Allen-Bradley ControlLogix Ethernet ドライバー

目次

Allen-Bradley ControlLogix Ethernet ドライバー ヘルプへようこそ

これは、Kepware Allen-Bradley ControlLogix Ethernetドライバー のユーザードキュメントです。このヘルプセンターは、最新の機能と情報を反映して定期的に更新されます。

バージョン 1.199

概要

Allen-Bradley ControlLogix Ethernet ドライバー は Allen-Bradley ControlLogix Ethernet コントローラが HMI、SCADA、Historian、MES、ERP や多数のカスタムアプリケーションを含む OPC クライアントアプリケーションに接続するための簡単かつ信頼性の高い手段を提供します。

サポートされる Allen-Bradley コントローラ

ControlLogix® 5500 シリーズ

ControlLogix との通信は、EtherNet/IP 通信モジュール (イーサネット通信の場合) または 1761-NET-ENI モジュール (コントローラのシリアルポートを使用したイーサネット/シリアル間通信の場合) を介して確立できます。

CompactLogix™ 5300 および 5400 シリーズ

CompactLogix とのイーサネット通信には、1769-L35E などの内蔵 EtherNet/IP ポートがあるプロセッサが必要です。これがない場合、CompactLogix との通信には、コントローラのシリアルポートを使用したイーサネット/シリアル間通信用 1761-NET-ENI モジュールが必要です。

FlexLogix 5400 シリーズ

FlexLogix との通信は、1788-ENBT ドーターカード (イーサネット通信の場合) または 1761-NET-ENI モジュール (コントローラのシリアルポートを使用したイーサネット/シリアル間通信の場合) を介して確立できます。

SoftLogix 5800

このドライバーは Allen-Bradley SoftLogix 5800 シリーズのコントローラをサポートしており、SoftLogix PC でイーサネットカードを必要とします。

Data Highway Plus ゲートウェイ

このドライバーは Data Highway Plus インタフェースを備えた PLC-5 シリーズおよび SLC 500 シリーズをサポートしています。これは DH+ ゲートウェイを介して確立され、前述のいずれかの PLC、EtherNet/IP 通信モジュール、および 1756-DHRIO インタフェースモジュール (どちらも ControlLogix ラック内) を必要とします。

ControlNet ゲートウェイ

このドライバーは PLC-5C シリーズをサポートしています。これは ControlNet ゲートウェイを介して確立され、前述の PLC、EtherNet/IP 通信モジュール、および 1756-CNB/CNBR インタフェースモジュール (どちらも ControlLogix ラック内) を必要とします。

1761-NET-ENI

このドライバーは 1761-NET-ENI デバイスとの通信をサポートしています。ENI デバイスによって全二重 DF1 コントローラと Logix コントローラの両方にイーサネット/シリアル間インタフェースが提供されることで、デバイスのネットワークと通信の柔軟性が向上します。ENI デバイスと併用した場合、このドライバーは以下をサポートします。

- ControlLogix 5500 シリーズ*
- CompactLogix 5300 シリーズ*
- FlexLogix 5400 シリーズ*
- MicroLogix シリーズ
- SLC 500 固定 I/O プロセッサ
- SLC 500 モジュラー I/O シリーズ
- PLC-5 シリーズ

*これらのモデルでは 1761-NET-ENI シリーズ B 以上が必要です。

MicroLogix 1100

このドライバーは EtherNet/IP を使用した MicroLogix 1100 (チャネル 1 イーサネット) との通信をサポートしています。

設定

チャネルとデバイスの制限値

このドライバーでサポートされているチャネルの最大数は 1024 です。このドライバーでサポートされているデバイスの最大数は、1 つのチャネルにつき 1024 です。

サポートされるデバイス

デバイスファミリー	通信
ControlLogix 5550 / 5553 / 5555 / 5561 / 5562 / 5563 / 5564 / 5565 / 5571 / 5572 / 5573 / 5574 / 5575 / 5580 プロセッサ (GuardLogix モデルを含む)	1756-ENBT / ENET / EN2F / EN2T / EN2TR / EN3TR / EWEB / EN2TXT イーサネットモジュール経由 シリアルゲートウェイ経由 チャネル 0 を使用した 1761-NET-ENI シリーズ B 以上経由 (シリアル)
CompactLogix 5320 / 5323 / 5330 / 5331 / 5332 / 5335 / 5343 / 5345 / 5370 / 5380 / 5480 (GuardLogix モデルを含む)	サフィックス E が付いたプロセッサ上の内蔵 Ethernet/IP ポート* シリアルゲートウェイ経由 チャネル 0 を使用した 1761-NET-ENI シリーズ B 以上経由 (シリアル)
FlexLogix 5433 / 5434 プロセッサ	1788-ENBT イーサネットドーターカード経由 シリアルゲートウェイ経由 チャネル 0 を使用した 1761-NET-ENI シリーズ B 以上経由 (シリアル)
SoftLogix 5810 / 5830 / 5860 プロセッサ	SoftLogix Ethernet / IP メッセージングモジュール経由 シリアルゲートウェイ経由
MicroLogix 1000 / 1200 / 1500	1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由
MicroLogix 1100 / 1400	MicroLogix 1100 / 1400 チャネル 1 経由 (イーサネット) 1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由
SLC 500 固定 I/O プロセッサ	1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由
SLC 500 モジュラー I/O プロセッサ (SLC 5/01、SLC 5/02、SLC 5/03、SLC 5/04、SLC 5/05)	DH+ ゲートウェイ経由** 1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由
PLC-5 シリーズ (PLC5/250 シリーズを除く)	DH+ ゲートウェイ経由 1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由
PLC-5/20C、PLC-5/40C、PLC-5/80C	ControlNet ゲートウェイ経由

デバイスファミリー	通信
	1761-NET-ENI 経由 EtherNet/IP ゲートウェイ経由

*たとえば、1769-L35E。
**このドライバーは、DH+ をサポートするか DH+ ネットワーク (KF2 インタフェースモジュールなど) にインタフェース接続可能なすべての SLC 500 シリーズ PLC をサポートしています。

ファームウェアのバージョン

デバイスファミリー	バージョン
ControlLogix 5550 (1756-L1)	11.035 - 13.034
ControlLogix 5553 (1756-L53)	11.028
ControlLogix 5555 (1756-L55)	11.032 - 16.004
ControlLogix 5561 (1756-L61)	12.031 - 20.011
ControlLogix 5562 (1756-L62)	12.031 - 20.011
ControlLogix 5563 (1756-L63)	11.026 - 20.011
ControlLogix 5564 (1756-L64)	16.003 - 20.011
ControlLogix 5565 (1756-L65)	16.003 - 20.011
ControlLogix 5571 (1756-L71)	20.011 - 37.011
ControlLogix 5572 (1756-L72)	19.011 - 37.011
ControlLogix 5573 (1756-L73)	18.012 - 37.011
ControlLogix 5574 (1756-L74)	19.011 - 37.011
ControlLogix 5575 (1756-L75)	18.012 - 37.011
ControlLogix 5580 (1756-L8)	28.011 - 37.011
CompactLogix 5370 (1769-L1)	20.011 - 37.011
CompactLogix 5370 (1769-L2)	20.011 - 37.011
CompactLogix 5370 (1769-L3)	20.011 - 37.011
CompactLogix 5320 (1769-L20)	11.027 - 13.018
CompactLogix 5323 (1769-L23)	17.005 - 20.011
CompactLogix 5330 (1769-L30)	11.027 - 13.018
CompactLogix 5331 (1769-L31)	16.022 - 20.011
CompactLogix 5332 (1769-L32)	16.022 - 20.011
CompactLogix 5335 (1769-L35)	16.022 - 20.011
CompactLogix 5343 (1768-L43)	15.007 - 20.011
CompactLogix 5345 (1768-L45)	16.024 - 20.011
CompactLogix 5380 (5069-L3)	28.011 - 37.011
CompactLogix 5480 (5069-L4)	32.012 - 37.011
FlexLogix 5433 (1794-L33)	11.025 - 13.033
FlexLogix 5434 (1794-L34)	11.025 - 16.002
SoftLogix 5800 (1789-L60)	16.000 - 20.001
ControlLogix、CompactLogix、および FlexLogix シリア	1761-NET-ENI シリーズ B 以上またはシリアルゲートウェ

デバイスファミリー	バージョン
ル通信	イ
MicroLogix 1100 (1763-L16AWA/BWA/BBB)	1.1

通信プロトコル

通信プロトコルは、TCP/IP を使用した EtherNet/IP (イーサネットを介した CIP) です。

Logix モデルとゲートウェイモデル

Logix モデルとゲートウェイモデルは以下をサポートしています。

- 接続メッセージング
- シンボリック読み取り
- シンボリック書き込み
- シンボルインスタンス読み取り (V21 以上)
- 物理 (DMA) 読み取り (V20 以下)
- シンボルインスタンス書き込み

ENI モデル

ENI モデルは非接続メッセージングをサポートしています。

チャネルのプロパティ - 一般

このサーバーでは、複数の通信ドライバーを同時に使用することができます。サーバープロジェクトで使用される各プロトコルおよびドライバーをチャネルと呼びます。サーバープロジェクトは、同じ通信ドライバーまたは一意の通信ドライバーを使用する多数のチャネルから成ります。チャネルは、OPC リンクの基本的な構成要素として機能します。このグループは、識別属性や動作モードなどの一般的なチャネルプロパティを指定するときに使用します。

***関連項目**: プロパティ情報に関する API ドキュメントについては、/config/v1/doc/drivers エンドポイントを参照してください。*

識別

「名前」: このチャネルのユーザー定義識別情報を指定します。各サーバープロジェクトで、それぞれのチャネル名が一意でなければなりません。名前は最大 256 文字ですが、一部のクライアントアプリケーションでは OPC サーバーのタグ空間をブラウズする際の表示ウィンドウが制限されています。チャネル名は OPC ブラウザ情報の一部です。チャネルの作成にはこのプロパティが必要です。

予約済み文字の詳細については、サーバーのヘルプで「チャネル、デバイス、タグ、およびタググループに適切な名前を付ける方法」を参照してください。

「説明」: このチャネルに関するユーザー定義情報を指定します。

「説明」などのこれらのプロパティの多くには、システムタグが関連付けられています。

「ドライバー」: このチャネル用のプロトコル/ドライバーを指定します。チャネル作成時に選択されたデバイスドライバーを指定します。チャネルのプロパティではこの設定を変更することはできません。チャネルの作成にはこのプロパティが必要です。

注記: サーバーがオンラインで常時稼働している場合、これらのプロパティをいつでも変更できます。これには、クライアントがデータをサーバーに登録できないようにチャネル名を変更することも含まれます。チャネル名を変更する前にクライアントがサーバーからアイテムをすでに取得している場合、それらのアイテムは影響を受けません。チャネル名が変更された後で、クライアントアプリケーションがそのアイテムを解放し、古いチャネル名を使用して再び取得しようとしても、そのアイテムは取得されません。大規模なクライアントアプリケーションを開発した場合は、プロパティを変更しないようにしてください。オペレータがプロパティを変更したりサーバーの機能にアクセスしたりすることを防ぐため、適切なユーザー役割を使用し、権限を正しく管理する必要があります。

診断

「診断取り込み」: このオプションが有効な場合、チャネルの診断情報が OPC アプリケーションに取り込まれます。サーバーの診断機能は最小限のオーバーヘッド処理を必要とするので、必要なときにだけ利用し、必要がないときには無効にしておくことをお勧めします。デフォルトでは無効になっています。

注記: ドライバーまたはオペレーティングシステムが診断をサポートしていない場合、このプロパティは使用できません。

ヒント: 診断情報を ASCII で表示できます。

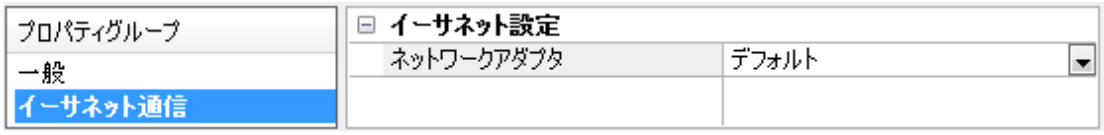
詳細については、サーバーのヘルプで「通信診断」と「統計タグ」を参照してください。

タグ数

「静的タグ」: デバイスレベルまたはチャネルレベルで定義される静的タグの数を指定します。この情報は、トラブルシューティングと負荷分散を行う場合に役立ちます。

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

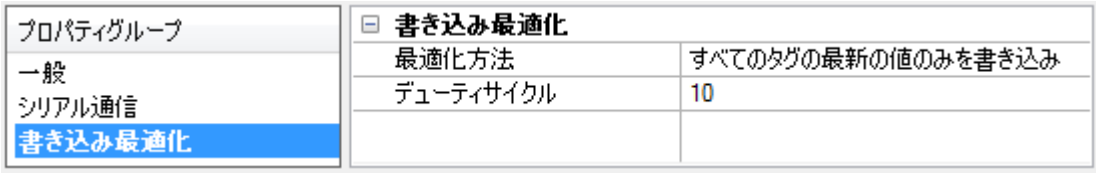


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or Default is selected, the operating system selects the default adapter.

Channel Properties — Write Optimizations

The server must ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties to meet specific needs or improve application responsiveness.



Write Optimizations

Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest

value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

Duty Cycle: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.
Note: It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

Channel Properties — Advanced

This group is used to specify advanced channel properties. Not all drivers support all properties; so the Advanced group does not appear for those devices.

Non-Normalized Float Handling: A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. Descriptions of the options are as follows:

- **Replace with Zero**: This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified**: This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

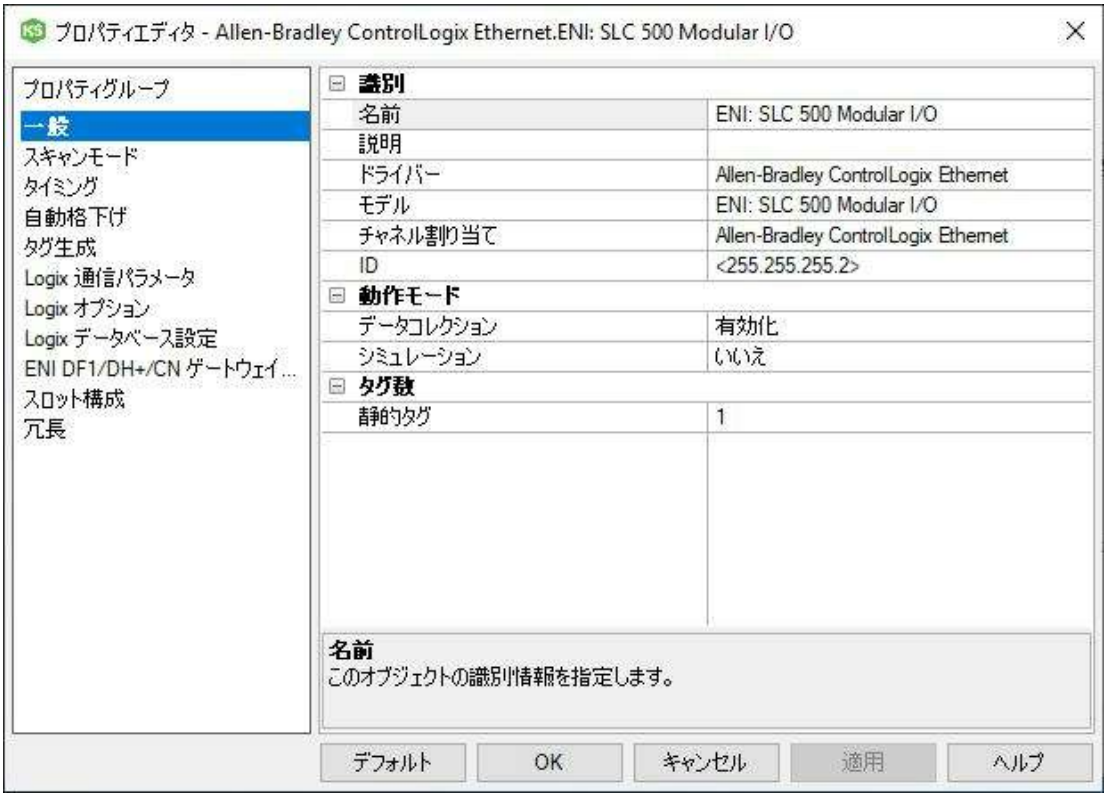
Note: This property is disabled if the driver does not support floating-point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

For more information on the floating-point values, refer to "How To ... Work with Non-Normalized Floating-Point Values" in the server help.

Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

Note: This property is not available for all drivers, models, and dependent settings.

デバイスのプロパティ - 一般



識別

「名前」: このデバイスのユーザー定義の識別情報。

「説明」: このデバイスに関するユーザー定義の情報。

「チャネル割り当て」: このデバイスが現在属しているチャネルのユーザー定義の名前。

「ドライバー」: このデバイスに設定されているプロトコルドライバー。

「モデル」: このデバイスのバージョン。

ヒント: GuardLogix には、ControlLogix または CompactLogix を使用します (**「サポートされるデバイス」**を参照)。

「ID」: デバイスの一意のネットワークアドレスを、通常は <IP またはホスト名>,1, <ルーティングパス>,<スロット> というフォーマットで入力します。

アドレス指定の規則はモデルとルーティングによって異なります。詳細については、**参考資料**でモデル固有のアドレス指定のトピックを参照してください。

Operating Mode

Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

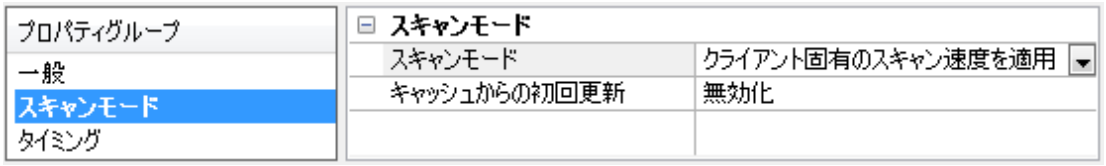
注記:

1. クライアントが切断して再接続するまで、更新は適用されません。
2. システムタグ (_Simulated) は読み取り専用であり、ランタイム保護のため、書き込みは禁止されています。このシステムタグを使用することで、このプロパティをクライアントからモニターできます。
3. シミュレーションモードでは、アイテムのメモリマップはクライアントの更新レート (OPC クライアントではグループ更新レート、ネイティブおよび DDE インタフェースではスキャン速度) に基づきます。つまり、異なる更新レートで同じアイテムを参照する 2 つのクライアントは異なるデータを返します。
4. デバイスをシミュレートしたときに、クライアントで更新が 1 秒未満で表示されない場合があります。

シミュレーションモードはテストとシミュレーションのみを目的としています。本番環境では決して使用しないでください。

Device Properties — Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.



Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the OPC client's

responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.

- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

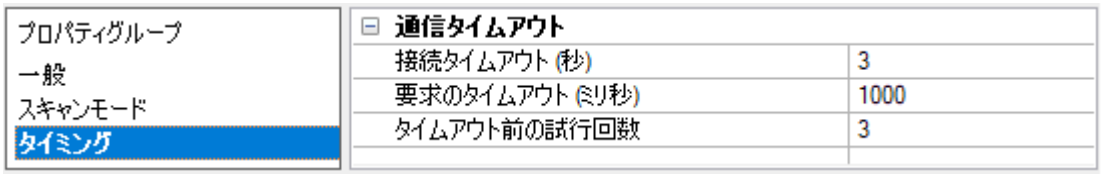
Initial Updates from Cache: When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

Tag Counts

Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

Device Properties — Timing

The device Timing properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Timing properties are specific to each configured device.



Communications Timeouts

Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds, but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.

Note: Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

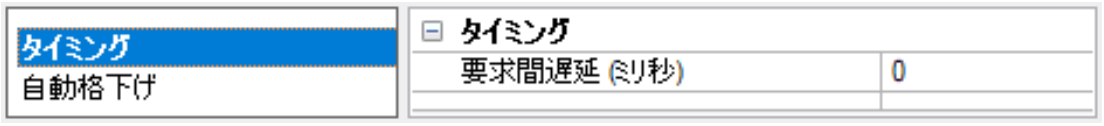
Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most

serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

Attempts Before Timeout: Specify how many times the driver issues a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number of attempts configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

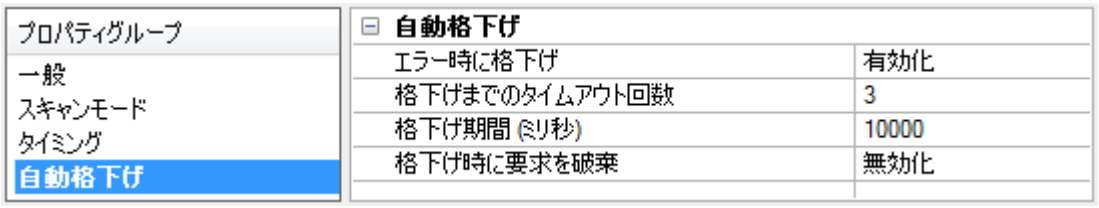
Timing

Inter-Request Delay: Specify how long the driver waits before sending the next request to the target device after receiving the response to the previous request. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround times and in cases where network load is a concern. Configuring a delay for a device affects communications with all other devices on the channel. It is recommended that users separate any device that requires an inter-request delay to a separate channel if possible. Other communications properties (such as communication serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may limit the maximum value due to a function of their particular design. The default is 0, which indicates no delay between requests with the target device.
Note: Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.



Device Properties — Auto-Demotion

The Auto-Demotion properties can temporarily place a device off-scan in the event that a device is not responding. By placing a non-responsive device offline for a specific time period, the driver can continue to optimize its communications with other devices on the same channel. After the time period has been reached, the driver re-attempts to communicate with the non-responsive device. If the device is responsive, the device is placed on-scan; otherwise, it restarts its off-scan time period.



Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

Demotion Period: Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for

another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.

デバイスのプロパティ - タグ生成

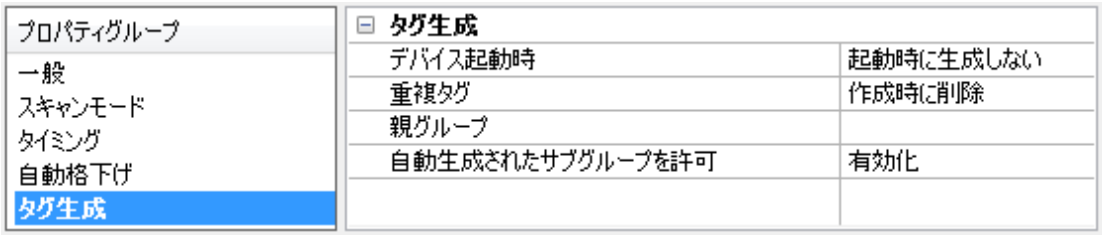
自動タグデータベース生成機能によって、アプリケーションの設定がプラグアンドプレイ操作になります。デバイス固有のデータに対応するタグのリストを自動的に構築するよう通信ドライバーを設定できます。これらの自動生成されたタグ (サポートしているドライバーの特性によって異なる) をクライアントからブラウズできます。

一部のデバイスやドライバーは自動タグデータベース生成のフル機能をサポートしていません。また、すべてのデバイスやドライバーが同じデータ型をサポートするわけではありません。詳細については、データ型の説明を参照するか、各ドライバーがサポートするデータ型のリストを参照してください。

ターゲットデバイスが独自のローカルタグデータベースをサポートしている場合、ドライバーはそのデバイスのタグ情報を読み取って、そのデータを使用してサーバー内にタグを生成します。デバイスが名前付きのタグをネイティブにサポートしていない場合、ドライバーはそのドライバー固有の情報に基づいてタグのリストを作成します。この 2 つの条件の例は次のとおりです。

1. データ取得システムが独自のローカルタグデータベースをサポートしている場合、通信ドライバーはデバイスで見つかったタグ名を使用してサーバーのタグを構築します。
2. イーサネット I/O システムが独自の使用可能な I/O モジュールタイプの検出をサポートしている場合、通信ドライバーはイーサネット I/O ラックにプラグイン接続している I/O モジュールのタイプに基づいてサーバー内にタグを自動的に生成します。

注記: 自動タグデータベース生成の動作モードを詳細に設定できます。詳細については、以下のプロパティの説明を参照してください。



「プロパティ変更時」: デバイスが、特定のプロパティが変更された際の自動タグ生成をサポートする場合、**「プロパティ変更時」**オプションが表示されます。これはデフォルトで**「はい」**に設定されていますが、**「いいえ」**に設定してタグ生成を実行する時期を制御できます。この場合、タグ生成を実行するには**「タグを作成」**操作を手動で呼び出す必要があります。

「デバイス起動時」: OPC タグを自動的に生成するタイミングを指定します。オプションの説明は次のとおりです。

- **「起動時に生成しない」**: このオプションを選択した場合、ドライバーは OPC タグをサーバーのタグ空間に追加しません。これはデフォルトの設定です。
- **「起動時に常に生成」**: このオプションを選択した場合、ドライバーはデバイスのタグ情報を評価します。さらに、サーバーが起動するたびに、サーバーのタグ空間にタグを追加します。
- **「最初の起動時に生成」**: このオプションを選択した場合、そのプロジェクトが初めて実行されたときに、ドライバーがデバイスのタグ情報を評価します。さらに、必要に応じて OPC タグをサーバーのタグ空間に追加します。

注記: OPC タグを自動生成するオプションを選択した場合、サーバーのタグ空間に追加されたタグをプロジェクトとともに保存する必要があります。ユーザーは**「ツール」**|**「オプション」**メニューから、自動保存するようプロジェクトを設定できます。

「重複タグ」: 自動タグデータベース生成が有効になっている場合、サーバーが以前に追加したタグや、通信ドライバーが最初に作成した後で追加または修正されたタグを、サーバーがどのように処理するかを設定する必要があります。この設定では、自動生成されてプロジェクト内に現在存在する OPC タグをサーバーがどのように処理するかを制御します。これによって、自動生成されたタグがサーバーに累積することもなくなります。

たとえば、**「起動時に常に生成」**に設定されているサーバーのラックで I/O モジュールを変更した場合、通信ドライバーが新しい I/O モジュールを検出するたびに新しいタグがサーバーに追加されます。古いタグが削除されなかった場合、多数の未使用タグがサーバーのタグ空間内に累積することがあります。以下のオプションがあります。

- **「作成時に削除」**: このオプションを選択した場合、新しいタグが追加される前に、以前にタグ空間に追加されたタグがすべて削除されます。これはデフォルトの設定です。
- **「必要に応じて上書き」**: このオプションを選択した場合、サーバーは通信ドライバーが新しいタグに置き換えているタグだけ除去します。上書きされていないタグはすべてサーバーのタグ空間に残ります。
- **「上書きしない」**: このオプションを選択した場合、サーバーは以前に生成されたタグやサーバーにすでに存在するタグを除去しません。通信ドライバーは完全に新しいタグだけを追加できます。
- **「上書きしない、エラーを記録」**: このオプションには上記のオプションと同じ効果がありますが、タグの上書きが発生した場合にはサーバーのイベントログにエラーメッセージも書き込まれます。

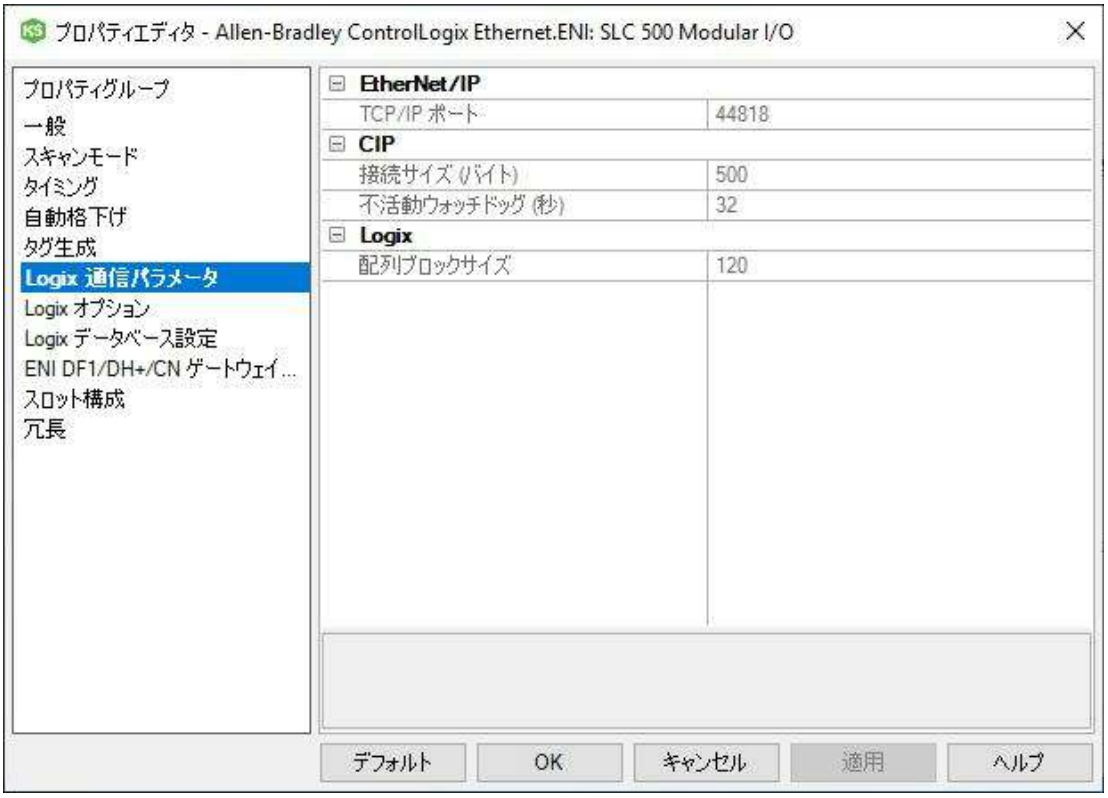
注記: OPC タグの除去は、通信ドライバーによって自動生成されたタグ、および生成されたタグと同じ名前を使用して追加されたタグに影響します。ドライバーによって自動生成されるタグと一致する可能性がある名前を使用してサーバーにタグを追加しないでください。

「親グループ」: このプロパティでは、自動生成されたタグに使用するグループを指定することで、自動生成されたタグと、手動で入力したタグを区別します。グループの名前は最大 256 文字です。この親グループは、自動生成されたすべてのタグが追加されるルートブランチとなります。

「自動生成されたサブグループを許可」: このプロパティでは、自動生成されたタグ用のサブグループをサーバーが自動的に作成するかどうかを制御します。これはデフォルトの設定です。無効になっている場合、サーバーはグループを作成しないで、デバイスのタグをフラットリスト内に生成します。サーバープロジェクトで、生成されたタグには名前としてアドレスの値が付きます。たとえば、生成プロセス中はタグ名は維持されません。
注記: サーバーがタグを生成しているときに、タグに既存のタグと同じ名前が割り当てられた場合、タグ名が重複しないようにするため、番号が自動的に 1 つ増分します。たとえば、生成プロセスによってすでに存在する "AI22" という名前のタグが作成された場合、代わりに "AI23" としてタグが作成されます。

「作成」: 自動生成 OPC タグの作成を開始します。**「タグを作成」**が有効な場合、デバイスの構成が修正されると、ドライバーはタグ変更の可能性についてデバイスを再評価します。システムタグからアクセスできるため、クライアントアプリケーションはタグデータベース作成を開始できます。
注記: 構成がプロジェクトをオフラインで編集する場合、**「タグを作成」**は無効になります。

Device Properties — Logix Communications Parameters



EtherNet/IP

TCP/IP Port: Specifies the TCP/IP port number that the device is configured to use. The default is 44818.

CIP

Connection Size: Specify the number of bytes available on the CIP connection for data requests and responses. The valid range is 500 to 4000 bytes. The default is 500 bytes.

Note: Only the ControlLogix 5500 and CompactLogix 5300 device models support this feature. To support connection sizes greater than 500 bytes, the device must support Firmware version 20 or later controllers and Ethernet bridge EN3x, EN2x, or EN5.x. Older Ethernet modules like ENBT and ENET do not support this feature. Devices that do not meet the necessary requirements automatically fall back to the default setting of 500 bytes, although the requested size is re-attempted after communications failure.

The Connection Size value may also be requested through the System tag "_CIPConnectionSizeRequested." For more information, refer to ***Internal Tags****.*

Inactivity Watchdog: Specify the amount of time, in seconds, a connection remains idle (without read/write transactions) before being closed by the controller. The larger the value, the more time it takes for connection resources to be released by the controller and vice versa. The default is 32 seconds.

Note: If an error about the CIP connection timing out while uploading a project occurs frequently, increase the Inactivity Watchdog value. Otherwise, the default value is suggested.

Logix

Array Block Size: This property specifies the maximum number of array elements to read in a single transaction. The value is adjustable and ranges from 30 to 3840 elements. The default is 120 elements.

Tip: For Boolean arrays, a single element is considered a 32-element bit array. Setting the block size to 30 elements translates to 960-bit elements, whereas 3840 elements translate to 122880 bit elements.

Device Properties — Logix Options

Protocol Options

Protocol Mode: Select how Logix tag data is read from the controller: Logical Non-Blocking, Logical Blocking, and Symbolic. The default is Logical Non-Blocking. This option should only be changed by advanced users looking to increase client / server tag update performance.

For more information, refer to ***Choosing a Protocol Mode****.*

Note: Logical Non-Blocking and Logical Blocking are not available to Serial Gateway models.

Synchronize After Online Edits: When enabled, the driver synchronizes its project image with that of the controller project when an online project edit (or project download from RSLogix/Studio5000) is detected. This option prevents unnecessary errors from occurring during a project change. It is only available when the selected Protocol Mode is Logical Non-Blocking or Logical Blocking. The default is Yes.

Synchronize After Offline Edits: When enabled, the driver synchronizes its own project image with that of the controller project when an offline project edit (or project download from RSLogix/Studio5000) is detected. This option prevents unnecessary errors from occurring during a project change. It is only available when the selected protocol is Logical Non-Blocking or Logical Blocking. The default is Yes.

Failure to synchronize with project changes can lead to reading from and writing to the wrong Native Tag address.

Terminate String Data at LEN: When enabled, the driver automatically reads the LEN member of the STRING structure whenever the DATA member is read. The DATA string is terminated at the first null character encountered, the character whose position equals the value of LEN, or the maximum string length of DATA (whichever occurs first). When disabled, the driver bypasses the LEN member read and terminates the DATA string at either the first null character encountered or the maximum string length of DATA

(whichever occurs first). Therefore, if LEN is reduced by an external source without modification to DATA, the driver does not terminate DATA according to this reduced length. The default is Enable.

Project Options

Default Data Type: Select the data type assigned to a client/server tag when the default type is selected during tag addition, modification, or import. The default is Default.

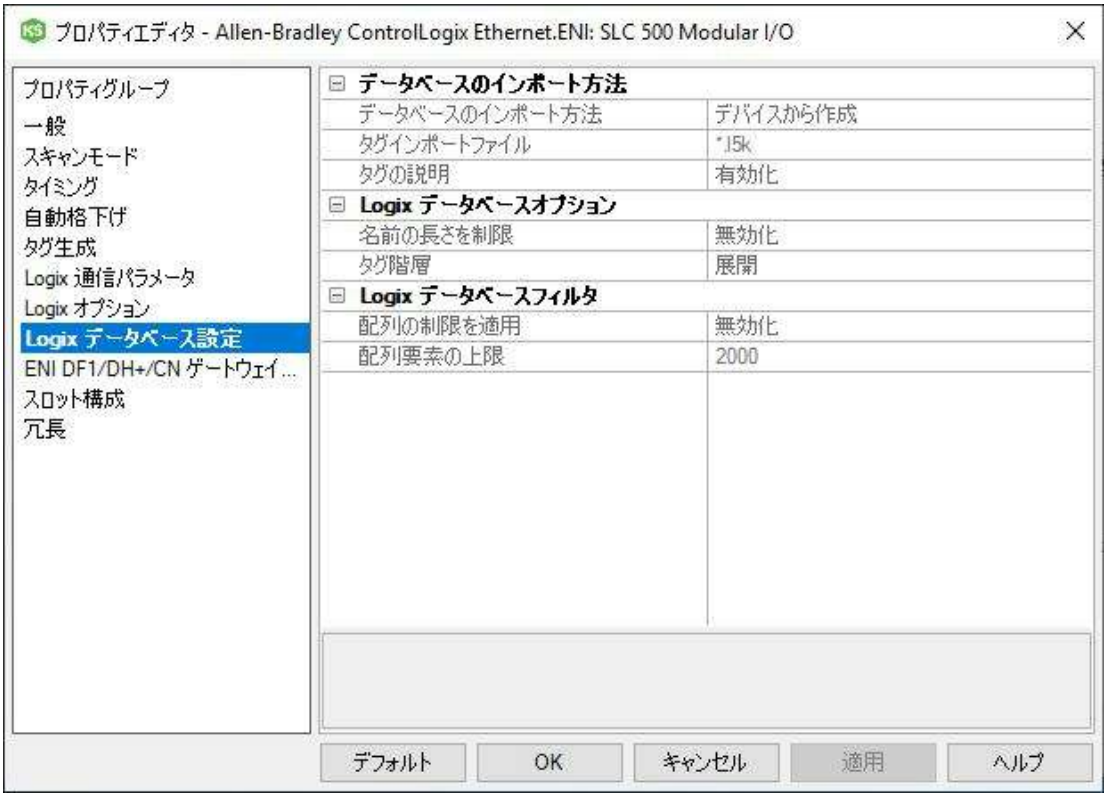
For more information, refer to ***Default Data Type Conditions***.

Performance Statistics: The Allen-Bradley ControlLogix Ethernet ドライバー has the ability to gather communication statistics to help determine the driver's performance. The driver tracks the number and types of client/server tag updates. On restart of the server application, the results are displayed in the server's Event Log. The default is Disable.

Note: Once a project configuration is designed for optimal performance, it is recommended that users disable Performance Statistics. Because the statistics are written to the Event Log on shutdown, the server must be re-launched to view the results.

See Also: ***Detecting a Change in the Controller Project***

Device Properties — Logix Database Settings



Database Import Method

Database Import Method: Select how the tag database should be populated:

- **Create from Device**: retrieves tags directly from the controller over the same Ethernet connection that is used for data access, which is fast and imports most tags, but requires access to the controller and does not import descriptions. Tags that are not imported include Add-On Instruction

(AOI) InOut properties.
Note: This feature is not available to Serial Gateway models.

- **Create from Import File**: retrieves tags from a selected RSLogix L5K/L5X file. Controller access is not necessary, descriptions are imported, and users can work offline; however, this option is slower and does not import all the tags in the controller. Tags that are not imported include:
  - I/O tags
  - Add-On Instruction (AOI) InOut properties
  - AOI properties that alias other properties
  - Equipment Phase properties that alias properties from another Equipment Phase or Program
  - Program properties that alias properties from another Program or Equipment Phase
  - Timer / Counter CTL bits

Tag Import File: Click the browse (**...**) button to locate and select the L5K/L5X file from which tags are to be imported. This file is used when Automatic Tag Database Generation is instructed to create the tag database. All tags, including Global and Program, are imported and expanded according to their respective data types.

Tag Descriptions: Choose **Enable** to import tag descriptions for non-structure, non-array tags. If necessary, a description is given to tags with long names stating the original tag name.

Tag Generation Using the API

The config API properties for CLX offline ATG are:

```
"controllogix_ethernet.DEVICE_DATABASE_IMPORT_METHOD": 1, "controllogix_
ethernet.DEVICE_TAG_IMPORT_FILE": "myFile.l5x", "controllogix_ethernet.DEVICE_
DISPLAY_DESCRIPTIONS": true,
```

where the enumeration for import method is:
0 for Create from Device;
and
1 for Create from File

Logix Database Options

Limit Name Length: Set to Enable to constrain the tag and group names to 31 characters. The default is **Disable**.

1. Before OPC server version 4.70, tag and group name lengths were restricted to 31 characters. The current length restriction of 256 characters can fit Logix 40-character Logix Tag names.
2. If an older server version was used to import tags via L5K/L5X import, inspect the Event Log or scan the server project to see if any tags were truncated due to the character limit. If so, Enable this property to preserve the server tag names. OPC client tag references are not affected. If disabled, longer tag names are created and clients referencing the clipped tag must be changed to reference the new tag name.

3. If an older OPC server version was used to import tags via L5K/L5X import and no tags were truncated due to the 31-character limit, leave this option disabled.
4. If tags were imported via L5K/L5X with server version 4.70 or above, leave this option disabled.
   See Also: ***Controller-to-Server Name Conversions***

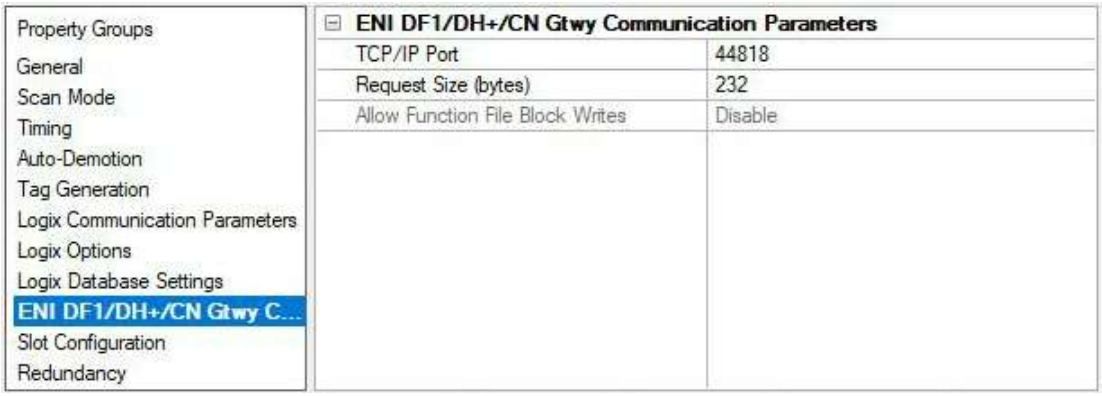
Tag Hierarchy: This property specifies the tree organization of the tag hierarchy. When **Condensed**, the server tags created by automatic tag generation follow a group/tag hierarchy consistent with the tag's address. Groups are created for every segment preceding the period. When **Expanded**, the server tags created by automatic tag generation follow a group/tag hierarchy consistent with the tag hierarchy in RSLogix 5000. Groups are created for every segment preceding the period and to represent logical groupings. To use this functionality, enable **Allow Sub Groups** in **Tag Generation** properties.
For more information on the groups created, refer to ***Tag Hierarchy*** *and* ***Controller-to-Server Name Conversions****.*

Logix Database Filtering

Impose Array Limit: Select Enable to constrain the number of array elements. Tags in the controller can be declared with very large array dimensions. By default, arrays are completely expanded during the tag generation process, which becomes time consuming for large arrays. By imposing a limit, only a specified number of elements from each dimension are generated. Limits only takes effect when the array dimension size exceeds the limit. The default is **Disable**.

Array Count Upper Limit: Specify the array count limit. The default is 2000.

Device Properties — ENI DF1/DH+/CN Gateway Communications Parameters



TCP/IP Port: Specify the port number that the remote device is configured to use (such as 1756-ENBT). The default is 44818.

Request Size: Select the number of bytes that may be requested from a device at one time to refine performance. Options are 32, 64, 128, or 232. The default is 232 bytes.

Allow Function File Block Writes: Function files are structure-based files (much like PD and MG data files) and are unique to the MicroLogix 1100, 1200, and 1500. For applicable function files, data can be written to the device in a single operation. By default, when data is written to a function file sub element (field within the function file structure), a write operation occurs immediately for that tag. For such files as the RTC file, whose sub elements include hour (HR), minute (MIN), and second (SEC), individual writes are not always

acceptable. With such sub elements relying solely on time, values must be written in one operation to avoid time elapsing between sub elements writes. For this reason, there is the option to block write these sub elements. The default is disabled.

*For more information, refer to **Block Writes** and **Function Files**.*

Block Writes

Block writing involves writing to the device the values of every read/write sub element in the function file in a single write operation. It is not necessary to write to every sub element before performing a block write. Sub elements that are not affected (written to) have their current value written back to them. For example, if the current (last read) date and time is 1/1/2001, 12:00.00, DOW = 3 and the hour is changed to 1 o'clock, the values written to the device are 1/1/2001, 1:00.00, DOW = 3. For more information, refer to the instructions below.

1. To start, locate **ENI DF1/DH+/CN Gateway Communications Parameters** in **Device Properties**.
2. Enable **Allow Function Files Block Writes** to notify the driver to utilize block writes on function files that support block writes.
3. Clicking **OK** or **Apply**.
4. Write the desired value to the sub element tag in question. The sub element tag immediately takes on the value written to it.

   Note: After a sub element is written to at least once in block write mode, the tag's value does not originate from the controller, but instead from the driver's write cache. After the block write is done, all sub element tag values originate from the controller.
5. Once the entire desired sub elements are written, perform the block write that sends these values to the controller. To instantiate a block write, reference tag address *RTC:<element>._SET*. Setting this tag's value to 'true' causes a block write to occur based on the current (last read) sub elements and the sub elements affected (written to). Immediately after setting the tag to 'true', it is automatically reset to "false." This is the default state and performs no action.

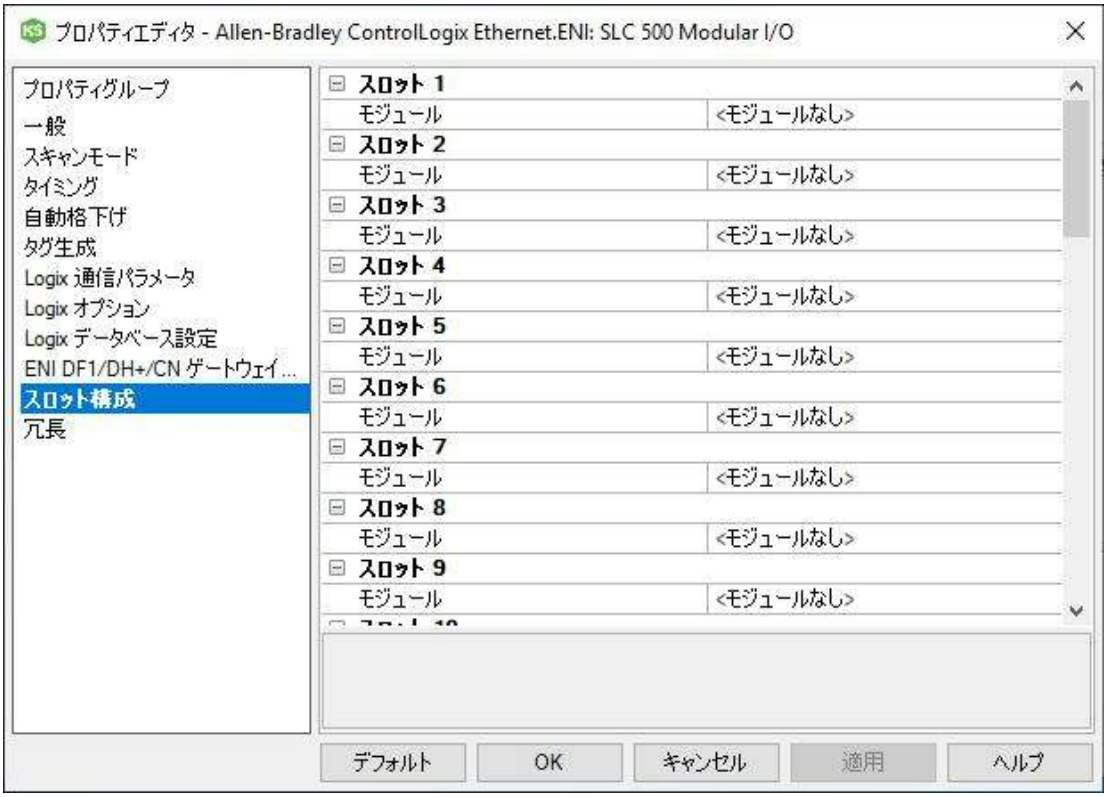
Applicable Function Files / Sub Elements

RTC	
Year	YR
Month	MON
Day	DAY
Day of Week	DOW
Hour	HR
Minute	MIN
Second	SEC

***See Also**: Function File Listing*

Device Properties — SLC 500 Slot Configuration

For I/O to be accessed, SLC5/01/02/03/04/05 models (modular I/O racks) must be configured for use with the Allen-Bradley ControlLogix Ethernet ドライバー. Up to 30 slots can be configured per device.



Slot *n*: the physical slot being configured. Use the plus icon to expand the properties.

Module: Set the type of module in the slot from the options available in the drop-down list.
*To configure slots through the **Configuration API Service, see this example**.*

Input Words: If required by the module selected, enter the maximum number of Input Words for this module.

Output Words: If required by the module selected, enter the maximum number of Output Words for this module.

To use slot configuration, follow the instructions below:

1. Select the slot to be configured by clicking on the row in the module list box.
2. To select a module, click on it from the available modules drop-down list.
3. Configure the Input Words and Output Words if necessary.
4. To remove a slot / module, select **No Module** from the available modules drop-down list.
5. When complete, click **OK**.

Tips:

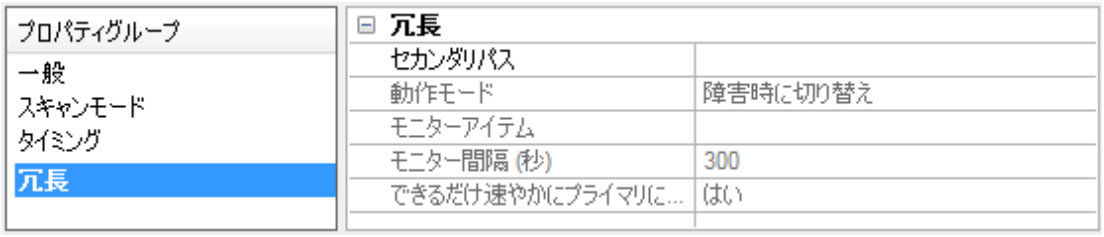
- Use the 0000-Generic Module to configure I/O that is not contained in the list of Available Modules.
- The module selections available are the same as those in the Allen Bradley APS software.

Note: It is common to have open slots in the rack that do not contain a physical module. To correctly access data for the various slots that do contain a module, the preceding module(s) must have the correct number of words mapped. For example, if only interested in the I/O in slot 3, but slots 1 and 2 contain I/O modules, the correct modules must be selected for slots 1, 2, and 3 from this slot configuration group.

0000-Generic Module
Use the Generic Module to map Input and Output words for modules that are not represented in the list of available modules. To correctly use the Generic Module, users must know the number of Input and Output words required for each module.

Consult Allen-Bradley I/O user manual documentation to confirm Input and Output requirements and be aware that requirements may be different based on Class 1 or Class 3 operation.
*For information on the number of input and output words available for each I/O module, refer to **Modular I/O Selection Guide**.*

Device Properties — Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.
*Consult the website, a sales representative, or the **user manual** for more information.*

SLC 500 Modular I/O Selection Guide

The following table lists the number of input and output words available for each I/O module in the Slot Configuration list.

Module ID	Module Type	Input Words	Output Words
0	1746-I*8 Any 8 pt Discrete Input Module	1	0
1	1746-I*16 Any 16 pt Discrete Input Module	1	0
2	1746-I*32 Any 32 pt Discrete Input Module	2	0
3	1746-O*8 Any 8 pt Discrete Output Module	0	1
4	1746-O*16 Any 16 pt Discrete Output Module	0	1
5	1746-O*32 Any 32 pt Discrete Output Module	0	2
6	1746-IA4 4 Input 100 / 120 VAC	1	0
7	1746-IA8 8 Input 100 / 120 VAC	1	0
8	1746-IA16 16 Input 100/120 VAC	1	0
9	1746-IB8 8 Input (Sink) 24 VDC	1	0
10	1746-IB16 16 Input (Sink) 24 VDC	1	0
11	1746-IB32 32 Input (Sink) 24 VDC	2	0
12	1746-IG16 16 Input [TTL] (Source) 5 VDC	1	0
13	1746-IM4 4 Input 200 / 240 VAC	1	0
14	1746-IM8 8 Input 200 / 240 VAC	1	0
15	1746-IM16 16 Input 200/240 VAC	1	0
16	1746-IN16 16 Input 24 VAC / VDC	1	0
17	1746-ITB16 16 Input [Fast] (Sink) 24 VDC	1	0
18	1746-ITV16 16 Input [Fast] (Source) 24 VDC	1	0
19	1746-IV8 8 Input (Source) 24 VDC	1	0
20	1746-IV16 16 Input (Source) 24 VDC	1	0
21	1746-IV32 32 Input (Source) 24 VDC	2	0
22	1746-OA8 8 Output (TRIAC) 100 / 240 VAC	0	1
23	1746-OA16 16 Output (TRIAC) 100 / 240 VAC	0	1
24	1746-OB8 8 Output [Trans] (Source) 10 / 50 VDC	0	1
25	1746-OB16 16 Output [Trans] (Source) 10 / 50 VDC	0	1
26	1746-OB32 32 Output [Trans] (Source) 10/50 VDC	0	2
27	1746-OBP16 16 Output [Trans 1 Amp] (SRC) 24 VDC	0	1
28	1746-OV8 8 Output [Trans] (Sink) 10/50 VDC	0	1
29	1746-OV16 16 Output [Trans] (Sink) 10/50 VDC	0	1
30	1746-OV32 32 Output [Trans] (Sink) 10/50 VDC	0	2
31	1746-OW4 4 Output [Relay] VAC/VDC	0	1
32	1746-OW8 8 Output [Relay] VAC/VDC	0	1
33	1746-OW16 16 Output [Relay] VAC/VDC	0	1
34	1746-OX8 8 Output [Isolated Relay] VAC/VDC	0	1

Module ID	Module Type	Input Words	Output Words
35	1746-OVP16 16 Output [Trans 1 Amp] (Sink) 24 VDC3	0	1
36	1746-IO4 2 In 100 / 120 VAC 2 Out [Rly] VAC / VDC3	1	1
37	1746-IO8 4 In 100 / 120 VAC 4 Out [Rly] VAC / VDC4	1	1
38	1746-IO12 6 In 100 / 120 VAC 6 Out [Rly] VAC / VDC	1	1
39	1746-NI4 4 Ch Analog Input	4	0
40	1746-NIO4I Analog Comb 2 in & 2 Current Out	2	2
41	1746-NIO4V Analog Comb 2 in & 2 Voltage Out	2	2
42	1746-NO4I 4 Ch Analog Current Output	0	4
43	1746-NO4V 4 Ch Analog Voltage Output	0	4
44	1746-NT4 4 Ch Thermocouple Input Module	8	8
45	1746-NR4 4 Ch Rtd / Resistance Input Module	8	8
46	1746-HSCE High-Speed Counter/Encoder	8	1
47	1746-HS Single Axis Motion Controller	4	4
48	1746-OG16 16 Output [TLL] (SINK) 5 VDC	0	1
49	1746-BAS Basic Module 500 5/01 Configuration	8	8
50	1746-BAS Basic Module 5/02 Configuration	8	8
51	1747-DCM Direct Communication Module (1/4 Rack)	2	2
52	1747-DCM Direct Communication Module (1/2 Rack)	4	4
53	1747-DCM Direct Communication Module (3/4 Rack)	6	6
54	1747-DCM Direct Communication Module (Full Rack)	8	8
55	1747-SN Remote I/O Scanner	32	32
56	1747-DSN Distributed I/O Scanner 7 Blocks	8	8
57	1747-DSN Distributed I/O Scanner 30 Blocks	32	32
58	1747-KE Interface Module, Series A	1	0
59	1747-KE Interface Module, Series B	8	8
60	1746-NI8 8 Ch Analog Input, Class 1	8	8
61	1746-NI8 8 Ch Analog Input, Class 3	16	12
62	0000-Generic Module	-	-
63	1746-IC16 16 Input (Sink) 48 VDC	1	0
64	1746-IH16 16 Input [Trans] (Sink) 125 VDC	1	0
65	1746-OAP12 12 Output [Triac] 120/240 VDC	0	1
66	1746-OB6EI 6 Output [Trans] (Source) 24 VDC	0	1
67	1746-OB16E 16 Output [Trans] (Source) Protected	0	1
68	1746-OB32E 32 Output [Trans] (Source) 10 / 50 VDC	0	2
69	1746-OBP8 8 Output [Trans 2 amp] (Source) 24 VDC	0	1
70	1746-IO12DC 6 Input 12 VDC, 6 Output [Rly]	1	1
71	1746-INI4I Analog 4 Ch. Isol. Current Input	8	8
72	1746-INI4VI Analog 4 Ch. Isol. Volt/Current Input	8	8

Module ID	Module Type	Input Words	Output Words
73	1746-INO4I Analog 4 Ch. Isol. Current Output	8	8
74	1746-INO4VI Analog 4 Ch. Isol. Volt/Current Output	8	8
75	1746-INT4 4 Ch. Isolated Thermocouple Input	8	8
76	1746-NT8 Analog 8 Ch Thermocouple Input	8	8
77	1746-HSRV Motion Control Module	12	8
78	1746-HSTP1 Stepper Controller Module	8	8
79	1747-MNET MNET Network Comm Module	0	0
80	1746-QS Synchronized Axes Module	32	32
81	1747-QV Open Loop Velocity Control	8	8
82	1747-RCIF Robot Control Interface Module	32	32
83	1747-SCNR ControlNet SLC Scanner	32	32
84	1747-SDN DeviceNet Scanner Module	32	32
85	1394-SJT GMC Turbo System	32	32
86	1203-SM1 SCANport Comm Module - Basic	8	8
87	1203-SM1 SCANport Comm Module - Enhanced	32	32
88	AMCI-1561 AMCI Series 1561 Resolver Module	8	8

Configuration API — Allen-Bradley ControlLogix Ethernet Example

For a list of channel and device definitions and enumerations, access the following endpoints with the REST client or refer to the appendices.

Channel Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-Bradley%20ControlLogix%20Ethernet/channels
```

Device Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-Bradley%20ControlLogix%20Ethernet/devices
```

Create Allen-Bradley ControlLogix Ethernet Channel

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels
```

Body:

```
{
   "common.ALLTYPES_NAME": "MyChannel",
   "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet"
}
```

See Also: ***Appendix*** *for a list of channel properties.*

Create Allen-Bradley ControlLogix Ethernet Device

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/MyChannel/devices
```

Body:

```
{
   "common.ALLTYPES_NAME": "MyDevice",
   "servermain.DEVICE_ID_STRING": "<IP Address>,0,1",
   "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet",
   "servermain.DEVICE_MODEL": <model enumeration>
}
```

where <IP Address> is the device's IP address.

Note: The format of the value for servermain.DEVICE_ID_STRING may vary depending on the model enumeration specified for servermain.DEVICE_MODEL. The device ID string format shown above is for the ControlLogix 5500 model.

See Also: ***Device Model Enumerations*** *and* ***Device Properties.***

Create Allen-Bradley ControlLogix Ethernet Tags

Endpoint (POST):

```
https://<hostname_or_
ip>:<port>/config/v1/project/channels/MyChannel/devices/MyDevice/tags
```

Body:

```
[
{
   "common.ALLTYPES_NAME": "MyTag1",
   "servermain.TAG_ADDRESS": "My_Tag_Address"
}
{
   "common.ALLTYPES_NAME": "MyTag2",
   "servermain.TAG_ADDRESS": "My_Tag_Address"
}
]
```

***See Also**: Appendix for a list of tag properties.*

See server and driver-specific help for more information on configuring tags and tag groups using the Configuration API.

列挙

デバイスモデルなどの一部のプロパティには、列挙にマッピングされる値が含まれています。列挙とその値の有効なリストを確認するには、'content=property_definitions' を使用してデバイスのエンドポイントをクエリーするか、ドキュメント定義のエンドポイントをクエリーします。

たとえば、"MyChannel" というチャネルの下にある "MyDevice" というデバイスのプロパティ定義を表示するには、GET リクエストを以下の URL に送信します。

```
https://<ホスト名または IP>:<ポート
>/config/v1/project/channels/MyChannel/devices/MyDevice/?content=property_definitions
```

プロパティ定義は、チャネル、タグなど、その他のオブジェクトでも使用することができます。

または、ドライバーのチャネルとデバイスのプロパティ定義を Configuration API の設定内で有効にすると、以下の URL でそれらのプロパティ定義を表示することができます。

```
https://<ホスト名または IP>:<ポート>/config/v1/doc/drivers/<ドライバー名>/Channels
```

```
https://<ホスト名または IP>:<ポート>/config/v1/doc/drivers/<ドライバー名>/Devices
```

データ型列挙の例

ドキュメントエンドポイントでタグのデータ型のクエリーを実行すると、以下の列挙が表示されます。

```
{
   "Default": -1,
   "String": 0,
   "Boolean": 1,
   "Char": 2,
   "Byte": 3,
   "Short": 4,
   "Word": 5,
   "Long": 6,
   "DWord": 7,
   "Float": 8,
```

```
   "Double": 9,
   "BCD": 10,
   "LBCD": 11,
   "Date": 12,
   "LLong": 13,
   "QWord": 14,
   "String Array": 20,
   "Boolean Array": 21,
   "Char Array": 22,
   "Byte Array": 23,
   "Short Array": 24,
   "Word Array": 25,
   "Long Array": 26,
   "DWord Array": 27,
   "Float Array": 28,
   "Double Array": 29,
   "BCD Array": 30,
   "LBCD Array": 31,
   "Date Array": 32,
   "LLong Array": 33,
   " QWord Array": 34
}
```

注記: サポートされるデータ型は、プロトコルとドライバーによって異なります。

Device Model Enumerations

The Device Model property has values mapped to the following enumerations. The below table is for reference only; the information at the device endpoint is the complete and current source of information:

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-
Bradley%20ControlLogix%20Ethernet/Channels
```

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-
Bradley%20ControlLogix%20Ethernet/Devices
```

Enumeration	Device Model
0	ControlLogix 5500
1	CompactLogix 5300
2	FlexLogix 5400
3	SoftLogix 5800
4	DH+ Gateway: PLC-5
5	DH+ Gateway: SLC 5/04
6	ControlNet Gateway: PLC-5C
7	EIP Gateway: MicroLogix
8	EIP Gateway: SLC Fixed
9	EIP Gateway: SLC Modular
10	EIP Gateway: PLC-5
11	Serial Gateway: ControlLogix
12	Serial Gateway: CompactLogix

Enumeration	Device Model
13	Serial Gateway: CompactLogix
14	Serial Gateway: FlexLogix
15	Serial Gateway: SoftLogix
16	ENI: ControlLogix 5500
17	ENI: FlexLogix 5400
18	ENI: MicroLogix
19	ENI: SLC 500 Fixed I/O
20	ENI: SLC 500 Modular I/O
21	ENI: PLC-5
22	MicroLogix 1100
23	MicroLogix 1400

Configuration API — Slot Configuration

Given this Example Slot Configuration:

Slot 1: 1746-OV32 32 Output [Trans] (Sink) 10/50 V_{DC}
Slot 2: 1746-I*16 Any 16 pt Discrete Input Module
Slot 3: <No Module>
Slot 4: 0000-Generic Module [1 Input Word, 2 Output Words]
Slot 5: 1746-NT8 Analog 8 Ch Thermocouple Input
Slot 6 – 30: <No Module>

Configure the Slots with the API:

```
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION": {
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_MODULE": [
30,
1,
255,
62,
76,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255
],

"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_INPUT_WORDS": [
0,
1,
0,
1,
8,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0
],

"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_OUTPUT_WORDS": [
2,
0,
0,
2,
8,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0
]
```

自動タグデータベース生成

デバイス固有のデータに対応するサーバー内のサーバータグのリストを自動生成するよう Allen-Bradley ControlLogix Ethernetドライバーを設定できます。自動生成されたタグは Logix デバイスで定義されている Logix タグに基づいており、OPC クライアントからブラウズできます。Logix タグはアトミックまたは構造体です。構造体タグと配列タグではインポートされるタグの数 (したがってサーバーで使用可能なタグの数) が急速に増えることがあります。

注記: ENI/DH+、ControlNet ゲートウェイ、および MicroLogix モデルでは自動タグデータベース生成はサポートされていません。これは ENI ControlLogix、CompactLogix、および FlexLogix モデルでのみサポートされています。

アトミックタグ -> **1 対 1** -> サーバータグ
構造体タグ -> **1 対多** -> サーバータグ
配列 -> **1 対多** -> サーバータグ

データベース作成の設定の詳細については、サーバーのヘルプファイルを参照してください。

注記: RSLogix5000 プログラミング環境で監視されたコントローラタグでは、「External Access」プロパティを「Read Only」または「Read/Write」に設定してタグを読み取る必要があります。自動的に生成されたタグでは、デフォルトで「External Access」が「None」に設定されている可能性があります。コントローラタグを読み取るには、RSLogix の Add-On Instruction パラメータで、必要に応じて External Access を再構成します。製造メーカーのドキュメントを参照してください。

Tag Hierarchy

The server tags created by automatic tag generation can follow one of two hierarchies: Expanded or Condensed. To use this functionality, enable Allow Sub Groups in device properties.

Expanded Mode

When Expanded, the server tags created by automatic tag generation follow a group / tag hierarchy consistent with the tag hierarchy in RSLogix 5000. Groups are created for every segment preceding the period as when Condensed, but are also created in logical groupings. Groups created include the following:

- Global (controller) scope
- Program scope
- Structures and substructures
- Arrays

Note: Groups are not created for .bit addresses.

The root-level groups (or subgroup levels of the group specified in Parent Group) are "Prgm_<program name>" and "Global". Each program in the controller has its own "Prgm_<program name>" group. The driver recognizes this as the first group level.

Basic Global Tags (or non-structure, non-array tags) are placed under the Global group; basic Program tags are placed under their respective program group. Each structure and array tag is provided in its own subgroup of the parent group. By organizing the data in this fashion, the server's tag view mimics RSLogix5000.

The name of the structure / array subgroup also provides a description of the structure / array. For instance, an array tag1[1,6] defined in the controller would have a subgroup name "tag1_x_y"; x signifies dimension 1 exists, and y signifies dimension 2 exists. The tags within an array subgroup are all the elements of that array (unless explicitly limited). The tags within a structure subgroup are the structure members themselves. If a structure contains an array, an array subgroup of the structure group is created as well.

With a complex project, the tag hierarchy can require a number of group levels. The maximum number of group levels created by automatic tag generation is seven. This does not include the group specified in "Add generated tags to the following group". When more than seven levels are required, the tags are placed in the seventh group (causing the hierarchy to plateau).

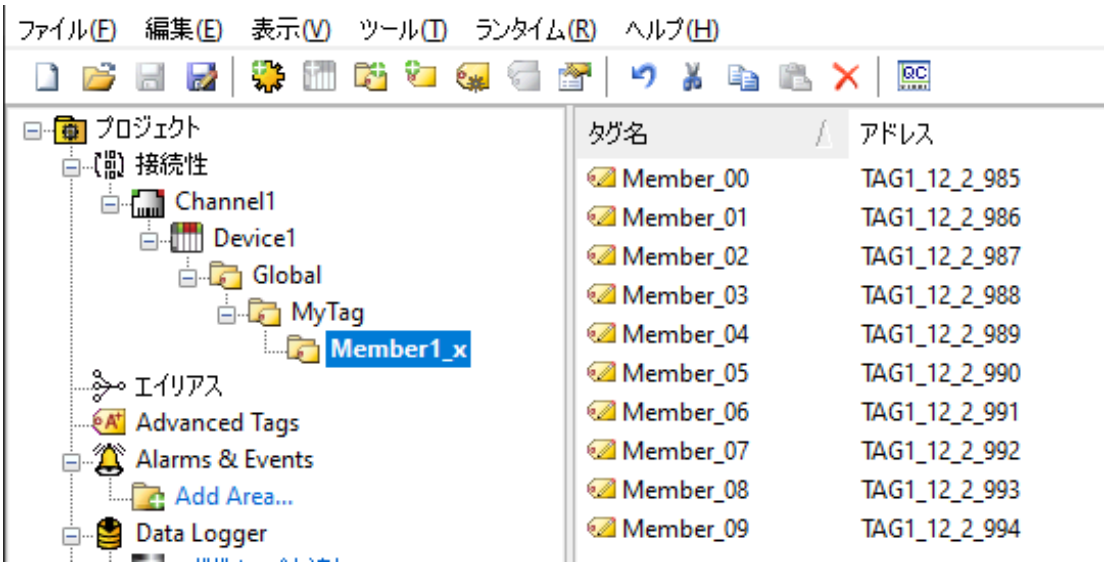
Array Tags
A group is created for each array that contains the array's elements. Group names have the notation: <*array name*>_*x_y_z* where:

x_y_z = 3 dimensional array
x_y = 2 dimensional array
x = 1 dimensional array

Array tags have the notation: <*tag element*>_*XXXXX_YYYYY_ZZZZZ*. For example, element tag1[12,2,987] would have the tag name "tag1_12_2_987".

Simple Example

Name	Value	Force Mask	Style	Data Type
MyTag	{...}	{...}		MyDataType
MyTag.Member1	{...}	{...}	Decimal	DINT[10]
MyTag.Member1[0]	0		Decimal	DINT
MyTag.Member1[1]	0		Decimal	DINT
MyTag.Member1[2]	0		Decimal	DINT
MyTag.Member1[3]	0		Decimal	DINT



Complex Example
A Logix tag is defined with the address "Local:1:O.Slot[9].Data". This would be represented in the groups "Global" - "Local_1_O" - "Slot_x" - "Slot_09". Within the last group would be the tag "Data".

The static reference to "Data" would be "Channel1.Device1.Global.Local_1_O.Slot_x.Slot_09.Data". The dynamic reference to "Data" would be "Channel1.Device1.Local:1:O.Slot[9].Data".

Condensed Mode

In Condensed Mode, the server tags created by automatic tag generation follow a group/tag hierarchy consistent with the tag's address. Groups are created for every segment preceding the period. Groups created include the following:

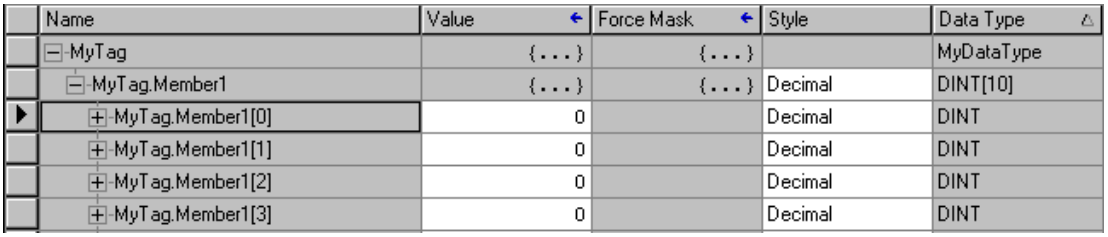
- Program scope
- Structures and substructures

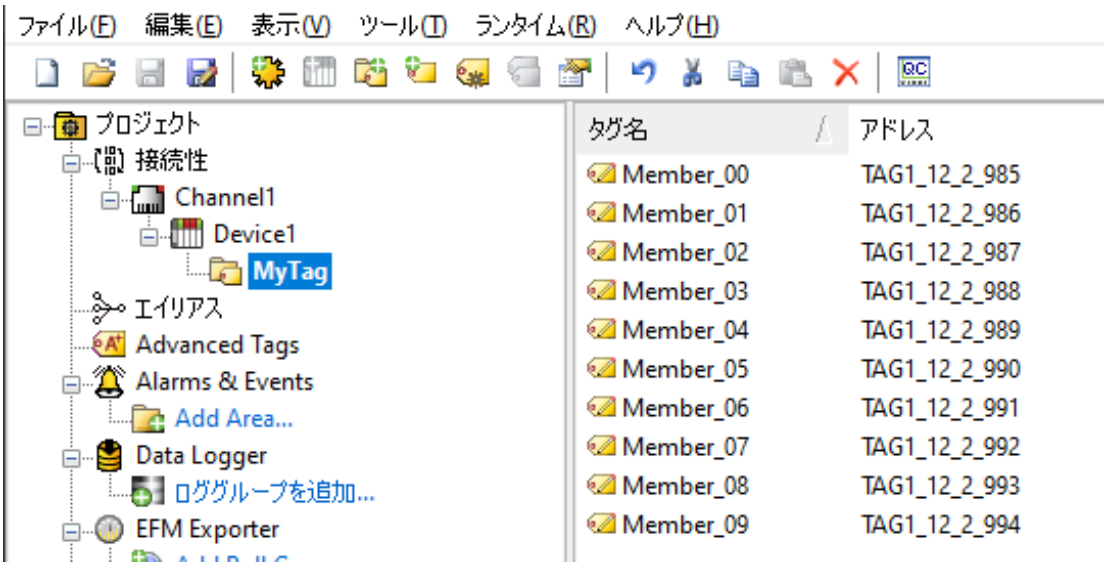
Note: Groups are not created for arrays or .bit addresses.

With a complex project, it is easy to see how the tag hierarchy can require a number of group levels. The maximum number of group levels created by automatic tag generation is seven. This does not include the group specified in "Add generated tags to the following group". When more than seven levels are required, the tags are placed in the seventh group (causing the hierarchy to plateau).

Note: Tag or structure member names leading off with an underscore is converted to "U_". This is required because the server does not support leading underscores. *For more information, refer to* ***Controller-to-Server Name Conversion***.

Simple Example





Complex Example

Logix tag is defined with address "Local:1:O.Slot[9].Data". This would be represented in the groups "Local:1:O" -> "Slot[9]". Within the last group would be the tag "Data".

The static reference to "Data" would be "Channel1.Device1.Local:1:O.Slot[9].Data". The dynamic reference would be "Channel1.Device1.Local:1:O.Slot[9].Data".

Note: I/O module tags cannot be directly imported in Offline mode. Since aliases can be imported, it is recommended that they be created for I/O module tags of interest in RSLogix5000.

Controller-to-Server Name Conversions

Leading Underscores

Leading underscores "_" in tag or program names are replaced with "U_". This is required because the server does not accept tag or group names beginning with an underscore.

Long Names (OPC Server Version 4.64 and below)

Under older OPC server versions, the Allen-Bradley ControlLogix Ethernet ドライバー was limited to 31 characters in group and tag names. Therefore, if a controller program or tag name exceeded 31 characters, it had to be clipped. OPC server Version 4.70 and above has a 256-character limit, so the rules do not apply. Names are clipped as follows:

Non-Array

1. Determine a 5-digit unique ID for this tag.
2. Given a tag name: ThisIsALongTagNameAndProbablyExceeds31
3. Clip tag at 31: ThisIsALongTagNameAndProbablyEx
4. Room is made for the unique ID: ThisIsALongTagNameAndProba#####
5. Insert this ID: ThisIsALongTagNameAndProba00000

Array

1. Determine a 5-digit unique ID for this array.
2. Given an array tag name: ThisIsALongTagNameAndProbablyExceeds31_23_45_8
3. Clip tag at 31 while holding on to the element values: ThisIsALongTagNameAndPr_23_45_8
4. Room is made for the unique ID: ThisIsALongTagName#####_23_45_8
5. Insert this ID: ThisIsALongTagName00001_23_45_8

Long program names are clipped in the same manner as long non-array tag names. For every tag or program name that is clipped, the unique ID is incremented. Array tag names (elements) of a clipped array name have the same unique ID. This provides for 100000 unique tag/program names.

Note: If Limit Name Length is enabled, the rules apply even if the 256-character names are supported. For more information, refer to **Logix Database Settings**.

Preparing for Automatic Tag Database Generation

For information on using Automatic Tag Database Generation, follow the instructions below.

Online

It is recommended that all communications to the Logix CPU of interest cease during the database creation process.

In RSLogix5000

Set the project OFFLINE.

In the server

1. Review the device properties of the device for which tags will be generated.
2. Within **Logix Database Settings** and select **Create from Device** for **Database Import Method**.
3. In **Logix Database Options**, make any desired changes and click **OK**.
4. In **Logix Database Filtering**, make any desired changes and click **OK**.
5. Select **Tag Generation** and, under Create, click the blue link to **Create tags**.

Note: In **Logix Options**, set **Protocol Mode** to **Symbolic** and **Default Data Type** to **Default** so that the tags are imported with the data types used in the controller.

Offline

The Allen-Bradley ControlLogix Ethernetドライバー uses a file generated from RSLogix5000 called an L5K/L5X import/export file to generate the tag database.

In RSLogix5000

1. Open the project containing the tags to be ported over to the OPC server.
2. Click **File | Save As**.
3. Select **L5K/L5X Import/Export File** and specify a name. RSLogix will export the project's contents into this L5K/L5X file.

In the OPC Server

1. Open the device properties of the device for which tags will be generated.
2. Select **Logix Database Settings** and select **Create from Import File** for **Database Import Method**.
3. Enter or browse for the location of the file previously created.
4. In **Logix Database Options**, make any desired changes and click **OK**.
5. In **Logix Database Filtering**, make any desired changes and click **OK**.
6. Select **Tag Generation** and, under Create, click the blue link to **Create tags**.

Note: Imported pre-defined tag types are based on the latest version supported by the driver. For more information, refer to Firmware Versions.

Performance Optimization

For more information about optimization of communication and application, select a link below.

Optimizing Communications
Optimizing Application
Performance Statistics and Tuning
Performance Tuning Example

Optimizing Communications

As with any programmable controller, there are a variety of ways to enhance the performance and system communications.

Protocol Mode

The Protocol Mode determines how Logix tag data is accessed from the controller. There are three types of protocol modes: Symbolic, Logical Non-Blocking and Logical Blocking. Descriptions are as follows:

- **Symbolic Mode:** Each client/server tag address is represented in the packet by its ASCII character name.
- **Logical Non-Blocking Mode:** Each client/server tag is represented by its logical memory address in the PLC.
- **Logical Blocking Mode:** The Logix tag is accessed as a single chunk of data. Each client/server tag (such as MYTIMER.ACC) has a corresponding Logix tag (MYTIMER). Many client/server tags can belong to the same Logix tag, as in the case of structures. On every read cycle, the Logix tag is read, its block is updated in the driver cache and all client/server tags are updated from this cache.

Logical Non-Blocking Mode is generally recommended because it is the most efficient mode for gathering and processing Logix tag data. Symbolic Mode is recommended for backward compatibility, whereas Logical Non-Blocking Mode is recommended for projects containing a small number of references to UDT and/or predefined structure Logix tags. Although Logical Blocking Mode can be efficient, it can also hurt performance if used incorrectly. For more information on each mode's benefits and detriments, refer to **Choosing a Protocol Mode**.

Tag Division Tips

Users should designate one or more devices for Logical Blocking purposes and one or more devices for Logical Non-Blocking purposes. This improves performance because different tags in a project are often better suited for different modes. When utilizing tag division, users should do the following:

1. Assign server tags referencing Atomic Logix tags (array or non-array) to the Logical Non-Blocking device.
2. Assign server tags referencing a Structure Logix tag composed of one-third* or less of the Structure tag to the Logical Non-Blocking device(s). For example, if there are 55** or less member tags referencing a PID_ENHANCED Logix tag, all these tags should be assigned to the Logical Non-Blocking device.
3. Assign server tags referencing a Structure Logix tag composed of one-third* or more of the Structure tag to the Logical Blocking device(s). For example, if there are more than 55** member tags referencing a PID_ENHANCED Logix tag, all of those tags should be assigned to the Logical Blocking device.

*One-third is not an exact limit, but rather a figure that has held true in a number of studies.
**A PID_ENHANCED structure has 165 tags, so one-third equals 55 tags.

Connection Size

Increasing the Connection Size allows more read/write requests per data packet, which provides greater throughput. Although it also increases the CPU load and response turnaround time, it significantly improves performance. The Connection Size property may be modified in the ControlLogix 5500 and CompactLogix 5300 device models only. For more information, refer to **Logix Communications Parameters**.

UDT Substructure Aliasing

If a UDT contains large substructures and one-third or more of the substructure members are referenced in the client, refer to the following instructions to optimize reads for the substructure.

1. Create an alias of the substructure in RSLogix 5000. Then, assign server tags referencing the rest of the UDT substructure to a Logical Blocking device.
2. Next, assign the server tags referencing the rest of the UDT (but not the substructure) to a Logical Non-Blocking device.

System Overhead Time Slice

The System Overhead Time Slice (SOTS) is the percentage of time allocated to perform communication tasks (such as OPC driver communications) that is set in RSLogix 5000. 100% SOTS is the percentage of time for controller tasks (such as ladder logic). The default SOTS is 10%. In every 10 ms program scan that occurs, the controller spends 1 ms processing driver requests (if the controller has a continuous task). The value of SOTS defines the task's priority. If controller tasks are a high priority, the SOTS should be set below 30%. If the communication tasks are high priority, the SOTS should be set at or above 30%. For the best balance of communications performance and CPU utilization, set the SOTS to 10% to 40%.

Multi-Request Packets

The Allen-Bradley ControlLogix Ethernetドライバー has been designed to optimize reads and writes. For non-array, non-string tags (which only request one element), requests are blocked into a single transaction. This provides drastic improvement in performance over single tag transactions. The only limitation is the number of data bytes that can fit in a single transaction.

Important: In Symbolic Mode, each tag's ASCII string value is inserted into the request packet until no more tag requests fit. For optimum performance, users should keep the tag names' size to a minimum. The smaller the tag name, the more tags that fit in a single transaction and the fewer transactions needed to process all tags.

Array Elements Blocked (Symbolic and Logical Non-Blocking Modes Only)

To optimize the reading of atomic array elements, read a block of the array in a single request instead of individually. The more elements read in a block, the greater the performance. Since transaction overhead and processing consumes the most time, do as few transactions as possible while scanning as many desired tags as possible. This is the essence of array element blocking.

Block sizes are specified as an element count. A block size of 120 elements means that a maximum of 120 array elements are read in one request. The maximum block size is 3840 elements. Boolean arrays are treated differently: in protocol, a Boolean array is a 32-bit array. Thus, requesting element 0 is requesting bits 0 through 31. To maintain consistency in discussion, a Boolean array element is considered a single bit. In summary, the maximum number of array elements (based on block size of 3840) that can be requested is as follows: 122880 BOOL, 3840 SINT, 3840 INT, 3840 DINT, 3840 REAL, 3840 LINT, 3840 UINT, 3840 USINT, 3840 ULINT, 3840 LREAL, 3840 TIME32, 3840 TIME, and 3840 LTIME.

As discussed in **Logix Communication Parameters**, the block size is adjustable and should be chosen based on the project at hand. For example, if array elements 0-26 and element 3839 are tags to be read, then using a block size of 3840 is overly large and detrimental to the driver's performance. This is because all elements between 0 and 3839 are read on each request, even though only 28 of those elements are of importance. In this case, a block size of 30 is more appropriate. Elements 0-26 would be serviced in one request and element 3839 would be serviced on the next.

Optimizing Strings

In the Logical Addressing modes, a write to STRING.DATA also writes to STRING.LEN with the proper length value.

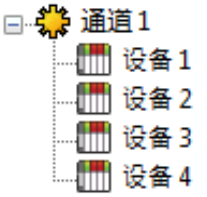
Terminate String Data at LEN
In this driver, string tags are structures with separate character data and length components. As such, the driver automatically reads a string tag in two transactions: one in Logical Protocol Mode for the string character data (DATA) and one in Symbolic Mode for the string length (LEN). When the Terminate String Data at LEN option is disabled, a single transaction is made to read the string character data. In this case, the Symbolic Mode read for string length is bypassed. In a project with many string tags, this can significantly reduce the time required to read all tags.

*For more information on the Terminate String Data at LEN option, refer to **Logix Options**.*

Optimizing the Application

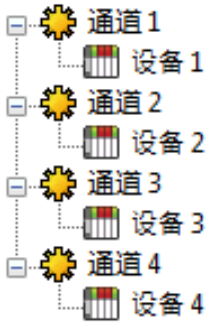
The Allen-Bradley ControlLogix Ethernet ドライバー has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Allen-Bradley ControlLogix Ethernet ドライバー is fast, there are a couple of guidelines that can be used to optimize the application and gain maximum performance.

The server refers to communications protocols, like Allen-Bradley ControlLogix Ethernet, as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices must then be defined under that channel. Each of these devices represents a single Allen-Bradley Logix CPU from which data is collected. While this approach to defining the application provides a high level of performance, it doesn't take full advantage of the Allen-Bradley ControlLogix Ethernet ドライバー or the network. An example of how the application may appear when configured using a single channel is shown below.



Each device is defined under a single channel. In this configuration, the driver moves from one device to the next as quickly as possible to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the driver could only define a single channel, the above would be the only option available; however, the driver can define up to 1024 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device can be defined under its own channel. In this configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has fewer devices, it can be optimized in this manner.

The performance improves even if the application has more devices. While fewer devices may be ideal, the application still benefits from additional channels. Although by spreading the device load across all channels causes the server to move from device to device again, it can now do so with far less devices to process on a single channel.

See Also**: **Performance Tuning Example**, **Optimizing Communications**, and **Performance Statistics and Tuning

Performance Statistics and Tuning

The Performance Statistics feature provides benchmarks and statistics about the application's performance. Because Performance Statistics is an additional layer of processing, it can affect the server's performance. As such, the default is off. To enable the Performance Statistics feature, access Device Properties and set **Logix Options** to enable **Enable Performance Statistics**.

Types of Performance Statistics

Performance Statistics provide meaningful numerical results across three scopes: device, channel, and driver. Descriptions of the types are as follows:

- **Device:** These statistics provide the data access performance on a particular device.
- **Channel:** These statistics provide the average data access performance for all the devices under a given channel with Performance Statistics enabled.
- **Driver:** These statistics provide the average data access performance for all devices using the Allen-Bradley ControlLogix Ethernetドライバー with Performance Statistics enabled.

Choosing a Statistic Type

The type of statistics needed depends on the application. In general, driver statistics provide a true measure of the application's performance, whereas channel and device statistics are most relevant while tuning the application. For example, will moving 10 certain tags from Device A to Device B increase the performance of Device A? Will moving Device A from Channel 1 to Channel 2 increase the performance of Channel 1? These questions are good examples of situations when device and channel statistics should be used.

Locating Statistics

Server statistics are output to the server's Event Log on shutdown. To view the results, shut down the server and restart it.

Differences Between Server Statistics and Performance Statistics

Performance Statistics provide the makeup of the types of reads performed (such as symbolic vs. symbol instance vs. physical, or device reads vs. cache reads) whereas server statistics provide a general read count value.

Tuning the Application for Increased Performance

For information on increasing device and channel statistic results, refer to the instructions below. For more information, refer to ***Optimizing Communications****.*

- Server tags referencing Atomic Logix tags (array or non-array) should be assigned to Logical Non-Blocking devices.
- Server tags referencing a Structure Logix tag composed of one-third or less of the Structure tag should be assigned to Logical Non-Blocking devices.
- Server tags referencing a Structure Logix tag composed of one-third or more of the Structure tag should be assigned to Logical Blocking devices.
- If Symbolic Mode is used, Logix names should be kept to a minimum length.
- Logix arrays should be used as often as possible.
- Only the necessary amount of System Overhead Time Slice for Ladder Logic/FBD should allocated to leave the rest for driver communications.
- For projects that read a large number of string tags in Logical Mode, disable the **Terminate String Data at LEN** option located under **Logix Options** in **Device Properties**.

*For information on increasing driver statistic results, refer to the instructions below. For more information, refer to **Optimizing Application**.*

- Devices should be spread across channels. More than one device should not be put on a channel unless necessary.
- Load should be spread evenly across devices. A single device should not be overloaded unless necessary.
- The same Logix tag should not be referenced across different devices.

Note: Although these general rules can help optimize performance, it ultimately depends on the application. The scan rate can obscure results: if tag requests are light, read and write transactions can complete before the next request comes in. In this case, Logical Blocking and Logical Non-Blocking will have the same Performance Statistics results. If tag requests are high (many tags or high scan rates), transaction completion time may take longer. This is when the strengths and weaknesses of Logical Blocking and Logical Non-Blocking become apparent. Performance Statistics can help tune the application for maximum performance. For an example, refer to **Performance Tuning Example**.

Performance Tuning Example

Statistics can be applied to any application. In the example below, the Quick Client is used in the performance tuning process. The idea is that all the tags used in the project are read at the same time at a fast scan rate. Although this is not realistic, it does provide an accurate benchmark to the project layout in the server (tags belonging to specific devices, devices belonging to specific channels, and so forth).

The statistics gathered are relative. Users should start with a server project layout, gather the statistics, and then tune. It is recommended that more than one trial be used to properly assess the results for a given layout. Once the most efficient layout is determined, the client application can be built with reassurance that the server is optimal.

Performance results obtained using the Quick Client do not equate to performance results obtained using a client application: several factors produce discrepancies. Although performance tuning with the client application is more accurate than with the Quick Client, the tuning required not only affects the server project, but the client application as well. It is recommended that the Quick Client be used to tune the application before the client application is developed.

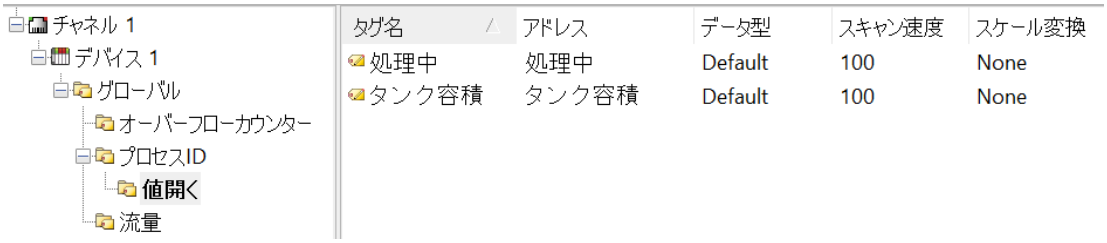
Note: The tuning process described below assumes that all tags are being read at a fast scan rate. Writes hinder the performance.

1. In the controller project displayed below, there are the following:

   2 Atomics
   1 Atomic Array
   1 UDT
   1 UDT Array
   1 Pre-Defined Type

   Note: Overhead Time Slice (OTS) = 10%.

2. After performing Automatic Tag Database Generation from this controller, the server produces the following project.

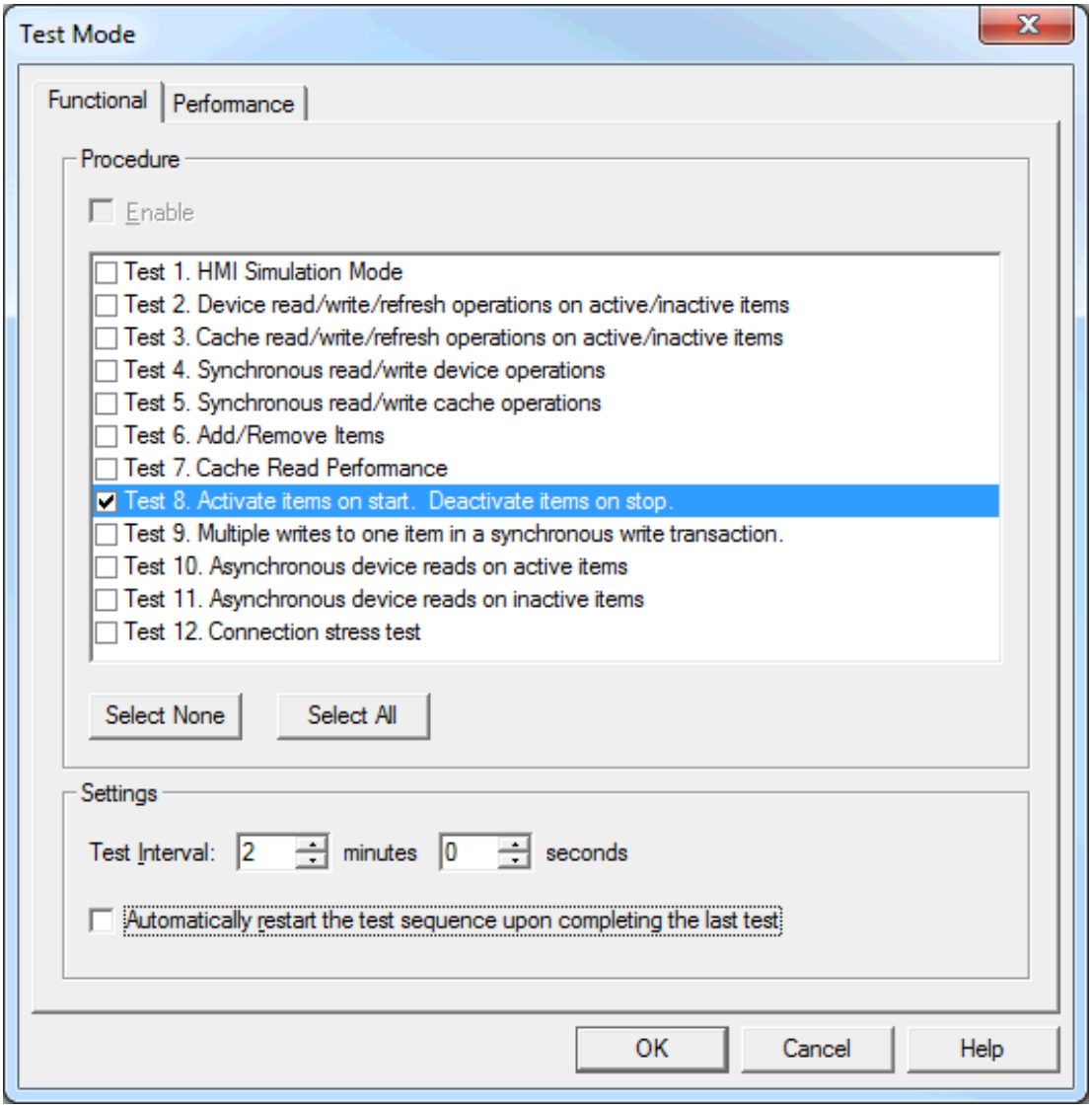


Note: The "Global" tag group contains 130 tags.

3. To illustrate the benefits of tag division, this example does not reference all tags. More than one-third of the ProcessPID tags, less than one-third of the FlowRates tags, and all other tags are referenced. As such, the new tag count is 105.
4. Prepare the client for the test. To do so, launch the Quick Client from the server application by clicking on the QuickClient icon.
5. Once the project is loaded, remove all groups except those containing tags of interest. Statistics and System tags, for example, are not needed.

   Note: For small projects, set the group **Update Rate** to 0-10 ms. For large projects, set the rate to 10-50 ms.
6. Select **Tools | Test Mode**.
7. Enable **Test 8. Activate items on start. Deactivate items on stop** and then set a test interval.

   Note: Since this project is fairly small, the interval has been set to 2 minutes. For larger projects, the interval should be increased to get a more accurate reading.



8. Return to **Tools | Test Mode** and disable test mode. All tags should be deactivated.
9. Disconnect the Quick Client so that time trials can begin.
10. Shutdown the server.
11. Launch the server and set the **Protocol Mode** to **Logical Blocking** under device properties.

12. Set **Performance Statistics** to Enable.

13. Connect to the server using the Quick Client. Select **Tools | Test Mode**. Enable Test Mode.

Note: Data reading begins. When the test interval expires, all tags are deactivated and the driver ceases statistics gathering. The results can then be viewed.

14. Disconnect the Quick Client from the server and then shutdown the server.

15. Re-launch the server and search its Event Log for statistics. The image below displays the first trial utilizing Logical Blocking for the device.

```
DEVICE Channel1:Device1 STATISTICS
Physical Block Device Reads = 40932
Physical Block Cache Reads = 661734
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 13644
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 716390, Elapsed Time = 119969 ms
DEVICE Channel1:Device1 PERFORMANCE: AvgTagReadPerSec = 5972.26
```

Note: The image below displays the first trial utilizing Logical Blocking for the channel and driver.

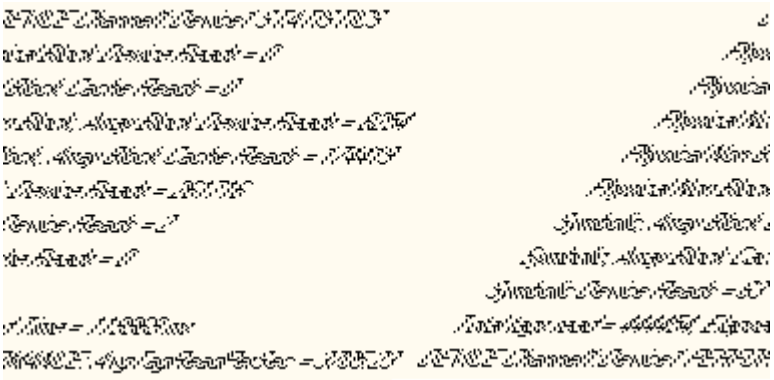
```
DRIVER STATISTICS (ALL CHANNELS)
Physical Block Device Reads = 40932
Physical Block Cache Reads = 661734
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 13644
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 716390, Elapsed Time = 119969 ms
DRIVER PERFORMANCE: AvgTagReadPerSec = 5972.26
Closing project C:\RDM\Support\ControlLogix Ethernet\CL_DEFAULT.opf
CHANNEL Channel1 STATISTICS
Physical Block Device Reads = 40932
Physical Block Cache Reads = 661734
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 13644
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 716390, Elapsed Time = 119969 ms
CHANNEL Channel1 PERFORMANCE: AvgTagReadPerSec = 5972.26
```

Note: This is the control set for comparisons.

16. In the server, set the **Protocol Mode** to **Logical Non-Blocking**.
17. Connect to the server using Quick Client. Select **Tools | Test Mode** and enable test mode.

Note: Data reading begins. When the test interval expires, all tags are deactivated and the driver ceases statistics gathering. The results can then be viewed.

18. Disconnect the Quick Client from the server and then shutdown the server.
19. Re-launch the server and then search its Event Log for statistics. The image below displays the second trial utilizing Logical Non-Blocking for the device.



Note: The image below displays the second trial utilizing Logical Non-Blocking for the channel and driver.

```
DRIVER STATISTICS (ALL CHANNELS)
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 8254
Physical Non Block, Array Block Cache Reads = 174419
Physical Non Block Device Reads = 261716
Symbolic, Array Block Device Reads = 2
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 63
Total tags read = 444454, Elapsed Time = 119969 ms
DRIVER PERFORMANCE: AvgTagReadPerSec = 3705.23
Closing project C:\RDM\Support\ControlLogix Ethernet\CL_DEFAULT.opf
CHANNEL Channel1 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 8254
Physical Non Block, Array Block Cache Reads = 174419
Physical Non Block Device Reads = 261716
Symbolic, Array Block Device Reads = 2
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 63
Total tags read = 444454, Elapsed Time = 119969 ms
CHANNEL Channel1 PERFORMANCE: AvgTagReadPerSec = 3705.23
```

20. From the server, set the **Protocol Mode** to **Symbolic** to see how the performance fared prior to Allen-Bradley ControlLogix Ethernetドライバー version 4.6.0.xx.
21. Connect to the server using the Quick Client. Then, click **Tools | Test Mode** and enable test mode.

    Note: Data reading begins. When the test interval expires, all tags are deactivated and the driver ceases statistics gathering. The results can then be viewed.
22. Disconnect the Quick Client from the server and then shutdown the server.
23. Re-launch the server and search its Event Log for statistics. The image below displays the third trial utilizing Symbolic for the device.

```
DEVICE Channel1:Device1 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 0
Symbolic, Array Block Device Reads = 1744
Symbolic, Array Block Cache Reads = 36613
Symbolic Device Reads = 54985
Total tags read = 93342, Elapsed Time = 120063 ms
DEVICE Channel1:Device1 PERFORMANCE: AvgTagReadPerSec = 777.442
```

The image below displays the third trial utilizing Symbolic for the channel and driver.

```
DRIVER STATISTICS (ALL CHANNELS)
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 0
Symbolic, Array Block Device Reads = 1744
Symbolic, Array Block Cache Reads = 36613
Symbolic Device Reads = 54985
Total tags read = 93342, Elapsed Time = 120063 ms
DRIVER PERFORMANCE: AvgTagReadPerSec = 777.442
Closing project C:\RDM\Support\ControlLogix Ethernet\CL_DEFAULT.opf
CHANNEL Channel1 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 0
Symbolic, Array Block Device Reads = 1744
Symbolic, Array Block Cache Reads = 36613
Symbolic Device Reads = 54985
Total tags read = 93342, Elapsed Time = 120063 ms
CHANNEL Channel1 PERFORMANCE: AvgTagReadPerSec = 777.442
```

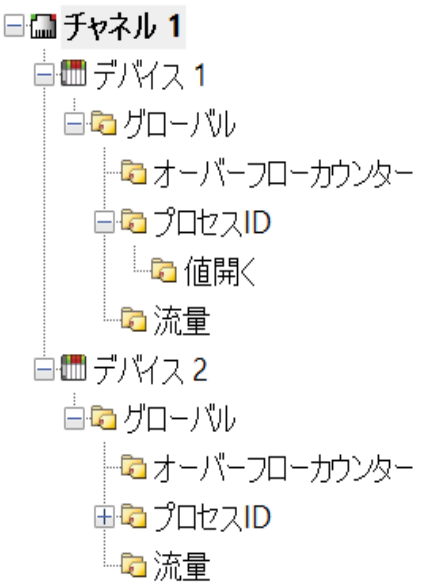
Note: It appears that Logical Blocking is most optimal for the given application.

Optimizing Channel Communications

Channel communications can be optimized by moving tags for Logical Blocking in one device and tags for Logical Non-Blocking in another. This is called tag division.

Logical Blocking (Device 1)
ProcessPID
OverflowCounter

Logical Non-Blocking (Device 2)
FlowRate
ValveOpen
InProcess
Tank Volume



1. Repeat Steps 4 through 15. In Step 11, make sure that Device 1 is Logical Blocking and Device 2 is Logical Non-Blocking.
2. Launch the server and search the server Event Log for statistics. The image below displays the fourth trial utilizing tag division for the device.

```
DEVICE Channel1:Device1 STATISTICS
Physical Block Device Reads = 13866
Physical Block Cache Reads = 610104
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 6933
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 76
Total tags read = 630979, Elapsed Time = 119782 ms
DEVICE Channel1:Device1 PERFORMANCE: AvgTagReadPerSec = 5268.43
DEVICE Channel1:Device2 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 6933
Physical Non Block, Array Block Cache Reads = 69375
Physical Non Block Device Reads = 27732
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 4
Total tags read = 104044, Elapsed Time = 119969 ms
DEVICE Channel1:Device2 PERFORMANCE: AvgTagReadPerSec = 867.373
```

Note: The image below displays the fourth trial utilizing tag division for the channel and driver.

```
DRIVER STATISTICS (ALL CHANNELS)
Physical Block Device Reads = 13866
Physical Block Cache Reads = 610104
Physical Non Block, Array Block Device Reads = 6933
Physical Non Block, Array Block Cache Reads = 69375
Physical Non Block Device Reads = 34665
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 735023, Elapsed Time = 119969 ms
DRIVER PERFORMANCE: AvgTagReadPerSec = 6126.77
Closing project C:\RDM\Support\ControlLogix Ethernet\CL_DEFAULT.opf
CHANNEL Channel1 STATISTICS
Physical Block Device Reads = 13866
Physical Block Cache Reads = 610104
Physical Non Block, Array Block Device Reads = 6933
Physical Non Block, Array Block Cache Reads = 69375
Physical Non Block Device Reads = 34665
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 735023, Elapsed Time = 119969 ms
CHANNEL Channel1 PERFORMANCE: AvgTagReadPerSec = 6126.77
```

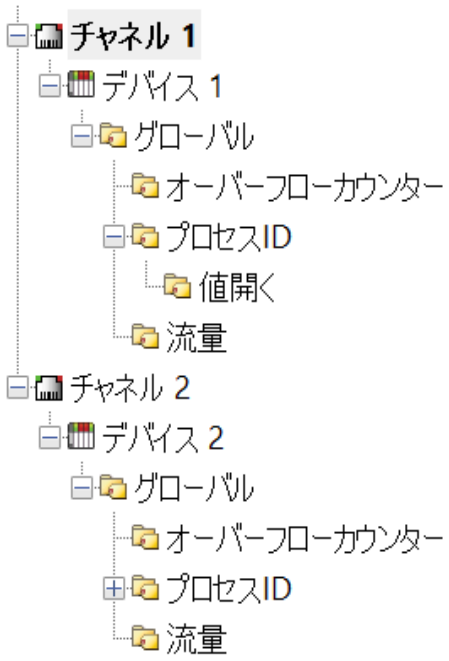
Note: The individual device statistics do not look impressive because the two devices are running on separate statistic counters. The key to this test is that the channel and driver statistics are better (6126) than using one channel/one device with either Logical Blocking (5972) or Logical Non-Blocking (3705).

Optimize Application

The application can be optimized by moving Device 1 to one channel and Device 2 to another.

Logical Blocking (Channel1.Device 1
ProcessPID
OverflowCounter

Logical Non-Blocking (Channel2.Device 2)
FlowRate
ValveOpen
InProcess
Tank Volume



1. Repeat Steps 4 through 15. In Step 11, make sure Channel1.Device 1 is Logical Blocking and Channel2.Device 2 is Logical Non-Blocking.
2. Launch the server and search the server Event Log for statistics. The image below displays the fifth trial utilizing Logix tag coupled with multiple channels for Channel 1.Device1.

```
CHANNEL Channel1 STATISTICS
Physical Block Device Reads = 14542
Physical Block Cache Reads = 639848
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 7271
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 661741, Elapsed Time = 119968 ms
CHANNEL Channel1 PERFORMANCE: AvgTagReadPerSec = 5517.4
DEVICE Channel1:Device1 STATISTICS
Physical Block Device Reads = 14542
Physical Block Cache Reads = 639848
Physical Non Block, Array Block Device Reads = 0
Physical Non Block, Array Block Cache Reads = 0
Physical Non Block Device Reads = 7271
Symbolic, Array Block Device Reads = 0
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 80
Total tags read = 661741, Elapsed Time = 119968 ms
DEVICE Channel1:Device1 PERFORMANCE: AvgTagReadPerSec = 5517.4
```

Note: The image below displays the fourth trial utilizing Logix tag for Channel2.Device2.

```
CHANNEL Channel2 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 7280
Physical Non Block, Array Block Cache Reads = 72800
Physical Non Block Device Reads = 29120
Symbolic, Array Block Device Reads = 1
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 4
Total tags read = 109205, Elapsed Time = 119968 ms
CHANNEL Channel2 PERFORMANCE: AvgTagReadPerSec = 910.52
DEVICE Channel2:Device2 STATISTICS
Physical Block Device Reads = 0
Physical Block Cache Reads = 0
Physical Non Block, Array Block Device Reads = 7280
Physical Non Block, Array Block Cache Reads = 72800
Physical Non Block Device Reads = 29120
Symbolic, Array Block Device Reads = 1
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 4
Total tags read = 109205, Elapsed Time = 119968 ms
DEVICE Channel2:Device2 PERFORMANCE: AvgTagReadPerSec = 910.52
```

Note: The image below displays the fourth trial utilizing tag division for the driver.

```
DRIVER STATISTICS (ALL CHANNELS)
Physical Block Device Reads = 14542
Physical Block Cache Reads = 639848
Physical Non Block, Array Block Device Reads = 7280
Physical Non Block, Array Block Cache Reads = 72800
Physical Non Block Device Reads = 36391
Symbolic, Array Block Device Reads = 1
Symbolic, Array Block Cache Reads = 0
Symbolic Device Reads = 84
Total tags read = 770946, Elapsed Time = 119968 ms
DRIVER PERFORMANCE: AvgTagReadPerSec = 6426.26
```

Results

Server Project Layout	Driver Performance (Reads / Second)	Improvement Over Symbolic
Single Channel / Single Device with Logical Blocking	5972	768%
Single Channel / Single Device with Logical Non-Blocking	3705	476%
Single Channel / Single Device with Symbolic	777	N/A
Single Channel / Multiple Devices with Tag Division	6126	788%
Multiple Channels / Multiple Devices with Tag Division	6426	827%

Conclusions

The project began with a single channel and a single device, which is the default behavior for a single controller. All tags were imported from this controller to this channel.device. All three protocol modes were

then tested to see which would provide the best performance. In this case, Logical Blocking Protocol was the best. The best protocol depends on the application at hand. When performance is crucial, it is worth performing Logical Blocking and Logical Non-Blocking trials to determine which is the best protocol mode for the application. Symbolic protocol is not necessary because it never meets the performance caliber of either of the other protocol modes. It is shown here for the sake of the example.

Measures were taken to optimize communications using the tips outlined in **Optimizing Communications**. Most notably, tag division was used to place the Logical Blocking type tags in a device assigned Logical Blocking and the Logical Non-Blocking type tags in a device assigned Logical Non-Blocking. Furthermore, both devices resided on the same channel. The results show an improvement over using Logical Blocking on a single device. This is because some tags lend themselves better to one protocol mode over another. For example, reading an entire COUNTER benefits from Logical Blocking over Logical Non-Blocking since it's much faster reading the COUNTER as a block then as individual members.

Measures were also taken to optimize the application by placing devices on their own channel. Using the devices created in the previous trial, a Logical Blocking device was placed on one channel and a Logical Non-Blocking device on another. The results show improvement over the single channel / multiple devices scenario from the previous trial. This reinforces the idea that performance is improved by having as few devices per channel and as many channels as necessary.

Tip: When multiple devices are connected to the same PLC, different outcomes have been observed when using the L8 processor versus the L7 processor, potentially influenced by the firmware version in use. Testing various setups is advisable to identify the configuration that best suits your specific devices and firmware. In some scenarios, grouping all devices connected to the same PLC under a single channel may offer improved performance.

After using these optimization methods, the project has a significant performance increase (especially in older versions). The performance increase is more apparent with larger projects.

Data Types Description

Data Types	Description
Boolean	Single bit
Byte	Unsigned 8-bit value
Char	Signed 8-bit value
Word	Unsigned 16-bit value
Short	Signed 16-bit value
DWord	Unsigned 32-bit value
Long	Signed 32-bit value
BCD	Two byte packed BCD, four decimal digits
LBCD	Four byte packed BCD, eight decimal digits
Float	32-bit IEEE floating point
Double	64-bit IEEE floating point
Date	64-bit Date/Time
String	Null-terminated character array

- *For a description of Logix platform-specific data types, refer to **Logix Advanced Addressing**.*

- *For specifics about Boolean arrays in firmware V30, visit the PTC website, log in to eSupport, and search for article cs332995.*

Default Data Type Conditions

Client / server tags are assigned the default data type when any of the following conditions occur:

1. A dynamic tag is created in the client with Native as its assigned data type.
2. A static tag is created in the server with Default as its assigned data type.
3. In offline automatic tag generation, when an unknown data type is encountered in the L5K/L5X file for UDT members and alias tags.
4. In offline automatic tag generation, when an alias of the following type is encountered in the L5K/L5X:
    a. Alias of an alias.
    b. Alias of non bit-within-Word/DWord I/O module tag. For example, if tag "AliasTag" references I/O module tag "Local:5:C.ProgToFaultEn" @ BOOL, the data type for "AliasTag" cannot be resolved, so this default type is assigned to it. On the other hand, if "AliasTag" references I/O module tag "Local:5:C.Ch0Config.RangeType.0" @ BOOL, the data type can be resolved because of the . (dot) BIT that defines it as a bit-within-Word/DWord. Aliases of bit-within-Word/DWord I/O module tags are automatically assigned the Boolean data type.

- **Notes:**

1. If Default is selected, the driver retrieves the Logix tag's data type from the controller when a client is accessing a tag dynamically and does not explicitly assign a data type to the item. For example, a tag exists in the controller that is called "MyTag" with a data type of REAL. The corresponding client item is specified as "Channel1.Device1.MyTag" with no data type assigned.

With Default selected as the default data type in the server, the driver reads "MyTag" from the controller and determine that it is a REAL in the response, providing the client a data type of Float.

2. Since the majority of I/O module tags are not bit-within-Word/DWord tags, it is advised that the default type be set to the majority data type as observed in the .ACD project. For example, if 75% of alias I/O module tags are INT tags, set the default type to INT.

Address Descriptions

Address specifications vary depending on the model in use. For the model of interest's address information, refer to the table below.

Model	Output	Input	Status	Binary	Timer	Counter	Control	Integer	Float	ASCII	String	BCD	Long	PID	Message	Block Transfer	Function
MicroLogix	X	X	X	X	X	X	X	X	X	X	X	X		X	X		
PLC5	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	
SLC5/05	X	X	X	X	X	X	X	X	X	X	X						

See Also:
Logix Addressing
MIcroLogix Addressing
PLC-5 Series Addressing
SLC 500 Modular I/O Addressing

Protocol Class	Models	Help Link
Logix-Ethernet	ControlLogix 5500 Ethernet, CompactLogix 5300 Ethernet, FlexLogix 5400 Ethernet, SoftLogix 5800	Logix Addressing
DH+ Gateway	DH+ Gateway: PLC-5 DH+ Gateway: SLC 5/04	PLC-5 Series Addressing SLC 500 Modular I/O Addressing
ControlNet Gateway	ControlNet Gateway: PLC-5C	PLC-5 Series Addressing
1761-NET-ENI	ENI: ControlLogix 5500 ENI: CompactLogix 5300 ENI: FlexLogix 5400 ENI: MicroLogix ENI: SLC 500 Fixed I/O ENI: SLC 500 Modular I/O ENI: PLC-5	Logix Addressing MicroLogix Addressing SLC 500 Fixed I/O Addressing SLC 500 Modular I/O Addressing PLC-5 Series Addressing
MicroLogix 1100 Ethernet	MicroLogix 1100	MicroLogix Addressing
MicroLogix 1400 Ethernet	MicroLogix 1400	MicroLogix Addressing

For more information on the controller's pre-defined data types, refer to the device documentation.

Logix Addressing

*For more information on these models' tag-based addressing and relationship to the Allen-Bradley ControlLogix Ethernetドライバー, refer to **Logix Tag-Based Addressing**.*

ControlLogix 5500 Addressing for Ethernet

ControlLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

ControlLogix 5500 Addressing for ENI

ControlLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

ControlLogix 5500 Addressing for Serial Gateway

ControlLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

CompactLogix 5300 Addressing for Ethernet

CompactLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

CompactLogix 5300 Addressing for ENI

CompactLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

CompactLogix 5300 Addressing for Serial Gateway

CompactLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

FlexLogix 5400 Addressing for Ethernet

FlexLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

FlexLogix 5400 Addressing for ENI

FlexLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

FlexLogix 5400 Addressing for Serial Gateway

FlexLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

SoftLogix 5800 Addressing

SoftlLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

SoftLogix 5800 Addressing for Serial Gateway

SoftlLogix is a member of the Logix family and part of Rockwell Automation's Integrated Architecture. This means it uses a tag or symbol-based addressing structure. Logix tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

MicroLogix Addressing

MicroLogix Addressing for EtherNet/IP Gateway

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
Long Files
MicroLogix PID Files
MicroLogix Message Files

For information on function files, select a link from the list below.

High-Speed Counter File (HSC)
Real-Time Clock File (RTC)
Channel 0 Communication Status File (CS0)
Channel 1 Communication Status File (CS1)
I/O Module Status File (IOS)

MicroLogix Addressing for ENI

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files

ASCII Files
String Files
Long Files
MicroLogix PID Files
MicroLogix Message Files

For information on function files, select a link from the list below.

High-Speed Counter File (HSC)
Real-Time Clock File (RTC)
Channel 0 Communication Status File (CS0)
Channel 1 Communication Status File (CS1)
I/O Module Status File (IOS)

MicroLogix 1100 Addressing

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
String Files
Long Files
MicroLogix PID Files
MicroLogix Message Files

For information on function files, select a link from the list below.

High-Speed Counter File (HSC)
Real-Time Clock File (RTC)
Channel 0 Communication Status File (CS0)
Channel 1 Communication Status File (CS1)
I/O Module Status File (IOS)

MicroLogix 1400 Addressing

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not

present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
Long Files
MicroLogix PID Files
MicroLogix Message Files

For information on function files, select a link from the list below.

High-Speed Counter File (HSC)
Real-Time Clock File (RTC)
Channel 0 Communication Status File (CS0)
Channel 1 Communication Status File (CS1)
I/O Module Status File (IOS)

SLC 500 Fixed I/O Addressing

SLC 500 Fixed I/O Addressing for EtherNet/IP Gateway

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files

SLC 500 Fixed I/O Addressing for ENI

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files

Binary Files
Timer Files
Counter Files
Control Files
Integer Files

SLC 500 Modular I/O Addressing

SLC 500 Modular I/O Addressing for DH+

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files

SLC 500 Modular I/O Addressing for EtherNet/IP Gateway

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files

SLC 500 Modular I/O Addressing for ENI

The actual number of addresses available depends on the model of the PLC. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and then removes the tag from its scan list. For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files

PLC-5 Series Addressing

PLC-5 Series Addressing for ControlNet

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
BCD Files
PID Files
Message Files
Block Transfer Files

PLC-5 Series Addressing for DH+

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files

Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
BCD Files
PID Files
Message Files
Block Transfer Files

PLC-5 Series Addressing for EtherNet/IP Gateway

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
BCD Files
PID Files
Message Files
Block Transfer Files

PLC-5 Series Addressing for ENI

For more information on file-specific addressing, select a link from the list below.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files
BCD Files

PID Files
Message Files
Block Transfer Files

Logix Tag-Based Addressing

Rockwell Automation's Integrated Architecture uses a tag or symbol-based addressing structure that is commonly referred to as Logix tags (or Native Tags). These tags differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.
Note: Throughout this help file, Logix tags are assumed to be global in nature unless specified otherwise.

The Allen-Bradley ControlLogix Ethernetドライバー allows users to access the controller's atomic data types: BOOL, SINT, INT, DINT, LINT, REAL, LREAL, USINT, UINT, UDINT, ULINT, TIME32, TIME, and LTIME. Although some of the pre-defined types are structures, they are ultimately based on these atomic data types. Thus, all non-structure (atomic) members of a structure are accessible. For example, a TIMER cannot be assigned to a server tag but an atomic member of the TIMER can be assigned to the tag (such as TIMER.EN, TIMER.ACC, and so forth). If a structure member is a structure itself, both structures would have to be expanded to access an atomic member of the substructure. This is more common with user and module-defined types and is not found in any of the pre-defined types.

Atomic Data Type	Description		Range
BOOL	Single-bit value	VT_BOOL	0, 1
SINT	Signed 8-bit value	VT_UI1	-128 to 127
INT	Signed 16-bit value	VT_I2	-32,768 to 32,767
DINT	Signed 32-bit value	VT_I4	-2,147,483,648 to 2,147,483,647
LINT	Signed 64-bit value	VT_I8	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
USINT	Unsigned 8-bit value	VT_UI1	0 to 255
UINT	Unsigned 16-bit value	VT_UI2	0 to 65535
UDINT	Unsigned 32-bit value	VT_UI4	0 to 4294967295
ULINT	Unsigned 64-bit value	VT_I8	0 to 18,446,744,073,709,551,615
REAL	32-bit IEEE floating point	VT_R4	±1.1754943508222875E-38 to ±3.4028234663852886E+38 (normalized) 0 ±1.4012984643248170E-45 to ±1.1754942106924411E-38 (denormalized)
LREAL	Signed 64-bit value	VT_R8	±2.2250738585072014E-308 to ±1.7976931348623157E+308 (normalized) 0, ±4.9406564584124654E-324 to ±2.2250738585072010E-308 (denormalized)
TIME32	Duration of time in microseconds as string	VT_BSTR	T32#-35m_47s_483ms_647us to T32#35m_47s_483ms_647us

Atomic Data Type	Description		Range
	Duration of time in microseconds as signed 32-bit value	VT_I4	-2147483647 to 2147483647
TIME	Duration of time in microseconds as string	VT_BSTR	T#-106750d_12h_59m_59s_999ms_999us to T#106750d_12h_59m_ 59s_999ms_999us
	Duration of time in microseconds as signed 64-bit value	VT_I8	-9223246799999999 to 9223246799999999
LTIME	Duration of time in nanoseconds as string	VT_BSTR	LT#-106750d_12h_59m_59s_999ms_999us_999ns to LT#106750d_12h_59m_ 59s_999ms_999us_ 999ns
	Duration of time in nanoseconds as signed 64-bit value	VT_I8	-9223246799999999999 to 9223246799999999999

***See Also**: Logix Advanced Addressing*

Client / Server Tag Address Rules
Logix tag names correspond to client / server tag addresses. Logix tag names (entered via RSLogix5000) follow the IEC 1131-3 identifier rules. Client / server tag addresses follow these same rules. They are as follows:

- Must begin with an alphabetic (A-Z, a-z) character or an underscore (_).
- Can only contain alphanumeric characters and underscores.
- Can have as many as 40 characters.
- Cannot have consecutive underscores.
- Are not case sensitive.

Client / Server Tag Name Rules
Tag name assignment in the server differs from address assignment in that names cannot begin with an underscore.

Note: Logix tag names should be kept to a minimum in size for optimum performance. The smaller the name, the more requests that are able fit in a single transaction.

Symbolic Mode users should keep the client / server tag addresses below 400 characters. For example, tagarray[1,2,4].somestruct.substruct_array[3].basetag.[4] is 57 characters in length. Since a packet can only hold 500 data bytes, any overhead bytes that need to be added to the packet can greatly diminish the room available to the characters themselves. By keeping the address below 400, the tag request remains complete and valid.

***See Also**: Performance Optimizations*

アドレスのフォーマット

サーバーで静的にまたはクライアントから動的に Logix タグのアドレスを指定するにはいくつかの方法があります。使用するフォーマットはタグのタイプと使用法によって異なります。たとえば、SINT 型タグ内のビットにアクセスする場合にはビット

フォーマットを使用します。アドレスのフォーマットと構文については、以下の表を参照してください。

注記: 配列とString 型を除くフォーマットはすべて RSLogix5000 ネイティブです。したがって、アトミックデータ型を参照する場合、RSLogix 5000 のタグ名をコピーしてサーバーのタグアドレスフィールドに貼り付けることでそのタグ名が有効になります。

フォーマット	構文	例	注記
標準	<Logix タグ名>	tag_1	タグが配列であってはなりません。
配列要素	<Logix 配列タグ名> [次元 1, 次元 2, 次元 3]	tag_1 [2, 58, 547] tag_1 [0, 3]	次元の範囲 = 1 から 3 | 要素の範囲 = 0 から 65535
オフセットがない配列*	<Logix 配列タグ名> {列数} <Logix 配列タグ名> {行数}{列数}	tag_1 {8} tag_1 {2} {4}	次元の範囲 = 1 から 2 | 要素の範囲 = 1 から 65535 読み書きする要素の数は行数 x 列数です。行数が指定されていない場合、行数はデフォルトで 1 になります。 配列はゼロオフセットで開始します (すべての次元で配列のインデックスが 0)。
オフセットがある配列*	<Logix 配列要素タグ> [offset1,offset2] {列数} <Logix 配列要素タグ> [offset1,offset2] {行数} {列数}	tag_1 [2, 3] {10} tag_1 [2, 3]{2} {5}	配列は配列要素タグで次元ごとに指定されているオフセットで開始します。この配列では必ず最大の次元がカバーされます。Tag_1[2,3]{10} では要素 tag_1[2,3] -> tag_1[2,13] の配列が生成されます
ビット	<Logix タグ名>.ビット <Logix タグ名>.[ビット]	tag_1.0 tag_1.[0]	ビット範囲 = 0 から 31 タグが配列である場合、必ず BOOL 配列になります。BOOL 配列でなければタグが配列になることはできません。
文字列	<Logix タグ名>/<最大文字列長>	tag_1.Data/4 Stringtag_1.Data/82 SINTarraytag_1/16	長さの範囲 = 1 から 65535 この文字列との間で読み取り/書き込み可能な最大文字数。

*このフォーマットでは複数の要素が要求されることがあるため、配列データが渡される順序は Logix 配列タグの次元によって異なります。たとえば、行数 x 列数 = 4 でコントローラタグが 3X3 要素の配列である場合、array_tag [0,0]、array_tag [0,1]、array_tag [0,2]、array_tag [1,0] の順序で要素が参照されます。コントローラタグが 2X10 要素の配列であった場合には結果が異なります。

1 次元、2 次元、および 3 次元配列で要素がどのように参照されるかについては、***配列データの順序***を参照してください。

Tag Scope

Global Tags

Global tags are Logix tags that have global scope in the controller. Any program or task can access Global tags; however, the number of ways a Global tag can be referenced depends on its Logix data type and the address format being used.

Program Tags

Program tags are identical to Global tags except that a Program tag's scope is local to the program in which it is defined. Program tags follow the same addressing rules and limitations as Global tags, but are prefixed with the following notation:

Program: <program name>.

For example, Logix tag "tag_1" in program "prog_1" would be addressed as "Program:prog_1.tag_1" in a client/server tag address.

Structure Tag Addressing

Logix Structure tags (Global or Program) are tags with one or more member tags. Member tags can be atomic or structured in nature.

<structure name>. <atomic-type tag>

This implies that a substructure would be addressed as:

<structure name> . <substructure name> .<atomic-type tag>

Arrays of structures would be addressed as:

<structure array name> [dim1, dim2, dim3] . <atomic-type tag>

This implies that an array of substructures would be addressed as:

<structure name> . <substructure array name> [dim1, dim2, dim3] . <atomic-type tag>

Note: The examples above are only a few of the addressing possibilities that involve structures and are displayed to provide an introduction to structure addressing. For more information, refer to Allen-Bradley or Rockwell documentation.

Internal Tags

Internal tags are not visible in the server configuration, but can be browsed by the OPC client and found under the *<Channel Name>.<Device Name>* group. The following tags are only valid for the ControlLogix 5500 and CompactLogix 5300 device models.

Tag Name	Support	Data Type	Access
_CIPConnectionSizeRequested	The CIP connection size that was last requested.	Word	Read/Write*
_CIPConnectionSizeActual	The actual CIP connection size that is in use. Its value differs from _ CIPConnectionSizeRequested if the value requested is not supported by the device.	Word	Read Only
_LogicalAddressUploadCount	Number of controller project uploads that have occurred since startup or reset. Write 0 to reset.	Word	Read/Write
_LogicalAddressUploadDuration	Amount of time it took to perform the last controller project upload. Tag will hold	Double	Read Only

Tag Name	Support	Data Type	Access
	this value until the next upload occurs.		
_LastEditDetectedTimestamp	Timestamp for when the last controller project edit (online or offline) was detected.	String	Read Only
_LastLogicalAddressUploadTimestamp	Timestamp for when the last controller project upload began.	String	Read Only

*This tag is read only for ENI Logix models.

Changing the CIP Connection Size

The _CIPConnectionSizeRequested tag allows users to change the CIP connection size property in real time. Both the connection size property (located under **Logix Communication Parameters** in **Device Properties**) and the System tag are configurable while clients are connected. Changes are applied before the next read/Write request is performed.

Predefined Term Tags

The tags displayed in the table below can be used to obtain general processor information from a PLC running firmware version 13 or higher.

Tag Name	Description
#MODE	A description of the PLC's current key switch mode. Possible string values include Program, Run, Remote Program, Remote Run, and Remote Debug. Supported data types include string.
#PLCTYPE	An integer value that corresponds to the "ProdType" attribute specified in the PLC's EDS file. Supported data types include all but string.
#REVISION	Firmware revision displayed as "<major>.<minor>". Supported data types include string.
#PROCESSORNAME	The processor name that corresponds to the "ProdName" attribute specified in the PLC's EDS file. Supported data types include string.
#STATUS	Indicates the PLC's status. Possible values include OK (1) and Faulted (0). Supported data types include all but date.
#PRODUCTCODE	An integer value that corresponds to the "ProdCode" attribute specified in the PLC's EDS file. Supported data types include all but string.
#VENDORID	An integer value that corresponds to the "VendCode" attribute specified in the PLC's EDS file. Supported data types include all but string.

Addressing Atomic Data Types

Below are suggested usages and addressing possibilities for a Logix data type given the address formats available. Examples are also given for reinforcement. Click on **Advanced** for advanced addressing possibilities for the given atomic data type.

Tip: If multiple server data types are supported for an Atomic Data Type, the default used in **Automatic Tag Database Generation** is shown in **bold**.

Note: Empty cells do not necessarily indicate a lack of support.

Atomic Data Type	Standard	Array Element	Array with or without Offset	Bit	String
BOOL					
Client / Server Data Type **Advanced**	Boolean	Boolean (BOOL 1 dimensional array)	Boolean Array (BOOL 1 dimensional array)		*See **Advanced Addressing BOOL***
Client / Server Tag Example	BOOLTAG	BOOLARR[0]	BOOLARR[0]{32}		
SINT					
Client / Server Data Type **Advanced**	Byte, Char	Byte, Char	Byte Array, Char Array (SINT 1/2/3 dimensional array)	Boolean (Bit w/i SINT)	String (SINT 1/2/3 dimensional array) *See **Advanced Addressing SINT***
Client / Server Tag Example	SINTTAG	SINTARR[0]	SINTARR[0]{4}	SINTTAG.0	SINTARR/4
INT					
Client / Server Data Type **Advanced**	Word, Short	Word, Short	Word Array, Short Array (INT 1/2/3 dimensional array)	Boolean (Bit w/i INT)	*See **Advanced Addressing INT***
Client / Server Tag Example	INTTAG	INTARR[0]	INTARR[0]{4}	INTTAG.0	
DINT					
Client / Server Data Type **Advanced**	DWord, Long	DWord, Long	DWord Array, Long Array	Boolean (Bit w/i DINT)	*See **Advanced Addressing DINT***
Client / Server Tag Example	DINTTAG	DINTARR[0]	DINTARR[0]{4}	DINTTAG.0	
LINT					
Client/Server Data Type **Advanced**	Double, Date	Double, Date	Double Array		*See **Advanced Addressing LINT***
Client/Server Tag Example	LINTTAG	LINTARR[0]	LINTARR[0]{4}		
REAL					
Client/Server Data Type **Advanced**	Float	Float	Float Array		*See **Advanced Addressing REAL***
Client/Server Tag Example	REALTAG	REALARR[0]	REALARR[0]{4}		
USINT					

Atomic Data Type	Standard	Array Element	Array with or without Offset	Bit	String
Client/Server Data Type Advanced	Byte	Byte	Byte Array	Boolean (Bit w/i USINT)	*See Advanced Addressing USINT*
Client/Server Data Type	USINTTAG	USINTTARR[0]	USINTTARR[0]{4}	USINTTAG.0	
UINT					
Client / Server Data Type Advanced	Word, BCD	Word, BCD	Word Array, BCD Array	Boolean (Bit w/i UINT	*See Advanced Addressing UINT*
Client / Server Tag Example	UINTTAG	UINTARR[0]	UINTARR[0]{4}	UINTTAG.0	
UDINT					
Client / Server Data Type Advanced	DWord, LBCD	DWord, LBCD	DWord Array, LBCD Array	Boolean	*See Advanced Addressing UDINT*
Client / Server Tag Example	UDINTTAG	UDINTARR[0]	UDINTARR[0]{4}	UDINTAG.0	
ULINT					
Client / Server Data Type Advanced	Double	Double	Double Array		*See Advanced Addressing ULINT*
Client / Server Tag Example	ULINTTAG	ULINTARR[0]	ULINTARR[0]{4}		
LREAL					
Client / Server Data Type Advanced	Double	Double	Double Array		*See Advanced Addressing LREAL*
Client / Server Tag Example	LREALTAG	LREALARR[0]	LREALARR[0]{4}		
TIME32					
Client / Server Data Type Advanced	**String**, Long	**String**, Long	Long Array		*See Advanced Addressing TIME32*
Client/Server Tag Example	TIME32TAG	TIME32ARR[0]	TIME32ARR[0]{4}		
TIME					
Client / Server Data Type Advanced	**String**, LLong	**String**, LLong	LLong Array		*See Advanced Addressing TIME*
Client / Server Tag Example	TIMETAG	TIMEARR[0]	TIMEARR[0]{4}		

Atomic Data Type	Standard	Array Element	Array with or without Offset	Bit	String
LTIME					
Client / Server Data Type **Advanced**	**String**, LLong	**String**, LLong	LLong Array		*See **Advanced Addressing LTIME***
Client / Server Tag Example	LTIMETAG	LTIMEARR[0]	LTIMEARR[0]{4}		

Addressing Structure Data Types

Only the atomic structure members can be addressed at the structure level. For more information, refer to the examples below.

Logix Tag
MyTimer @ TIMER

Client/Server Tag
1. Invalid

TimerTag address = MyTimer
TimerTag data type = ??

2. Valid
TimerTag address = MyTimer.ACC
TimerTag data type = DWord

Addressing STRING Data Type

STRING is a pre-defined Logix data type whose structure contains two members: DATA and LEN. DATA is an array of SINTs and stores the characters of the STRING. LEN is a DINT and represents the number of characters in DATA to display to a client.
Note: String arrays are not supported.

Because LEN and DATA are atomic members, they must be referenced independently from a client/server. The syntax is as shown below.

Description	Syntax	Example
STRING Value	DATA/<Maximum STRING length >	MYSTRING.DATA/82
Actual STRING length	LEN	MYSTRING.LEN

Reads

The STRING read from DATA is terminated by the following:

a. The first null terminator encountered.

b. The value in LEN if a) doesn't occur first.

c. The <Maximum STRING length > if either a) or b) doesn't occur first.

Example
MYSTRING.DATA contains "Hello World" in the PLC, but LEN is manually set to 5. A read of MYSTRING.DATA/82 displays "Hello". If LEN is set to 20, MYSTRING.DATA/82 displays "Hello World".

Writes

When a STRING value is written to DATA, the driver also writes to LEN with the length of DATA written. If the write to LEN fails for any reason, the write operation to DATA is considered failed as well (despite the fact that the DATA write to the controller succeeded).

Note: This behavior was designed specifically for Logix tags of type STRING or a custom derivative of it. The following precautions apply to users who wish to implement their own STRING in UDTs.

- If a UDT exists that has a DATA member referenced as a STRING and a LEN member referenced as a DINT, the write to LEN succeeds regardless of the intentions of LEN for the given UDT. Care must be taken when designing UDTs to avoid this possibility if LEN is not intended to be the length of DATA.
- If a UDT exists that has a DATA member referenced as a STRING but does not have a LEN member, the write to LEN fails silently without consequence to DATA.

Example
MYSTRING.DATA/82 holds the value "Hello World." MYSTRING.LEN holds 11. If the value "Alarm Triggered" is written to MYSTRING.DATA/82, 15 is written to MYSTRING.LEN. If the write to MYSTRING.LEN fails, MYSTRING.LEN holds its previous value of 11 while MYSTRING.DATA/82 displays the first 11 characters ("Alarm Trigg"). If the write to MYSTRING.DATA/82 fails, neither tag is affected.

Terminate String Data at LEN

In the logical addressing modes, reading STRING.DATA causes an automatic read of STRING.LEN in Symbolic Mode. This may be bypassed by disabling the Terminate String Data at LEN option. *For more information, refer to* ***Logix Options***.

Ordering of Logix Array Data

One-Dimensional Arrays - array [dim1]
One-dimensional array data is passed to and from the controller in ascending order.
for (dim1 = 0; dim1 < dim1_max; dim1++)

Example: 3 element array
array [0]
array [1]
array [2]

Two-Dimensional Arrays - array [dim1, dim2]
Two-dimensional array data is passed to and from the controller in ascending order.
for (dim1 = 0; dim1 < dim1_max; dim1++)
for (dim2 = 0; dim2 < dim2_max; dim2++)

Example: 3X3 element array
array [0, 0]
array [0, 1]
array [0, 2]
array [1, 0]
array [1, 1]
array [1, 2]
array [2, 0]
array [2, 1]
array [2, 2]

Three-Dimensional Arrays - array [dim1, dim2, dim3]
Three-dimensional array data is passed to and from the controller in ascending order.
for (dim1 = 0; dim1 < dim1_max; dim1++)
for (dim2 = 0; dim2 < dim2_max; dim2++)
for (dim3 = 0; dim3 < dim3_max; dim3++)

Example: 3X3x3 element array
array [0, 0, 0]
array [0, 0, 1]

array [0, 0, 2]
array [0, 1, 0]
array [0, 1, 1]
array [0, 1, 2]
array [0, 2, 0]
array [0, 2, 1]
array [0, 2, 2]
array [1, 0, 0]
array [1, 0, 1]
array [1, 0, 2]
array [1, 1, 0]
array [1, 1, 1]
array [1, 1, 2]
array [1, 2, 0]
array [1, 2, 1]
array [1, 2, 2]
array [2, 0, 0]
array [2, 0, 1]
array [2, 0, 2]
array [2, 1, 0]
array [2, 1, 1]
array [2, 1, 2]
array [2, 2, 0]
array [2, 2, 1]
array [2, 2, 2]

Logix Advanced Addressing

Advanced Addressing is available for the following atomic data types. Select a link from the list below for more information on a specific data type.

BOOL
SINT
INT
DINT
LINT
REAL
USINT
UINT
UDINT
ULINT
LREAL
TIME32
TIME
LTIME

Advanced Addressing: BOOL

Format	Supported Data Types	Notes
Standard	Boolean, Byte, Char, Word, Short, BCD, DWord, Long, LBCD, Float*	None
	Boolean	The Controller tag must be a one-dimensional array.
Array w/o Offset	Boolean Array	1. The Controller tag must be a one-dimensional array. 2. The number of elements must be a factor of 8.
Array w/o Offset	Byte Array, Char Array, Word Array, Short Array, BCD Array, DWord Array, Long Array, LBCD Array, Float Array*	Not supported.
Array w/ Offset	Boolean Array	1. The Controller tag must be a one-dimensional array. 2. The offset must lie on 32-bit boundary. 3. The number of elements must be a factor of 8.
Bit	Boolean	1. The Controller tag must be a one-dimensional array. 2. The range is limited from 0 to 31.
String	String	Not supported

*The float value equals the face value of the Controller tag in float form (non-IEEE floating-point number).

Examples

Examples highlighted signify common use cases.

BOOL Controller Tag - booltag = true

Server Tag Address	Format	Data Type	Notes
booltag	Standard	Boolean	Value = true
booltag	Standard	Byte	Value = 1
booltag	Standard	Word	Value = 1
booltag	Standard	DWord	Value = 1
booltag	Standard	Float	Value = 1.0
booltag [3]	Array Element	Boolean	Invalid: Tag not an array
booltag [3]	Array Element	Word	Invalid: Tag not an array
booltag {1}	Array w/o Offset	Word	Invalid: Not supported
booltag {1}	Array w/o Offset	Boolean	Invalid: Not supported
booltag [3] {32}	Array w/ Offset	Boolean	Invalid: Tag not an array
booltag . 3	Bit	Boolean	Invalid: Tag not an array
booltag / 1	String	String	Invalid: Not supported

Server Tag Address	Format	Data Type	Notes
booltag / 4	String	String	Invalid: Not supported

BOOL Array Controller Tag - bitarraytag = [0,1,0,1]

Server Tag Address	Format	Data Type	Notes
bitarraytag	Standard	Boolean	Invalid: Tag cannot be an array
bitarraytag	Standard	Byte	Invalid: Tag cannot be an array
bitarraytag	Standard	Word	Invalid: Tag cannot be an array
bitarraytag	Standard	DWord	Invalid: Tag cannot be an array
bitarraytag	Standard	Float	Invalid: Tag cannot be an array
bitarraytag [3]	Array Element	Boolean	Value = true
bitarraytag [3]	Array Element	Word	Invalid: Bad data type
bitarraytag {3}	Array w/o Offset	Word	Invalid: Tag cannot be an array
bitarraytag {1}	Array w/o Offset	Word	Invalid: Tag cannot be an array
bitarraytag {1}	Array w/o Offset	Boolean	Invalid: Array size must be a factor of 8
bitarraytag {32}	Array w/o Offset	Boolean	Value = [0,1,0,1,...]
bitarraytag [3] {32}	Array w/ Offset	Boolean	Offset must begin on 32-bit boundary
bitarraytag[0]{32}	Array w/ Offset	Boolean	Value = [0,1,0,1,...]
bitarraytag[32]{64}	Array w/ Offset	Boolean	Value = [...] *values not provided above*
bitarraytag . 3	Bit	Boolean	Value = true
bitarraytag / 1	String	String	Invalid: Not supported
bitarraytag / 4	String	String	Invalid: Not supported

Advanced Addressing: SINT

Format	Supported Data Types	Notes
Standard	Boolean*, Byte, Char, Word, Short, BCD, DWord, Long, LBCD, Float***	None
Array Element	Byte, Char, Word, Short, BCD, DWord, Long, LBCD, Float***	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within a SINT in array form. **Note:** This is not an array of SINTs in Boolean notation. 2. Applies to bit-within-SINT only. Example: tag_1.0{8}. 3. .bit + array size cannot exceed 8 bits. Example: tag_1.1{8} exceeds an SINT, tag_1.0{8} does not. 4. Array size must be a multiple of

Format	Supported Data Types	Notes
		eight (8).
Array w/o Offset	Byte Array, Char Array, Word Array, Short Array, BCD Array**, DWord Array, Long Array, LBCD Array**, Float Array**,***	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Byte Array, Char Array, Word Array, Short Array, BCD Array**, DWord Array, Long Array, LBCD Array**, Float Array**,***	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 7. 2. If the Controller tag is an array, the bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0.
String	String	1. If accessing a single element, the Controller tag need not be an array. **Note:** The value of the string is the ASCII equivalent of the SINT value. Example: SINT = 65 dec = "A". 2. If accessing more than a single element, the Controller tag must be an array. The value of the string is the null-terminated ASCII equivalent of all the SINTs in the string. 1 character in string = 1 SINT.

*non-zero values are clamped to true.
**Each element of the array corresponds to an element in the SINT array. Arrays are not packed.
*** Float value equals the face value of Controller tag in float form (non-IEEE floating-point number).

Examples

Examples highlighted signify common use cases.

SINT Controller Tag - sinttag = 122 (decimal)

Server Tag Address	Format	Data Type	Notes
sinttag	Standard	Boolean	Value = true
sinttag	Standard	Byte	Value = 122
sinttag	Standard	Word	Value = 122
sinttag	Standard	DWord	Value = 122
sinttag	Standard	Float	Value = 122.0
sinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Also, Boolean is

Server Tag Address	Format	Data Type	Notes
			invalid.
sinttag [3]	Array Element	Byte	Invalid: Tag not an array
sinttag {3}	Array w/o Offset	Byte	Invalid: Tag not an array
sinttag {1}	Array w/o Offset	Byte	Value = [122]
sinttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
sinttag [3] {1}	Array w/ Offset	Byte	Invalid: Tag not an array
sinttag . 3	Bit	Boolean	Value = true
sinttag . 0 {8}	Array w/o Offset	Boolean	Value = [0,1,0,1,1,1,1,0] Bit value of 122
sinttag / 1	String	String	Value = "z"
sinttag / 4	String	String	Invalid: Tag not an array

SINT Array Controller Tag - sintarraytag [4,4] = [[83,73,78,84],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

Server Tag Address	Format	Data Type	Notes
sintarraytag	Standard	Boolean	Invalid: Tag cannot be an array
sintarraytag	Standard	Byte	Invalid: Tag cannot be an array
sintarraytag	Standard	Word	Invalid: Tag cannot be an array
sintarraytag	Standard	DWord	Invalid: Tag cannot be an array
sintarraytag	Standard	Float	Invalid: Tag cannot be an array
sintarraytag [3]	Array Element	Byte	Invalid: Server tag missing dimension 2 address
sintarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements
sintarraytag [1,3]	Array Element	Byte	Value = 8
sintarraytag {10}	Array w/o Offset	Byte	Value = [83,73,78,84,5,6,7,8,9,10]
sintarraytag {2} {5}	Array w/o Offset	Word	Value = [83,73,78,84,5] [6,7,8,9,10]
sintarraytag {1}	Array w/o Offset	Byte	Value = 83
sintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
sintarraytag [1,3] {4}	Array w/ Offset	Byte	Value = [8,9,10,11]
sintarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location
sintarraytag [1,3] . 3	Bit	Boolean	Value = 1
sintarraytag [1,3] . 0 {8}	Array w/o Offset	Boolean	Value = [0,0,0,1,0,0,0,0]
sintarraytag / 1	String	String	Value = "S"
sintarraytag / 4	String	String	Value = "SINT"

Advanced Addressing: INT

Format	Supported Data Types	Notes
Standard	Boolean*, Byte, Char**, Word, Short, BCD, DWord, Long, LBCD, Float****	None
Array Element	Byte, Char**, Word, Short, BCD, DWord, Long, LBCD, Float****	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within an INT in array form. **Note:** This is not an array of INTs in Boolean notation. 2. Applies to bit-within-INT only. Example: tag_1.0{16}. 3. .bit + array size cannot exceed 16 bits. Example: tag_1.1{16} exceeds an INT, tag_1.0{16} does not. 4. Array size must be a multiple of eight (8).
Array w/o Offset	Byte Array, Char Array**, Word Array, Short Array, BCD Array, DWord Array, Long Array, LBCD Array***Float Array***,****	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Byte Array, Char Array** Word Array, Short Array, BCD Array, DWord Array, Long Array, LBCD Array***, Float Array***,****	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 15. 2. If the Controller tag is an array, the bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0.
String	String	1. If accessing a single element, the Controller tag need not be an array. **Note:** The value of the string is the ASCII equivalent of the INT value (clamped to 255). Example: INT = 65 dec = "A". 2. If accessing more than a single element, the Controller tag must be an array. The value of the string is the null-terminated ASCII equivalent of all the INTs

Format	Supported Data Types	Notes
		(clamped to 255) in the string. 1 character in string = 1 INT, clamped to 255 INT strings are not packed. For greater efficiency, use SINT strings or the STRING structure instead.

*non-zero values are clamped to true.
**Values exceeding 255 are clamped to 255.
***Each element of the array corresponds to an element in the INT array. Arrays are not packed.
****Float value equals the face value of Controller tag in float form (non-IEEE floating-point number).

Examples

Examples highlighted signify common use cases.

INT Controller Tag - inttag = 65534 (decimal)

Server Tag Address	Class	Data Type	Notes
inttag	Standard	Boolean	Value = true
inttag	Standard	Byte	Value = 255
inttag	Standard	Word	Value = 65534
inttag	Standard	DWord	Value = 65534
inttag	Standard	Float	Value = 65534.0
inttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
inttag [3]	Array Element	Word	Invalid: Tag not an array.
inttag {3}	Array w/o Offset	Word	Invalid: Tag not an array.
inttag {1}	Array w/o Offset	Word	Value = [65534]
inttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
inttag [3] {1}	Array w/ Offset	Word	Invalid: Tag not an array.
inttag . 3	Bit	Boolean	Value = true
inttag . 0 {16}	Array w/o Offset	Boolean	Value = [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] Bit value of 65534
inttag / 1	String	String	Value = unprintable character = 255 decimal.
inttag / 4	String	String	Invalid: Tag not an array.

INT Array Controller Tag - intarraytag [4,4] = [[73,78,84,255],[256,257,258,259],[9,10,11,12], [13,14,15,16]]

Server Tag Address	Class	Data Type	Notes
intarraytag	Standard	Boolean	Invalid: Tag cannot be an array.
intarraytag	Standard	Byte	Invalid: Tag cannot be an array.

Server Tag Address	Class	Data Type	Notes
intarraytag	Standard	Word	Invalid: Tag cannot be an array.
intarraytag	Standard	DWord	Invalid: Tag cannot be an array.
intarraytag	Standard	Float	Invalid: Tag cannot be an array.
intarraytag [3]	Array Element	Word	Invalid: Server tag missing dimension 2 address.
intarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
intarraytag [1,3]	Array Element	Word	Value = 259
intarraytag {10}	Array w/o Offset	Byte	Value = [73,78,84,255,255,255,255,255,9,10]
intarraytag {2} {5}	Array w/o Offset	Word	Value = [73,78,84,255,256] [257,258,259,9,10]
intarraytag {1}	Array w/o Offset	Word	Value = 73
intarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
intarraytag [1,3] {4}	Array w/ Offset	Word	Value = [259,9,10,11]
intarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location.
intarraytag [1,3] . 3	Bit	Boolean	Value = 0
intarraytag [1,3] . 0 {16}	Array w/o Offset	Boolean	Value = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit value for 259
intarraytag / 1	String	String	Value = "I"
intarraytag / 3	String	String	Value = "INT"

Advanced Addressing: DINT

Format	Supported Data Types	Notes
Standard	Boolean*, Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float ****	None
Array Element	Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float ****	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within a DINT in array form. **Note:** This is not an array of DINTs in Boolean notation. 2. Applies to bit-within-DINT only. Example: tag_1.0{32}. 3. .bit + array size cannot exceed 32 bits. Example: tag_1.1{32} exceeds a DINT, tag_1.0{32} does not.

Format	Supported Data Types	Notes
		4. Array size must be a multiple of eight (8).
Array w/o Offset	Byte Array, Char Array**, Word Array, Short Array, BCD Array***, DWord Array, Long Array, LBCD Array, Float Array ****	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Byte Array, Char Array**, Word Array, Short Array, BCD Array***, DWord Array, Long Array, LBCD Array, Float Array ****	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 31. 2. If Controller tag is an array, bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0.
String	String	1. If accessing a single element, the Controller tag need not be an array. **Note:** The value of the string is the ASCII equivalent of the DINT value (clamped to 255). Example: SINT = 65dec = "A". 2. If accessing more than a single element, the Controller tag must be an array. The value of the string is the null-terminated ASCII equivalent of all the DINTs (clamped to 255) in the string. 1 character in string = 1 DINT, clamped to 255. **Note:** DINT strings are not packed. For greater efficiency, use SINT strings or the STRING structure instead.

*non-zero values are clamped to true.
**Values exceeding 255 are clamped to 255.
***Values exceeding 65535 are clamped to 65535.
****Float value equals the face value of Controller tag in float form (non-IEEE floating-point number).

Examples

Examples highlighted signify common use cases.

DINT Controller Tag - dinttag = 70000 (decimal)

Server Tag Address	Format	Data Type	Notes
dinttag	Standard	Boolean	Value = true
dinttag	Standard	Byte	Value = 255
dinttag	Standard	Word	Value = 65535
dinttag	Standard	DWord	Value = 70000
dinttag	Standard	Float	Value = 70000.0
dinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
dinttag [3]	Array Element	DWord	Invalid: Tag not an array.
dinttag {3}	Array w/o Offset	DWord	Invalid: Tag not an array.
dinttag {1}	Array w/o Offset	DWord	Value = [70000]
dinttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
dintag [3] {1}	Array w/ Offset	DWord	Invalid: Tag not an array.
dinttag . 3	Bit	Boolean	Value = false
dinttag . 0 {32}	Array w/o Offset	Boolean	Value = [0,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,...0] Bit value for 70000
dinttag / 1	String	String	Value = unprintable character = 255 decimal
dinttag / 4	String	String	Invalid: Tag not an array.

DINT Array Controller Tag - dintarraytag [4,4] = [[68,73,78,84],[256,257,258,259],[9,10,11,12], [13,14,15,16]]

Server Tag Address	Format	Data Type	Notes
dintarraytag	Standard	Boolean	Invalid: Tag cannot be an array.
dintarraytag	Standard	Byte	Invalid: Tag cannot be an array.
dintarraytag	Standard	Word	Invalid: Tag cannot be an array.
dintarraytag	Standard	DWord	Invalid: Tag cannot be an array.
dintarraytag	Standard	Float	Invalid: Tag cannot be an array.
dintarraytag [3]	Array Element	DWord	Invalid: Server tag missing dimension 2 address.
dintarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
dintarraytag [1,3]	Array Element	DWord	Value = 259
dintarraytag {10}	Array w/o Offset	Byte	Value = [68,73,78,84,255,255,255,255,9,10]
dintarraytag {2}{5}	Array w/o Offset	DWord	Value = [68,73,78,84,256] [257,258,259,9,10]
dintarraytag {1}	Array w/o Offset	DWord	Value = 68
dintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
dintarraytag [1,3]	Array w/ Offset	DWord	Value = [259,9,10,11]

Server Tag Address	Format	Data Type	Notes
{4}			
dintarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location.
dintarraytag [1,3] . 3	Bit	Boolean	Value = 0
dintarraytag [1,3] .0 {32}	Array w/o Offset	Boolean	Value = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit value for 259
dintarraytag / 1	String	String	Value = "D"
dintarraytag / 3	String	String	Value = "DINT"

Advanced Addressing: LINT

Format	Supported Data Types	Notes
Standard	Double*, Date**	None
Array Element	Double*, Date**	The Controller tag must be an array.
Array w/o Offset	Double, Array*	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Double, Array*	The Controller tag must be an array.
Bit	N/A	Not supported.
String	N/A	Not supported.

*Double value equals face value of Controller tag in float form (non-IEEE floating-point number).
**Date values are in universal time (UTC), not localized time.

Examples

Examples highlighted signify common use cases.

LINT Controller Tag - linttag = 2007-01-01T16:46:40.000 (date) == 1.16767E+15 (decimal)

Server Tag Address	Format	Data Type	Notes
linttag	Standard	Boolean	Invalid: Boolean not supported.
linttag	Standard	Byte	Invalid: Byte not supported.
linttag	Standard	Word	Invalid: Word not supported.
linttag	Standard	Double	Value = 1.16767E+15
linttag	Standard	Date	Value = 2007-01-01T16:46:40.000*
linttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
linttag [3]	Array Element	Double	Invalid: Tag not an array.
linttag {3}	Array w/o Offset	Double	Invalid: Tag not an array.
linttag {1}	Array w/o Offset	Double	Value = [1.16767E+15]
linttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.

Server Tag Address	Format	Data Type	Notes
lintag [3] {1}	Array w/ Offset	Double	Invalid: Tag not an array.
linttag . 3	Bit	Boolean	Invalid: Syntax/data type not supported.
linttag / 1	String	String	Invalid: Syntax/data type not supported.

*Date values are in universal time (UTC), not localized time.

LINT Array Controller Tag -
lintarraytag [2,2] = [0, 1.16767E+15],[9.4666E+14, 9.46746E+14] where:
1.16767E+15 == 2007-01-01T16:46:40.000 (date)
9.4666E+14 == 1999-12-31T17:06:40.000
9.46746E+14 == 2000-01-1T17:00:00.000
0 == 1970-01-01T00:00:00.000

Server Tag Address	Format	Data Type	Notes
lintarraytag	Standard	Boolean	Invalid: Boolean not supported.
lintarraytag	Standard	Byte	Invalid: Byte not supported.
lintarraytag	Standard	Word	Invalid: Word not supported.
lintarraytag	Standard	Double	Invalid: Tag cannot be an array.
lintarraytag	Standard	Date	Invalid: Tag cannot be an array.
lintarraytag [1]	Array Element	Double	Invalid: Server tag missing dimension 2 address.
lintarraytag [1,1]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
lintarraytag [1,1]	Array Element	Double	Value = 9.46746E+14
lintarraytag [1,1]	Array Element	Date	Value = 2000-01-01T17:00:00.000*
lintarraytag {4}	Array w/o Offset	Double	Value = [0, 1.16767E+15, 9.4666E+14, 9.46746E+14]
lintarraytag {2} {2}	Array w/o Offset	Double	Value = [0, 1.16767E+15][9.4666E+14, 9.46746E+14]
lintarraytag {4}	Array w/o Offset	Date	Invalid: Date array not supported.
lintarraytag {1}	Array w/o Offset	Double	Value = 0
lintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
lintarraytag [0,1] {2}	Array w/ Offset	Double	Value = [1.16767E+15, 9.4666E+14]
lintarraytag . 3	Bit	Boolean	Invalid: Syntax/data type not supported.
lintarraytag / 1	String	String	Invalid: Syntax/data type not supported.

*Date values are in universal time (UTC), not localized time.

Advanced Addressing: REAL

Format	Supported Data Types	Notes
Standard	Boolean*, Byte, Char**, Word, Short,	None

Format	Supported Data Types	Notes
	BCD***, DWord, Long, LBCD, Float****	
Array Element	Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float****	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within a REAL in array form. **Note:** This is not an array of REALs in Boolean notation. 2. Applies to bit-within-REAL only. Example: tag_1.0{32}. 3. .bit + array size cannot exceed 32 bits. Example: tag_1.1{32} exceeds an REAL, tag_1.0{32} does not. 4. Array size must be a multiple of eight (8).
Array w/o Offset	Byte Array, Char Array**, Word Array, Short Array, BCD Array***, DWord Array, Long Array, LBCD Array, Float Array****	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Byte Array, Char Array**, Word Array, Short Array, BCD Array***, DWord Array, Long Array, LBCD Array, Float Array****	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 31. 2. If the Controller tag is an array, the bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0. **Note:** Float is casted to a DWord to allow referencing of bits.
String	String	1. If accessing a single element, the Controller tag need not be an array. **Note:** The value of the string is the ASCII equivalent of the REAL value (clamped to 255). Example: SINT = 65 dec = "A". 2. If accessing more than a single element, the Controller tag must be an array. The value of the string is the null-terminated ASCII equivalent of all the REALs (clamped to 255) in the string. 1

Format	Supported Data Types	Notes
		character in string = 1 REAL, clamped to 255. **Note:** REAL strings are not packed. For greater efficiency, use SINT strings or the STRING structure instead.

*non-zero values are clamped to true.
**Values exceeding 255 are clamped to 255.
***Values exceeding 65535 are clamped to 65535.
****Float value is a valid IEEE single precision floating point number.

Examples

Examples highlighted signify common use cases.

REAL Controller Tag - realtag = 512.5 (decimal)

Server Tag Address	Format	Data Type	Notes
realtag	Standard	Boolean	Value = true
realtag	Standard	Byte	Value = 255
realtag	Standard	Word	Value = 512
realtag	Standard	DWord	Value = 512
realtag	Standard	Float	Value = 512.5
realtag [3]	Array Element	Boolean	Invalid: Tag not an array. Also, Boolean is invalid.
realtag [3]	Array Element	DWord	Invalid: Tag not an array
realtag {3}	Array w/o Offset	DWord	Invalid: Tag not an array
realtag {1}	Array w/o Offset	Float	Value = [512.5]
realtag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
realtag [3] {1}	Array w/ Offset	Float	Invalid: Tag not an array
realtag . 3	Bit	Boolean	Value = true
realtag . 0 {32}	Array w/o Offset	Boolean	Value = [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,...0] Bit value for 512
realtag / 1	String	String	Value = unprintable character = 255 decimal
realtag / 4	String	String	Invalid: Tag not an array.

REAL Array Controller Tag - realarraytag [4,4] = [[82.1,69.2,65.3,76.4],[256.5,257.6,258.7,259.8], [9.0,10.0,11.0,12.0],[13.0,14.0,15.0,16.0]]

Server Tag Address	Format	Data Type	Notes
realarraytag	Standard	Boolean	Invalid: Tag cannot be an array.
realarraytag	Standard	Byte	Invalid: Tag cannot be an array.

Server Tag Address	Format	Data Type	Notes
realarraytag	Standard	Word	Invalid: Tag cannot be an array.
realarraytag	Standard	DWord	Invalid: Tag cannot be an array.
realarraytag	Standard	Float	Invalid: Tag cannot be an array.
realarraytag [3]	Array Element	Float	Invalid: Server tag missing dimension 2 address.
realarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
realarraytag [1,3]	Array Element	Float	Value = 259.8
realarraytag {10}	Array w/o Offset	Byte	Value = [82,69,65,76,255,255,255,255,9,10]
realarraytag {2} {5}	Array w/o Offset	Float	Value = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
realarraytag {1}	Array w/o Offset	Float	Value = 82.1
realarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
realarraytag [1,3] {4}	Array w/ Offset	Float	Value = [259.8,9.0,10.0,11.0]
realarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location.
realarraytag [1,3] . 3	Bit	Boolean	Value = 0
realarraytag [1,3] . 0 {32}	Array w/o Offset	Boolean	Value = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit value for 259
realarraytag / 1	String	String	Value = "R"
realarraytag / 3	String	String	Value = "REAL"

Advanced Addressing: USINT

Format	Supported Data Types	Notes
Standard	Byte	None
Array Element	Byte	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within an USINT in array form. **Note:** This is not an array of USINTs in Boolean notation. 2. Applies to bit-within-USINT only. Example: tag_1.0{8}. 3. .bit + array size cannot exceed 8 bits. Example: tag_1.1{8} exceeds an USINT, tag_1.0{8} does not.
Array w/o Offset	Byte Array	If accessing more than a single element, the Controller tag must be an array.

Format	Supported Data Types	Notes
Array w/ Offset	Byte Array	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 7. 2. If the Controller tag is an array, the bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0.
String	N/A	Not Supported

Examples

Examples highlighted signify common use cases.

USINT Controller Tag - usinttag = 122 (decimal)

Server Tag Address	Format	Data Type	Notes
usinttag	Standard	Boolean	Value = true
usinttag	Standard	Byte	Value = 122
usinttag	Standard	Word	Invalid: Word not supported
usinttag	Standard	DWord	Invalid: DWord not supported
usinttag	Standard	Float	Invalid: Float not supported
usinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Also, Boolean is invalid
usinttag [3]	Array Element	Byte	Invalid: Tag not an array
usinttag {3}	Array w/o Offset	Byte	Invalid: Tag not an array
usinttag {1}	Array w/o Offset	Byte	Value = [122]
usinttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
usinttag [3] {1}	Array w/ Offset	Byte	Invalid: Tag not an array
usinttag . 3	Bit	Boolean	Value = true
usinttag . 0 {8}	Array w/o Offset	Boolean	Value = [0,1,0,1,1,1,1,0] Bit value of 122
usinttag / 1	String	String	Value = "z"
usinttag / 4	String	String	Invalid: Tag not an array

USINT Array Controller Tag - usintarraytag [4,4] = [[83,73,78,84],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

Server Tag Address	Format	Data Type	Notes
usintarraytag	Standard	Boolean	Invalid: Tag cannot be an array
usintarraytag	Standard	Byte	Invalid: Tag cannot be an array
usintarraytag	Standard	Word	Invalid: Tag cannot be an array
usintarraytag	Standard	DWord	Invalid: Tag cannot be an array
usintarraytag	Standard	Float	Invalid: Tag cannot be an array
usintarraytag [3]	Array Element	Byte	Invalid: Server tag missing dimension 2

Server Tag Address	Format	Data Type	Notes
			address
usintarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements
usintarraytag [1,3]	Array Element	Byte	Value = 8
usintarraytag {10}	Array w/o Offset	Byte	Value = [83,73,78,84,5,6,7,8,9,10]
usintarraytag {2} {5}	Array w/o Offset	Word	Value = [83,73,78,84,5] [6,7,8,9,10]
usintarraytag {1}	Array w/o Offset	Byte	Value = 83
usintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type
usintarraytag [1,3] {4}	Array w/ Offset	Byte	Value = [8,9,10,11]
usintarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location
usintarraytag [1,3] . 3	Bit	Boolean	Value = 1
usintarraytag [1,3] . 0 {8}	Array w/o Offset	Boolean	Value = [0,0,0,1,0,0,0,0]
usintarraytag / 1	String	String	Invalid: Syntax / data type not supported
usintarraytag / 4	String	String	Invalid: Syntax / data type not supported

Advanced Addressing: UINT

Format	Supported Data Types	Notes
Standard	Word, BCD	None
Array Element	Word, BCD	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within a UINT in array form. **Note:** This is not an array of UINTs in Boolean notation. 2. Applies to bit-within-UINT only. Example: tag_1.0{16}. 3. .bit + array size cannot exceed 8 bits. Example: tag_1.1{16} exceeds an UINT, tag_1.0{16} does not. 4. Array size must be a multiple of eight (8).
Array w/o Offset	Word Array, BCD Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Word Array, BCD Array	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 15. 2. If the Controller tag is an array, the bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0.
String	N/A	Not Supported

Examples

Examples highlighted signify common use cases.

UINT Controller Tag - uinttag = 65534 (decimal)

Server Tag Address	Class	Data Type	Notes
uinttag	Standard	Boolean	Invalid: Boolean not supported.
uinttag	Standard	Byte	Invalid: Byte not supported.
uinttag	Standard	Word	Value = 65534
uinttag	Standard	DWord	Invalid: DWord not supported.
uinttag	Standard	Float	Invalid: Float not supported.
uinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
uinttag [3]	Array Element	Byte	Invalid: Tag not an array.
uinttag {3}	Array w/o Offset	Byte	Invalid: Tag not an array.

Server Tag Address	Class	Data Type	Notes
uinttag {1}	Array w/o Offset	Byte	Value = [65534]
uinttag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
uinttag [3] {1}	Array w/ Offset	Word	Invalid: Tag not an array.
uinttag . 3	Bit	Boolean	Value = true
uinttag . 0 {16}	Array w/o Offset	Boolean	Value = [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] Bit value of 65534
uinttag / 1	String	String	Invalid: Syntax / data type not supported.
uinttag / 4	String	String	Invalid: Syntax / data type not supported.

UINT Array Controller Tag - uintarraytag [4,4] = [[73,78,84,255],[256,257,258,259],[9,10,11,12], [13,14,15,16]]

Server Tag Address	Format	Data Type	Notes
uintarraytag	Standard	Boolean	Invalid: Tag cannot be an array.
uintarraytag	Standard	Byte	Invalid: Tag cannot be an array.
uintarraytag	Standard	Word	Invalid: Tag cannot be an array.
uintarraytag	Standard	DWord	Invalid: Tag cannot be an array.
uintarraytag	Standard	Float	Invalid: Tag cannot be an array.
uintarraytag [3]	Array Element	Word	Invalid: Server tag missing dimension 2 address.
uintarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
uintarraytag [1,3]	Array Element	Word	Value = 259
uintarraytag {10}	Array w/o Offset	BCD	Value = [49,54,54,165,100,101,102,103,9,10]
uintarraytag {2} {5}	Array w/o Offset	Word	Value = [73,78,84,255,256] [257,258,259,9,10]
uintarraytag {1}	Array w/o Offset	Word	Value = 73
uintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
uintarraytag [1,3] {4}	Array w/ Offset	Byte	Value = [259,9,10,11]
uintarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location.
uintarraytag [1,3] . 3	Bit	Boolean	Value = 0
uintarraytag [1,3] . 0 {16}	Array w/o Offset	Boolean	Value = [0,0,0,1,0,0,0,0]Value = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit value for 259
uintarraytag / 1	String	String	Invalid: Syntax / data type not supported.
uintarraytag / 3	String	String	Invalid: Syntax / data type not supported.

Advanced Addressing: UDINT

Format	Supported Data Types	Notes
Standard	DWord, LBCD	None
Array Element	DWord, LBCD	The Controller tag must be an array.
Array w/o Offset	Boolean Array	1. Use this case to have the bits within an UDINT in array form. **Note**: This is not an array of UDINTs in Boolean notation. 2. Applies to bit-within-UDINT only. Example: tag_1.0{32}. 3. .bit + array size cannot exceed 32 bits. Example: tag_1.1{32} exceeds an UINT, tag_1.0{32} does not. 4. Array size must be a multiple of (eight) 8.
Array w/o Offset	DWord Array, LBCD Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	DWord Array, LBCD Array	The Controller tag must be an array.
Bit	Boolean	1. The range is limited from 0 to 31. 2. If Controller tag is an array, bit class reference must be prefixed by an array element class reference. Example: tag_1 [2,2,3].0
String	N/A	Not supported

Examples

Examples highlighted signify common use cases.

UDINT Controller Tag - udinttag = 70000 (decimal)

Server Tag Address	Format	Data Type	Notes
udinttag	Standard	Boolean	Invalid: Boolean not supported
udinttag	Standard	Byte	Invalid: Byte not supported
udinttag	Standard	Word	Invalid: Word not supported
udinttag	Standard	DWord	Value = 70000
udinttag	Standard	LBCD	Value = 11170
udinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
udinttag [3]	Array Element	DWord	Invalid: Tag not an array
udinttag {3}	Array w/o Offset	DWord	Invalid: Tag not an array

Server Tag Address	Format	Data Type	Notes
udinttag {1}	Array w/o Offset	DWord	Value = [70000]
udinttag {1}	Array w/o Offset	Boolean	Invalid: Boolean Array not supported
udintag [3] {1}	Array w/ Offset	DWord	Invalid: Tag not an array
udinttag . 3	Bit	Boolean	Value = False
udinttag . 0 {32}	Array w/o Offset	Boolean	Value = [0,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,...0] Bit value for 70000
udinttag / 1	String	String	Invalid: Syntax/data type not supported
udinttag / 4	String	String	Invalid: Syntax/data type not supported

UDINT Array Controller Tag - udintarraytag [4,4] = [[68,73,78,84],[256,257,258,259],[9,10,11,12], [13,14,15,16]]

Server Tag Address	Format	Data Type	Notes
udintarraytag	Standard	Boolean	Invalid: Boolean not supported
udintarraytag	Standard	Byte	Invalid: Byte not supported
udintarraytag	Standard	Word	Invalid: Word not supported
udintarraytag	Standard	DWord	Invalid: Tag cannot be an array
udintarraytag	Standard	Float	Invalid: Float not supported
udintarraytag [3]	Array Element	DWord	Invalid: Server tag missing dimension 2 address.
udintarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements
udintarraytag [1,3]	Array Element	DWord	Value = 259
udintarraytag {10}	Array w/o Offset	LCBD	Value = [44,49,54,54,100,101,102,103,9,10]]
udintarraytag {2} {5}	Array w/o Offset	DWord	Value = [68,73,78,84,256] [257,258,259,9,10]
udintarraytag {1}	Array w/o Offset	DWord	Value = 68
udintarraytag {1}	Array w/o Offset	Boolean	Invalid: Bad data type.
udintarraytag [1,3] {4}	Array w/ Offset	DWord	Value = [259,9,10,11]
udintarraytag . 3	Bit	Boolean	Invalid: Tag must reference atomic location.
udintarraytag [1,3] . 3	Bit	Boolean	Value = False
udintarraytag [1,3] .0 {32}	Array w/o Offset	Boolean	Value = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit value for 259
udintarraytag / 1	String	String	Invalid: Syntax/data type not supported
udintarraytag / 3	String	String	Invalid: Syntax/data type not supported

Advanced Addressing: ULINT

Format	Supported Data Types	Notes
Standard	Double*	None
Array Element	Double*	The Controller tag must be an array.
Array w/o Offset	Double, Array*	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Double, Array*	The Controller tag must be an array.
Bit	N/A	Not supported
String	N/A	Not supported

*Double value equals face value of Controller tag in float form (non-IEEE floating-point number).

Examples

Examples highlighted signify common use cases.

ULINT Controller Tag - ulinttag = 1.8446744073709560e+19 (decimal)

Server Tag Address	Format	Data Type	Notes
ulinttag	Standard	Boolean	Invalid: Boolean not supported
ulinttag	Standard	Byte	Invalid: Byte not supported
ulinttag	Standard	Word	Invalid: Word not supported
ulinttag	Standard	Double	Value = 1.8446744073709560e+19
ulinttag [3]	Array Element	Boolean	Invalid: Tag not an array. Boolean is invalid.
ulinttag [3]	Array Element	Double	Invalid: Tag not an array
ulinttag {3}	Array w/o Offset	Double	Invalid: Tag not an array
ulinttag {1}	Array w/o Offset	Double	Value = [1.8446744073709560e+19]
ulinttag {1}	Array w/o Offset	Boolean	Invalid: Boolean array not supported
ulintag [3] {1}	Array w/ Offset	Double	Invalid: Tag not an array
ulinttag . 3	Bit	Boolean	Invalid: Syntax/data type not supported
ulinttag / 1	String	String	Invalid: Syntax/data type not supported

ULINT Array Controller Tag - ulintarraytag [2,2] = [0, 1.16767E+15],[9.4666E+14, 1.8446744073709560e+19]

Server Tag Address	Format	Data Type	Notes
ulintarraytag	Standard	Boolean	Invalid: Boolean not supported
ulintarraytag	Standard	Byte	Invalid: Byte not supported
ulintarraytag	Standard	Word	Invalid: Word not supported
ulintarraytag	Standard	Double	Invalid: Tag cannot be an array
ulintarraytag	Standard	Date	Invalid: Date not supported
ulintarraytag [1]	Array Element	Double	Invalid: Server tag missing dimension 2 address.

Server Tag Address	Format	Data Type	Notes
ulintarraytag [1,1]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
ulintarraytag [1,1]	Array Element	Double	Value = 1.8446744073709560e+19
ulintarraytag {4}	Array w/o Offset	Double	Value = [0, 1.16767E+15, 9.4666E+14, 1.8446744073709560e+19]
ulintarraytag {2} {2}	Array w/o Offset	Double	Value =[0, 1.16767E+15][9.4666E+14, 1.8446744073709560e+19]
ulintarraytag {4}	Array w/o Offset	Date	Invalid: Date array not supported
ulintarraytag {1}	Array w/o Offset	Double	Value = 0
ulintarraytag {1}	Array w/o Offset	Boolean	Invalid: Boolean array not supported
ulintarraytag [0,1] {2}	Array w/ Offset	Double	Value = [1.16767E+15, 9.4666E+14]
ulintarraytag . 3	Bit	Boolean	Invalid: Syntax/data type not supported
ulintarraytag / 1	String	String	Invalid: Syntax/data type not supported

Advanced Addressing: LREAL

Format	Supported Data Types	Notes
Standard	Double	None
Array Element	Double	The Controller tag must be an array.
Array w/o Offset	Double Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Double Array	The Controller tag must be an array.
Bit	N/A	Not supported
String	N/A	Not supported

Examples

Examples highlighted signify common use cases.

LREAL Controller Tag - lrealtag = 1.7976931348623157E+308 (decimal)

Server Tag Address	Format	Data Type	Notes
lrealtag	Standard	Boolean	Invalid: Boolean not supported
lrealtag	Standard	Byte	Invalid: Byte not supported
lrealtag	Standard	Word	Invalid: Word not supported
lrealtag	Standard	Double	Value = 1.7976931348623157E+308
lrealtag [3]	Array Element	Boolean	Invalid: Tag not an array. Also, Boolean is invalid.
lrealtag {1}	Array w/o Offset	Double	Value = [1.7976931348623157E+308]
lrealtag {1}	Array w/o Offset	Boolean	Invalid: Boolean array not supported

Server Tag Address	Format	Data Type	Notes
lrealtag [3] {1}	Array w/ Offset	Double	Invalid: Tag not an array
lrealtag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
lrealtag . 0 {32}	Array w/o Offset	Boolean	Invalid: Syntax / data type not supported
lrealtag / 1	String	String	Invalid: Syntax / data type not supported
lrealtag / 4	String	String	Invalid: Syntax / data type not supported

LREAL Array Controller Tag - lrealarraytag [4,4] = [[82.1,69.2,65.3,76.4],[256.5,257.6,258.7,259.8], [9.0,10.0,11.0,12.0],[13.0,14.0,15.0,16.0]]

Server Tag Address	Format	Data Type	Notes
lrealarraytag	Standard	Boolean	Invalid: Tag cannot be an array.
lrealarraytag	Standard	Byte	Invalid: Tag cannot be an array.
lrealarraytag	Standard	Word	Invalid: Tag cannot be an array.
lrealarraytag	Standard	Double	Invalid: Tag cannot be an array.
lrealarraytag [3]	Array Element	Double	Invalid: Server tag missing dimension 2 address.
lrealarraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
lrealarraytag [1,1]	Array Element	Double	Value = 257.6
lrealarraytag {2} {5}	Array w/o Offset	Double	Value = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
lrealarraytag {1}	Array w/o Offset	Double	Value = 82.1
lrealarraytag {1}	Array w/o Offset	Boolean	Invalid: Boolean Array not supported
lrealarraytag [1,3] {4}	Array w/ Offset	Double	Value = [259.8,9.0,10.0,11.0]
lrealarraytag . 3	Bit	Boolean	Invalid: Syntax / data type not supported.
lrealarraytag / 1	String	String	Invalid: Syntax / data type not supported.

Advanced Addressing: TIME32

Format	Supported Data Types	Notes
Standard	**String**, Long	None
Array Element	**String**, Long	The Controller tag must be an array.
Array w/o Offset	Long Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	Long Array	The Controller tag must be an array.
Bit	N/A	Not supported
String	N/A	String with length specifier is not supported.

Tip: Data type in **bold** represents the default data type.

Examples

Examples highlighted signify common use cases.

TIME32 Controller Tag - time32tag = -2147483647 (decimal)

Server Tag Address	Format	Data Type	Notes
time32tag	Standard	String	Value = T32#-35m_47s_483ms_647us
time32tag	Standard	Long	-2147483647
time32tag	Standard	DWord	Invalid: DWord not supported
time32tag	Standard	Boolean	Invalid: Boolean not supported
time32tag [3]	Array Element	String	Invalid: Tag not an array
time32tag {1}	Array w/o Offset	Long Array	Value = [-2147483647]
time32tag {1}	Array w/o Offset	String Array	Invalid: String array not supported
time32tag [3] {1}	Array w/ Offset	Long Array	Invalid: Tag not an array
time32tag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
time32tag . 0 {32}	Array w/o Offset	Boolean Array	Invalid: Syntax / data type not supported
time32tag / 1	String	String	Invalid: String length not supported
time32tag / 4	String	String	Invalid: String length not supported

TIME32 Array Controller Tag - time32arraytag [4,4] = [[1,2,3,4],[500,600,700,800], [90000,100000,110000,120000],[13000000,14000000,15000000,16000000]] (decimal)

Server Tag Address	Format	Data Type	Notes
time32arraytag	Standard	String	T32#1us (only first element is read)
time32arraytag	Standard	Long	1 (only first element is read)
time32arraytag	Standard	Boolean	Invalid: Tag cannot be an array. Boolean not supported.
time32arraytag [1,1]	Array Element	String	T32#600us
time32arraytag [2,2]	Array Element	Long	Value = 110000
time32arraytag [3]	Array Element	Long	Invalid: Server tag missing dimension 2 address.
time32arraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
time32arraytag {2} {5}	Array w/o Offset	Long Array	Value = [1,2,3,4,500] [600,700,800,90000,100000]
time32arraytag {1}	Array w/o Offset	Long Array	Value = [1]
time32arraytag {1}	Array w/o Offset	String Array	Invalid: String Array not supported
time32arraytag [1,2] {3}	Array w/ Offset	Long Array	Value = [700,800,90000]

Server Tag Address	Format	Data Type	Notes
time32arraytag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
time32arraytag / 1	String	String	Invalid: String length not supported

Advanced Addressing: TIME

Format	Supported Data Types	Notes
Standard	**String**, LLong	None
Array Element	**String**, LLong	The Controller tag must be an array.
Array w/o Offset	LLong Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	LLong Array	The Controller tag must be an array.
Bit	N/A	Not supported
String	N/A	String with length specifier is not supported.

Tip: Data type in **bold** represents the default data type.

Examples

Examples highlighted signify common use cases.

TIME Controller Tag - timetag = -2725199999999 (decimal)

Server Tag Address	Format	Data Type	Notes
timetag	Standard	String	Value = T#-31d_12h_59m_59s_999ms_999us
timetag	Standard	LLong	-2725199999999
timetag	Standard	QWord	Invalid: QWord not supported
timetag	Standard	Boolean	Invalid: Boolean not supported
timetag [3]	Array Element	String	Invalid: Tag not an array
timetag {1}	Array w/o Offset	LLong Array	Value = [-2725199999999]
timetag {1}	Array w/o Offset	String Array	Invalid: String array not supported
timetag [3] {1}	Array w/ Offset	LLong Array	Invalid: Tag not an array
timetag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
timetag . 0 {32}	Array w/o Offset	Boolean Array	Invalid: Syntax / data type not supported
timetag / 1	String	String	Invalid: String length not supported
timetag / 4	String	String	Invalid: String length not supported

TIME Array Controller Tag - timearraytag [4,4] = [[1,2,3,4],[500,600,700,800], [90000,100000,110000,120000],[13000000,14000000,15000000,16000000]] (decimal)

Server Tag Address	Format	Data Type	Notes
timearraytag	Standard	String	T#1us (only first element is read)
timearraytag	Standard	LLong	1 (only first element is read)
timearraytag	Standard	Boolean	Invalid: Tag cannot be an array. Boolean not supported.
timearraytag [1,1]	Array Element	String	T#600us
timearraytag [2,2]	Array Element	LLong	Value = 110000
timearraytag [3]	Array Element	LLong	Invalid: Server tag missing dimension 2 address.
timearraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
timearraytag {2} {5}	Array w/o Offset	LLong Array	Value = [1,2,3,4,500] [600,700,800,90000,100000]
timearraytag {1}	Array w/o Offset	LLong Array	Value = [1]
timearraytag {1}	Array w/o Offset	String Array	Invalid: String Array not supported
timearraytag [1,3] {4}	Array w/ Offset	LLong Array	Value = [800,90000,100000,110000]
timearraytag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
timearraytag / 1	String	String	Invalid: String length not supported

Advanced Addressing: LTIME

Format	Supported Data Types	Notes
Standard	**String**, LLong	None
Array Element	**String**, LLong	The Controller tag must be an array.
Array w/o Offset	LLong Array	If accessing more than a single element, the Controller tag must be an array.
Array w/ Offset	LLong Array	The Controller tag must be an array.
Bit	N/A	Not supported
String	N/A	String with length specifier is not supported.

Tip: Data type in **bold** represents the default data type.

Examples

Examples highlighted signify common use cases.

LTIME Controller Tag - ltimetag = -2725199999999999 (decimal)

Server Tag Address	Format	Data Type	Notes
ltimetag	Standard	String	Value = LT#-31d_12h_59m_59s_999ms_999us_999ns

Server Tag Address	Format	Data Type	Notes
ltimetag	Standard	LLong	-2725199999999999
ltimetag	Standard	QWord	Invalid: QWord not supported
ltimetag	Standard	Boolean	Invalid: Boolean not supported
ltimetag [3]	Array Element	String	Invalid: Tag not an array
ltimetag {1}	Array w/o Offset	LLong Array	Value = [-2725199999999999]
ltimetag {1}	Array w/o Offset	String Array	Invalid: String array not supported
ltimetag [3] {1}	Array w/ Offset	LLong Array	Invalid: Tag not an array
ltimetag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
ltimetag . 0 {32}	Array w/o Offset	Boolean Array	Invalid: Syntax / data type not supported
ltimetag / 1	String	String	Invalid: String length not supported
ltimetag / 4	String	String	Invalid: String length not supported

LTIME Array Controller Tag - ltimearraytag [4,4] = [[1,2,3,4],[500,600,700,800], [90000,100000,110000,120000],[13000000,14000000,15000000,16000000]] (decimal)

Server Tag Address	Format	Data Type	Notes
ltimearraytag	Standard	String	T#1us (only first element is read)
ltimearraytag	Standard	LLong	1 (only first element is read)
ltimearraytag	Standard	Boolean	Invalid: Tag cannot be an array. Boolean not supported.
ltimearraytag [1,1]	Array Element	String	T#600ns
ltimearraytag [2,2]	Array Element	LLong	Value = 110000
ltimearraytag [3]	Array Element	LLong	Invalid: Server tag missing dimension 2 address.
ltimearraytag [1,3]	Array Element	Boolean	Invalid: Boolean not allowed for array elements.
ltimearraytag {2} {5}	Array w/o Offset	LLong Array	Value = [1,2,3,4,500] [600,700,800,90000,100000]
ltimearraytag {1}	Array w/o Offset	LLong Array	Value = [1]
ltimearraytag {1}	Array w/o Offset	String Array	Invalid: String Array not supported
ltimearraytag [2,2] {4}	Array w/ Offset	LLong Array	Value = [110000,120000,13000000,14000000]
ltimearraytag . 3	Bit	Boolean	Invalid: Syntax / data type not supported
ltimearraytag / 1	String	String	Invalid: String length not supported

File Listing

Select a link from the list below for information on a specific file supported by various device models.

Function File Listing

*For more information on device models and their supported files, refer to **Address Descriptions**.*

Output Files

The syntax for accessing data in the output file differs depending on the PLC model. Arrays are not supported for output files. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
O:<word>	Short, **Word**, BCD	Read/Write
O:<word>/<bit>	**Boolean**	Read/Write
O/bit	**Boolean**	Read/Write

Note : Word and bit address information is in octal for PLC-5 models. This follows the convention of the programming software.

MicroLogix Syntax

Syntax	Data Type	Access
O:<word>	Short, **Word**, BCD	Read/Write
O:<word>/<bit>	**Boolean**	Read/Write
O/bit	**Boolean**	Read/Write

MicroLogix models have two types of I/O: embedded I/O and expansion I/O (not applicable for MicroLogix 1000). Embedded I/O resides with the CPU base unit while Expansion I/O plugs into the CPU base unit. The table below lists the I/O capabilities of each MicroLogix model.

MicroLogix Model	Embedded I/O	Expansion I/O
1000	Slot 0	N/A
1100	Slot 0	Slots 1-4
1200	Slot 0	Slots 1-6
1400	Slot 0	Slots 1-7
1500	Slot 0	Slots 1-16

The address syntax for MicroLogix I/O references a zero-based word offset, not a slot. Users must determine the word offset to a particular slot. This requires knowledge of the modules and their respective size in words. The table below specifies the size of some available modules; however, it is recommended that users consult both the MicroLogix documentation and the controller project to determine the module's true word size.

MicroLogix Embedded I/O Word Sizes

MicroLogix Model	# Input Words	# Output Words
1000	2	1
1100	6	4
1200	4	4
1400	8	6
1500	4	4

MicroLogix Expansion I/O Word Sizes

Modules	# Input Words	# Output Words
1769-HSC	35	34
1769-IA8I	1	0
1769-IA16	1	0
1769-IF4	6	0
1769-IF4XOF2	8	2
1769-IF8	12	1
1769-IM12	1	0
1769-IQ16	1	0
1769-IQ6XOW4	1	1
1769-IQ16F	1	0

Modules	# Input Words	# Output Words
1769-IQ32	2	0
1769-IR6	8	0
1769-IT6	8	0
1769-OA8	0	1
1769-OA16	0	1
1769-OB8	0	1
1769-OB16	0	1
1769-OB16P	0	1
1769-OB32	0	2
1769-OF2	2	2
1769-OF8C	11	9
1769-OF8V	11	9
1769-OV16	0	1
1769-OW8	0	1
1769-OW16	0	1
1769-OW8I	0	1
1769-SDN	66	2
1769-SM1	12	12
1769-SM2	7	7
1769-ASCII	108	108
1762-IA8	1	0
1762-IF2OF2	6	2
1762-IF4	7	0
1762-IQ8	1	0
1762-IQ8OW6	1	1
1762-IQ16	1	0
1762-OA8	0	1
1762-OB8	0	1
1762-OB16	0	1
1762-OW8	0	1
1762-OW16	0	1
1762-IT4	6	0
1762-IR4	6	0
1762-OF4	2	4
1762-OX6I	0	1

Calculation

Output Word Offset for slot x = # Output Words in slot 0 through slot (x-1).

Notes:

1. The Embedded I/O needs to be taken into account when offsetting to Expansion I/O.
2. The number of Input words does not factor into the calculation for Output Word Offset.

I/O Example

Let
Slot 0 = MicroLogix 1500 LRP Series C = 4 Output Words
Slot 1 = 1769-OF2 = 2 Output Words
Slot 2 = 1769-OW8 = 1 Output Word
Slot 3 = 1769-IA16 = 0 Output Word
Slot 4 = 1769-OF8V = 9 Output Word
Bit 5 of Slot 4 = 4 + 2 + 1 = 7 words = O:7/5

SLC 500 Syntax

The default data types are shown in **bold**.

Syntax	Data Type	Access
O:<slot>	Short, **Word**, BCD	Read Only
O:<slot>.<word>	Short, **Word**, BCD	Read Only
O:<slot>/<bit>	**Boolean**	Read Only
O:<slot>.<word>/<bit>	**Boolean**	Read Only

Ranges

PLC Model	Min. Slot	Max. Slot	Max. Word
MicroLogix	N/A	N/A	2047
SLC 500 Fixed I/O	N/A	N/A	1
SLC 500 Modular I/O	1	30	*
PLC-5 Series	N/A	N/A	277 (octal)

*The number of Input or Output words available for each I/O module can be found in the **SLC 500 Modular I/O Selection Guide**.

Examples

MicroLogix	Description
O:0	word 0
O/2	bit 2
O:0/5	bit 5

SLC 500 Fixed I/O	Description
O:0	word 0
O:1	word 1
O/16	bit 16
O:1/0	bit 0 word 1 (same as O/16)

PLC5*	Description
O:0	word 0
O:37	word 31 (37 octal = 31 decimal)
O/42	bit 34 (42 octal = 34 decimal)
O:2/2	bit 2 word 2 (same as O/42)

*Addresses are in Octal.

SLC 500 Modular I/O	Description
O:1	word 0 slot 1
O:1.0	word 0 slot 1 (same as O:1)
O:12	word 0 slot 12
O:12.2	word 2 slot 12
O:4.0/0	bit 0 word 0 slot 4
O:4/0	bit 0 slot 4 (same as O:4.0/0)
O:4.2/0	bit 0 word 2 slot 4
O:4/32	bit 32 slot 4 (same as O:4.2/0)

Input Files

The syntax for accessing data in the input file differs depending on the PLC model. Arrays are not supported for input files. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
I:<word>	Short, **Word**, BCD	Read/Write
I:<word>/<bit>	**Boolean**	Read/Write
I/bit	**Boolean**	Read/Write

Note: Word and bit address information is in octal for PLC-5 models. This follows the convention of the programming software.

MicroLogix Syntax

Syntax	Data Type	Access
I:<word>	Short, **Word**, BCD	Read/Write
I:<word>/<bit>	**Boolean**	Read/Write
I/bit	**Boolean**	Read/Write

MicroLogix models have two types of I/O: embedded I/O and expansion I/O (not applicable for MicroLogix 1000). Embedded I/O resides with the CPU base unit while Expansion I/O plugs into the CPU base unit. The table below lists the I/O capabilities of each MicroLogix model.

MicroLogix Model	Embedded I/O	Expansion I/O
1000	Slot 0	N/A
1100	Slot 0	Slots 1-4
1200	Slot 0	Slots 1-6
1400	Slot 0	Slots 1-7
1500	Slot 0	Slots 1-16

The address syntax for MicroLogix I/O references a zero-based word offset, not a slot. Users must determine the word offset to a particular slot. This requires knowledge of the modules and their respective size in words. The table below specifies the size of some available modules; however, it is recommended that the MicroLogix documentation and controller project be consulted to determine a module's true word size.

MicroLogix Embedded I/O Word Sizes

MicroLogix Model	# Input Words	# Output Words
1000	2	1
1100	6	4
1200	4	4
1400	8	6
1500	4	4

MicroLogix Expansion I/O Word Sizes

Modules	# Input Words	# Output Words
1769-HSC	35	34
1769-IA8I	1	0
1769-IA16	1	0
1769-IF4	6	0
1769-IF4XOF2	8	2
1769-IF8	12	1
1769-IM12	1	0
1769-IQ16	1	0
1769-IQ6XOW4	1	1
1769-IQ16F	1	0
1769-IQ32	2	0
1769-IR6	8	0
1769-IT6	8	0
1769-OA8	0	1
1769-OA16	0	1
1769-OB8	0	1
1769-OB16	0	1
1769-OB16P	0	1

Modules	# Input Words	# Output Words
1769-OB32	0	2
1769-OF2	2	2
1769-OF8C	11	9
1769-OF8V	11	9
1769-OV16	0	1
1769-OW8	0	1
1769-OW16	0	1
1769-OW8I	0	1
1769-SDN	66	2
1769-SM1	12	12
1769-SM2	7	7
1769-ASCII	108	108
1762-IA8	1	0
1762-IF2OF2	6	2
1762-IF4	7	0
1762-IQ8	1	0
1762-IQ8OW6	1	1
1762-IQ16	1	0
1762-OA8	0	1
1762-OB8	0	1
1762-OB16	0	1
1762-OW8	0	1
1762-OW16	0	1
1762-IT4	6	0
1762-IR4	6	0
1762-OF4	2	4
1762-OX6I	0	1

Calculation

Input Word Offset for slot x = # Input Words in slot 0 through slot (x-1).

Notes:

1. The Embedded I/O needs to be taken into account when offsetting to Expansion I/O.
2. The number of Output words does not factor into the calculation for Input Word Offset.

I/O Example

Let
Slot 0 = MicroLogix 1500 LRP Series C = 4 Input Words
Slot 1 = 1769-OF2 = 2 Input Words
Slot 2 = 1769-OW8 = 0 Input Word
Slot 3 = 1769-IA16 = 1 Input Word

Slot 4 = 1769-OF8V = 11 Input Word
Bit 5 of Slot 3 = 4 + 2 = 6 words = I:6/5

SLC 500 Syntax

Syntax	Data Type	Access
I:<slot>	Short, **Word**, BCD	Read Only
I:<slot>.<word>	Short, **Word**, BCD	Read Only
I:<slot>/<bit>	**Boolean**	Read Only
I:<slot>.<word>/<bit>	**Boolean**	Read Only

Ranges

PLC Model	Min. Slot	Max. Slot	Max. Word
MicroLogix	N/A	N/A	2047
SLC 500 Fixed I/O	N/A	N/A	1
SLC 500 Modular I/O	1	30	*
PLC-5 Series	N/A	N/A	277 (octal)

*The number of Input or Output words available for each I/O module can be found in the **SLC 500 Modular I/O Selection Guide**.

Examples

MicroLogix	Description
I:0	Word 0
I/2	Bit 2
I:1/5	Bit 5 word 1

SLC 500 Fixed I/O	Description
I:0	Word 0
I:1	Word 1
I/16	bit 16
I:1/0	Bit 0 word 1 (same as I/16)

PLC5*	Description
I:0	Word 0
I:10	Word 8 (10 octal = 8 decimal)
I/20	Bit 16 (20 octal = 16 decimal)
I:1/0	Bit 0 word 1 (same as I/20)

*Addresses are in Octal.

SLC 500 Modular I/O	Description
I:1	Word 0 slot 1

SLC 500 Modular I/O	Description
I:1.0	Word 0 slot 1 (same as I:1)
I:12	Word 0 slot 12
I:12.2	Word 2 slot 12
I:4.0/0	Bit 0 word 0 slot 4
I:4/0	Bit 0 slot 4 (same as I:4.0/0)
I:4.2/0	Bit 0 word 2 slot 4
I:4/32	Bit 32 slot 4 (same as I:4.2/0)

Status Files

To access status files, specify a word and an optional bit in the word. The default data types are shown in **bold**.

Syntax	Data Type	Access
S:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Read/Write
S:<word> [rows][cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write
S:<word> [cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write
S:<word>/<bit>	**Boolean**	Read/Write
S/bit	**Boolean**	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that the array size cannot exceed 16 words given a block request size of 32 bytes.

Ranges

PLC Model	Max. Word
MicroLogix	999
SLC 500 Fixed I/O	96
SLC 500 Modular I/O	999
PLC-5 Series	999

The maximum word location is one less when accessing as a 32-bit data type (such as Long, DWord, or Long BCD).

Examples

Example	Description
S:0	Word 0
S/26	Bit 26
S:4/15	Bit 15 word 4
S:10 [16]	16 element array starting at word 10
S:0 [4] [8]	4 by 8 element array starting at word 0

Binary Files

To access binary files, specify a file number, a word and optional bit in the word. The default data types are shown in **bold**.

Syntax	Data Type	Access
B<file>:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Read/Write
B<file>:<word> [rows][cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write

Syntax	Data Type	Access
B<file>:<word> [cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write
B<file>:<word>/<bit>	**Boolean**	Read/Write
B<file>/bit	**Boolean**	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 words given a block request size of 32 bytes.

Ranges

PLC Model	File Number	Max. Word
MicroLogix	3, 9-999	999
SLC 500 Fixed I/O	3, 9-255	255
SLC 500 Modular I/O	3, 9-999	999
PLC-5 Series	3-999	1999

The maximum word location is one less when accessing as a 32-bit data type (such as Long, DWord, or Long BCD).

Examples

Example	Description
B3:0	Word 0
B3/26	Bit 26
B12:4/15	Bit 15 word 4
B3:10 [20]	20 element array starting at word 10
B15:0 [6] [6]	6 by 6 element array starting at word 0

Timer Files

Timer files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data types are shown in **bold**.

Syntax	Data Type	Access
T<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
ACC	**Short**, Word	Read/Write
PRE	**Short**, Word	Read/Write
DN	**Boolean**	Read Only
TT	**Boolean**	Read Only
EN	**Boolean**	Read Only

Ranges

PLC Model	File Number	Max. Element
MicroLogix	4, 9-999	999
SLC 500 Fixed I/O	4, 9-255	255
SLC 500 Modular I/O	4, 9-999	999
PLC-5 Series	3-999	1999

Examples

Example	Description
T4:0.ACC	Accumulator of timer 0 file 4
T4:10.DN	Done bit of timer 10 file 4
T15:0.PRE	Preset of timer 0 file 15

Counter Files

Counter files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

Syntax	Data Type	Access
C<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
ACC	**Word**, Short	Read/Write
PRE	**Word**, Short	Read/Write
UA	**Boolean**	Read Only
UN	**Boolean**	Read Only
OV	**Boolean**	Read Only
DN	**Boolean**	Read Only
CD	**Boolean**	Read Only
CU	**Boolean**	Read Only

Ranges

PLC Model	File Number	Max. Element
MicroLogix	5, 9-999	999
SLC 500 Fixed I/O	5, 9-255	255
SLC 500 Modular I/O	5, 9-999	999
PLC-5 Series	3-999	1999

Examples

Example	Description
C5:0.ACC	Accumulator of counter 0 file 5
C5:10.DN	Done bit of counter 10 file 5
C15:0.PRE	Preset of counter 0 file 15

Control Files

Control files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

Syntax	Data Type	Access
R<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC documentation.

Element Field	Data Type	Access
LEN	**Word**, Short	Read/Write
POS	**Word**, Short	Read/Write
FD	**Boolean**	Read Only
IN	**Boolean**	Read Only
UL	**Boolean**	Read Only
ER	**Boolean**	Read Only
EM	**Boolean**	Read Only
DN	**Boolean**	Read Only
EU	**Boolean**	Read Only
EN	**Boolean**	Read Only

Ranges

PLC Model	File Number	Max. Element
MicroLogix	6, 9-999	999
SLC 500 Fixed I/O	6, 9-255	255
SLC 500 Modular I/O	6, 9-999	999
PLC-5 Series	3-999	1999

Examples

Example	Description
R6:0.LEN	Length field of control 0 file 6
R6:10.DN	Done bit of control 10 file 6
R15:18.POS	Position field of control 18 file 15

Integer Files

To access integer files, specify a file number, a word, and an optional bit in the word. The default data types are shown in **bold**.

Syntax	Data Type	Access
N<file>:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Read/Write
N<file>:<word> [rows][cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write
N<file>:<word> [cols]	Short, **Word**, BCD, DWord, Long, LBCD (array type)	Read/Write
N<file>:<word>/<bit>	**Boolean**	Read/Write
N<file>/bit	**Boolean**	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 words given a block request size of 32 bytes.

Ranges

PLC Model	File Number	Max. Word
MicroLogix	7, 9-999	999
SLC 500 Fixed I/O	7, 9-255	255
SLC 500 Modular I/O	7, 9-999	999
PLC-5 Series	3-999	1999

The maximum word location is one less when accessing as a 32-bit data type (such as Long, DWord, or Long BCD).

Examples

Example	Description
N7:0	Word 0
N7/26	Bit 26
N12:4/15	Bit 15 word 4
N7:10 [8]	8 element array starting at word 10
N15:0 [4] [5]	4 by 5 element array starting at word 0

Float Files

To access float files, specify a file number and an element. The default data types are shown in **bold**.

Syntax	Data Type	Access
F<file>:<element>	**Float**	Read/Write
F<file>:<element> [rows][cols]	**Float** (array type)	Read/Write
F<file>:<element> [cols]	**Float** (array type)	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 8 floats given a block request size of 32 bytes.

Ranges

PLC Model	File Number	Max. Word
MicroLogix	8-999	999
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	8-999	999
PLC-5 Series	3-999	1999

Examples

Example	Description
F8:0	Float 0
F8:10 [16]	16 element array starting at word 10
F15:0 [4] [4]	16 element array starting at word 0

ASCII Files

To access ASCII file data, specify a file number and a character location. The default data types are shown in **bold**.

Syntax	Data Type	Access
A<file>:<char>	**Char**, Byte*	Read/Write
A<file>:<char> [rows][cols]	**Char**, Byte*	Read/Write
A<file>:<char> [cols]	**Char**, Byte*	Read/Write
A<file>:<word offset>/length	String**	Read/Write

*The number of array elements cannot exceed the block request size specified. Internally, the PLC packs two characters per word in the file, with the high byte containing the first character and the low byte containing the second character. The PLC programming software allows access at the word level or two-character level. TheAllen-Bradley ControlLogix Ethernetドライバー allows accessing to the character level.

Using the programming software, "A10:0 = AB," would result in 'A' being stored in the high byte of A10:0 and 'B' being stored in the low byte. Using the Allen-Bradley ControlLogix Ethernetドライバー, two assignments would be made: "A10:0 = A" and "A10:1 = B." This would result in the same data being stored in the PLC memory.

**Referencing this file as string data allows access to data at word boundaries like the programming software. The length can be up to 232 characters. If a string that is sent to the device is smaller in length than the length specified by the address, the driver null terminates the string before sending it down to the controller.

Ranges

PLC Model	File Number	Max. Character
MicroLogix	3-255	511

PLC Model	File Number	Max. Character
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	9-999	1999
PLC-5 Series	3-999	1999

Note: Not all MicroLogix and SLC 500 PLC devices support ASCII file types. For more information, refer to the PLC's documentation.

Examples

Example	Description
A9:0	character 0 (high byte of word 0)
A27:10 [80]	80 character array starting at character 10
A15:0 [4] [16]	4 by 16 character array starting at character 0
A62:0/32	32 character string starting at word offset 0

String Files

To access string files, specify a file number and an element. Strings are 82 character null terminated arrays. The driver places the null terminator based on the string length returned by the PLC. The default data types are shown in **bold**.

Note: Arrays are not supported for string files.

Syntax	Data Type	Access
ST<file>:<element>.<field>	**String**	Read/Write

Ranges

PLC Model	File Number	Max. Word
MicroLogix	9-999	999
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	9-999	999
PLC-5 Series	3-999	999

Examples

Example	Description
ST9:0	String 0
ST18:10	String 10

BCD Files

To access BCD files, specify a file number and a word. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
D<file>:<word>	**BCD**, LBCD	Read/Write

Syntax	Data Type	Access
D<file>:<word> [rows][cols]	**BCD**, LBCD (array type)	Read/Write
D<file>:<word> [cols]	**BCD**, LBCD (array type)	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 BCD, given a block request size of 32 bytes.

Ranges

PLC Model	File Number	Max. Word
MicroLogix	N/A	N/A
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	N/A	N/A
PLC-5 Series	3-999	999

Examples

Example	Description
D9:0	Word 0
D27:10 [16]	16 element array starting at Word 10
D15:0 [4][8]	32 element array starting at Word 0

Long Files

To access long integer files, specify a file number and an element. The default data types are shown in **bold**.

Syntax	Data Type	Access
L<file>:<DWord>	Long, **DWord**, LBCD	Read/Write
L<file>:<DWord> [rows][cols]	Long, **DWord**, LBCD (array type)	Read/Write
L<file>:<DWord> [cols]	Long, **DWord**, LBCD (array type)	Read/Write

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 8 longs given a block request size of 32 bytes.

Ranges

PLC Model	File Number	Max. Word
MicroLogix	9-999	999
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	N/A	N/A
PLC-5 Series	N/A	N/A

Examples

Example	Description
L9:0	word 0
L9:10 [8]	8 element array starting at word 10

Example	Description
L15:0 [4] [5]	4 by 5 element array starting at word 0

MicroLogix PID Files

PID files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

Syntax	Data Type	Access
PD<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation for the meaning of each field.

Element Field	Data Type	Access
SPS	**Word**, Short	Read/Write
KC	**Word**, Short	Read/Write
TI	**Word**, Short	Read/Write
TD	**Word**, Short	Read/Write
MAXS	**Word**, Short	Read/Write
MINS	**Word**, Short	Read/Write
ZCD	**Word**, Short	Read/Write
CVH	**Word**, Short	Read/Write
CVL	**Word**, Short	Read/Write
LUT	**Word**, Short	Read/Write
SPV	**Word**, Short	Read/Write
CVP	**Word**, Short	Read/Write
TM	**Boolean**	Read/Write
AM	**Boolean**	Read/Write
CM	**Boolean**	Read/Write
OL	**Boolean**	Read/Write
RG	**Boolean**	Read/Write
SC	**Boolean**	Read/Write
TF	**Boolean**	Read/Write
DA	**Boolean**	Read/Write
DB	**Boolean**	Read/Write
UL	**Boolean**	Read/Write
LL	**Boolean**	Read/Write
SP	**Boolean**	Read/Write
PV	**Boolean**	Read/Write
DN	**Boolean**	Read/Write
EN	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	3-255	255
All SLC	N/A	N/A
PLC-5	PID Files	PID Files

Examples

Example	Description
PD14:0.KC	Proportional gain of PD 0 file 14
PD18:6.EN	PID enable bit of PD 6 file 18

PID Files

PID files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
PD<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
SP	**Real**	Read/Write
KP	**Real**	Read/Write
KI	**Real**	Read/Write
KD	**Real**	Read/Write
BIAS	**Real**	Read/Write
MAXS	**Real**	Read/Write
MINS	**Real**	Read/Write
DB	**Real**	Read/Write
SO	**Real**	Read/Write
MAXO	**Real**	Read/Write
MINO	**Real**	Read/Write
UPD	**Real**	Read/Write
PV	**Real**	Read/Write
ERR	**Real**	Read/Write
OUT	**Real**	Read/Write
PVH	**Real**	Read/Write
PVL	**Real**	Read/Write
DVP	**Real**	Read/Write

Element Field	Data Type	Access
DVN	**Real**	Read/Write
PVDB	**Real**	Read/Write
DVDB	**Real**	Read/Write
MAXI	**Real**	Read/Write
MINI	**Real**	Read/Write
TIE	**Real**	Read/Write
FILE	**Word**, Short	Read/Write
ELEM	**Word**, Short	Read/Write
EN	**Boolean**	Read/Write
CT	**Boolean**	Read/Write
CL	**Boolean**	Read/Write
PVT	**Boolean**	Read/Write
DO	**Boolean**	Read/Write
SWM	**Boolean**	Read/Write
CA	**Boolean**	Read/Write
MO	**Boolean**	Read/Write
PE,	**Boolean**	Read/Write
INI	**Boolean**	Read/Write
SPOR	**Boolean**	Read/Write
OLL	**Boolean**	Read/Write
OLH	**Boolean**	Read/Write
EWD	**Boolean**	Read/Write
DVNA	**Boolean**	Read/Write
DVHA	**Boolean**	Read/Write
PVLA	**Boolean**	Read/Write
PVHA	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	N/A
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	N/A	N/A
PLC-5 Series	3-999	999

Examples

Example	Description
PD14:0.SP	Set point field of PD 0 file 14
PD18:6.EN	Status enable bit of PD 6 file 18

MicroLogix Message Files

Message files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

Syntax	Data Type	Access
MG<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
IA	**Word**, Short	Read/Write
RBL	**Word**, Short	Read/Write
LBN	**Word**, Short	Read/Write
RBN	**Word**, Short	Read/Write
CHN	**Word**, Short	Read/Write
NOD	**Word**, Short	Read/Write
MTO	**Word**, Short	Read/Write
NB	**Word**, Short	Read/Write
TFT	**Word**, Short	Read/Write
TFN	**Word**, Short	Read/Write
ELE	**Word**, Short	Read/Write
SEL	**Word**, Short	Read/Write
TO	**Boolean**	Read/Write
CO	**Boolean**	Read/Write
EN	**Boolean**	Read/Write
RN	**Boolean**	Read/Write
EW	**Boolean**	Read/Write
ER	**Boolean**	Read/Write
DN	**Boolean**	Read/Write
ST	**Boolean**	Read/Write
BK	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	3-255	255
All SLC	N/A	N/A
PLC5	Message Files	Message Files

Examples

Example	Description
MG14:0.TO	Time out bit for MSG element 0 in data file 14
MG18:6.CO	Continue bit for MSG element 6 in data file 18

Message Files

Message files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
MG<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
ERR	**Short**, Word	Read/Write
RLEN	**Short**, Word	Read/Write
DLEN	**Short**, Word	Read/Write
EN	**Boolean**	Read/Write
ST	**Boolean**	Read/Write
DN	**Boolean**	Read/Write
ER	**Boolean**	Read/Write
CO	**Boolean**	Read/Write
EW	**Boolean**	Read/Write
NR	**Boolean**	Read/Write
TO	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	N/A
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	N/A	N/A
PLC-5 Series	3-999	999

Examples

Example	Description
MG14:0.RLEN	Requested length field of MG 0 file 14
MG18:6.CO	Continue bit of MG 6 file 18

Block Transfer Files

Block transfer files are a structured type whose data is accessed by specifying a file number, an element, and a field. The default data types are shown in **bold**.

PLC-5 Syntax

Syntax	Data Type	Access
BT<file>:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For more information on the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
RLEN	**Word**, Short	Read/Write
DLEN	**Word**, Short	Read/Write
FILE	**Word**, Short	Read/Write
ELEM	**Word**, Short	Read/Write
RW	**Boolean**	Read/Write
ST	**Boolean**	Read/Write
DN	**Boolean**	Read/Write
ER	**Boolean**	Read/Write
CO	**Boolean**	Read/Write
EW	**Boolean**	Read/Write
NR	**Boolean**	Read/Write
TO	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	N/A
SLC 500 Fixed I/O	N/A	N/A
SLC 500 Modular I/O	N/A	N/A
PLC-5 Series	3-999	1999

Examples

Example	Description
BT14:0.RLEN	Requested length field of BT 0 file 14
BT18:6.CO	Continue bit of BT 6 file 18

Function Files

For information on the files supported by the ENI MicroLogix and MicroLogix 1100 device models, select a link from the list below.

High-Speed Counter File (HSC)
Real-Time Clock File (RTC)
Channel 0 Communication Status File (CS0)
Channel 1 Communication Status File (CS1)
I/O Module Status File (IOS)

*For more information on device models and their supported files, refer to **Address Descriptions**.*

High-Speed Counter File (HSC)

The HSC files are a structured type whose data is accessed by specifying an element and a field. The default data types are shown in **bold**.

***See Also:** ENI DF1/ DH+/ControlNet Gateway Communications Parameters*

Syntax	Data Type	Access
HSC:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Default Type	Access
ACC	**DWord**, Long	Read Only
HIP	**DWord**, Long	Read/Write
LOP	**DWord**, Long	Read/Write
OVF	**DWord**, Long	Read/Write
UNF	**DWord**, Long	Read/Write
PFN	**Word**, Short	Read Only
ER	**Word**, Short	Read Only
MOD	**Word**, Short	Read Only
OMB	**Word**, Short	Read Only
HPO	**Word**, Short	Read/Write
LPO	**Word**, Short	Read/Write
UIX	**Boolean**	Read Only
UIP	**Boolean**	Read Only
AS	**Boolean**	Read Only
ED	**Boolean**	Read Only
SP	**Boolean**	Read Only
LPR	**Boolean**	Read Only
HPR	**Boolean**	Read Only

Element Field	Default Type	Access
DIR	**Boolean**	Read Only
CD	**Boolean**	Read Only
CU	**Boolean**	Read Only
UIE	**Boolean**	Read/Write
UIL	**Boolean**	Read/Write
FE	**Boolean**	Read/Write
CE	**Boolean**	Read/Write
LPM	**Boolean**	Read/Write
HPM	**Boolean**	Read/Write
UFM	**Boolean**	Read/Write
OFM	**Boolean**	Read/Write
LPI	**Boolean**	Read/Write
HPI	**Boolean**	Read/Write
UFI	**Boolean**	Read/Write
OFI	**Boolean**	Read/Write
UF	**Boolean**	Read/Write
OF	**Boolean**	Read/Write
MD	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	254
All SLC	N/A	N/A
PLC5	N/A	N/A

Examples

Example	Description
HSC:0.OMB	Output mask setting for high-speed counter 0
HSC:1.ED	Error detected indicator for high-speed counter 1

Real-Time Clock File (RTC)

The RTC files are a structured type whose data is accessed by specifying an element and a field. The default data types are shown in **bold**.

See Also: *ENI DF1/ DH+/ControlNet Gateway Communications Parameters*

Syntax	Data Type	Access
RTC:<element>.<field>	Depends on field	Depends on field

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

Element Field	Data Type	Access
YR	**Word**, Short	Read/Write
MON	**Word**, Short	Read/Write
DAY	**Word**, Short	Read/Write
HR	**Word**, Short	Read/Write
MIN	**Word**, Short	Read/Write
SEC	**Word**, Short	Read/Write
DOW	**Word**, Short	Read/Write
DS	**Boolean**	Read Only
BL	**Boolean**	Read Only
_SET (for block writes)	**Boolean**	Read/Write

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	254
All SLC	N/A	N/A
PLC5	N/A	N/A

Examples

Example	Description
RTC:0.YR	Year setting for real-time clock 0.
RTC:0.BL	Battery low indicator for real-time clock 0.

Channel 0 Communication Status File (CS0)

To access the communication status file for channel 0, specify a word (and optionally a bit in the word). The default data types are shown in **bold**.

See Also: ***ENI DF1/ DH+/ControlNet Gateway Communications Parameters***

Syntax	Data Type	Access
CS0:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Depends on <word> and <bit>
CS0:<word>/<bit>	**Boolean**	Depends on <word> and <bit>
CS0/bit	**Boolean**	Depends on <word> and <bit>

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	254
All SLC	N/A	N/A
PLC5	N/A	N/A

Examples

Example	Description
CS0:0	Word 0
CS0:4/2	Bit 2 word 4 = MCP

For more information on CS0 words/bit meanings, refer to the Rockwell documentation.

Channel 1 Communication Status File (CS1)

To access the communication status file for channel 1, specify a word (and optionally a bit in the word). The default data types are shown in **bold**.

***See Also**: ENI DF1/ DH+/ControlNet Gateway Communications Parameters*

Syntax	Data Type	Access
CS1:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Depends on <word> and <bit>
CS1:<word>/<bit>	**Boolean**	Depends on <word> and <bit>
CS1/bit	**Boolean**	Depends on <word> and <bit>

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	254
All SLC	N/A	N/A
PLC5	N/A	N/A

Examples

Example	Description
CS1:0	Word 0
CS1:4/2	Bit 2 word 4 = MCP

For more information on CS1 words/bit meanings, refer to the Rockwell documentation.

I/O Module Status File (IOS)

To access an I/O module status file, specify a word and optionally a bit. The default data types are shown in **bold**.

***See Also:** ENI DF1/ DH+/ControlNet Gateway Communications Parameters*

Syntax	Data Type	Access
IOS:<word>	Short, **Word**, BCD, DWord, Long, LBCD	Depends on <word> and <bit>
IOS:<word>/<bit>	**Boolean**	Depends on <word> and <bit>
IOS/bit	**Boolean**	Depends on <word> and <bit>

Ranges

PLC Model	File Number	Max. Element
MicroLogix	N/A	254
All SLC	N/A	N/A
PLC5	N/A	N/A

Examples

Example	Description
IOS:0	Word 0
IOS:4/2	Bit 2 word 4

For a listing of 1769 expansion I/O status codes, refer to the manufacturer's instruction manual.

Error Codes

The following sections define error codes that may be encountered in the server's Event Log. For more information on a specific error code type, select a link from the list below.

Encapsulation Error Codes
CIP Error Codes

Encapsulation Error Codes

The following error codes are in hexadecimal.

Error Code	Description
0001	Command not handled
0002	Memory not available for command
0003	Poorly formed or incomplete data
0064	Invalid session ID
0065	Invalid length in header
0069	Requested protocol version not supported
0070	Invalid target ID

CIP Error Codes

The following error codes are in hexadecimal.

Error Code	Log Code	Description
0001	0x01	Connection Failure*
0002	0x02	Insufficient resources
0003	0x03	Value invalid
0004	0x04	IOI could not be deciphered or tag does not exist **Note**: This error may also occur if the address is valid but the security permissions do not allow client access.”
0005	0x05	Unknown destination
0006	0x06	Data requested would not fit in response packet
0007	0x07	Loss of connection
0008	0x08	Unsupported service
0009	0x09	Error in data segment or invalid attribute value
000A	0x0A	Attribute list error
000B	0x0B	State already exists
000C	0x0C	Object model conflict
000D	0x0D	Object already exists
000E	0x0E	Attribute not settable
000F	0x0F	Permission denied
0010	0x10	Device state conflict

Error Code	Log Code	Description
0011	0x11	Reply does not fit
0012	0x12	Fragment primitive
0013	0x13	Insufficient command data / parameters specified to execute service
0014	0x14	Attribute not supported
0015	0x15	Too much data specified
001A	0x1A	Bridge request too large
001B	0x1B	Bridge response too large
001C	0x1C	Attribute list shortage
001D	0x1D	Invalid attribute list
001E	0x1E	Embedded service error
001F	0x1F	Failure during connection**
0022	0x22	Invalid reply received
0025	0x25	Key segment error
0026	0x26	Number of IOI words specified does not match IOI word count
0027	0x27	Unexpected attribute in list

- ***See Also:** *0x0001 Extended Error Codes*
- ****See Also:** *0x001F Extended Error Codes*

Logix5000-Specific (1756-L1) Error Codes

The following error codes are in hexadecimal.

Error Code	Description
00FF	General Error*

- ***See Also**: *0x00FF Extended Error Codes*
- ***See Also**: For unlisted error codes, refer to the Rockwell documentation.*

0x0001 Extended Error Codes

The following error codes are in hexadecimal.

Error Code	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

Error Code	Description
0114	Wrong module
0115	Wrong device type
0116	Wrong 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
0318	Link address to self is invalid

For unlisted error codes, refer to the Rockwell documentation.

0x001F Extended Error Codes

The following error codes are in hexadecimal.

Error Code	Description
0203	Connection timed out

For unlisted error codes, refer to the Rockwell documentation.

0x00FF Extended Error Codes

The following error codes are in hexadecimal.

Error Code	Description
2104	Address out of range
2105	Attempt to access beyond end of data object
2106	Data in use
2107	Data type is invalid or not supported

For unlisted error codes, refer to the Rockwell documentation.

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the OPC server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Tip: Messages that originate from a data source (such as third-party software, including databases) are presented through the Event Log. Troubleshooting steps should include researching those messages online and in vendor documentation.

デバイスからコントローラプロジェクトをアップロード中に次のエラーが発生しました。シンボリックプロトコルを使用します。

エラータイプ:
エラー

同期化中に無効または破損したコントローラプロジェクトが検出されました。まもなく同期化を再試行します。

エラータイプ:
エラー

考えられる原因:
同期化中に無効または破損したコントローラプロジェクトが検出されました。

解決策:
操作は必要ありません。ドライバーは 30 秒経過すると再び同期化を試みます。

注記:
論理アドレス指定モードではプロジェクトの同期化が必要です。

同期化中にプロジェクトのダウンロードが検出されました。まもなく同期化を再試行します。

エラータイプ:
エラー

考えられる原因:
デバイスがコントローラプロジェクトと同期化しているときにプロジェクトのダウンロードが試みられました。

解決策:
操作は必要ありません。ドライバーは 30 秒経過すると再び同期化を試みます。

注記:
論理アドレス指定モードではプロジェクトの同期化が必要です。

データベースエラー。参照タグのデータ型が不明です。エイリアスタグのデータ型をデフォルトに設定します。| 参照タグ = '<タグ>'、エイリアスタグ = '<タグ>'、デフォルトデータ型 = '<タイプ>'。

エラータイプ:
エラー

考えられる原因:
エイリアスタグの宣言で参照される "Alias For" タグのデータ型がタグインポートファイルで見つかりませんでした。エイリアスタグを正しく生成するためにはこのデータ型が必要です。

解決策:
エイリアスタグはデフォルトの型として指定されているデータ型を使用します。

注記:
RSLogix5000 で、「Edit Tags」タブのタグビューに "Alias For" 列があり、このタグへの参照、構造タグメンバー、またはエイリアスタグが表すビットが入力されています。

関連項目:
Logix オプション

データベースエラー。タグインポートファイルでメンバーのデータ型が見つかりません。データ型をデフォルトに設定します。| メンバーのデータ型 = '<タイプ>'、UDT = '<タイプ>'、デフォルトデータ型 = '<タイプ>'。

エラータイプ:
エラー

考えられる原因:
ユーザー定義型のメンバーのデータ型の定義がタグインポートファイルで見つかりませんでした。メンバーはデバイスのプロパティで指定されているデフォルトの型をとります。

解決策:
指定されているタグのユーザー定義データ型の定義を確認して修正し、インポートを再試行してください。

関連項目:
Logix オプション

データベースエラー。タグインポートファイルでデータ型が見つかりません。タグは追加されません。| データ型 = '<タイプ>'、タグ名 = '<タグ>'。

エラータイプ:
エラー

考えられる原因:
指定されているタグのデータ型の定義がタグインポートファイルで見つかりませんでした。タグはデータベースに追加されません。

解決策:

指定されているタグのデータ型の定義を確認して修正し、インポートを再試行してください。

データベースエラー。エイリアスタグの処理中にエラーが発生しました。タグは追加されませんでした。| エイリアスタグ = '<タグ>'。

エラータイプ:
エラー

考えられる原因:
エイリアスタグの処理中に内部エラーが発生しました。エイリアスタグを生成できませんでした。

解決策:
指定されているタグのデータ型の定義を確認して修正し、インポートを再試行してください。

データベースエラー。レジスタセッションの要求時にカプセル化エラーが発生しました。| カプセル化エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
要求時にEthernet/IP パケットのカプセル化部分の範囲でデバイスがエラーを返しました。要求内のすべての読み取りと書き込みが失敗しました。

解決策:
このようなエラーからの回復はドライバーが自動的に試みます。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。エラー 0x02 はドライバー関連ではなくデバイス関連なので除外されます。

関連項目:
カプセル化エラーコード

データベースエラー。レジスタセッションの要求時にフレーミングエラーが発生しました。

エラータイプ:
エラー

データベースエラー。フォワードオープンの要求時にカプセル化エラーが発生しました。| カプセル化エラー = <コード>。

エラータイプ:
エラー

データベースエラー。フォワードオープンの要求時にフレーミングエラーが発生しました。

エラータイプ:
エラー

データベースエラー。フォワードオープンの要求時にエラーが発生しました。| CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:

エラー

データベースエラー。プロジェクト情報のアップロード中にカプセル化エラーが発生しました。| カプセル化エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中にEthernet/IPパケットのカプセル化部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
カプセル化エラーコード

データベースエラー。プロジェクト情報のアップロード中にエラーが発生しました。| CIPエラー = <コード>、拡張エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中にEthernet/IPパケットのCIP部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
CIPエラーコード

データベースエラー。プロジェクト情報のアップロード中にフレーミングエラーが発生しました。

エラータイプ:
エラー

考えられる原因:

1. パケットに不整列が発生しています (原因は PC とデバイス間の接続/切断)。
2. デバイスのケーブル接続の不良によりノイズが発生しています。

解決策:

1. ノイズが少ないネットワーク上にデバイスを配置してください。
2. 要求タイムアウト、再試行回数、またはその両方の値を増やしてください。
3. サーバーを再起動してから、もう一度試してください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

データベースエラー。内部エラーが発生しました。

エラータイプ:
エラー

データベースエラー。プログラム情報のアップロード中にカプセル化エラーが発生しました。| プログラム名 = '<名前>'、カプセル化エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中に Ethernet/IP パケットのカプセル化部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
カプセル化エラーコード

データベースエラー。プログラム情報のアップロード中にエラーが発生しました。| プログラム名 = '<名前>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:

返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
CIP エラーコード

データベースエラー。プログラム情報のアップロード中にフレーミングエラーが発生しました。| プログラム名 = '<名前>'。

エラータイプ:
エラー

考えられる原因:

1. パケットに不整列が発生しています(原因はPCとデバイス間の接続/切断)。
2. デバイスのケーブル接続の不良によりノイズが発生しています。

解決策:

1. ノイズが少ないネットワーク上にデバイスを配置してください。
2. 要求タイムアウト、再試行回数、またはその両方の値を増やしてください。
3. サーバーを再起動してから、もう一度試してください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

データベースエラー。タグの CIP データ型を解決できません。デフォルトの型に設定します。| CIP データ型 = <タイプ>、タグ名 = '<タグ>'、デフォルトデータ型 = '<タイプ>'。

エラータイプ:
エラー

考えられる原因:

1. インポートファイル内の CIP データ型が不明です。
2. インポートファイルにエラーが含まれている可能性があります。

解決策:
RSLogix 内のエラーを解決し、タグエクスポートプロセスを再試行して新しいタグインポートファイルを生成してください。

関連項目:
自動タグデータベース生成の準備

プロジェクト情報のアップロード中にカプセル化エラーが発生しました。| カプセル化エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中に Ethernet/IP パケットのカプセル化部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
カプセル化エラーコード

プロジェクト情報のアップロード中にエラーが発生しました。| CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
エラー

考えられる原因:
コントローラプロジェクトをアップロード中に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

関連項目:
CIP エラーコード

プロジェクト情報のアップロード中にフレーミングエラーが発生しました。

エラータイプ:
エラー

考えられる原因:

1. パケットに不整列が発生しています (原因は PC とデバイス間の接続/切断)。
2. デバイスのケーブル接続の不良によりノイズが発生しています。

解決策:

1. ノイズが少ないネットワーク上にデバイスを配置してください。
2. 要求タイムアウト、再試行回数、またはその両方の値を増やしてください。
3. サーバーを再起動してから、もう一度試してください。

注記:
論理アドレス指定モードではプロジェクトのアップロードが必要です。

プログラム情報のアップロード中にカプセル化エラーが発生しました。| プログラム名 = '<名前>'、カプセル化エラー = <コード>。

エラータイプ:
エラー

プログラム情報のアップロード中にエラーが発生しました。| プログラム名 = '<名前>'、CIPエラー = <コード>、拡張エラー = <コード>。

エラータイプ:
エラー

プログラム情報のアップロード中にフレーミングエラーが発生しました。| プログラム名 = '<名前>'。

エラータイプ:
エラー

コントローラプログラム情報のアップロード中にカプセル化エラーが発生しました。カプセル化エラー = <コード>。

エラータイプ:
エラー

コントローラプログラム情報のアップロード中にエラーが発生しました。CIPエラー = <コード>、拡張エラー = <コード>。

エラータイプ:
エラー

コントローラプログラム情報のアップロード中にフレーミングエラーが発生しました。

エラータイプ:
エラー

プロジェクト情報のアップロード中にCIP接続がタイムアウトしました。

エラータイプ:
エラー

考えられる原因:

不活動ウォッチドッグに設定されている値が小さすぎるため、プロジェクトをロードできません。

解決策:
不活動ウォッチドッグの値を増やしてから、もう一度試してください。

データベースエラー。プロジェクト情報のアップロード中に CIP 接続がタイムアウトしました。

エラータイプ:
エラー

考えられる原因:
不活動ウォッチドッグに設定されている値が小さすぎるため、プロジェクトをロードできません。

解決策:
不活動ウォッチドッグの値を増やしてから、もう一度試してください。

データベースエラー。フォワードオープンの要求に利用可能な接続はもうありません。

エラータイプ:
エラー

タグデータベースのインポート用のファイルを開くときにエラーが発生しました。| OS エラー = '<コード>'。

エラータイプ:
エラー

サポートされていないコントローラです。| ベンダー ID = <ID>、製品タイプ = <タイプ>、製品コード = <コード>、製品名 = '<名前>'。

エラータイプ:
警告

デバイスから受信したフレームにエラーが含まれています。

エラータイプ:
警告

考えられる原因:

1. PC とデバイス間の接続/切断によってパケットに不整列が発生しています。
2. デバイスのケーブル接続の不良によりノイズが発生しています。

解決策:

1. ノイズが少ないネットワーク上にデバイスを配置してください。
2. 要求タイムアウト、再試行回数、またはその両方の値を増やしてください。

フレーミングエラーにより書き込み要求が失敗しました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 不正な要求サービスコードがあります。
2. ドライバーは予想されるバイト数よりも多いかまたは少ないバイト数を受信しました。
3. このエラーが頻繁に発生する場合、ケーブル接続またはデバイスに問題がある可能性があります。

解決策:

1. ドライバーがこのエラーから回復できるようにするには、再試行回数を増やしてください。
2. ケーブル接続とデバイスが適切に機能していることを確認してください。

フレーミングエラーによりタグの読み取り要求が失敗しました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 不正な要求サービスコードがあります。
2. ドライバーは予想されるバイト数よりも多いかまたは少ないバイト数を受信しました。
3. このエラーが頻繁に発生する場合、ケーブル接続またはデバイスに問題がある可能性があります。

解決策:

1. ドライバーがこのエラーから回復できるようにするには、再試行回数を増やしてください。
2. ケーブル接続とデバイスが適切に機能していることを確認してください。

フレーミングエラーによりブロック読み取り要求が失敗しました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間のイーサネット接続が切断しています。
2. イーサネット接続の通信パラメータが不正です。
3. この名前のデバイスに不正な IP アドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定した IP アドレスが実際のデバイスのアドレスと一致することを確認してください。

フレーミングエラーによりブロック読み取り要求が失敗しました。| ブロックサイズ = <数値> (バイト)、ブロック名 = '<名前>'。

エラータイプ:
警告

考えられる原因:

1. 不正な要求サービスコードがあります。
2. ドライバーは予想されるバイト数よりも多いかまたは少ないバイト数を受信しました。
3. このエラーが頻繁に発生する場合、ケーブル接続またはデバイスに問題がある可能性があります。

解決策:

1. ドライバーがこのエラーから回復できるようにするには、再試行回数を増やしてください。
2. ケーブル接続とデバイスが適切に機能していることを確認してください。

タグに書き込めません。| タグアドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたタグの書き込み要求時に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。

関連項目:
CIP エラーコード

タグを読み取れません。| タグアドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたタグの読み取り要求時に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:

返されたエラーコードによって解決策が異なります。

関連項目:
CIP エラーコード

ブロックを読み取れません。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

ブロックを読み取れません。| ブロックサイズ = <数値> (バイト)、タグ名 = '<タグ>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

タグに書き込めません。コントローラタグのデータ型が不明です。| タグアドレス = '<アドレス>'、データ型 = <タイプ>。

エラータイプ:
警告

考えられる原因:
コントローラタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

関連項目:
アトミックデータ型のアドレス指定

タグを読み取れません。コントローラタグのデータ型が不明です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、データ型 = <タイプ>。

エラータイプ:
警告

考えられる原因:
コントローラタグのデータ型がサポートされていないため、指定されたタグの読み取り要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

関連項目:
アトミックデータ型のアドレス指定

ブロックを読み取れません。コントローラタグのデータ型が不明です。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
ブロック内のコントローラタグのデータ型がサポートされていないため、指定されたブロックの読み取り要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

関連項目:
アトミックデータ型のアドレス指定

タグに書き込めません。データ型がサポートされていません。| タグアドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。

関連項目:
アトミックデータ型のアドレス指定

タグを読み取れません。データ型がサポートされていません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
コントローラタグのデータ型がサポートされていないため、指定されたタグの読み取り要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

関連項目:
アトミックデータ型のアドレス指定

ブロックを読み取れません。データ型がサポートされていません。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
ブロック内のコントローラタグのデータ型がサポートされていないため、指定されたブロックの読み取り要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

関連項目:
アトミックデータ型のアドレス指定

タグに書き込めません。このタグには不正なデータ型です。| タグアドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型が示されたコントローラタグに対して不正であるため、指定されたタグの書き込み要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のコントローラタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

タグを読み取れません。このタグには不正なデータ型です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型が示されたコントローラタグに対して不正であるため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のコントローラタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題を解決できます。

関連項目:

アトミックデータ型のアドレス指定

ブロックを読み取れません。このブロックには不正なデータ型です。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型が示されたコントローラタグに対して不正であるため、ブロックの読み取り要求は失敗しました。

解決策:
このブロック内のタグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のコントローラタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

タグに書き込めません。タグは複数要素の配列をサポートしません。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
複数要素の配列から示されたコントローラタグへのアクセスをドライバーがサポートしないため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型またはアドレスをサポート対象のものに変更してください。

関連項目:
アトミックデータ型のアドレス指定

タグを読み取れません。タグは複数要素の配列をサポートしません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
複数要素の配列から示されたコントローラタグへのアクセスをドライバーがサポートしないため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型またはアドレスをサポート対象のものに変更してください。このエラーによりタグは非アクティブ化され、再度処理されることはありません。

関連項目:

アトミックデータ型のアドレス指定

ブロックを読み取れません。ブロックは複数要素の配列をサポートしません。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
複数要素の配列から示されたコントローラタグへのアクセスをドライバーがサポートしないため、このブロック内のタグの読み取り要求は失敗しました。

解決策:
このブロック内のタグのデータ型またはアドレスをサポートされているものに変更してください。このエラーによりブロックの <count> 個の要素は非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

タグに書き込めません。ネイティブタグのサイズが不一致です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
ネイティブタグのサイズ (フットプリント) が、プロジェクトのアップロードによって決まった予想されるサイズと一致しません。

解決策:

1. プロトコルモードをシンボリックモードに変更してから、もう一度試してください。
2. テクニカルサポートまでご連絡の上、この問題についてご報告ください。

タグを読み取れません。ネイティブタグのサイズが不一致です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
ネイティブタグのサイズ (フットプリント) が、プロジェクトのアップロードによって決まった予想されるサイズと一致しません。

解決策:

1. プロトコルモードをシンボリックモードに変更してから、もう一度試してください。
2. テクニカルサポートまでご連絡の上、この問題についてご報告ください。

ブロックを読み取れません。ネイティブタグのサイズが一致しません。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
ネイティブタグのブロックのサイズ (フットプリント) が、プロジェクトのアップロードによって決まった予想されるサイズと一致しません。

解決策:

1. プロトコルモードをシンボリックモードに変更してから、もう一度試してください。
2. テクニカルサポートまでご連絡の上、この問題についてご報告ください。

ブロックを読み取れません。ネイティブタグのサイズが一致しません。| ブロックサイズ = <数値> (バイト)、ブロック名 = '<名前>'。

エラータイプ:
警告

考えられる原因:
ネイティブタグのブロックのサイズ (フットプリント) が、プロジェクトのアップロードによって決まった予想されるサイズと一致しません。

解決策:

1. プロトコルモードをシンボリックモードに変更してから、もう一度試してください。
2. テクニカルサポートまでご連絡の上、この問題についてご報告ください。

タグに書き込めません。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間のイーサネット接続が切断しています。
2. イーサネット接続の通信パラメータが不正です。
3. この名前のデバイスに不正な IP アドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定した IP アドレスが実際のデバイスのアドレスと一致することを確認してください。

タグを読み取れません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間のイーサネット接続が切断しています。
2. イーサネット接続の通信パラメータが不正です。
3. この名前のデバイスに不正な IP アドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定した IP アドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりタグは非アクティブ化され、再度処理されることはありません。

ブロックを読み取れません。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間のイーサネット接続が切断しています。
2. イーサネット接続の通信パラメータが不正です。
3. この名前のデバイスに不正な IP アドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定した IP アドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりブロックの要素は非アクティブ化され、再度処理されることはありません。

ブロックを読み取れません。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (バイト)、タグ名 = '<タグ>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間のイーサネット接続が切断しています。
2. イーサネット接続の通信パラメータが不正です。
3. この名前のデバイスに不正な IP アドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定した IP アドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:

このエラーによりブロックの要素は非アクティブ化され、再度処理されることはありません。

デバイスへの要求中にエラーが発生しました。| CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:

警告

考えられる原因:

要求時に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。要求内のすべての読み取りと書き込みが失敗しました。

解決策:

返されたエラーコードによって解決策が異なります。

関連項目:

CIP エラーコード

デバイスへの要求中にカプセル化エラーが発生しました。| カプセル化エラー = <コード>。

エラータイプ:

警告

考えられる原因:

要求時に Ethernet/IP パケットのカプセル化部分の範囲でデバイスがエラーを返しました。要求内のすべての読み取りと書き込みが失敗しました。

解決策:

このようなエラーからの回復はドライバーが自動的に試みます。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。エラー 0x02 はドライバー関連ではなくデバイス関連なので除外されます。

関連項目:

カプセル化エラーコード

メモリをタグに割り当てることができませんでした。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
タグの構築に必要なリソースを割り当てることができませんでした。タグはプロジェクトに追加されません。

解決策:
使用していないアプリケーションを終了する、仮想メモリの量を増やすなどをした後でもう一度試してください。

ブロックを読み取れません。受信したフレームにエラーが含まれています。| ブロックサイズ = <数値> (要素)、開始アドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 不正な要求サービスコードがあります。
2. ドライバーは予想されるバイト数よりも多いかまたは少ないバイト数を受信しました。

解決策:

1. ドライバーがこのエラーから回復できるようにするには、再試行回数を増やしてください。
2. このエラーが頻繁に発生する場合、ケーブル接続またはデバイス自体に問題がある可能性があります。特定のタグでこのエラーが頻繁に発生する場合、テクニカルサポートまでご連絡ください。

デバイスからファンクションファイルを読み取れません。受信したフレームにエラーが含まれています。| ファンクションファイル = '<名前>'。

エラータイプ:
警告

ブロックを読み取れません。タグは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、開始アドレス = '<アドレス>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:

ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

デバイスからファンクションファイルを読み取れません。タグは非アクティブ化されました。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

アドレスに書き込めません。受信したフレームにエラーが含まれています。| アドレス = '<アドレス>'。

エラータイプ:
警告

ファンクションファイルに書き込めません。受信したフレームにエラーが含まれています。| ファンクションファイル = '<名前>'。

エラータイプ:
警告

ブロックを読み取れません。| ブロックサイズ = <数値> (要素)、開始アドレス = '<アドレス>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:

アドレスがPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KFモジュールが何らかの理由によってネットワーク上で宛先PLCを見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLCによって検出されたエラーを示します。これらのエラーは、データ位置がPLCで使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

ファンクションファイルを読み取れません。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスはPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KFモジュールが何らかの理由によってネットワーク上で宛先PLCを見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLCによって検出されたエラーを示します。これらのエラーは、データ位置がPLCで使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

ブロックを読み取れません。タグは非アクティブ化されました。| ブロックサイズ = <数値>(要素)、開始アドレス = '<アドレス>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスはPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

ファンクションファイルを読み取れません。タグは非アクティブ化されました。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

アドレスに書き込めません。| アドレス = '<アドレス>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

ファンクションファイルに書き込めません。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>、拡張ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスはPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KFモジュールが何らかの理由によってネットワーク上で宛先PLCを見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLCによって検出されたエラーを示します。これらのエラーは、データ位置がPLCで使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

ブロックを読み取れません。| ブロックサイズ = <数値> (要素)、開始アドレス = '<アドレス>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスはPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KFモジュールが何らかの理由によってネットワーク上で宛先PLCを見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLCによって検出されたエラーを示します。これらのエラーは、データ位置がPLCで使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

ファンクションファイルを読み取れません。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

アドレスに書き込めません。| アドレス = '<アドレス>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このアドレスは PLC に存在しません。

解決策:
PLC から返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは 16 進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KF モジュールが何らかの理由によってネットワーク上で宛先 PLC を見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLC によって検出されたエラーを示します。これらのエラーは、データ位置が PLC で使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

ファンクションファイルに書き込めません。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:

このアドレスはPLCに存在しません。

解決策:
PLCから返されたステータスコードと拡張ステータスコードを確認してください。拡張ステータスコードは必ず返るわけではなく、エラー情報はステータスコードに含まれています。これらのコードは16進数で表示されます。

注記:
ステータスコードの下位ニブルのステータスコードエラーは、ローカルノードによって検出されたエラーを示します。ローカルノードによって検出されたエラーは、KFモジュールが何らかの理由によってネットワーク上で宛先PLCを見つけられない場合に発生します。ステータスコードの上位ニブルのステータスコードエラーは、PLCによって検出されたエラーを示します。これらのエラーは、データ位置がPLCで使用できないか書き込み不能の場合に生成されます。

関連項目:
エラーコード定義に関するAllen-Bradleyドキュメント

タグを読み取れません。内部メモリが無効です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

タグを読み取れません。このタグには不正なデータ型です。| タグアドレス = '<アドレス>'、データ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型が示されたコントローラタグに対して不正であるため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL配列のコントローラタグにデータ型Shortは不正です。その場合、データ型をBooleanに変更すると問題を解決できます。

関連項目:
アトミックデータ型のアドレス指定

ブロックを読み取れません。内部メモリが無効です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

ブロックを読み取れません。内部メモリが無効です。ブロックは非アクティブ化されました。| ブロックサイズ = <数値> (要素)、ブロック開始アドレス = '<アドレス>'。

エラータイプ:
警告

アドレスに書き込めません。内部メモリが無効です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

ブロックを読み取れません。ブロックは非アクティブ化されました。| ブロックサイズ = <数値>(要素)、ブロック開始アドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたブロックの読み取り要求時に Ethernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。

関連項目:
CIP エラーコード

デバイスが応答していません。ローカルノードがエラーを返しました。| DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
PLC はローカルノードからの要求に応答しませんでした。ローカルノードが 1756-DHRIO、1756-CNB、1761-NET-ENI などの中間ノードである可能性があります。

解決策:
エラーコード定義に関する Allen-Bradley ドキュメントを参照してください。たとえば、STS コード '0x02'(hex) が返された場合、リモートノード (PLC) とローカルノード間のケーブル接続を確認してください。

関連項目:
エラーコード定義に関する Allen-Bradley ドキュメント

ファンクションファイルに書き込めません。ローカルノードがエラーを返しました。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このエラーは、PLC がローカルノードからの書き込み要求に応答しなかったことを意味します。ローカルノードが 1756-DHRIO、1756-CNB、1761-NET-ENI などの中間ノードである可能性があります。

解決策:

STS エラーコード定義に関する Allen-Bradleyドキュメントを参照してください。たとえば、STS コード '0x02'(hex) が返された場合、リモートノード (PLC) とローカルノード間のケーブル接続を確認してください。

関連項目:
エラーコード定義に関する Allen-Bradleyドキュメント

アドレスに書き込めません。ローカルノードがエラーを返しました。| ファンクションファイル = '<名前>'、DF1 ステータス = <コード>。

エラータイプ:
警告

考えられる原因:
このエラーは、PLC がローカルノードからの書き込み要求に応答しなかったことを意味します。ローカルノードが 1756-DHRIO、1756-CNB、1761-NET-ENI などの中間ノードである可能性があります。

解決策:
STS エラーコード定義に関する Allen-Bradleyドキュメントを参照してください。たとえば、STS コード '0x02'(hex) が返された場合、リモートノード (PLC) とローカルノード間のケーブル接続を確認してください。

関連項目:
エラーコード定義に関する Allen-Bradleyドキュメント

タグで予期しないオフセットが見つかりました。タグはシンボリックプロトコルを使用します。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

タグで予期しないオフセットが見つかりました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

タグで予期しないオフセット/スパンが見つかりました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

プロジェクトのダウンロードが進行中であるかプロジェクトが存在しません。

エラータイプ:
警告

プロジェクトのダウンロードが完了しました。

エラータイプ:
警告

プロジェクトのオンライン編集が検出されました。現在、シンボリックのアドレス指定を使用しています。

エラータイプ:
警告

プロジェクトのオフライン編集が検出されました。現在、シンボリックのアドレス指定を使用しています。

エラータイプ:
警告

デバイスからコントローラプロジェクトをアップロード中に次のエラーが発生しました。シンボリックプロトコルを使用します。

エラータイプ:
警告

デバイスの識別情報を取得できません。すべてのタグがシンボリックプロトコルを使用します。| カプセル化エラー = <コード>。

エラータイプ:
警告

考えられる原因:
要求時にEthernet/IP パケットのカプセル化部分の範囲でデバイスがエラーを返しました。問題が解決するまで、論理モードに設定されているデバイスはシンボリックモードに戻ります。

解決策:
このようなエラーからの回復はドライバーが自動的に試みます。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。エラー 0x02 はドライバー関連ではなくデバイス関連なので除外されます。

関連項目:
カプセル化エラーコード

デバイスの識別情報を取得できません。すべてのタグがシンボリックプロトコルを使用します。| CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
要求時にEthernet/IP パケットの CIP 部分の範囲でデバイスがエラーを返しました。問題が解決するまで、論理モードに設定されているデバイスはシンボリックモードに戻ります。

解決策:
返されたエラーコードによって解決策が異なります。問題が引き続き発生する場合、テクニカルサポートまでご連絡ください。

関連項目:

CIP エラーコード

デバイスの識別情報を取得できません。受信したフレームにエラーが含まれています。すべてのタグがシンボリックプロトコルを使用します。

エラータイプ:
警告

考えられる原因:

1. PC とデバイス間の接続/切断によってパケットに不整列が発生しています。
2. デバイス間のケーブル接続の不良によりノイズが発生しています。
3. 不正なフレームサイズを受信しました。
4. TNS の不一致があります。
5. デバイスから無効な応答コマンドが返されました。
6. このデバイスでは Ethernet/IP が有効になっていません。

解決策:

1. 介入しなくてもドライバーはこのエラーから回復します。このエラーが頻繁に発生する場合、ケーブル接続、ネットワーク、またはデバイス自体に問題がある可能性があります。
2. 通信先のデバイスがイーサネット対応デバイスであることを確認してください。

要求された CIP 接続サイズはこのデバイスによってサポートされていません。自動的に最大サイズにフォールバックします。| 要求されたサイズ = <数値> (バイト)、最大サイズ = <数値> (バイト)。

エラータイプ:
警告

考えられる原因:
要求された CIP 接続サイズはこのデバイスによってサポートされていません。

解決策:
このデバイスによってサポートされているサイズに CIP 接続サイズを変更してください。

関連項目:
Logix 通信パラメータ

タグのインポートファイル名が無効です。ファイルパスは使用できません。

エラータイプ:
警告

考えられる原因:
タグのインポートファイル名にはパスが含まれます。

解決策:
ファイル名からパスを除去します。

デバイスへの読み取り/書き込み要求が中止しました。デバイスプロジェクトからの論理アドレスを更新しています。

エラータイプ:
警告

デバイスへの読み取り/書き込み要求が再開しました。デバイスからの論理アドレスの更新が完了しました。現在、論理アドレス指定を使用しています。

エラータイプ:
警告

タグに書き込めません。書き込まれた値には構文エラーが含まれています。 | タグアドレス = '<アドレス>'、必要なフォーマット = '<フォーマット>'。

エラータイプ:
警告

考えられる原因:
書き込まれた文字列の値は、必要な '<format>' に準拠していません。

解決策:

1. 必要なフォーマットに準拠している文字列の値を、余分な文字や埋め込みの空白なしで書き込んでください。
2. 時間タイプでは、ドライバーは Studio 5000 によってサポートされているものと同じリテラル文字列をサポートしているため、参照として使用できます。たとえば、すべての時間の部分は必要ありませんが、少なくとも 1 つの部分が必要です: T32#0us は使用できますが、T32# は使用できません。

タグに書き込めません。書き込まれた値は範囲外です。 | タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 書き込まれた文字列の値は、コントローラタグのデータ型で許可されている最小値または最大値を超えます。
2. 書き込まれた整数値は、コントローラタグのデータ型で許可されている最小値または最大値を超えています。

解決策:
コントローラタグのデータ型の範囲を超えていない文字列の値または整数値を書き込みます。たとえば、TIME 整数最大値は 2725199999999 です。これは、文字列リテラル 'T#31d_12h_59m_59s_999ms_999us' と関連付けられています; 'T#31d_13h' は範囲外になります。

プログラム情報のアップロード中に無効な属性が検出されました。このプログラムのすべてのタグがシンボリックプロトコルを使用します。|プログラム名 = '<名前>'。

エラータイプ:
警告

考えられる原因:
コントローラのファームウェアは論理プロトコルモードでのプログラムタグへのアクセスをサポートしていません。

解決策:
論理プロトコルモードアクセス用のプログラムタグデータをコントローラレベルのタグに移動する、コントローラ論理を追加します。

データベースエラー。サポートされているプログラムの最大数に達しました。このプログラムに対してタグは作成されません。| プログラム名 = '<名前>'、最大プログラム数 = <数>。

エラータイプ:
警告

考えられる原因:
「データベースのインポート方法: デバイスから作成」を使用してタグを自動生成している状態で、サポートされているプログラムの最大数に達しました。

解決策:

1. 「データベースのインポート方法: インポートファイルから作成」(L5K または L5X から) を使用します。サーバーデバイスが論理プロトコルモード用に構成されている場合、最大数を超えるプログラムに属するタグはシンボリックプロトコルモードを使用することに注意してください。
2. テクニカルサポートまでご連絡の上、この問題についてご報告ください。

サポートされているプログラムの最大数に達しました。このプログラムのすべてのタグはシンボリックプロトコルを使用します。| プログラム名 = '<名前>'、最大プログラム数 = <数>。

エラータイプ:
警告

考えられる原因:
デバイスが論理プロトコルモードに構成されている状態で、サポートされているプログラムの最大数に達しました。

解決策:
テクニカルサポートまでご連絡の上、この問題についてご報告ください。

データベースステータス。非エイリアスタグをインポートしています。

エラータイプ:
情報

データベースステータス。エイリアスタグをインポートしています。

エラータイプ:

情報

データベースステータス。タグプロジェクトを構築しています。お待ちください。| タグプロジェクト数 = <数値>。

エラータイプ:
情報

データベースエラー。最大文字長さを超えているため、タグ名が変更されました。| タグ名 = '<タグ>'、最大長さ = <数値>、新しいタグ名 = '<タグ>'。

エラータイプ:
情報

データベースエラー。最大文字長さを超えているため、配列タグの名前が変更されました。| 配列タグ = '<タグ>'、最大長さ = <数値>、新しい配列タグ = '<tags>'。

エラータイプ:
情報

データベースエラー。プログラムグループの名前が最大文字長さを超えています。プログラムグループの名前が変更されました。| グループ名 = '<名前>'、最大長さ = <数値>、新しいグループ名 = '<名前>'。

エラータイプ:
情報

データベースステータス。コントローラプロジェクトを読み込んでいます。

エラータイプ:
情報

データベースステータス。| プログラムの数 = <数値>、データ型の数 = <数値>、インポートされたタグの数 = <数値>。

エラータイプ:
情報

データベースステータス。OPC タグを生成しています。

エラータイプ:
情報

メモリリソース量が低下しています。

エラータイプ:
情報

不明なエラーが発生しました。

エラータイプ:

情報

データベースステータス。.L5X ファイルからタグをインポートしています。| スキーマリビジョン = '<値>'、ソフトウェアリビジョン = '<値>'。

エラータイプ:
情報

詳細。| IP = '<アドレス>'、ベンダー ID = <ベンダー>、製品タイプ = <タイプ>、製品コード = <コード>、リビジョン = '<値>'、製品名 = '<名前>'、製品シリアル番号 = <数値>。

エラータイプ:
情報

経過時間 = <数値> (秒)。

エラータイプ:
情報

シンボリックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリック配列ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリック配列ブロックキャッシュの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリックインスタンス非ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリックインスタンス非ブロック、配列ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリックインスタンス非ブロック、配列ブロックキャッシュの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリックインスタンスブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

シンボリックインスタンスブロックキャッシュの読み取り回数 = <数値>。

エラータイプ:
情報

物理非ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

物理非ブロック、配列ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

物理非ブロック、配列ブロックキャッシュの読み取り回数 = <数値>。

エラータイプ:
情報

物理ブロックデバイスの読み取り回数 = <数値>。

エラータイプ:
情報

物理ブロックキャッシュの読み取り回数 = <数値>。

エラータイプ:
情報

読み取りタグ数 = <数値>。

エラータイプ:
情報

送信パケット数 = <数値>。

エラータイプ:
情報

受信パケット数 = <数値>。

エラータイプ:
情報

初期化トランザクション数 = <数値>。

エラータイプ:
情報

読み取り/書き込みトランザクション数 = <数値>。

エラータイプ:

情報

1 秒あたり平均送信パケット数 = <数値>。

エラータイプ:
情報

1 秒あたり平均受信パケット数 = <数値>。

エラータイプ:
情報

1 秒あたり平均タグ読み取り回数 = <数値>。

エラータイプ:
情報

1 トランザクションあたり平均タグ数 = <数値>。

エラータイプ:
情報

エラータイプ:
情報

%s | デバイス統計

エラータイプ:
情報

デバイス平均ターンアラウンドタイム = <数値> (ミリ秒)

エラータイプ:
情報

%s | チャネル統計

エラータイプ:
情報

ドライバー統計

エラータイプ:
情報

デバイスタグのインポートが中断しました。

エラータイプ:
情報

インポートファイル '%s' はパス '%s' に見つかりません。

エラータイプ:
情報

コントローラプロジェクトの読み込み中にエラーが発生しました。

エラータイプ:
情報

内部ドライバーエラーが発生しました。

エラータイプ:
情報

同期化中に無効または破損したコントローラプロジェクトが検出されました。後でもう一度試してください。

エラータイプ:
情報

同期化中にプロジェクトのダウンロードが検出されました。後でもう一度試してください。

エラータイプ:
情報

メモリリソース量が低下しています。

エラータイプ:
情報

L5K ファイルが無効であるか破損しています。

エラータイプ:
情報

不明なエラーが発生しました。

エラータイプ:
情報

データベースエラー。PLC5/SLC/MicroLogix デバイスはこの機能をサポートしていません。

エラータイプ:
情報

L5X ファイルが無効であるか破損しています。

エラータイプ:
情報

インポートファイル '<空>' はパス '<空>' に見つかりません。

エラータイプ:
情報

インポートファイル '%s' はパス '<空>' に見つかりません。

エラータイプ:
情報

インポートファイル '<空>' はパス '%s' に見つかりません。

エラータイプ:
情報

XML 要素がポストスキーマの検証に失敗しました。デバイスからのタグのインポートはこのモデルではサポートされていません。代替要素を使用してください。| XML 要素 = '{<名前空間>}<要素>'、サポートしていないモデル = '<モデル>'、代替 XML 要素 = '{<名前空間>}<要素>'。

エラータイプ:
セキュリティ

この値はこのモデルの XML 要素ではサポートされていません。新しい値に自動的に設定します。| 値 = '<値>'、XML 要素 = '{<名前空間>}<要素>'、モデル = '<モデル>'、新しい値 = '<値>'。

エラータイプ:
セキュリティ

Appendices

Select a link from the list below for more information on a specific topic.

Allen-Bradley ControlLogix Ethernet Channel Properties

Below is a list of Allen-Bradley ControlLogix Ethernet channel-level properties.

```
{
"common.ALLTYPES_NAME": "MyChannel",
"common.ALLTYPES_DESCRIPTION": "",
"servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet",
"servermain.CHANNEL_DIAGNOSTICS_CAPTURE": false,
"servermain.CHANNEL_UNIQUE_ID": 4126021724,
"servermain.CHANNEL_ETHERNET_COMMUNICATIONS_NETWORK_ADAPTER_STRING": "",
"servermain.CHANNEL_WRITE_OPTIMIZATIONS_METHOD": 2,
"servermain.CHANNEL_WRITE_OPTIMIZATIONS_DUTY_CYCLE": 10,
"servermain.CHANNEL_NON_NORMALIZED_FLOATING_POINT_HANDLING": 0,
}
```

Allen-Bradley ControlLogix Ethernet Device Properties

Below is a list of Allen-Bradley ControlLogix Ethernet device-level properties.

```
{
"common.ALLTYPES_NAME": "MyDevice",
"common.ALLTYPES_DESCRIPTION": "",
"servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet",
"servermain.DEVICE_MODEL": 0,
"servermain.DEVICE_UNIQUE_ID": 1286734974,
"servermain.DEVICE_CHANNEL_ASSIGNMENT": "Channel1",
"servermain.DEVICE_ID_FORMAT": 0,
"servermain.DEVICE_ID_STRING": "<10.10.110.15>,1,0",
"servermain.DEVICE_ID_HEXADECIMAL": 0,
"servermain.DEVICE_ID_DECIMAL": 0,
"servermain.DEVICE_ID_OCTAL": 0,
```

```
"servermain.DEVICE_DATA_COLLECTION": true,
"servermain.DEVICE_SIMULATED": false,
"servermain.DEVICE_SCAN_MODE": 0,
"servermain.DEVICE_SCAN_MODE_RATE_MS": 1000,
"servermain.DEVICE_SCAN_MODE_PROVIDE_INITIAL_UPDATES_FROM_CACHE": false,
"servermain.DEVICE_CONNECTION_TIMEOUT_SECONDS": 3,
"servermain.DEVICE_REQUEST_TIMEOUT_MILLISECONDS": 1000,
"servermain.DEVICE_RETRY_ATTEMPTS": 3,
"servermain.DEVICE_INTER_REQUEST_DELAY_MILLISECONDS": 0,
"servermain.DEVICE_AUTO_DEMOTION_ENABLE_ON_COMMUNICATIONS_FAILURES": false,
"servermain.DEVICE_AUTO_DEMOTION_DEMOTE_AFTER_SUCCESSIVE_TIMEOUTS": 3,
"servermain.DEVICE_AUTO_DEMOTION_PERIOD_MS": 10000,
"servermain.DEVICE_AUTO_DEMOTION_DISCARD_WRITES": false,
"servermain.DEVICE_TAG_GENERATION_ON_STARTUP": 0,
"servermain.DEVICE_TAG_GENERATION_DUPLICATE_HANDLING": 0,
"servermain.DEVICE_TAG_GENERATION_GROUP": "",
"servermain.DEVICE_TAG_GENERATION_ALLOW_SUB_GROUPS": true,
"controllogix_ethernet.DEVICE_PORT_NUMBER": 44818,
"controllogix_ethernet.DEVICE_CONNECTION_SIZE_BYTES": 500,
"controllogix_ethernet.DEVICE_INACTIVITY_WATCHDOG_SECONDS": 32,
"controllogix_ethernet.DEVICE_ARRAY_BLOCK_SIZE_ELEMENTS": 120,
"controllogix_ethernet.DEVICE_PROTOCOL_MODE": 1,
"controllogix_ethernet.DEVICE_ONLINE_EDITS": true,
"controllogix_ethernet.DEVICE_OFFLINE_EDITS": true,
"controllogix_ethernet.DEVICE_AUTOMATICALLY_READ_STRING_LENGTH": true,
"controllogix_ethernet.DEVICE_DEFAULT_DATA_TYPE": -1,
"controllogix_ethernet.DEVICE_ENABLE_PERFORMANCE_STATISTICS": false,
"controllogix_ethernet.DEVICE_DATABASE_IMPORT_METHOD": 1,
"controllogix_ethernet.DEVICE_TAG_IMPORT_FILE": "Veracode2.L5X",
"controllogix_ethernet.DEVICE_DISPLAY_DESCRIPTIONS": true,
"controllogix_ethernet.DEVICE_LIMIT_TAG_NAMES": false,
"controllogix_ethernet.DEVICE_TAG_HIERARCHY": 1,
"controllogix_ethernet.DEVICE_IMPOSE_ARRAY_ELEMENT_COUNT_LIMIT": false,
"controllogix_ethernet.DEVICE_ARRAY_ELEMENT_COUNT_LIMIT": 2000,
"controllogix_ethernet.DEVICE_CL_ENET_PORT_NUMBER": 44818,
"controllogix_ethernet.DEVICE_REQUEST_SIZE": 232,
"controllogix_ethernet.DEVICE_PERFORM_BLOCK_WRITES": false,
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION": {
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_MODULE": [
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
```

```
255,
255,
255,
255,
255,
255,
255,
255,
255,
255,
255
],
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_INPUT_WORDS": [
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0
],
"controllogix_ethernet.DEVICE_SLOT_CONFIGURATION_OUTPUT_WORDS": [
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0
]
}
}
```

Allen-Bradley ControlLogix Ethernet Tag Properties

Below is a list of Allen-Bradley ControlLogix Ethernet tag properties.

```
{
"common.ALLTYPES_NAME": "MyTag1",
"common.ALLTYPES_DESCRIPTION": "",
"servermain.TAG_ADDRESS": "MyTag1",
"servermain.TAG_DATA_TYPE": 6,
"servermain.TAG_READ_WRITE_ACCESS": 1,
"servermain.TAG_SCAN_RATE_MILLISECONDS": 100,
"servermain.TAG_AUTOGENERATED": true,
"servermain.TAG_SCALING_TYPE": 0,
"servermain.TAG_SCALING_RAW_LOW": 0,
"servermain.TAG_SCALING_RAW_HIGH": 1000,
"servermain.TAG_SCALING_SCALED_DATA_TYPE": 9,
"servermain.TAG_SCALING_SCALED_LOW": 0,
"servermain.TAG_SCALING_SCALED_HIGH": 1000,
"servermain.TAG_SCALING_CLAMP_LOW": false,
"servermain.TAG_SCALING_CLAMP_HIGH": false,
"servermain.TAG_SCALING_NEGATE_VALUE": false,
"servermain.TAG_SCALING_UNITS": ""
}
```

Logix Device IDs

For information on ENI device ID setup, refer to **1761-NET-ENI Setup**.

ControlLogix 5500 Ethernet

The device ID specifies the device IP address, as well as the slot number in which the controller CPU resides. Device IDs are specified as the following:

<IP or hostname>,1,[<optional routing path>],<CPU Slot>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	IP Address or host name.	Decimal	0-255
1	Port ID	Port to backplane.	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the EtherNet/IP interface module and must equal 1 (port to the backplane).	Decimal	*
CPU Slot	Link Address	Slot number of the ControlLogix processor.	Decimal	0-255

For more information, refer to **Connection Path Specification.*

Example
123.123.123.123,1,0

This equates to an Ethernet/IP of 123.123.123.123. The port ID is 1 and the CPU resides in slot 0.

CompactLogix 5300 Ethernet Device ID

The device ID specifies the device IP address, as well as the slot number in which the controller CPU resides. Device IDs are specified as the following:

<IP or hostname>,1,[<optional routing path>],<CPU Slot>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	CompactLogix Ethernet IP Address or host name.	Decimal	0-255
1	Port ID	Port to backplane.	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the Ethernet port and must equal 1 (port to the backplane).	Decimal	*
CPU Slot	Link Address	Slot number of the CompactLogix processor.	Decimal	0-255

For more information, refer to **Connection Path Specification.*

Example
123.123.123.123,1,0

This equates to CompactLogix IP of 123.123.123.123. The port ID is 1 and the CPU resides in slot 0.

FlexLogix 5400 Ethernet Device ID

The device ID specifies the device IP address, as well as the slot number in which the controller CPU resides. Device IDs are specified as the following:

<IP or hostname>,1,[<optional routing path>],<CPU Slot>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	1788-ENBT IP Address or host name.	Decimal	0-255
1	Port ID	Port to backplane.	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the 1788-ENBT interface module and must equal 1 (port to the backplane).	Decimal	*
CPU Slot	Link Address	Slot number of the FlexLogix processor.	Decimal	0-255

For more information, refer to **Connection Path Specification.*

Example
123.123.123.123,1,0

This equates to 1788-ENBT IP of 123.123.123.123. The port ID is 1 and the CPU resides in slot 0.

SoftLogix 5800 Device ID

The device ID specifies the SoftLogix PC IP address, as well as the virtual slot number in which the controller CPU resides. Device IDs are specified as the following:

<IP or hostname>,1,[<optional routing path>],<CPU Slot>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	SoftLogix PC NIC IP Address or host name.	Decimal	0-255
1	Port ID	Port to backplane.	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the EtherNet/IP Messaging module and must equal 1 (port to the virtual backplane).	Decimal	*
CPU Slot	Link Address	Slot number of the SoftLogix processor in the virtual backplane.	Decimal	0-255

For more information, refer to **Connection Path Specification.*

Example
123.123.123.123,1,1

This equates to SoftLogix PC IP Address of 123.123.123.123. The port ID is 1 and the CPU resides in slot 1.

*For information on supplementing a device ID with a routing path to a remote backplane, refer to **Communications Routing**.*

See Also:** **SoftLogix 5800 Connection Notes

1761-NET-ENI Setup

1761-NET-ENI provides a means of communicating with ControlLogix, CompactLogix, FlexLogix, MicroLogix, SLC 500, and PLC-5 Series PLCs on Ethernet with the Allen-Bradley ControlLogix Ethernet ドライバー.

Requirements

MicroLogix, SLC 500, or PLC-5 series PLC supporting Full Duplex DF1 utilizing the CH0 RS232 channel.
1761-NET-ENI Device Series A, B, C, or D.

ControlLogix, CompactLogix or FlexLogix PLC utilizing the CH0 RS232 channel.
1761-NET-ENI Device Series B and newer.

Notes:

1. For communications parameters, database settings, and project/protocol options, ENI ControlLogix, CompactLogix, and FlexLogix users should refer to the "Logix Setup" book in the Table of Contents.
2. To turn on the **CompactLogix Routing** option (located in the utility's **ENI IP Addr** tab), use the ENI / ENIW utility supplied by Allen-Bradley. This was tested on an ENI module with Firmware revision 2.31.

The ENI module has a limited number of TCP connections. As such, users should avoid applications that communicate with the module (such as RSLinx/RSWho) so that connections are available for the driver.

ENI Device ID

The device ID specifies the IP address of the 1761-NET-ENI. Device IDs are specified as the following:

<IP Address>

Designator	Designator Type	Description	Formats	Range
IP Address	N/A	1761-NET-ENI IP address	Decimal	0-255

Example
123.123.123.123

This equates to an ENI IP of 123.123.123.123. Since the device only supports Full Duplex DF1, a node ID is not required.

For more information on communications parameters, refer to ***Logix Communications Parameters***.

Data Highway™ Plus Gateway Setup

DH+ Gateway provides a means of communicating with SLC 500 and PLC-5 series PLC on DH+ with the Allen-Bradley ControlLogix Ethernet ドライバー.

Requirements

EtherNet/IP Interface module.
1756-DHRIO Interface Module with appropriate channel configured for DH+.
SLC500 or PLC-5 series PLC on DH+ network.

Note: DH+ Gateway models do not support automatic tag database generation.

DH+ Gateway Device ID

The device ID specifies the device IP address as well as the DH+ parameters necessary for making a connection. Device IDs are specified as the following:

<IP or hostname>,1,[<optional routing path>],<DHRIO Slot>.<DHRIO Channel>.<DH+ Node ID (dec)>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	IP Address or host name	Decimal	0-255
1	Port ID	Port to backplane	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the EtherNet/IP interface module and must equal 1 (port to the backplane)	Decimal	*
DHRIO Slot	Link Address	Slot number of the 1756-DHRIO interface module	Decimal	0-255
DHRIO Channel		DH+ channel to use	Alpha	A and B
DH+ Node ID		DH+ node ID of target PLC in Decimal Format**	Decimal	0-99

- **For more information, refer to **Connection Path Specification**.*
- ***For more information, refer to "Node ID Octal Addressing" below.*

Example
123.123.123.123,1,2.A.3

This equates to an Ethernet/IP of 123.123.123.123. The DH+ card resides in slot 2: use DH+ channel A and addressing target DH+ Node ID 3 (dec).

Node ID Octal Addressing

The DH+ node ID is specified in Octal format in the PLC and requires a conversion to Decimal format for use in the DH+ Gateway device ID. The node ID can be located in RSWho within RSLinx. It is displayed in Octal format.

Example
DH+ Node 10 (octal) in RSWho = DH+ Node 8 (decimal) in DH+ Gateway device ID.

It is important to verify communications with the proper controller. In the example above, if 10 was entered as the DH+ node ID in the DH+ Gateway device ID, then communications would take place with Node 12 (octal equivalent of 10 decimal) and not Node 10 (octal). If Node 12 (octal) does not exist, then the DHRIO module would return DF1 STS 0x02. This means that the link layer cannot guarantee delivery of the packet. In short, the DH+ node cannot be located on the DH+ network.

- *For information on supplementing a device ID with a routing path to a remote DH+ node, refer to **Communications Routing**.*
- *For more information on communications parameters, refer to **ENI DF1/DH+/ControlNet Gateway Communications Parameters**.*

ControlNet ™ ゲートウェイの設定

ControlNet ゲートウェイは Allen-Bradley ControlLogix Ethernet ドライバーを使用して ControlNet 上の PLC-5C シリーズ PLC と通信する手段を提供します。

要件

イーサネット/IP インタフェースモジュール。
1756-CNB または 1756-CNBR インタフェースモジュール。
ControlNet ネットワーク上の PLC-5C シリーズ PLC。

注記: ControlNet ゲートウェイモデルは自動タグデータベース生成をサポートしていません。

ControlNet ゲートウェイのデバイス ID

デバイス ID では、接続を行うために必要な ControlNet パラメータに加え、デバイスの IP アドレスを指定します。デバイス ID は次のように指定します。

<IP またはホスト名>,1,[<オプションのルーティングパス>],<CNB スロット>.<CNB チャネル>.<ControlNet ノード ID (10 進)>

指定子	指定子のタイプ *	説明	フォーマット	範囲
IP/ホスト名	該当なし	IP アドレスまたはホスト名	10 進	0-255
1	ポート ID	バックプレーンへのポート	10 進	1
オプションのルーティングパス	リンクとポートの複数のペア	イーサネット/IP 通信モジュールからの経路として 1 (バックプレーンへのポート) を指定します	10 進	*
CNB スロット	リンクアドレス	1756-CNB/CNBR インタフェースモジュールのスロット番号	10 進	0-255
CNB チャネル	ポート ID	使用する ControlNet チャネル	英字	A と B
ControlNet ノード ID	リンクアドレス	ターゲット PLC の ControlNet ノード ID (10 進フォーマット)**	10 進	0-99

*詳細については、*接続パスの指定*を参照してください。
**詳細については、以下の「8 進フォーマットでのノード ID のアドレス指定」を参照してください。

例
123.123.123.123,1,2.A.3

これはイーサネット/IP 123.123.123.123 に相当します。ControlNet カードはスロット 2 にあります。ControlNet チャネル A およびアドレス指定ターゲットとして ControlNet ノード ID 3 を使用します。

8 進フォーマットでのノード ID のアドレス指定

PLC では ControlNet ノード ID が 8 進フォーマットで指定されているため、ControlNet ゲートウェイデバイス ID で使用するためには 10 進フォーマットに変換する必要があります。ノード ID は RSLinx 内の RSWho にあります。これは 8 進フォーマットで表示されます。

例

RSWho での CN ノード 10 (8 進) = ControlNet ゲートウェイデバイス ID での CN ノード 8 (10 進)。

適切なコントローラとの通信を確認することが重要です。上記の例では、ControlNet ゲートウェイデバイス ID での ControlNet ノード ID として 10 を入力した場合、ノード 10 (8 進数) ではなくノード 12 (10 進数の 10 に相当する 8 進数) との通信が確立されます。ノード 12 (8 進) が存在しない場合、CNB モジュールは DF1 STS 0x02 を返します。これはリンクレイヤーがパケットの送信を保証できなかったことを意味します。つまり、ControlNet ネットワーク上で ControlNet ノードが見つかりませんでした。

- *デバイス ID にリモート ControlNet ノードへのルーティングパスを追加する方法については、**通信のルーティング**を参照してください。*
- *通信パラメータの詳細については、**ENI DF1/DH+/ControlNet ゲートウェイ通信パラメータ**を参照してください。*

EtherNet/IP ゲートウェイの設定

EtherNet/IP ゲートウェイは Allen-Bradley ControlLogix Ethernetドライバーを使用して EtherNet/IP 上の MicroLogix、SLC 500、および PLC-5 シリーズ PLC と通信する手段を提供します。

要件

2 つ以上の EtherNet/IP インタフェースモジュール (1756-ENBT など)。
EtherNet/IP 接続の MicroLogix、SLC500、または PLC-5 シリーズ PLC。

注記: Ethernet/IP ゲートウェイモデルは自動タグデータベース生成をサポートしていません。

イーサネット/IP ゲートウェイのデバイス ID

デバイス ID では、接続を行うために必要なリモート EtherNet/IP アドレスに加え、ローカルデバイスの IP アドレスを指定します。デバイス ID は次のように指定します。

<IP またはホスト名>,1,[<オプションのルーティングパス>],<ENBT スロット>.<ENBT チャネル>.<リモート IP>

指定子	指定子のタイプ*	説明	フォーマット	範囲
IP/ホスト名	該当なし	ローカル EtherNet/IP インタフェースモジュールの IP アドレスまたはホスト名	10 進	0-255
1	ポート ID	バックプレーンへのポート	10 進	1
オプションのルーティングパス	リンクとポートの複数のペア	EtherNet/IP インタフェースモジュールからの経路として 1 (バックプレーンへのポート) を指定します	10 進	*
ENBT スロット	リンクアドレス	2 つ目の EtherNet/IP インタフェースモジュールのスロット番号	10 進	0-255
ENBT チャネル	ポート ID	使用する Ethernet/IP ポート	英字	A と B
リモート IP アドレス	リンクアドレス	ターゲット PLC のリモート IP アドレス	10 進	0-255

- **詳細については、**接続パスの指定**を参照してください。*

例
123.123.123.123,1,2.A.192.168.1.10

これはローカル IP 123.123.123.123 に相当します。2 つ目の Ethernet/IP カードはスロット 2 にあります。ポート A およびアドレス指定ターゲットとして IP 192.168.1.10 のデバイスを使用します。

注記: デバイス ID を設定する場合は、RSLinx で同じルートを使用してそのデバイスを検出できるかどうかを確認してください。

*デバイス ID にリモート Ethernet/IP デバイスへのルーティングパスを追加する方法については、**通信のルーティング**を参照してください。*
*通信パラメータの詳細については、**ENI DF1/DH+/ControlNet ゲートウェイ通信パラメータ**を参照してください。*

Serial Gateway Setup

Serial Gateway provides a means of communicating with ControlLogix, CompactLogix, FlexLogix, and SoftLogix PLCs on a serial network with the Allen-Bradley ControlLogix Ethernet ドライバー.

Requirements

EtherNet/IP Interface module
Local CPU with a serial port
Remote ControlLogix, CompactLogix, FlexLogix, or SoftLogix CPU with a serial port

Notes:

1. Local and Remote CPUs must be on the same serial network.
2. Serial Gateway models do not support automatic tag database generation.

Serial Gateway Device ID

The device ID specifies the local device IP address as well as the remote device station ID necessary for making a connection. Device IDs are specified as the following:

<IP or hostname>,1,[<Optional Routing Path>],<CPU Slot>.<Serial Port Channel>.<Station ID (dec)>

Designator	Designator Type*	Description	Formats	Range
IP/Host Name	N/A	IP address or host name	Decimal	0-255
1	Port ID	Port to backplane	Decimal	1
Optional Routing Path	Multiple Link, port pairs	Specifies a way out of the EtherNet/IP interface module and must equal 1 (port to the backplane)	Decimal	*
CPU Slot	Link Address	Slot number of the CPU module that contains the serial port used for communications	Decimal	0-255
Serial Port Channel		Serial port channel to use	Alpha	A and B
Station ID		DF1 station ID of target PLC in Decimal Format**	Decimal	0-255

For more information, refer to **Connection Path Specification.*

Example
123.123.123.123,1,0.A.3

This equates to an Ethernet/IP of 123.123.123.123. The CPU card resides in slot 0: use Channel A (serial port) and addressing target station ID 3 (dec).

Notes:

1. For information on supplementing a Device ID with a routing path to a remote serial node, refer to **Communications Routing**.
2. For more information on communications parameters, refer to **Logix Communications Parameters**.
3. When configuring the Device ID, users should verify that the device can be detected using the same route through RSLinx.

MicroLogix 1100 Setup

MicroLogix 1100 Device ID

The Device ID specifies the IP address of the MicroLogix 1100. Device IDs are specified as the following:

<IP or hostname>

Designator	Designator Type	Description	Formats	Range
IP/Host Name	N/A	IP Address or host name	Decimal	0-255

Example
123.123.123.123

This equates to an IP of 123.123.123.123.

*For more information on communications parameters, refer to **ENI DF1/DH+/ControlNet Gateway Communications Parameters**.*

Communications Routing

Routing provides a way to communicate with a remote device over various networks. It can be thought of as a bridge between the local device and a remote device even if they are on two different field bus networks. Access to a remote (destination) backplane allows for direct communication with the supported modules located on this backplane. Supported modules include the following:

- ControlLogix 5500 processor for ControlLogix applications.
- SoftLogix 5800 processor for SoftLogix applications.
- 1756-DHRIO interface module for DH+ Gateway applications.
- 1756-CNB or 1756-CNBR interface module for ControlNet Gateway applications.

A routing path is a series of backplane hops, whose last hop points to the destination backplane. Each hop requires a Logix backplane (not a Logix processor). An individual hop can utilize one of the following networks as its medium:

- ControlNet
- DH+
- TCP/IP (Ethernet/IP)

Important: Routing is not supported for ENI and MicroLogix 1100 models.

Connection Path Specification

The routing path is specified in the device ID. As with non-routing applications, communication originates from the Allen-Bradley ControlLogix Ethernet ドライバー on the PC and is directed at the local Ethernet module. Once at this local Ethernet module, the device ID specifies a way out of the module and onto the backplane, just like with non-routing applications. The routing path directs the message to the desired Logix backplane. The device ID also determines what device is communicated with (such as the ControlLogix processor, SoftLogix processor, DH+ node, or ControlNet node).

The routing path specification begins and ends with the left and right bracket respectively ([ ]). The path itself is a series of port/link address pairs, identical to the communication path syntax in RSLogix 5000 Message Configuration dialog.

Designator Type	Description	Formats	Range
Port ID	Specifies a way out of the interface module in question.*	Decimal	0-65535
Link Address	If the corresponding port is the backplane, the link address is the slot number of the interface module that goes out. If the corresponding port is an interface module port, the link address specifies a destination node as follows. - DH+/ControlNet: node ID - EtherNet/IP communication module: IP address - SoftLogix EtherNet/IP module: IP address	Decimal	0-255

*****For more information, refer to "Port Reference" below.*

Single Hop
IP Address, Port ID0, [Link Address0, Port ID1, Link Address1, Port ID2], Link Address2.

Multi-Hop (N Hops)
IP Address, Port ID0, [Link Address0, Port ID1, Link Address1, Port ID2, Link Address2, ... Port ID(N+1), Link Address(N+1), Port ID(N+2)], Link Address(N+2).

Notes:

1. The last port ID in the path (Port ID2 and Port ID(N+2) for single-hop and multi-hop respectively) must be 1 (port for backplane).
2. Port ID0 must be 1 (port for backplane). Link Address2 and Link Address (N+2) are the slot numbers of the remote Logix processor/1756-DHRIO module/1756-CNB module.

Port Reference

Interface Module	Port 1	Port 2	Port 3
EtherNet/IP Communication Module	Backplane	Ethernet Network	N/A
SoftLogix EtherNet/IP Messaging Module	Virtual Backplane	Ethernet Network	N/A

Interface Module	Port 1	Port 2	Port 3
1756-DHRIO	Backplane	DH+ Network on Ch. A	DH+ Network on Ch. B
1756-CNB	Backplane	ControlNet Network	N/A

Application Notes

1. Messages cannot be routed in or out of the same interface module channel more than once within the path. Doing so results in CIP error 0x01 Ext. error 0x100B.
2. For multiple channel interface modules, messages cannot be routed into and then immediately out of that same module (using different channels), regardless of whether the message is directed to the backplane first or avoids the backplane all together. As previously mentioned, the latter is not supported since each hop requires a ControlLogix backplane. An example would be to route a DH+ message from one DH+ link (such as Channel A of 1756-DHRIO) to another DH+ link (such as Channel B of same 1756-DHRIO) through one 1756-DHRIO-interface module. This is commonly referred to as Remote DH+ messaging and is not supported.

Routing Examples

The routing examples below include the entire device ID minus the IP of the local 1756-ENBT. The perspective of the device ID/routing path is from the local 1756-ENBT Module. Hop descriptions are in the following form:

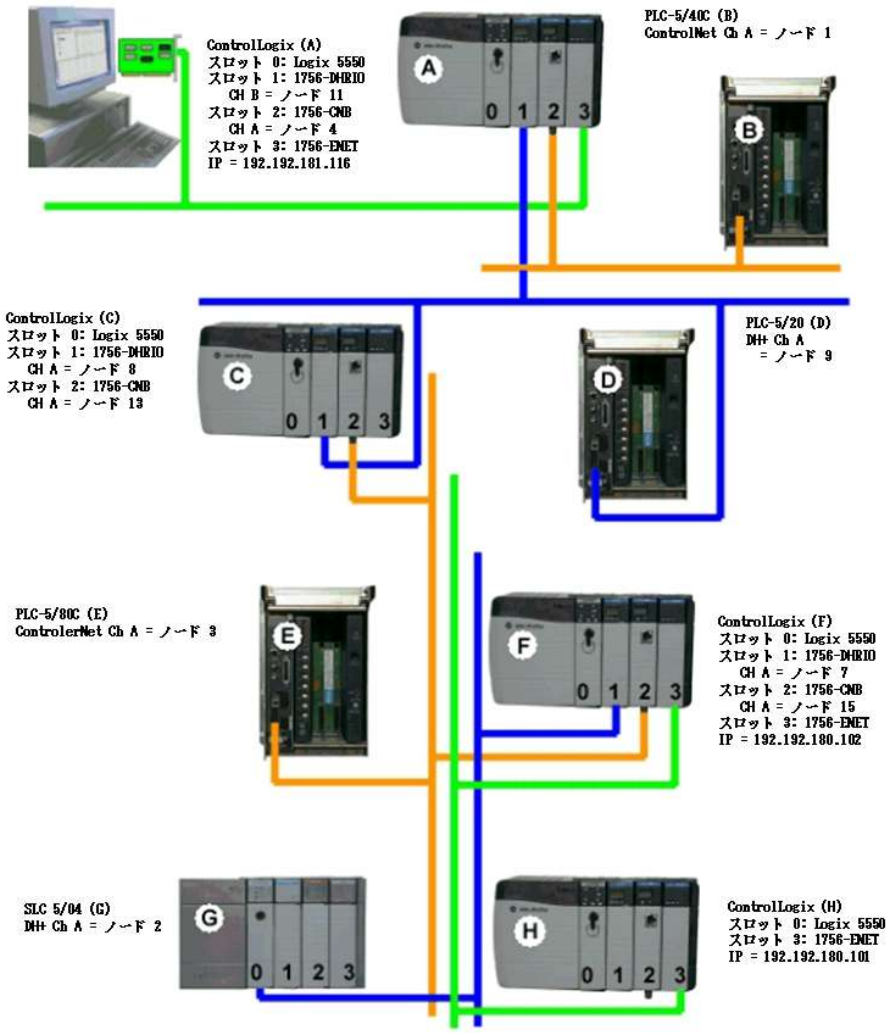
Link Address (N), Port ID(N+1), Link Address(N+1), Port ID(N+2)

*For more information, refer to **Connection Path Specification**. For further details on building a connection/routing path, refer to Allen-Bradley Publication 1756-6.5.14, pp. 4-5 through 4-8.*

In the illustration below, all DH+/ControlNet node IDs are specified in decimal format. The node ID specified in the PLC and displayed in RSWho is in Octal format. Descriptions of the colors are as follows:

- Green = Ethernet
- Blue = DH+
- Orange = ControlNet

*For more information, refer to **Data Highway Plus Gateway Setup** and **ControlNet Gateway Setup**.*



Example 1

Logix5550 to PLC-5 via DH+ Gateway.

Destination Node	Model	Routing	Device ID less IP
PLC-5/20 (D)	DH+ Gateway	No	1,1.B.9

Example 2

Logix5550 to PLC-5C via CN Gateway.

Destination Node	Model	Routing	Device ID less IP
PLC-5/40C (B)	CN Gateway	No	1,2.A.1

Example 3

Logix5550 to Logix5550 via routing over DH+.

Destination Node	Model	Routing	Device ID less IP
Logix5550 (C)	ControlLogix 5550	Yes	1,[1,2,8,1],0

Routing Path Breakdown for Example 3.

Hop	Segment	Description
1	1,2,8,1	Slot 1 (DHRIO) -> Port 2 (DH+ Ch A) -> DH+ Node 8 -> Logix C backplane

Example 4

Logix5550 to PLC-5C via CN Gateway, routing over DH+.

Destination Node	Model	Routing	Device ID less IP
PLC-5/80C (E)	CN Gateway	Yes	1,[1,2,8,1],2.A.3

Routing Path Breakdown for Example 4.

Hop	Segment	Description
1	1,2,8,1	Slot 1 (DHRIO) -> Port 2 (DH+ Ch A) -> DH+ Node 8 -> Logix C backplane

Example 5

Logix5550 to Logix5550 via routing over DH+, ControlNet

Destination Node	Model	Routing	Device ID less IP
Logix5550 (F)	ControlLogix 5550	Yes	1,[1,2,8,1,2,2,15,1],0

Routing Path Breakdown for Example 5.

Hop	Segment	Description
1	1,2,8,1	Slot 1 (DHRIO) -> Port 2 (DH+ Ch A) -> DH+ Node 8 -

Hop	Segment	Description
		> Logix C backplane
2	2,2,15,1	Slot 2 (CNB) -> Port 2 (CN Ch A) -> CN Node 15 -> Logix F backplane

Example 6

Logix5550 to SLC 5/04 via routing over DH+, ControlNet.

Destination Node	Model	Routing	Device ID less IP
SLC 5/04 (G)	DH+ Gateway	Yes	1,[1,2,8,1,2,2,15,1],1.A.2

Routing Path Breakdown for Example 6.

Hop	Segment	Description
1	1,2,8,1	Slot 1 (DHRIO) -> Port 2 (DH+ Ch A) -> DH+ Node 8 -> Logix C backplane
2	2,2,15,1	Slot 2 (CNB) -> Port 2 (CN Ch A) -> CN Node 15 -> Logix F backplane

Example 7

Logix5550 to Logix5550 via routing over DH+, ControlNet, Ethernet.

Destination Node	Model	Routing	Device ID less IP
Logix5550 (H)	ControlLogix 5550	Yes	1, [1,2,8,1,2,2,15,1,3,2,192.192.180.101,1],0

Routing Path Breakdown for Example 7.

Hop	Segment	Description
1	1,2,8,1	Slot 1 (DHRIO) -> Port 2 (DH+ Ch A) -> DH+ Node 8 -> Logix C backplane
2	2,2,15,1	Slot 2 (CNB) -> Port 2 (CN Ch A) -> CN Node 15 -> Logix F backplane
3	3,2,192.192.180.101,1	Slot 3 (ENBT) -> Port 2 -> Remote1756-ENBT IP -> Logix H backplane

Choosing a Protocol Mode

Symbolic Mode

Symbolic Mode represents each client / server tag address in the packet by its ASCII character name.

Benefits	Detriments
1. All the information needed to make a data request lies in the client / server tag's	1. High device turnaround time when processing the symbolic addresses.

Benefits	Detriments
address. 2. Only the data that is being accessed in the client / server tags are requested from the PLC. 3. Backward compatible.	2. Less requests per multi-request packet because the size of each request varies.

Notes:

1. To take advantage of the multi-request packet optimization, as many tags should be represented in a single packet as possible. Since tag addresses are represented by their ASCII character name in the packet, the tag addresses should be as short as possible. For example, "MyTag" is preferred over "MyVeryLongTagNameThatContains36Chars."
2. When the default data type property is set to "Default," automatic tag generation creates tags with a data type that matches the type in the controller.

Logical Modes

Logical Non-Blocking and Logical Blocking encapsulate two read protocols. The protocol used is automatically determined by the driver and is based on the controller revision. The table below summarizes the modes and the protocols to which they map.

Protocol Mode	Read Protocol Used		Write Protocol Used
	FRN V21 and Higher	**FRN V20 and Lower**	**All FRN**
Symbolic	Symbolic (Non-Blocking)	Symbolic (Non-Blocking)	Symbolic
Logical Non-Blocking	Symbol Instance Non-Blocking	Physical Non-Blocking*	Symbol Instance
Logical Blocking	Symbol Instance Blocking	Physical Blocking*	Symbol Instance

*Deprecated in V21.

The information necessary to perform Logical reads is retrieved in a controller project upload sequence performed automatically by the driver. For the sake of brevity, the term "Logical Address" represents the Symbol Instance ID or Physical Address, depending on the protocol used. The Logical Modes avoid the time-consuming address parsing and lookups that are required for every symbolic request.

Note: These Logical Modes are not available to Serial Gateway models.

Logical Non-Blocking Mode

Logical Non-Blocking Mode requests all client / server tags individually and at a fixed size.

Benefits	Detriments
1. Contains the maximum request per multi-request packet because each request is a fixed size. 2. Low device turnaround time because the client / server tags are specified in the packet with the logical address.	Initialization overhead when uploading the project to determine the logical addresses.

Benefits	Detriments
3. Only the data that is being accessed in the client / server tags are requested from the PLC.	

Note: This mode is preferred when the minority of Structure tag members are referenced by a client / server.

Logical Blocking Mode
Logical Blocking retrieves all data for a Logix tag in a single request that may be initiated by only one client / server tag. When the data block is received, it is placed in a cache in the driver and then time stamped. Successive client / server tags that belong to the given Logix tag then get their data from this cache. When all tags are updated, a new request is initiated provided that the cache is not old. The cache is old when the current time > cache timestamp + tag scan rate. If this case holds, another block request is made to the device, the cache is refreshed, and the cycle repeats.

Benefits	Detriments
1. Contents are retrieved on every read. 2. Low device turnaround time because the client / server tags are specified in the packet with the logical address. 3. Contains the maximum request per multi-request packet because each request is a fixed size.	1. Initialization overhead when uploading the project to determine the logical addresses. 2. If the minority of Logix tags are referenced, it is slower than Logical Non-Blocking Mode (because more data is being accessed from the PLC than referenced in the client / server tags).

Note: This mode is preferred when the majority of Structure tag members are referenced by a client / server.

See Also: *Performance Statistics and Tuning*

Symbol Instance vs. Physical Protocol

Symbol Instance reads are CIP requests wherein the CIP Instance ID is used to specify a Native Tag in a read request. In Non-Blocking Mode, the CIP Member ID may be required to fully qualify the path to structure members and array elements. For example, the CIP Instance ID would represent the structure whereas the CIP Member ID represents the member within the structure. Because of the addition of CIP Member IDs required to fully qualify a client / server tag, requests can vary in size. The deeper nesting of structures means more CIP Member IDs are required to specify a tag and fewer requests fit into a single packet. Symbol Instance reads were introduced in FRN V21.

Physical reads are CIP requests wherein the DMA address is used to specify a Native Tag in a read request. In Non-Blocking Mode, the byte offset may be required to fully qualify the path to structure members and array element. For example, the starting DMA address would represent the structure whereas the byte offset represents the member within the structure. Ultimately the start + offset is the DMA address specified in the request: all requests are fixed in size (unlike Symbol Instance reads). No matter how deep structures are nested, the request is the same size in the packet. Physical reads have been deprecated as of FRN V21.

Detecting a Change in the Controller Project

The Allen-Bradley ControlLogix Ethernet ドライバー monitors for project changes and can detect downloads in progress, online edits, and offline edits. When the protocol is set to Logical, users have the option to

synchronize the driver's project image with that of the controller project. Synchronization ensures that the driver uses the current logical address for each Native Tag when performing reads and writes.

- **Downloads in Progress:** The driver monitors for both online and offline edits in every request. It detects if a download occurs while actively reading or writing to Native Tags, then follows a project-change procedure depending on its mode. To enable this synchronization, right-click on the device and select **Properties...**. In the **Logix Options** group, locate either **Synchronize After Online Edits** or **Synchronize After Offline Edits** and select **Yes**.
- **Synchronize After Online Edits**: The driver monitors for online edits in every request. It detects if an online edit occurs with the controller on the following read or write operation, then follows a project-change procedure depending on its mode. To enable this synchronization, right-click on the device and select **Properties...**. In **Logix Options**group, locate **Synchronize After Online Edits** and select **Yes**.
- **Synchronize After Offline Edits**: The driver monitors for offline edits in every request. It detects if an offline edit occurs with the controller on the following read or write operation, then follows a project-change procedure depending on its mode. To enable this synchronization, right-click on the device and select **Properties...**. In **Logix Options** group, locate **Synchronize After Offline Edits** and select **Yes**.

Project Change Procedure (Symbolic Mode)

1. A project change is detected.
2. A message is posted to the Event Log indicating that a change is detected.
3. During project change, the scenario for downloads is as follows:
    - All reads and writes in progress halt and fail.
    - The controller is polled every 2 seconds to monitor for project change completion.
    - The project change is no longer detected.
    - A message is posted to the Event Log indicating that a change is no longer detected.
4. During project change, the scenario for online and offline edits is as follows:
    - The response data is ignored.
    - All reads and writes in progress are retried.
5. The reads and writes resume using Symbolic Mode.

Project Change Procedure (Logical Modes)

1. A project change is detected.
2. A message is posted to the Event Log indicating that a change is detected.
3. During project change, the scenario for downloads is as follows:
    - All reads and writes in progress halt and fail.
    - The controller is polled every 2 seconds to monitor for project change completion.

- The project change is no longer detected.
- A message is posted to the Event Log indicating that the change is no longer detected.

4. During project change, the scenario for online and offline edits is as follows:
   - The response data is ignored.
   - All reads and writes in progress are retried.
5. The reads and writes resume using Symbolic Mode.
6. If the Synchronize with Controller options are enabled:
   - After 30 seconds of Symbolic Mode, the driver uploads (synchronizes) the project from the controller.
   - The reads and writes resume using Logical Mode with the new logical addresses.
7. If the Synchronize with Controller options are disabled, the reads and writes resume using Logical Mode with the old logical addresses.

SoftLogix 5800 Connection Notes

For proper operation, no Ethernet-based drivers (such as Ethernet devices, remote devices via Gateway, and so forth) should be installed in RSLinx on the SoftLogix PC. With one or more Ethernet-based drivers installed, requests return with CIP error 0x5, Ext. error 0x1, and CIP error 0x8.

Connecting to a SoftLogix Soft PLC on the Same PC as the OPC Server

To connect the Allen-Bradley ControlLogix Ethernet ドライバー to a SoftLogix Soft PLC running on the same PC as the server, follow the instructions below.

1. Ensure that there are no Ethernet-based drivers currently running in RSLinx on the PC.
2. Verify that the **EtherNet/IP Message Module** is installed in the SoftLogix virtual chassis.
3. In the **Device Properties**| **General** group, locate the device ID value. It should not be "127.0.0.1, 1, <PLC_CPU_slot>". The Device ID should be set to "<specific_IP_address_of_PC>, 1, <PLC_CPU_slot>".

   For example, if the PC's IP address is 192.168.3.4 and the SoftLogix CPU is in slot 2 of the virtual chassis, then the correct device ID would be "192.168.3.4, 1, 2".

索引

1

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W

X

あ

い

こ

さ

し

た

ち

て

と

ふ

め

も

漢字