Layer Protocol – SAC2PLC of individual project |
| 8 | ConnectionTimeoutMultiplier | Used to calculate the connection path (inactivity) timeout (= Multiplier x RPI). CIP connection will be closed if the connection path timeout elapsed. 0x00<br>(Refer to chapter: **4.3.1.1.8-Connection Timeout Multiplier**) | 00 |
| 9 | O-T_RPI | Requested RPI, originator to target, microseconds<br>(Refer to chapter: **4.3.1.1.9-Required Packet Interval (RPI)**)<br>30000ms (0x01C9C380) | 80 C3 C9 01 |
| 10 | T-O_RPI | Requested RPI, target to originator, microseconds<br>30000ms (0x01C9C380) | 80 C3 C9 01 |
| 11 | O-T_ConnectionParam | O->T Network Connection Parameters: 0x43F6<br>0... .... .... .... = Owner: Exclusive (0)<br>.10. .... .... .... = Connection Type: Point | F643 |

| | | | |
|---|---|---|---|
| | | to Point (2)<br>.... 00.. .... .... = Priority: Low Priority (0)<br>.... ..1. .... .... = Connection Size Type: Variable (1)<br>.... ...1 1111 0110 = Connection Size: 502<br>(Refer to chapter: **4.3.1.1.7-Network Connection Parameters**) | |
| 12 | T-O_ConnectionParam | Refer to O-T_ConnectionParam | F643 |
| 13 | TransportClassTrigger | **0xA3**, Server transport, class 3, application trigger<br>(Refer to chapter: **4.3.1.1.11 - Transport Class and Trigger**) | 0xA3 |
| 14 | ConnectionPathSize | Number of 16 bit words in Connection Path:<br>**0x03 – ControlLogix via backplane**<br>0x02 – direct network device | 03 |
| 15 | ConnectionPath | e.g. 0x010020022401<br>**01 – Backplane port of 1756-ENET**<br>**00 – Logix5550 in slot 0**<br>20 02 – Class segment, 02 is MR<br>24 01 – Instance segment, No.1 | 010020022401 |
| 16 | FwdCloseRequestPath | EPATH (or IOI) 20,06 (class, CM object);24,01 (Instance 1) | 20062401 |
| 17 | MSGTagOfPLC | The tag name in ControlLogix PLC for receiving the incoming application data from SAC | TAG_SAC |
| 18 | MSGTagOfSAC | The tag name in SAC for receiving the incoming application data from PLC | TAG_PLC |

# 6. APPENDIX 1

## 6.1 APPENDIX 1: EIP STATUS CODES

The value in the Status field shall indicate whether or not the receiver was able to execute the requested encapsulation command. A value of zero in a reply shall indicate successful execution of the command. In all requests issued by the sender, the Status field shall contain zero. If the receiver receives a request with a non-zero Status field, the request shall be ignored and no reply shall be generated. The status codes shall be as follows:

| No. | Status Code | Comment |
|---|---|---|
| 1 | 0x0000 | Success |
| 2 | 0x0001 | The sender issued an invalid or unsupported encapsulation command. |
| 3 | 0x0002 | Insufficient memory resources in the receiver to handle the command. This is not an application error. Instead, it only results if the encapsulation layer cannot obtain memory resources that it needs. |
| 4 | 0x0003 | Poorly formed or incorrect data in the data portion of the encapsulation message. |
| 5 | 0x0004 – 0x0063 | Reserved for legacy (RA). |
| 6 | 0x0064 | An originator used an invalid session handle when sending an encapsulation message to the target. |
| 7 | 0x0065 | The target received a message of invalid length |
| 8 | 0x0066 – 0x0068 | Reserved for legacy (RA). |
| 9 | 0x0069 | Unsupported encapsulation protocol revision. |
| 10 | 0x006A – 0xFFFF | Reserved for future expansion |

## 6.2 APPENDIX 2: CIP GENERAL STATUS CODES

### 6.2.1 General Status Code Defined in CIP Specification

The following table lists the Status Codes that may be present in the General Status Code field of an Error Response message. Note that the Extended Code Field is available for use in further describing any General Status Code. Extended Status Codes are unique to each General Status Code within each object. Each object shall manage the extended status values and value ranges (including vendor specific). All extended status values are reserved unless otherwise indicated within the object definition.

| General Status Code (in hex) | Status Name | Description of Status |
|---|---|---|
| 00 | Success | Service was successfully performed by the object specified. |
| 01 | Connection failure | A connection related service failed along the connection path. |
| 02 | Resource unavailable | Resources needed for the object to perform the requested service were unavailable |
| 03 | Invalid parameter value | See Status Code 0x20, which is the preferred value to use for this condition. |
| 04 | Path segment error | The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered. |
| 05 | Path destination unknown | The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered. |
| 06 | Partial transfer | Only part of the expected data was transferred. |
| 07 | Connection lost | The messaging connection was lost. |
| 08 | Service not supported | The requested service was not implemented or was not defined for this Object Class/Instance. |
| 09 | Invalid attribute value | Invalid attribute data detected |
| 0A | Attribute list error | An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status. |
| 0B | Already in requested mode/state | The object is already in the mode/state being requested by the service |
| 0C | Object state conflict | The object cannot perform the requested service in its current mode/state |
| 0D | Object already exists | The requested instance of object to be created already exists. |
| 0E | Attribute not settable | A request to modify a non-modifiable attribute was received. |
| 0F | Privilege violation | A permission/privilege check failed |
| 10 | Device state conflict | The device's current mode/state prohibits the execution of the requested service. |
| 11 | Reply data too large | The data to be transmitted in the response buffer is larger than the allocated response buffer |
| 12 | Fragmentation of a primitive value | The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type. |

| | | |
|---|---|---|
| 13 | Not enough data | The service did not supply enough data to perform the specified operation. |
| 14 | Attribute not supported | The attribute specified in the request is not supported |
| 15 | Too much data | The service supplied more data than was expected |
| 16 | Object does not exist | The object specified does not exist in the device. |
| 17 | Service fragmentation sequence not in progress | The fragmentation sequence for this service is not currently active for this data. |
| 18 | No stored attribute data | The attribute data of this object was not saved prior to the requested service. |
| 19 | Store operation failure | The attribute data of this object was not saved due to a failure during the attempt. |
| 1A | Routing failure, request packet too large | The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service. |
| 1B | Routing failure, response packet too large | The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service. |
| 1C | Missing attribute list entry data | The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior. |
| 1D | Invalid attribute value list | The service is returning the list of attributes supplied with status information for those attributes that were invalid. |
| 1E | Embedded service error | An embedded service resulted in an error. |
| 1F | Vendor specific error | A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error. |
| 20 | Invalid parameter | A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification. |
| 21 | Write-once value or medium already written | An attempt was made to write to a write-once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established. |
| 22 | Invalid Reply Received | An invalid reply is received (e.g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies. |
| 23 - 24 | | Reserved by CIP for future extensions |
| 25 | Key Failure in path | The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed. |
| 26 | Path Size Invalid | The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included. |
| 27 | Unexpected attribute in list | An attempt was made to set an attribute that is not able to be set at this time. |

| | | |
|---|---|---|
| 28 | Invalid Member ID | The Member ID specified in the request does not exist in the specified Class/Instance/Attribute |
| 29 | Member not settable | A request to modify a non-modifiable member was received |
| 2A | Group 2 only server general failure | This error code may only be reported by DeviceNet group 2 only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable. |
| 2B – CF | | Reserved by CIP for future extensions |
| D0 – FF | Reserved for Object Class and service errors | This range of error codes is to be used to indicate Object Class specific errors. Use of this range should only be performed when none of the Error Codes presented in this table accurately reflect the error that was encountered. |

### 6.2.2 General Status Code Defined in RSLogix Online Help

For ControlLogix (CIP) error codes, the programming software does not always display the full description.

| **Error code (hex)** | **Description** |
|---|---|
| 0001 | Connection failure (see extended error codes |
| 0002 | Insufficient resource |
| 0003 | Invalid value |
| 0004 | IOI syntax error (see extended error codes |
| 0005 | Destination unknown, class unsupported, instance undefined or structure element undefined (see extended error codes |
| 0006 | Insufficient packet space |
| 0007 | Connection lost |
| 0008 | Service unsupported |
| 0009 | Error in data segment or invalid attribute value |
| 000A | Attribute list error |
| 000B | State already exists |
| 000C | Object model conflict |
| 000D | Object already exists |
| 000E | Attribute not settable |
| 000F | Permission denied |
| 0010 | Device state conflict |
| 0011 | Reply will not fit |
| 0012 | Fragment primitive |
| 0013 | Insufficient command data |
| 0014 | Attribute not supported |
| 0015 | Too much data |
| 001A | Bridge request too large |
| 001B | Bridge response too large |
| 001C | Attribute list shortage |
| 001D | Invalid attribute list |
| 001E | Embedded service error |
| 001F | Connection related failure (see extended error codes |
| 0022 | Invalid reply received |
| 0025 | Key segment error |
| 0026 | Invalid IOI error |
| 0027 | Unexpected attribute in list |
| 0028 | DeviceNet error - invalid member ID |
| 0029 | DeviceNet error - member not settable |
| 00D1 | Module not in run state |
| 00FB | Message port not supported |
| 00FC | Message unsupported data type |
| 00FD | Message uninitialized |
| 00FE | Message timeout |
| 00FF | General error (see extended error codes |

## 6.3 APPENDIX 3: CIP ADDITIONAL STATUS CODES

Following additional status code are extracted from RSLogix5000 online help.

### 6.3.1 For CIP General Status Code 0001

| Extended error code (hex) | Description |
|---|---|
| 0100 | Connection in use |
| 0103 | Transport not supported |
| 0106 | Ownership conflict |
| 0107 | Connection not found |
| 0108 | Invalid connection type |
| 0109 | Invalid connection size |
| 0110 | Module not configured |
| 0111 | EPR not supported |
| 0113 | Module Configuration Invalid: data size too small. |
| 0114 | Wrong module |
| 0115 | Wrong device type |
| 0116 | Wring revision |
| 0118 | Invalid configuration format |
| 011A | Application out of connections |
| 0203 | Connection timeout |
| 0204 | Unconnected message timeout |
| 0205 | Unconnected send parameter error |
| 0206 | Message too large |
| 0301 | No buffer memory |
| 0302 | Bandwidth not available |
| 0303 | No screeners available |
| 0305 | Signature match |
| 0311 | Port not available |
| 0312 | Link address not available |
| 0315 | Invalid segment type |
| 0317 | Connection not scheduled |
| 0810 | No target application data available |

### 6.3.2 For CIP General Status Code 000F

| **Extended error code (hex)** | **Description** |
|---|---|
| 0203 | Connection timeout |

### 6.3.3 For CIP General Status Code 0004 AND 0005

| **Extended error code (hex)** | **Description** |
|---|---|
| 0000 | extended status out of memory |
| 0001 | extended status out of instances |

### 6.3.4 For CIP General Status Code 00FF

| Extended error code (hex) | Description |
|---|---|
| 2001 | Excessive IOI |
| 2002 | Bad parameter value |
| 2018 | Semaphore reject |
| 201B | Size too small |
| 201C | Invalid size |
| 2100 | Privilege failure |
| 2101 | Invalid keyswitch position |
| 2102 | Password invalid |
| 2103 | No password issued |
| 2104 | Address out of range |
| 2105 | Address and how many out of range |
| 2106 | Data in use |
| 2107 | Type is invalid or not supported |
| 2108 | Controller in upload or download mode |
| 2109 | Attempt to change number of array dimensions |
| 210A | Invalid symbol name |
| 210B | Symbol does not exist |
| 210E | Search failed |
| 210F | Task cannot start |
| 2110 | Unable to write |
| 2111 | Unable to read |
| 2112 | Shared routine not editable |
| 2113 | Controller in faulted mode |
| 2114 | Run mode inhibited |

## 6.4 APPENDIX 4: MESSAGE SAMPLES

**RegisterSession Request Message:**

65000400000000000000000000000000000000000000000001000000

**RegisterSession Reply Message:**

65000400000402130000000000000000000000000000000001000000

**Fwd_Open Request Message Sample:**

6F00400000060213000000005341433250 4C433100000000000000000000000200000000
000B200300054022006240107E80000000001000000524900F000100000000000000080
C3C901F64380C3C901F643A303010020022401

**Fwd_Open Successful Response Message Sample:**

6F002E0000040213000000005341433250 4C433100000000000000000000000200000000
000B2001E00D40000000243FD000100000052490F000100000080C3C90180C3C90100
00

**Fwd_Open Unsuccessful Response Message Sample:**

6F00200000040213000000005341433250 4C433100000000000000000000000200000000
000B2001000D40001010402FFFF0F00010000000300

**Fwd_Close Message Sample:**

6F00280000290212000000005341433250 4C433100000000000000000000000200000000
000B20018004E022006240107E800E00F00F19746000300010120022401

**CIP Data Table Write Request via SendUnitData Message Sample:**

70002D0000040213000000005341433250 4C4331000000000000000000000000200A1000
400020F0001B1001900010 0**4D**069108544C475F526563652800C20005003939393331

**CIP Data Table Write Response via SendUnitData Message Sample:**

70001A0000290212000000005341433250 4C4331000000000000000000000000200A1000
4000299FD00B1000600010 0**CD**000000

**UnRegisterSession Message Sample:**

66000000002902120000000053414332504C433100000000