EtherNet/IP Scanner – Help File

1.Introduction to EtherNet/IP Scanner-

EtherNet/IP is an industrial Ethernet protocol widely used for real-time communication between PLCs and devices such as **remote I/O modules, VFDs, servo drives, and other controllers**.

In EtherNet/IP communication:

- A **Scanner** actively initiates and manages connections
- An **Adapter** responds to the scanner and exchanges data

The **EtherNet/IP Scanner in OCS** allows the OCS controllers to act as a **master (scanner)** and establish **implicit (cyclic) I/O connections** with EtherNet/IP adapter devices on the network.

2.Enabling EtherNet/IP Scanner in Cscape-

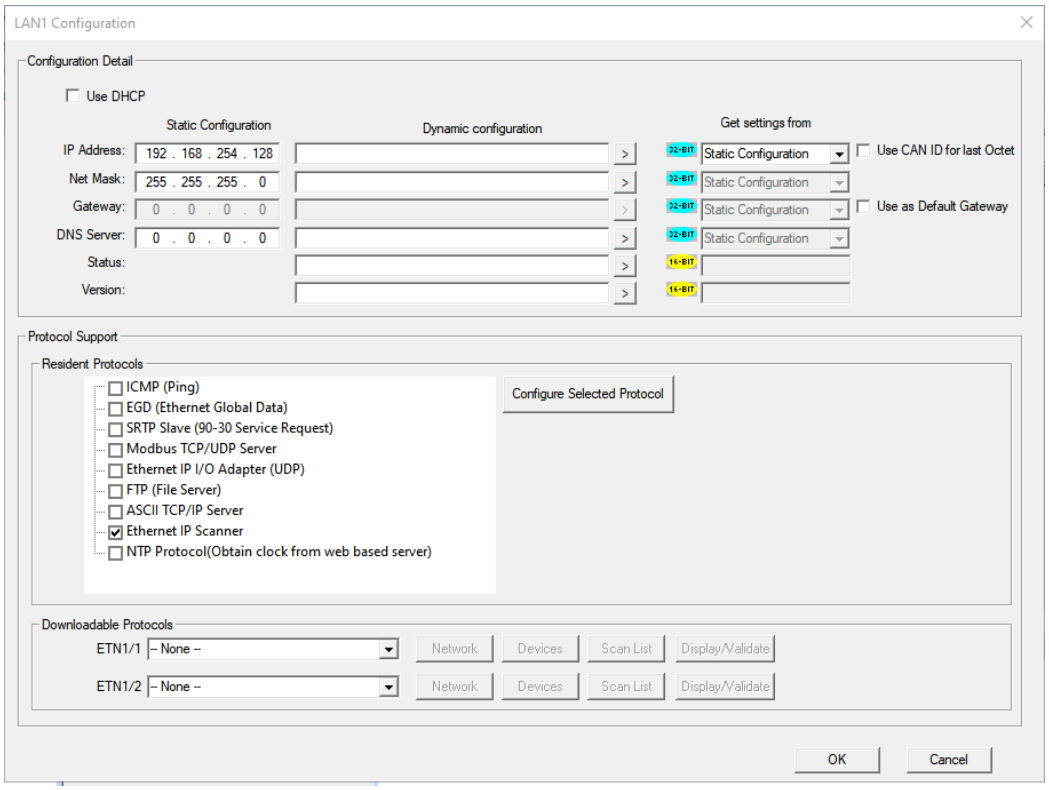
EtherNet/IP Scanner is configured under LAN Configuration as a Resident Protocol.

Navigation Path-

Hardware Configuration

→ LAN Configuration

→ Resident Protocols



3. EtherNet/IP Scanner – Resident Protocol Configuration

Step 1: Enable the Scanner Protocol

In the **Resident Protocols** section:

- Locate **Ethernet IP Scanner**
- **Check the checkbox** next to it

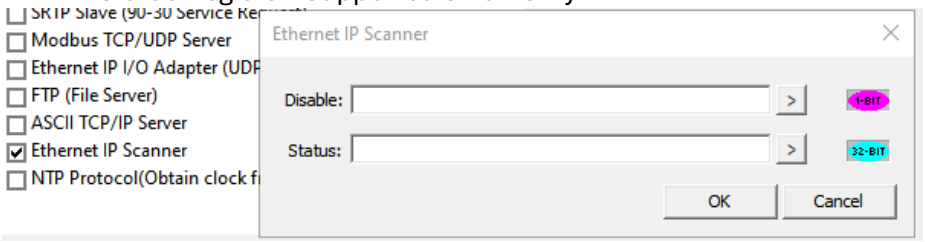
This action enables the EtherNet/IP Scanner service in the OCS firmware.

⚠ **Important**

Simply checking the box is **not sufficient**. Mandatory configuration is required after enabling.

Step 2: Open Scanner Configuration
Double-click on **Ethernet IP Scanner or Right Click (Click on Edit)**
This opens the **EtherNet IP Scanner configuration dialog**, which contains two mandatory fields:

- Disable Bit- Supported BOOLEAN Only
- Status Register- Supported 32 bit Only



3.1 Disable Bit- The Disable Bit allows the application logic to **enable or disable the EtherNet/IP Scanner at runtime**.

How it works

- Disable Bit = **0** → Scanner is **Enabled**
- Disable Bit = **1** → Scanner is **Disabled**

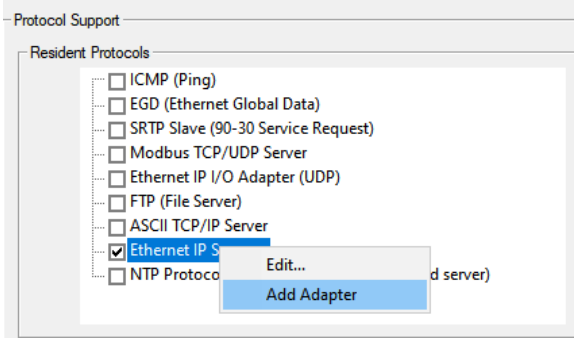
3.2 Status Register- The Status Register provides **runtime diagnostic information** about the EtherNet/IP Scanner.

⚠ **Important**
If these two are not configured, the EtherNet/IP Scanner **will not work**, even though it is enabled & adapters are configured.

Scanner Status Register (32 bit details)-

Status Bits	Description of Status
BIT 0-7	Number of Adapters Connected
BIT 8-31	Reserved for Future Use

4.Adding an EtherNet I/P Adapter Device



Once the EtherNet/IP Scanner is enabled and configured:
a.Under **Resident Protocols**, right-click on **Ethernet IP Scanner**
b.Select **Add Adapter**

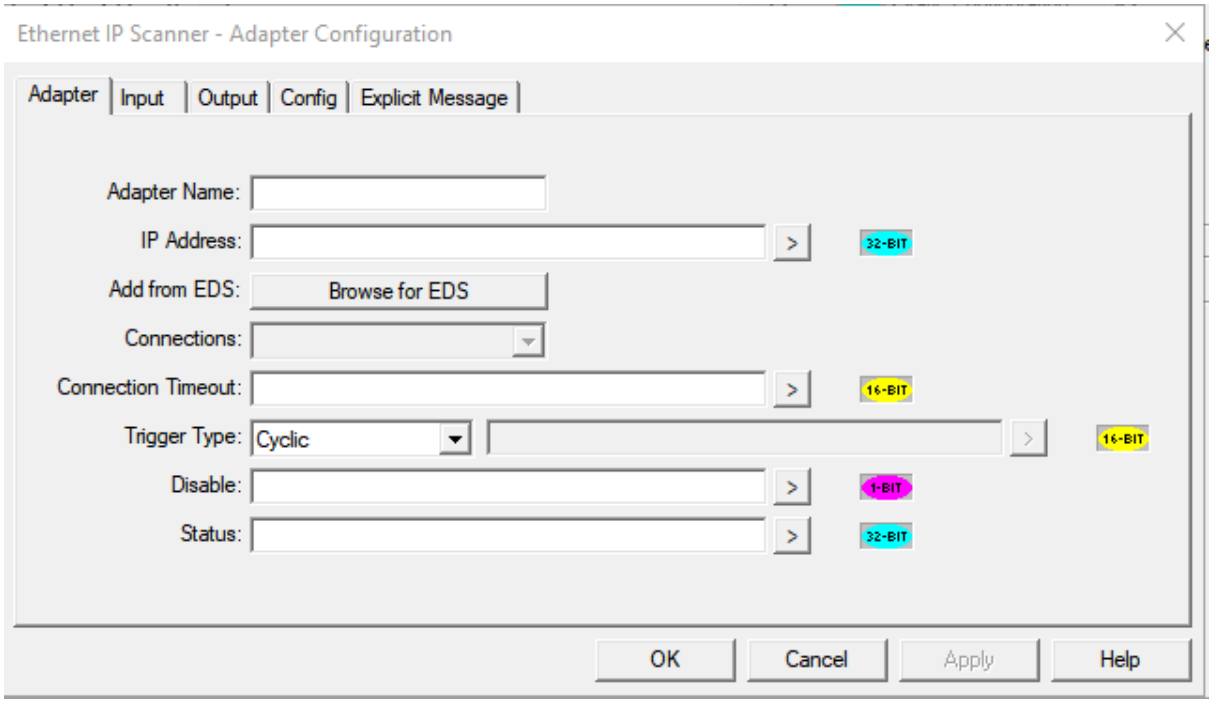
This action is used to add an EtherNet/IP Adapter device (such as a VFD, remote I/O, or drive) that the OCS scanner will communicate with.

Each adapter added here represents **one EtherNet/IP target device** on the network.

⚠ **Important**
Maximum 32 adapters can be configured.

5. EtherNet/IP Adapter Configuration – Adapter Tab

After clicking Add Adapter, the EtherNet IP Scanner – Adapter Configuration window opens.
The **Adapter tab** is used to configure the basic communication parameters required for the OCS EtherNet/IP Scanner to establish **Implicit messaging/Explicit Messaging** with an EtherNet/IP adapter device.
Each adapter configured here represents **one EtherNet/IP target device** on the network.



5.1 Adapter Name

Logical name used to identify the adapter inside the OCS project.

How to configure

- Enter a meaningful name (e.g. PowerFlex_525, RemoteIO_Station1)

Best practice

- Use names that clearly indicate the device type and location
- This name helps during:
  - Debugging
  - Maintenance
  - Multi-device projects

5.2 IP Address

Specifies the network address of the EtherNet/IP adapter device.

Configuration options

- **Static IP**
  - Enter a constant IP address directly
- **Dynamic IP (via variable)**
  - Assign a **32-bit register / variable**
  - Allows the IP address to be changed at runtime through logic or HMI

Important notes

- The adapter IP must be reachable from the OCS LAN interface
- Dynamic IP assignment is useful when:
  - Device IP's are set through logic or from graphics screen
  - Multiple identical systems are deployed

5.3 Add from EDS

Imports EtherNet/IP device information from an EDS (Electronic Data Sheet) file.

Behavior

- Clicking Browse for EDS allows the user to select an EDS file
- After selecting the EDS, a confirmation popup appears showing:
    - Vendor Name
    - Product Type
    - Product Name

This confirmation ensures the correct adapter EDS has been imported.
Benefit

- Automatically populates supported:
    - Connection definitions
    - Assembly information (used in later tabs)
- Reduces manual configuration errors

5.4 Connections

Selects the EtherNet/IP connection type supported by the adapter.

Behavior

- The dropdown lists all connection types defined in the imported EDS file
- Available options depend entirely on:
    - Adapter capability
    - EDS definition

Important

- The selected connection must match the device's expected I/O configuration
- Incorrect connection selection can lead to:
    - Forward Open failure
    - Data size mismatch errors

5.5 Connection Timeout

Defines the maximum time the scanner waits for valid I/O data before declaring a connection failure.

Configuration options

- **Static value**
    - Enter a constant timeout value
- **Dynamic value**
    - Assign a **16-bit register / variable** to change timeout at runtime

Supported range

- 50 milliseconds to 60 seconds

Usage notes

- Increase timeout for:
    - Devices with slow startup
    - Networks with higher latency
- Shorter timeout allows faster fault detection

5.6 Trigger Type

Defines how and when implicit I/O data exchange is triggered.

Available trigger types

- **Cyclic**
  - Data is exchanged continuously based on RPI
- **Change-of-State (COS)**
  - Data updates only when a change occurs
- **Application**
  - Triggered by application logic
- **Dynamic Config**
  - Trigger behavior defined dynamically via configuration

Typical usage

- Cyclic is the most commonly used mode for:
  - Drives
  - Remote I/O
  - Real-time control devices

5.7 Disable Bit

Allows the application to disable communication with this specific adapter.

Configuration

- Assign a Boolean variable or register

Behavior

- 0 → Adapter communication enabled
- 1 → Adapter communication disabled

Usage

- Disables only the selected adapter without affecting other EtherNet/IP connections

5.8 Status Register

Provides status information related to implicit messaging for this adapter.

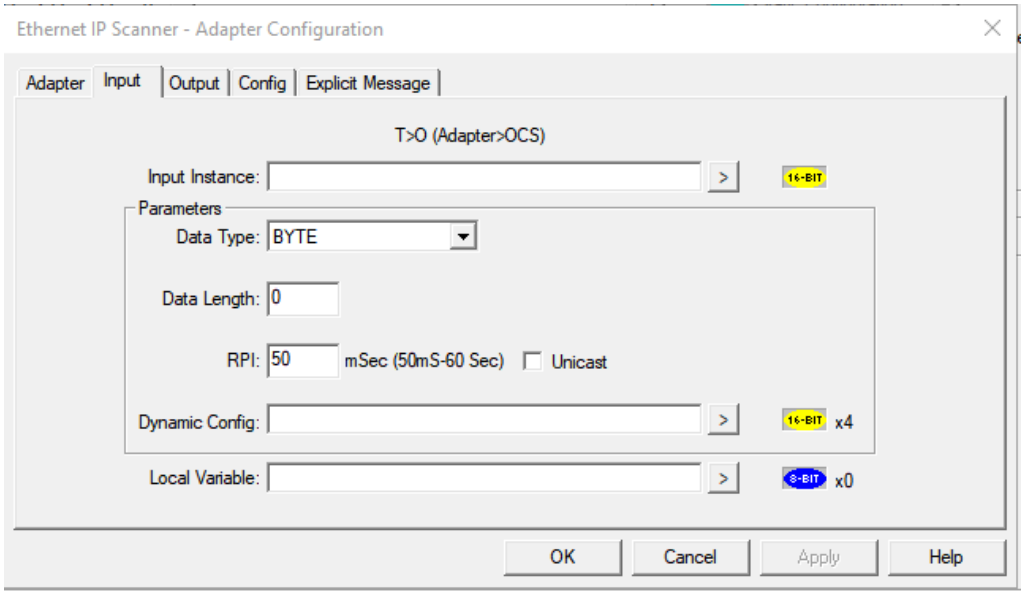
Notes

- This status register is specific to implicit messaging
- It is used for monitoring adapter communication state and errors
- Detailed status bit definitions are provided separately

Status Bits	Status Name	Description of Status
BIT 0	Implicit Connection Established	Implicit Messaging Connection is established
BIT 1	Implicit Connecting	Implicit Messaging Connecting
BIT 2	Invalid Configuration	Invalid Implicit Messaging Configuration
BIT 3	Bind Error	Socket Bind error
BIT 4	Connection Timeout	Timeout during connection establishment with adapter
BIT 5	Response Timeout	No response from Adapter
BIT 6	Transmit Buffer Error	Buffer allocation failed during transmit
BIT 7	Transmit Error	Unable to transmit data
BIT 8	Structure Fail	Structure Allocation Fail
BIT 9-15	Unused Bits	Reserved for future use
BIT 16-23	CIP Status Codes	Please refer this link "Volume 1: Common Industrial Protocol Specification, Appendix B: Status Codes"

In case of any errors, refer to troubleshooting section to find the cause of the error and suggestions to fix those.

6. EtherNet/IP Adapter Configuration – Input Tab



The Input tab defines how input (Adapter → OCS) data is received from the EtherNet/IP adapter to the OCS using implicit messaging.

This tab mainly controls:

- Input Assembly Instance
- Input data format and size
- Requested Packet Interval (RPI)
- Unicast / Multicast communication
- Optional dynamic configuration from logic or HMI

6.1 Input Instance

Specifies the **Input Assembly Instance number** used by the EtherNet/IP adapter.

Behavior based on EDS import

- **If EDS file is imported**
  - The **Input Instance field becomes hidden**
  - Assembly instance is automatically taken from the EDS definition
  - Manual configuration is not required
- **If EDS file is NOT imported**
  - The **Input Instance field remains visible**
  - User must manually enter the input assembly value
  - The value must match one of the assemblies supported by the adapter

Configuration options

- Enter a **constant value** matching the adapter's supported input assemblies
- Assign a **16-bit variable/register** to configure the Input Instance dynamically from:
  - logic
  - HMI

6.2 Parameters Section

The Parameters section defines how input data is formatted and how often it is updated.

6.2.1 Data Type

Defines the data type of the input assembly data.

Behavior

- The dropdown lists all supported data types
- Available options are shown in the configuration window

Supported data types

- BOOL
- BYTE
- INT
- UINT
- DINT
- UDINT
- REAL
- LREAL

Usage

- Must match the adapter's supported input data format
- Incorrect data type selection can cause data interpretation issues

6.2.2 Data Length

Specifies the length of the input data.

Behavior

- **If EDS file is imported**
  - Data Length is auto filled based on the EDS definition
- **If EDS file is NOT imported**
  - User can enter the value manually

Note

- Data length must match the adapter's actual input data size definition in the EDS

6.2.3 RPI (Requested Packet interval)

Defines how often input data is sent from the adapter to the OCS.

Behavior

- **If EDS file specifies a default RPI**
  - RPI is auto-filled from the EDS
- **If EDS file does NOT specify RPI**
  - Cscape sets a default value of 50 ms

Supported range

- 50 milliseconds to 60 seconds

User control

- RPI value can be changed manually
- The adapter must support the configured RPI value

Important

- Smaller RPI → faster updates, higher network load
- Larger RPI → slower updates, lower network load

6.2.4 Unicast/Multicast Checkbox

What is Multicast?

- Adapter sends data to multiple devices simultaneously
- Uses multicast IP addressing
- Efficient when multiple scanners need the same data

What is Unicast?

- Adapter sends data directly to a single scanner
- Uses point-to-point communication
- Reduces unnecessary network traffic

Unicast Checkbox Behavior

- Checked → Unicast communication
- Unchecked → Multicast communication

Important

- The Unicast option is selectable only if the adapter EDS supports both modes
- Selection should be based on network design and application needs

6.2.5 Parameters Dynamic Config

Allows the input parameters to be configured dynamically at runtime from logic or HMI.

Parameters controlled dynamically

- Data Type
- Data Length
- RPI
- Unicast / Multicast selection

Configuration

- Assign a base 16-bit register or variable
- Four consecutive 16-bit locations are used automatically

Example-
Register-Based Programs-

If Dynamic Config is assigned to %R1001:

Register	Function
%R1001	Datatype
%R1002	Data length
%R1003	RPI
%R1004	Unicast(value 1)/ Multicast(value 0)

Variable-Based, IEC & Enhanced IEC Programs-

If Dynamic Config is assigned to a variable array:

Input_Parameters_dynamic[0] → Data Type
Input_Parameters_dynamic[1] → Data Length
Input_Parameters_dynamic[2] → RPI
Input_Parameters_dynamic[3] → Unicast (1) / Multicast (0)

The array length must be **4**.

6.2.5.1 Datatype Mapping for dynamic configuration

Data Type	Value	Size
BOOL	0	1-bit
BYTE	1	8-bit
INT	2	16-bit
UINT	3	16-bit
DINT	4	32-bit
UDINT	5	32-bit
REAL	6	32-bit
LREAL	7	64-bit

UI behavior

- Once Dynamic Config is assigned:
  - Data Type
  - Data Length
  - RPI
  - Unicast checkbox

  are greyed out, as values are now controlled dynamically

6.3 Local Variable

Defines where the input data received from the adapter is stored in the OCS.

Configuration

- Assign a register or variable
- The variable type and size depend on:
  - Selected Data Type
  - Configured Data Length

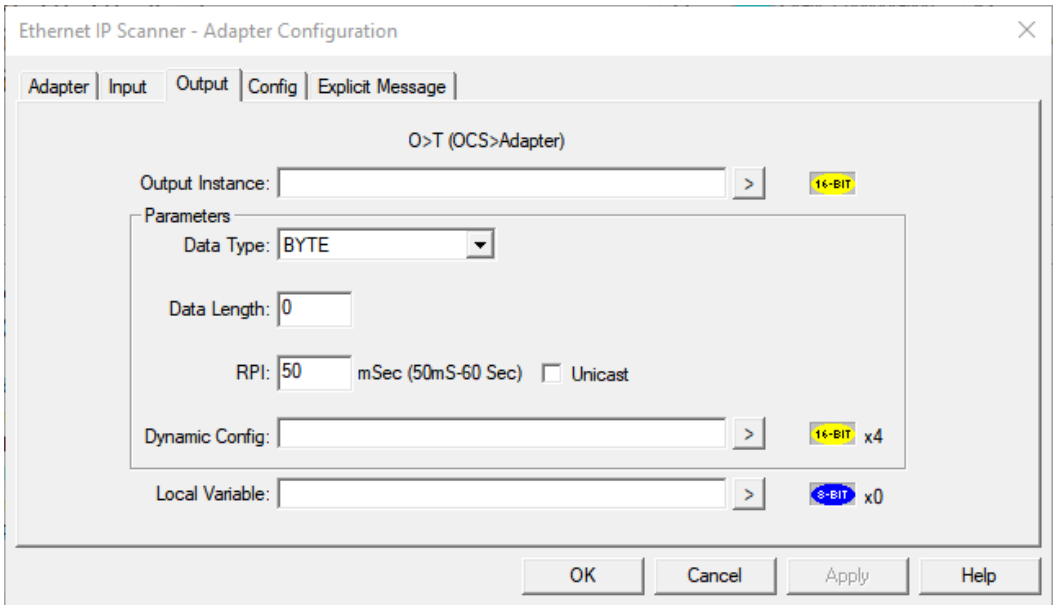
Behavior

- Input data from adapter to OCS is mapped into this variable
- Data is interpreted based on the configured parameters

Special case – Dynamic Configuration

- When Parameters are set dynamically:
  - The local variable is forced to a **16-bit register array** with a **maximum length of 256**, since the data format is determined dynamically.
- This allows logic to interpret data dynamically based on runtime configuration

7. EtherNet/IP Adapter Configuration – Output Tab



The **Output tab** is used to configure the cyclic output data that is sent **from the OCS to the Ethernet/IP adapter**.

This tab mainly controls:

- Output Assembly Instance
- Output data format and size
- Requested Packet Interval (RPI)
- Unicast / Multicast communication
- Optional dynamic configuration from logic or HMI

7.1 Output Instance

Specifies the **Output Assembly Instance number** used by the EtherNet/IP adapter.

Behavior based on EDS import

- **If EDS file is imported**
  - The **Output Instance field becomes hidden**
  - Assembly instance is automatically taken from the EDS definition
  - Manual configuration is not required
- **If EDS file is NOT imported**
  - The **Output Instance field remains visible**
  - User must manually enter the output assembly value
  - The value must match one of the assemblies supported by the adapter

Configuration options

- Enter a **constant value** matching the adapter's supported output assemblies
- Assign a **16-bit variable/register** to configure the Output Instance dynamically from:
  - logic
  - HMI

7.2 Parameters Section

The Parameters section defines how input data is formatted and how often it is updated.

7.2.1 Data Type

Defines the data type of the output assembly data.

Behavior

- The dropdown lists all supported data types
- Available options are shown in the configuration window

Supported data types

- BOOL
- BYTE
- INT
- UINT
- DINT
- UDINT
- REAL
- LREAL

Usage

- Must match the adapter's supported output data format
- Incorrect data type selection can cause data interpretation issues

7.2.2 Data Length

Specifies the length of the output data.

Behavior

- **If EDS file is imported**
  - Data Length is auto filled based on the EDS definition
- **If EDS file is NOT imported**
  - User can enter the value manually

Note

- Data length must match the adapter's actual output data size definition in the EDS

7.2.3 RPI (Requested Packet interval)

Defines how often output data is sent from the OCS to the adapter.

Behavior

- **If EDS file specifies a default RPI**
  - RPI is auto-filled from the EDS
- **If EDS file does NOT specify RPI**
  - Cscape sets a default value of 50 ms

Supported range

- 50 milliseconds to 60 seconds

User control

- RPI value can be changed manually
- The adapter must support the configured RPI value

Important

- Smaller RPI → faster updates, higher network load
- Larger RPI → slower updates, lower network load

7.2.4 Unicast/Multicast Checkbox

What is Multicast?

- Adapter sends data to multiple devices simultaneously
- Uses multicast IP addressing
- Efficient when multiple scanners need the same data

What is Unicast?

- Adapter sends data directly to a single scanner
- Uses point-to-point communication
- Reduces unnecessary network traffic

Unicast Checkbox Behavior

- Checked → Unicast communication
- Unchecked → Multicast communication

Important

- The Unicast option is selectable only if the adapter EDS supports both modes
- Selection should be based on network design and application needs

7.2.5 Parameters Dynamic Config

Allows the Output parameters to be configured dynamically at runtime from logic or HMI.

Parameters controlled dynamically

- Data Type
- Data Length
- RPI
- Unicast / Multicast selection

Configuration

- Assign a base 16-bit register or variable
- Four consecutive 16-bit locations are used automatically

Example-
Register-Based Programs-

If Dynamic Config is assigned to %R2001:

Register	Function
%R2001	Datatype
%R2002	Data length
%R2003	RPI
%R2004	Unicast(value 1)/ Multicast(value 0)

Variable-Based, IEC & Enhanced IEC Programs-

If Dynamic Config is assigned to a variable array:

Output_Parameters_dynamic[0] → Data Type
Output_Parameters_dynamic[1] → Data Length
Output_Parameters_dynamic[2] → RPI
Output_Parameters_dynamic[3] → Unicast (1) / Multicast (0)

The array length must be **4**.

7.2.5.1 Datatype Mapping for dynamic configuration

Data Type	Value	Size
BOOL	0	1-bit
BYTE	1	8-bit
INT	2	16-bit
UINT	3	16-bit
DINT	4	32-bit
UDINT	5	32-bit
REAL	6	32-bit
LREAL	7	64-bit

UI behavior

- Once Dynamic Config is assigned:
  - Data Type
  - Data Length
  - RPI
  - Unicast checkbox
    are greyed out, as values are now controlled dynamically

7.3 Local Variable

Defines the output data that needs to be sent from the OCS to the adapter.

Configuration

- Assign a register or variable
- The variable type and size depend on:
  - Selected Data Type
  - Configured Data Length

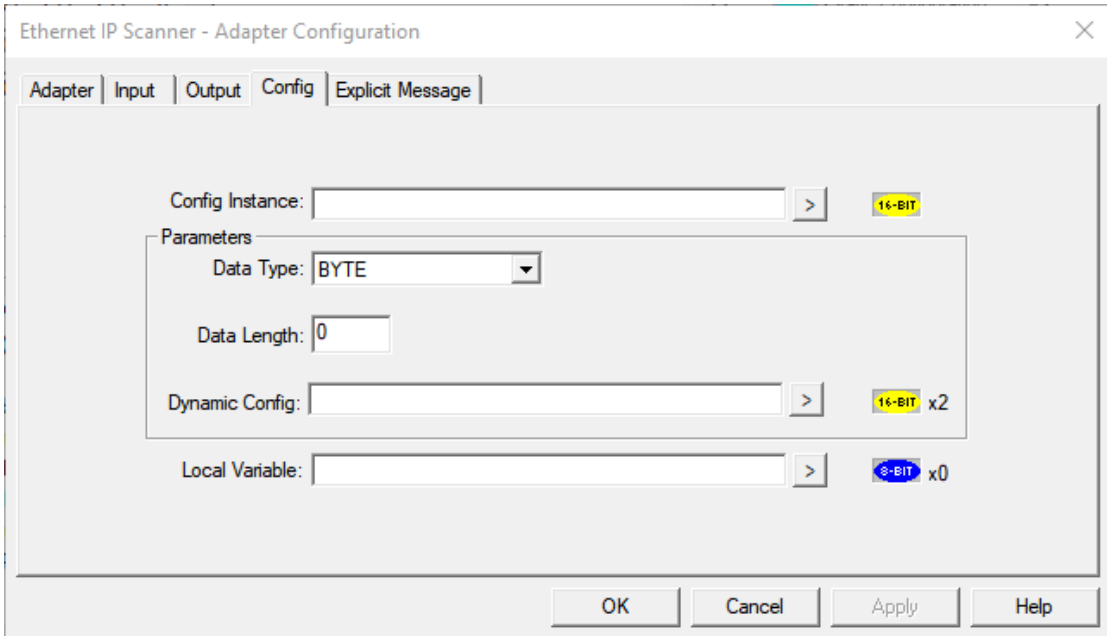
Behavior

- Output data from OCS to adapter is mapped into this variable
- Data is interpreted based on the configured parameters

Special case – Dynamic Configuration

- When Parameters are set dynamically:
  - The local variable is forced to a **16-bit register array** with a **maximum length of 256**, since the data format is determined dynamically.
- This allows the logic to interpret data dynamically based on runtime configuration

8. EtherNet/IP Adapter Configuration – Config Tab



In EtherNet/IP, the **Config Assembly** is used to send configuration data to an adapter during connection establishment.
Only adapters that **explicitly support a Config Assembly** (and expose it in their **EDS file**) can use this feature.
If the adapter does not support a Config Assembly or the EDS file does not define it, the Config tab will have no effect even if values are entered manually.

8.1 Config Instance

- The **Config Instance** field becomes **invisible** if an EDS file is imported using **Add from EDS**, *provided the adapter supports a Config Assembly*.
- If **no EDS file is imported**, this field remains **visible**, and the user can:
  - Enter the Config Assembly instance as a **constant value**, based on what the adapter supports.
  - Assign a **16-bit register/variable** to set the Config Instance **dynamically** from logic or HMI.

8.2 Parameters Section

The Parameters section defines how config data is formatted and how often it is updated.

8.2.1 Data Type

Defines the data type of the Config assembly data.
Behavior

- The dropdown lists all supported data types
- Available options are shown in the configuration window

Supported data types

- BOOL
- BYTE
- INT
- UINT

- DINT
- UDINT
- REAL
- LREAL

Usage

- Must match the adapter's supported config data format
- Incorrect data type selection can cause data interpretation issues

8.2.2 Data Length

Specifies the length of the Config data.

Behavior

- **If EDS file is imported**
  - Data Length is auto filled based on the EDS definition
- **If EDS file is NOT imported**
  - User can enter the value manually

Note

- Data length must match the adapter's actual config assembly size definition in the EDS

8.2.3 Parameters Dynamic Config

Allows the Output parameters to be configured dynamically at runtime from logic or HMI.

Parameters controlled dynamically

- Data Type
- Data Length

Configuration

- Assign a base 16-bit register or variable
- Two consecutive 16-bit locations are used automatically

Example-
Register-Based Programs-

If Dynamic Config is assigned to %R3001:

Register	Function
%R3001	Datatype
%R3002	Data length

Variable-Based, IEC & Enhanced IEC Programs-

If Dynamic Config is assigned to a variable array:

Config_Parameters_dynamic[0] → Data Type
Config_Parameters_dynamic[1] → Data Length

The array length must be **2**.

8.2.3.1 Datatype Mapping for dynamic configuration

Data Type	Value	Size
BOOL	0	1-bit
BYTE	1	8-bit
INT	2	16-bit
UINT	3	16-bit
DINT	4	32-bit
UDINT	5	32-bit
REAL	6	32-bit
LREAL	7	64-bit

8.3 Local variable

This Local Variable holds the Config data exchanged between the OCS and the adapter, based on the configured parameters.

- A **Local Variable/Register** must be assigned to store the Config Assembly data.
- The variable/register type is determined by the selected **Data Type** and **Data Length**.
- If **Dynamic Config** is enabled:
    - The local variable is forced to a **16-bit register array** with a **maximum length of 256**, since the data format is determined dynamically.

9. EtherNet/IP Adapter Configuration – Explicit Message

In EtherNet/IP, communication is divided into implicit and explicit messaging. While implicit messaging is used for continuous, cyclic I/O data exchange between the controller and adapter, explicit messaging is used for on-demand communication. Explicit messaging follows a request/response model, meaning data is exchanged only when triggered by the logic. It is typically used for reading or writing device parameters, accessing object attributes, performing diagnostics, or executing specific services supported by the adapter. Unlike implicit messaging, it is not time-critical and does not use cyclic RPI-based transmission.

Explicit Messaging is used for:

- Reading device parameters
- Writing configuration values
- Accessing object attributes
- Performing diagnostics
- Triggering device-specific services

Key Characteristics:

- Request/Response based communication
- Non-cyclic (not continuous)
- Triggered from logic
- Uses object model addressing

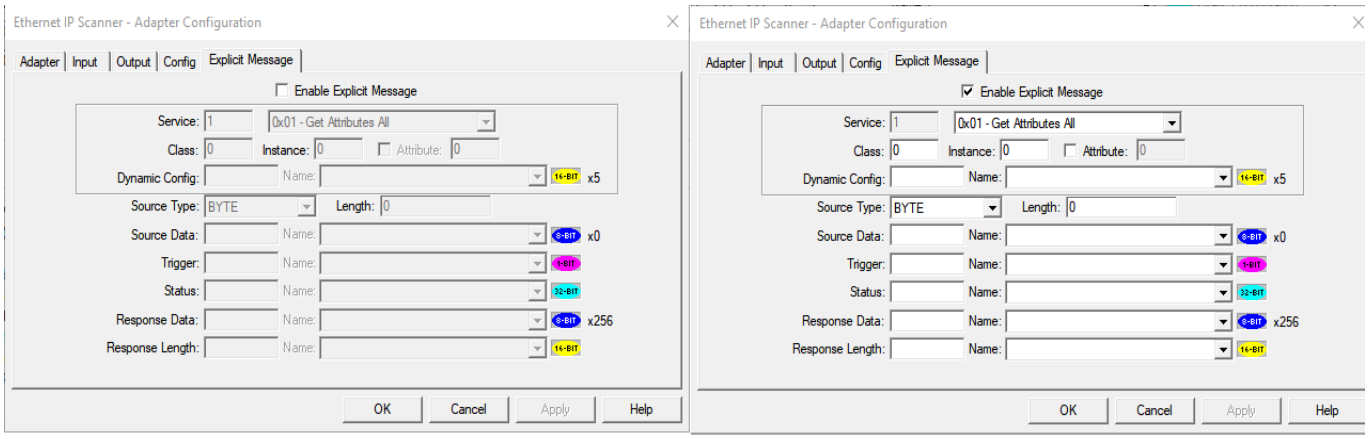
Required Addressing Information:

Explicit messaging operates using:

- Class ID
- Instance ID
- Attribute ID

- Service Code (e.g., Read, Write)

These parameters are defined in the adapter's object model and are usually available in the device documentation.



9.1 Enable Explicit Message

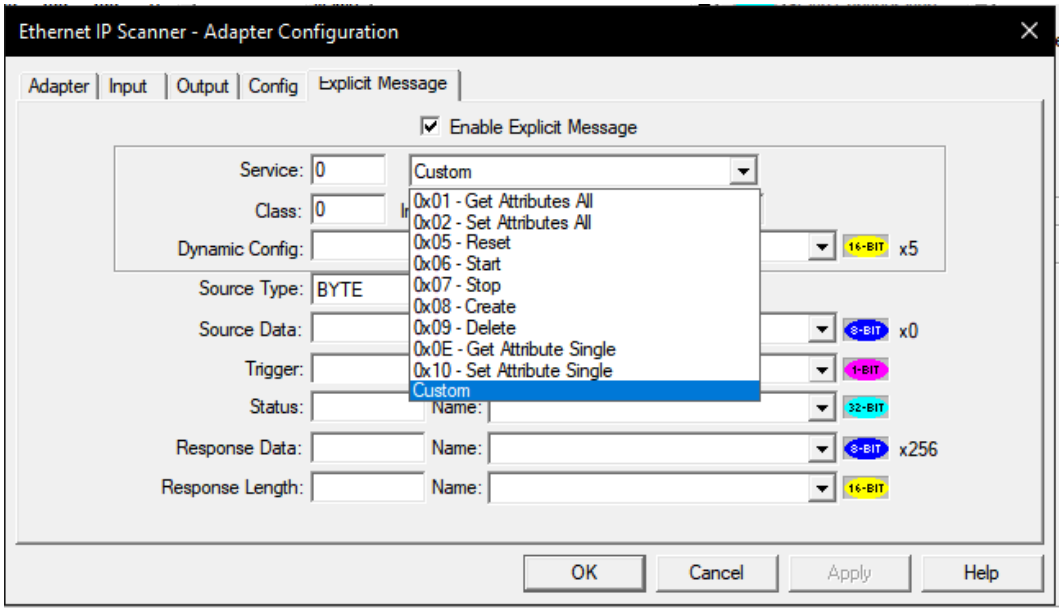
The Enable Explicit Message checkbox must be selected in order to use explicit messaging functionality.

- By default, explicit messaging is disabled.
- When the checkbox is unchecked, all explicit message configuration fields remain inactive (greyed out).
- When the checkbox is checked, explicit messaging is enabled and all related configuration fields become active, allowing the user to configure:
  - Service
  - Class
  - Instance
  - Attribute
  - Dynamic Configuration
  - Source and Response parameters

Explicit messaging works in a request/response mode. Whenever a message needs to be sent, the trigger bit must be set high. After the message is successfully transmitted, the trigger bit remains high; it must be reset to low and then set high again to send the next message.

when parameter access or device-level communication is required.
Once enabled, the user can configure the necessary addressing and data parameters based on the adapter's object model and supported services.

9.2 Service Types



The **Service** dropdown defines the type of explicit message that will be sent to the adapter.

- The dropdown lists all supported EtherNet/IP service types.
- When a predefined service is selected, the **Service code field is automatically filled** and becomes **read-only (greyed out)**.

- When **Custom** is selected:
  - The Service field becomes **editable**
  - The user must manually enter the required service code.

This allows standard services to be selected easily while still providing flexibility for vendor-specific or advanced services.

Available Service Types

Below is a simple explanation of each service type shown in the dropdown:

0x01 – Get Attributes All

- Reads all attributes of the specified object instance.
- Returns complete data for that object.
- Commonly used for diagnostics or retrieving full parameter sets.

0x02 – Set Attributes All

- Writes values to all attributes of an object instance.
- Used when multiple attributes must be configured at once.

0x05 – Reset

- Resets the device or object.
- May perform a soft reset or full reset depending on the adapter implementation.

0x06 – Start

- Starts the operation of the specified object.
- Typically used to enable or activate a function in the device.

0x07 – Stop

- Stops the operation of the specified object.
- Used to disable or halt a function.

0x08 – Create

- Creates a new instance of an object.
- Used in devices that support dynamic object creation.

0x09 – Delete

- Deletes an existing object instance.
- Used when removing dynamically created objects.

0x0E – Get Attribute Single

- Reads a single specific attribute of an object.
- Most commonly used service for reading individual parameters.

0x10 – Set Attribute Single

- Writes a value to a single specific attribute.

- Most commonly used service for updating one parameter.

Custom

- Allows manual entry of a service code.
- Used for vendor-specific or advanced services not listed in the predefined dropdown.

⚠ **Important Notes**

- The selected service must be supported by the adapter.
- Required Class, Instance, and Attribute values depend on the selected service.
- These values are typically defined in the device manual or object model documentation.

9.3 Class, Instance & Attribute Configuration

In explicit messaging, the **Class, Instance, and Attribute** fields define the exact object and parameter inside the adapter that the message will access.
These values must be entered based on:

- The adapter's supported object model
- Information provided in the device manual

Class

- Identifies the object type (for example, Identity Object, Assembly Object, etc.).
- The user must enter a valid Class ID supported by the adapter.

Instance

- Identifies a specific instance of the selected Class.
- The value must match an instance supported by the device.

Attribute

- Identifies a specific parameter within the selected Class and Instance.
- The **Attribute field remains greyed out by default**.
- It becomes **editable only when the Attribute checkbox is selected**.
- If the checkbox is:
  - **Checked** → Attribute value can be entered.
  - **Unchecked** → Attribute field remains disabled and is not used in the message.

This design ensures that the Attribute field is only used when required (for example, in *Get Attribute Single* or *Set Attribute Single* services).

9.3.1 Dynamic Configuration (Explicit Message)

The **Dynamic Config** field allows the following parameters to be set dynamically from logic or HMI:

- Service
- Class
- Instance
- Attribute enable/disable
- Attribute value

When a 16-bit register or variable is assigned to the Dynamic Config field:

- In **Register-based programs**, it consumes **five consecutive 16-bit registers**.
- In **Variable-based / IEC / Enhanced IEC programs**, it requires a variable array with a **length of 5**.

Once Dynamic Config is assigned:

- Manual entry of Service, Class, Instance, and Attribute is overridden.
- Values are taken from the assigned registers/variables.

Register-Based Example

If %R3501 is assigned in the Dynamic Config field:

- %R3501 → Service value
- %R3502 → Class
- %R3503 → Instance
- %R3504 → Attribute enable flag
  - 1 = Attribute enabled
  - 0 = Attribute disabled
- %R3505 → Attribute value

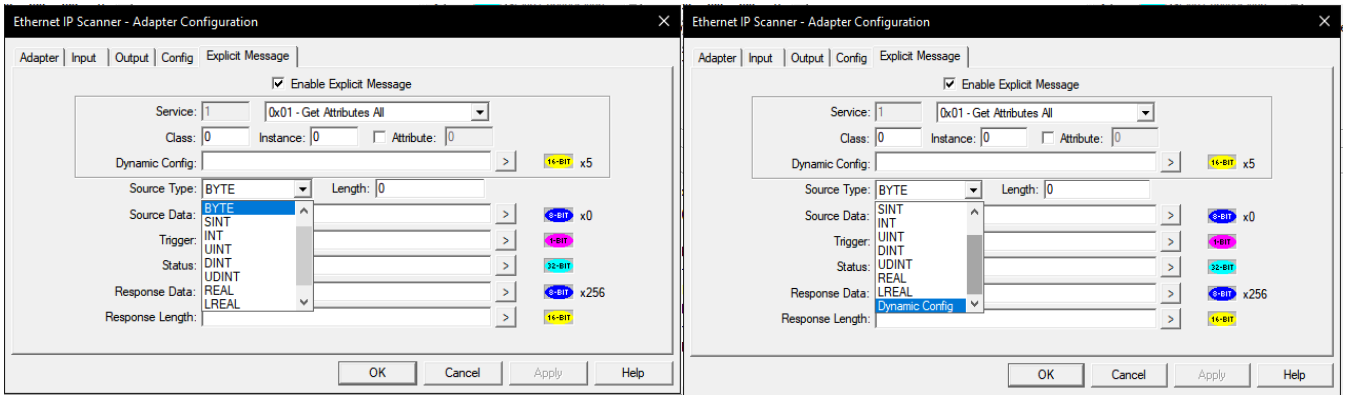
Variable-Based Example

If Explicit_Parameters_Dynamic[] variable is assigned:

- Explicit_Parameters_Dynamic[0] → Service
- Explicit_Parameters_Dynamic[1] → Class
- Explicit_Parameters_Dynamic[2] → Instance
- Explicit_Parameters_Dynamic[3] → Attribute enable flag
- Explicit_Parameters_Dynamic[4] → Attribute value

This structure allows full control of explicit message addressing directly from logic or HMI, making it flexible for advanced or runtime-configurable applications.

9.4 Source Type and Length



The **Source Type** defines the data format of the message being sent to the adapter (for example, when using *Set Attribute Single, Set Attribute All & Custom*).

The dropdown lists all supported data types:

- BOOL
- BYTE
- SINT
- INT
- UINT
- DINT
- UDINT
- REAL
- LREAL
- Dynamic Config

The user must select the **Source Data Type** based on:

- The parameter being written
- The format supported by the adapter

Static Configuration (Manual Selection)

When a standard data type (BYTE, INT, DINT, etc.) is selected:

- The **Length** field must be set according to the required data size.
- The length represents the total data size to be transmitted.

The system automatically adjusts the Length when the data type changes.

Example:

- If Source Type = BYTE and Length = 8
  → Total data size = 8 bytes

If the data type is changed:

- INT (16-bit) → Length automatically becomes 4
- DINT (32-bit) → Length automatically becomes 2

This ensures the total byte size remains consistent.

Register-Based Example-

If %R3601 is assigned:

- %R3601 → Source Data Type value
- %R3602 → Source Length

The Source Data Type must be written using the following mapping:

Datatype	Value	Size
BOOL	0	1-bit
BYTE	1	8-bit
INT	2	16-bit
UINT	3	16-bit
DINT	4	32-bit
UDINT	5	32-bit
REAL	6	32-bit
LREAL	7	64-bit

Variable-Based / IEC / Enhanced IEC Example-

If Explicit_Source_Dynamic[] is assigned:

- Explicit_Source_Dynamic[0] → Source Data Type (numeric value from table above)
- Explicit_Source_Dynamic[1] → Source Length

This allows the source data format and size to be modified at runtime through logic or HMI.

9.5 Source Data

The **Source Data** field is used to assign the register or variable that contains the actual data to be transmitted in the explicit message.

Key Points

- The **expected data size and length** displayed next to the Source Data field automatically changes based on the **Source Data Type** selected in the dropdown.
- The register or variable assigned in this field **must match the selected data type**.
  - For example:
    - If Source Type = INT, the assigned variable must be of INT type.
    - If Source Type = DINT, the assigned variable must be of DINT type.
- The size indicator helps the user verify that the correct variable type and length are configured.

9.5.1 Behavior in Static Configuration

When a fixed (non-dynamic) Source Type is selected:

- The expected data size displayed is calculated based on:
  - Selected data type
  - Configured length
- Any change in data type automatically updates the expected size shown next to the field.

This ensures proper alignment between:

- Selected data type
- Length
- Assigned variable/register

9.5.2 Behavior in Dynamic Configuration

When **Source Type is set to Dynamic Config**:

- The expected data size shown next to the Source Data field defaults to: **16-bit × 256 (maximum)**
- This is because the actual data type and length will be determined at runtime.
- In this case, the assigned register/variable must be capable of handling the maximum possible size.

This ensures sufficient memory allocation when the format is being controlled dynamically from logic or HMI.

9.6 Trigger field

The Trigger field is used to initiate the explicit messaging request. This field must be configured with a BOOLEAN register or variable. When this bit transitions to HIGH (1), the configured explicit request is sent to the adapter. The transmission of the explicit message depends entirely on this trigger signal, meaning the request will not be executed unless the trigger bit is set.

- Must be configured with a **BOOLEAN register/variable**.
- When the bit becomes **HIGH (1)**, the explicit request is sent.
- Acts as the execution control for explicit communication.
- No explicit message will be transmitted unless the trigger is activated.

9.7 Status Field

The Status field is dedicated exclusively to monitoring the status of the explicit messaging transaction. This field must be configured with a 32-bit register or variable. Each bit within this 32-bit value represents a specific execution state, success condition, or error related to the explicit request.

Detailed status bit definitions are provided separately below-

Status Bits	Status Name	Description of Status
BIT 0	Explicit Messaging Completed	Explicit Messaging is completed
BIT 1	Explicit Messaging Started	Explicit Messaging Started
BIT 2	Invalid Configuration	Invalid Explicit Messaging Configuration
BIT 3	Bind Error	Socket Bind error
BIT 4	Connection Timeout	Timeout during connection establishment with adapter
BIT 5	Response Timeout	No response from Adapter
BIT 6	Transmit Buffer Error	Buffer allocation failed during transmit
BIT 7	Transmit Error	Unable to transmit data
BIT 8	Structure Fail	Structure allocation fail
BIT 9-15	Unused Bits	Reserved for future use
BIT 16-23	CIP Status Codes	Please refer this link "Volume 1: Common Industrial Protocol Specification, Appendix B: Status Codes"

BIT 24-31	Additional Status Size	Number of additional status bytes (Additional bytes will be part of response buffer)

9.8 Response Data

The Response Data field is used to store the data received from the adapter after an explicit request is executed. This field must be assigned to a register or variable capable of handling up to 8-bit × 256 elements.
The response returned by services such as Get Attribute Single, Get Attribute All, or Custom services will be stored in this assigned register/variable area.

9.9 Response Length

The Response Length field indicates the total amount of response data received from the adapter. This field must be configured with a 16-bit register/variable. It displays the total response received in terms of data length, including the number of characters returned in the explicit response.

Status Register Expected Behaviour & Troubleshooting-

Implicit Messaging-

Expected Behaviour-

Status Bits	Status Name	Expected Behaviour
BIT 0	Implicit Connection Established	❖ Bit turns HIGH once the implicit communication is successfully established.
BIT 1	Implicit Connecting	❖ Bit turns HIGH when the implicit connection is in progress between OCS & adapter. ❖ Bit turns LOW when the implicit communication is successfully established.

Troubleshooting-

Status Bits	Status Name	Troubleshooting
BIT 1	Implicit Connecting	❖ If Bit remains HIGH continuously it means implicit communication is not getting established, please check the following- ➢ Check the IP address of both the OCS & the adapter in adapter configuration in Cscape ➢ Verify the LAN Connectivity
BIT 2	Invalid Configuration	❖ Bit turns HIGH when the downloaded implicit configuration has any invalid parameters configured. Please check the following in Cscape- ➢ Check the EDS file of adapter imported is correct or not. ➢ Check for Input Instance, Output Instance & Config Instance values in Cscape Ethernet IP Adapter

		configuration (applicable if EDS file is not imported) ➢ Check for Data length & RPI values in Ethernet IP Adapter configuration in Cscape
BIT 3	Bind Error	❖ Contact Tech Support for further assistance
BIT 4	Connection Timeout	❖ Bit turns HIGH when there is a connection timeout. Please check following- ➢ Check LAN connectivity is proper between both OCS & Adapter ➢ Check IP address of both OCS & Adapter
BIT 5	Response Timeout	❖ Bit turns HIGH when no response is received from the adapter. ➢ Check the adapter side configuration.
BIT 6	Transmit Buffer Error	❖ Contact Tech Support for further assistance
BIT 7	Transmit Error	❖ Contact Tech Support for further assistance
BIT 8	Structure Fail	❖ Contact Tech Support for further assistance
BIT 9-15	Unused Bits	❖ Reserved for future use
BIT 16-23	CIP Status Codes	❖ Please refer this link "Volume 1: Common Industrial Protocol Specification, Appendix B: Status Codes"

Explicit Messaging-

Expected Behaviour-

Status Bits	Status Name	Expected Behaviour
BIT 0	Explicit Messaging Completed	❖ Bit turns HIGH once the explicit messaging is successful after triggering the explicit request
BIT 1	Explicit Messaging Started	❖ Bit turns HIGH immediately after triggering the explicit request and while the explicit connection is in progress between OCS & adapter. ❖ Bit turns LOW when the explicit messaging is successful.

Troubleshooting-

Status Bits	Status Name	Troubleshooting
BIT 1	Explicit Messaging Started	❖ If Bit remains HIGH continuously after triggering the explicit request it means explicit communication is not getting established, please check the following- ➢ Check the IP address of both the OCS & the adapter in adapter configuration in Cscape ➢ Verify the LAN Connectivity
BIT 2	Invalid Configuration	❖ Bit turns HIGH when the downloaded explicit configuration has any invalid parameters configured. Please check the following- ➢ Check the values provided in Explicit parameters like Service, Class, Instance, Attribute in Ethernet IP adapter Explicit Message configuration in Cscape.

		➢ If dynamic configuration is downloaded then Check the values of Explicit parameters like Service, Class, Instance, Attribute provided in registers in OCS. ➢ Check valid source length is provided in case of services like Set Attribute single, Set attribute all, Custom etc.
BIT 3	Bind Error	❖ Contact Tech Support for further assistance
BIT 4	Connection Timeout	❖ Bit turns HIGH when there is a connection timeout. Please check following- ➢ Check LAN connectivity is proper between both OCS & Adapter ➢ Check IP address of both OCS & Adapter
BIT 5	Response Timeout	❖ Bit turns HIGH when no response is received from the adapter. ➢ Check the adapter side.
BIT 6	Transmit Buffer Error	❖ Contact Tech Support for further assistance
BIT 7	Transmit Error	❖ Contact Tech Support for further assistance
BIT 8	Structure Fail	❖ Contact Tech Support for further assistance
BIT 9-15	Unused Bits	❖ Reserved for future use
BIT 16-23	CIP Status Codes	❖ Please refer this link "Volume 1: Common Industrial Protocol Specification, Appendix B: Status Codes"