

## Produce and Consume Tags (interlock controllers)

This chapter explains how to interlock (produce and consume tags) controllers via a ControlNet network.

Topic	Page
Terminology	71
Set Up the Hardware	72
Determine Connections for Produced and Consumed Tags	73
Organize Tags for Produced or Consumed Data	75
Adjust for Bandwidth Limitations	76
Produce a Tag	77
Consume a Tag	79

Interlocking controllers is the preferred method of sharing scheduled data between controllers when data needs to be delivered regularly, quickly and at a set interval.

### Terminology

A Logix5000 controller lets you produce (broadcast) and consume (receive) system-shared tags.

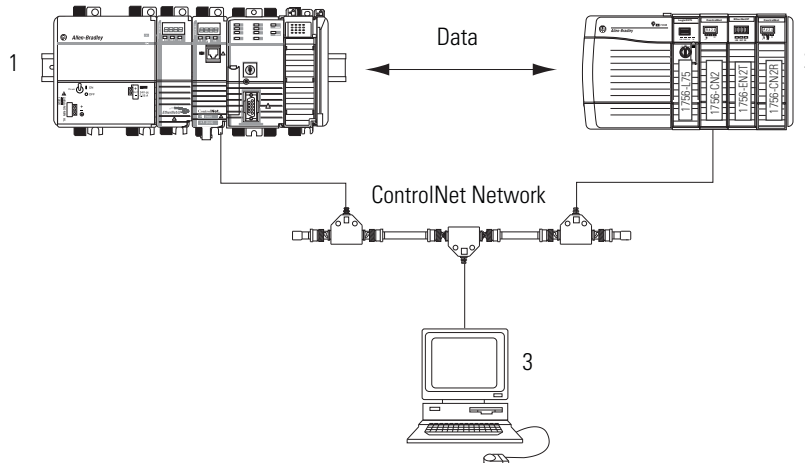
Term	Definition
Produced tag	A tag that a controller makes available for use by other controllers. Multiple controllers can simultaneously consume (receive) the data. A produced tag sends its data to one or more consumed tags (consumers) without using logic. The produced tag sends its data at the RPI of the fastest consuming tag.
Consumed tag	A tag that receives a produced tag's data. The data type of the consumed tag must match the data type, including any array dimensions, of the produced tag. The RPI of the fastest consumed tag determines the rate at which the produced tag is produced.

For two controllers to share produced or consumed tags, they must reside on the same ControlNet network.

## Set Up the Hardware

In this example, the controller in the first chassis produces a tag that is consumed by the controller in the second chassis.

**Figure 11 - Interlocking Controllers Example**



Item	Description
1	<p>Chassis 1 can contain any of these combinations:</p> <ul style="list-style-type: none"> <li>• 1756 ControlLogix controller with a 1756-CN2 or 1756-CN2R communication module in the chassis.</li> <li>• 1756 ControlLogix controller with a 1756-CNB or 1756-CNBR communication module in the chassis.</li> <li>• 1768-L43 CompactLogix controller with a 1768-CNB or 1768-CNBR communication module in the chassis.</li> <li>• 1769-L32C or 1769-L35CR CompactLogix controller.</li> <li>• 1789 SoftLogix controller with a 1784-PCICS communication card.</li> <li>• PowerFlex 700S with DriveLogix controller and a 1788-CNx ControlNet communication card.</li> <li>• Non-Logix5000 controller or other device connected to ControlNet via a ControlNet scanner card.</li> </ul>
2	<p>Chassis 2 can contain any of these combinations:</p> <ul style="list-style-type: none"> <li>• 1756 ControlLogix controller with a 1756-CN2 or 1756-CN2R communication module in the chassis.</li> <li>• 1756 ControlLogix controller with a 1756-CNB or 1756-CNBR communication module in the chassis.</li> <li>• 1768-L43 CompactLogix controller with a 1768-CNB or 1768-CNBR communication module in the chassis.</li> <li>• 1769-L32C or 1769-L35CR CompactLogix controller.</li> <li>• 1789 SoftLogix controller with a 1784-PCICS communication card.</li> <li>• PowerFlex 700S with DriveLogix controller and a 1788-CNx ControlNet communication card.</li> <li>• Non-Logix5000 controller or other device connected to ControlNet via a ControlNet scanner card.</li> </ul>
3	Programming terminal

Make sure of the following:

- The ControlNet communication modules are connected to a scheduled ControlNet network.
- All wiring and cabling are properly connected.
- The communication driver is configured for the programming workstation.

**TIP** If you are sharing tags only between ControlLogix controllers, the controllers are not controlling any I/O modules. You can set the communication format of the 1756-CN2, 1756-CN2R, 1756-CNB, or 1756-CNBR modules in the remote chassis to None. This limits connection usage and network traffic.

## Determine Connections for Produced and Consumed Tags

Logix controllers can produce (broadcast) and consume (receive) system-shared tags that are sent and received via the ControlNet communication module. Each produced and consumed tag requires connections.

**Table 17 - Tag Type and Connections**

Tag Type	Required Connections
Produced	The produced tag requires two connections. The producing controller must have one connection for the produced tag and the first consumer and one connection for each additional consumer (heartbeat). The heartbeat is a small scheduled packet the consumer sends to indicate that it is getting the produced data. As you increase the number of controllers that can consume a produced tag, you also reduce the number of available controller connections for other operations, such as communication and I/O.
Consumed	Each consumed tag requires one connection for the controller that is consuming the tag.

All ControlNet modules support at least 32 connections. The number of available connections limits the number of tags that can be produced or consumed. If the communication module uses all of its connections for I/O and other communication modules, no connections are left for produced and consumed tags.

**Table 18 - Produced and Consumed Tags and Number of Connections**

<b>Controller</b>	<b>Available Connections</b>	<b>Connections Used by a Produced Tag</b>	<b>Connections Used by a Consumed Tag</b>
CompactLogix PowerFlex 700S with DriveLogix software	100	Number of consumers + 1	1
ControlLogix SoftLogix5800	250		
<b>Communication Card</b>	<b>Available Connections</b>	<b>Connections Used by a Produced Tag</b>	<b>Connections Used by a Consumed Tag</b>
ControlNet port on the CompactLogix controller	32	Number of consumers	1
1768-CNB and 1768-CNBR CompactLogix ControlNet modules	48		
1788-CN <sub>x</sub> card in PowerFlex 700S with DriveLogix controller	32 total ControlNet connections, 22 of which can be scheduled and used for producing and consuming tags.		
1756-CN2 and 1756-CN2R series B ControlNet modules in the local chassis of a ControlLogix controller	131 Note that 3 of the 131 connections are always reserved for redundant control. Therefore, 128 connections are available for standard use.		
1756-CNB and 1756-CNBR ControlNet modules in the local chassis of a ControlLogix controller	64 We recommend that you do not use more than 40...48 scheduled connections.		
1784-PCICS card in a SoftLogix5800 controller	127		

## Organize Tags for Produced or Consumed Data

Follow these guidelines as you organize your tags for produced or consumed data (shared data).

**Table 19 - Guidelines for Produced or Consumed Data Tags**

Function	Guidelines			
Create the tags at the controller scope.	You can produce and consume only controller-scoped tags.			
Produce and consume specific tags.	You cannot produce or consume these types: <ul style="list-style-type: none"> <li>• Alias</li> <li>• Axis type</li> <li>• BOOL</li> <li>• Consumed</li> <li>• I/O</li> <li>• INT</li> <li>• Message</li> </ul>			
Use one of these data types: <ul style="list-style-type: none"> <li>• DINT</li> <li>• REAL</li> <li>• Array of DINTs or REALs</li> <li>• User-defined</li> </ul>	To share other data types, create a user-defined data type that contains the required data. Use the same data type for the produced tag and corresponding consumed tag or tags.			
Limit the size of the tag to ≤480 bytes.	If you must transfer more than 480 bytes, create logic to transfer the data in smaller packets or create multiple produce/consume tags.			
To share tags with a PLC-5C controller, use a user-defined data type.	Produce	Integers, BOOLS or combinations of both	Create a user-defined data type that contains an array of INTs with an even number of elements, such as INT[2].	
		Only one REAL value	Use the REAL data type.	
		More than one REAL value	Create a user-defined data type that contains an array of REALs.	
	Consume	Integers	Create a user-defined data type that contains these members:	
			<b>Data type</b>	<b>Description</b>
			DINT	Status BIT 0 <ul style="list-style-type: none"> <li>• 0 PLC5 in PROG mode</li> <li>• 1 PLC5 in RUN mode</li> </ul>
		INT[x], where x is the output size of the data from the PLC-5C controller. If you are consuming only one INT, omit x.	Data produced by a PLC-5C controller	
Use the highest permissible RPI for your application.	If the controller consumes the tag over a ControlNet network, use a binary multiple of the ControlNet network update time (NUT). For example, if the NUT is 5 ms, use an RPI of 5, 10, 20, or 40 ms.			
Combine data that goes to the same controller.	If you are producing several tags for the same controller, group the data in these ways: <ul style="list-style-type: none"> <li>• To reduce the number of connections, group the data into one or more user-defined data types.</li> <li>• To conserve network bandwidth, group the data according to similar update intervals.</li> </ul>			

## Adjust for Bandwidth Limitations

When you share a tag over a ControlNet network, the tag must fit within the bandwidth of the network:

- As the number of connections over a ControlNet network increases, several connections, including produced or consumed tags, may need to share a network update time (NUT).
- A ControlNet node can transmit approximately 500 bytes of scheduled data in a single NUT.

Depending on system size, your ControlNet network may lack the bandwidth for large tags. If a tag is too large for your ControlNet network, make one or more of these adjustments.

**Table 20 - Tag Adjustments**

Adjustment	Description						
Increase the requested packet interval (RPI) of your connections. This is the recommended method.	At higher RPIs, connections can take turns sending data during an update period.						
Reduce your network update time (NUT).	At a faster NUT, fewer connections have to share an update period.						
For a ControlNet 1756-CN2, 1756-CN2R, 1756-CNB, or 1756-CNBR bridge module in a remote chassis, choose the most efficient communication format for that chassis.	<table border="1"> <thead> <tr> <th>Are most of the modules in the chassis nondiagnostic, digital I/O modules?</th> <th>Then choose this communication format for the remote 1756-CN2 or 1756-CNB module</th> </tr> </thead> <tbody> <tr> <td>Yes</td> <td>Rack optimization</td> </tr> <tr> <td>No</td> <td>None</td> </tr> </tbody> </table>	Are most of the modules in the chassis nondiagnostic, digital I/O modules?	Then choose this communication format for the remote 1756-CN2 or 1756-CNB module	Yes	Rack optimization	No	None
	Are most of the modules in the chassis nondiagnostic, digital I/O modules?	Then choose this communication format for the remote 1756-CN2 or 1756-CNB module					
	Yes	Rack optimization					
No	None						
The rack optimization format uses an additional eight bytes for each slot in its chassis. Analog modules or modules that are sending or receiving diagnostic, fuse, timestamp, or schedule data require direct connections and cannot take advantage of the rack-optimized form. Selecting None frees up the eight bytes per slot for other uses, such as produced or consumed tags.							
Separate the tag into two or more smaller tags.	<ol style="list-style-type: none"> <li>1. Group the data according to similar update rates. For example, you could create one tag for data that is critical and another tag for data that is not as critical.</li> <li>2. Assign a different RPI to each tag.</li> </ol>						
Create logic to transfer the data in smaller sections (packets).	For information on how to do this, see the Logix5000 Controllers Common Procedures Programming Manual, publication <a href="#">1756-PM001</a> .						

## Produce a Tag

A Logix5000 controller can produce only controller-scoped, user-created tags in the local controller's tag structure. Logix5000 controllers cannot produce I/O tags or tags aliased to I/O tags.

To produce a tag, perform this procedure.

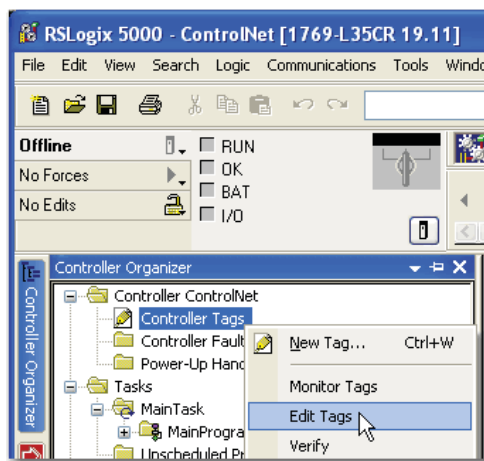
1. Open the RSLogix 5000 project containing the tag you want to produce.

---

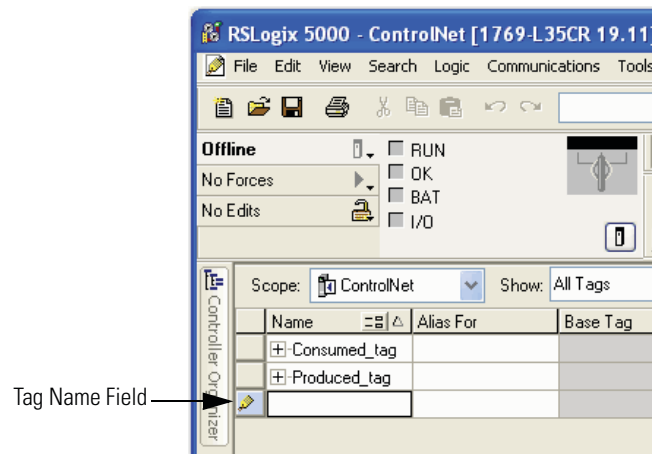
**IMPORTANT** You can create produced tags only when your RSLogix 5000 project is offline.

---

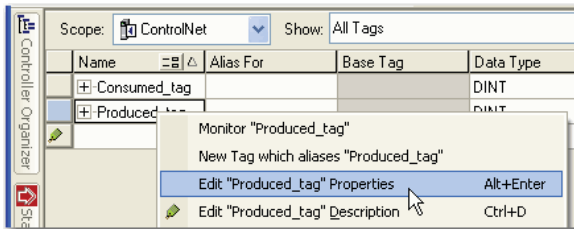
2. Within the Controller Organizer of RSLogix 5000 software, right-click Controller Tags and choose Edit Tags.



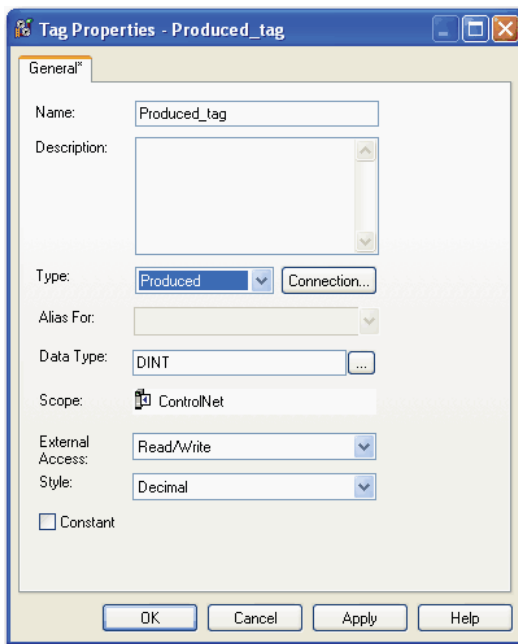
3. From the Controller Tags dialog box, type the name of the new tag in an available Tag Name field.



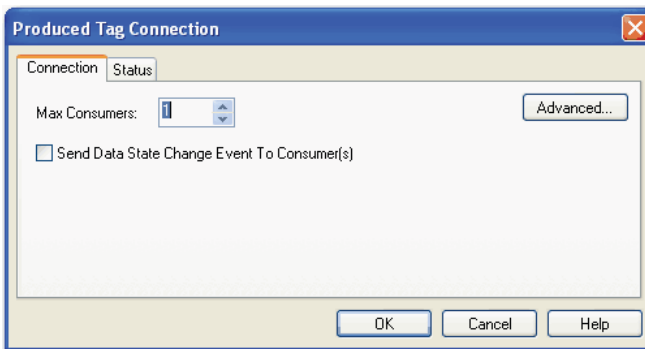
- Right-click the new tag name and choose Edit Tag Properties.



- On the Tag Properties dialog box, from the Type pull-down menu, choose Produced.



- In the Data Type field, type a data type that the controller can produce. A controller cannot produce a tag by using MSG or INT data types.
- Click the Connection tab.





8. In the Maximum Consumers field, type a number of consumers.

If you are unsure of the number of consumers, use a number higher than the actual number of consumers. Unused connections are deducted from the number of available controller connections.

9. Click OK.

---

**IMPORTANT** When your Logix5000 controller produces a tag, any device that interfaces with a ControlNet network can consume the tag. However, when a non-Logix controller, such as a personal computer using a 1784-PKTCS card, consumes the tag produced by a Logix controller, you must perform additional tasks in RSNetWorx for ControlNet software.

---

## Consume a Tag

Logix5000 controllers can consume only controller-scoped user-created tags from another controller's tag structure. The Logix5000 controllers cannot consume I/O tags or tags aliased to I/O tags.

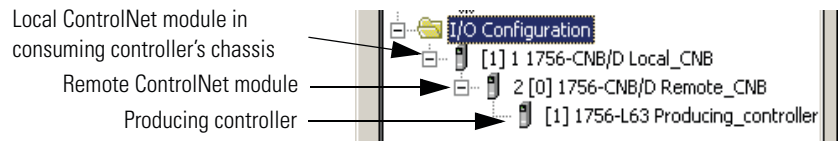
---

**IMPORTANT** You can create consumed tags only when your RSLogix 5000 project is offline.

---

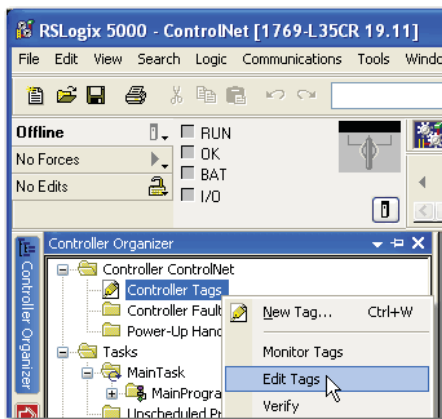
To consume a tag, perform this procedure.

1. Open the RSLogix 5000 project that contains the controller that you want to consume the produced tag.
2. Make sure the controller producing the tag to be consumed is in the consuming controller's I/O configuration, as shown in this example.

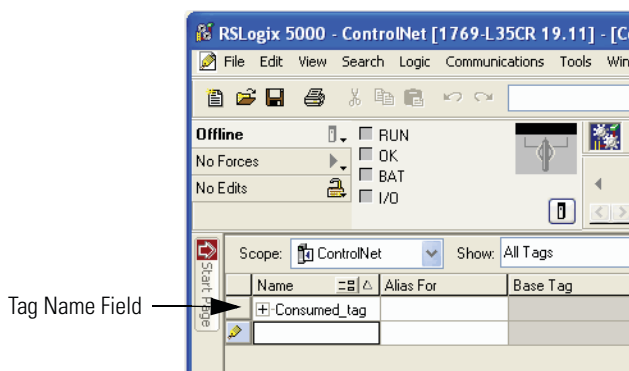


3. Make sure the communication format for the remote ControlNet module is None.

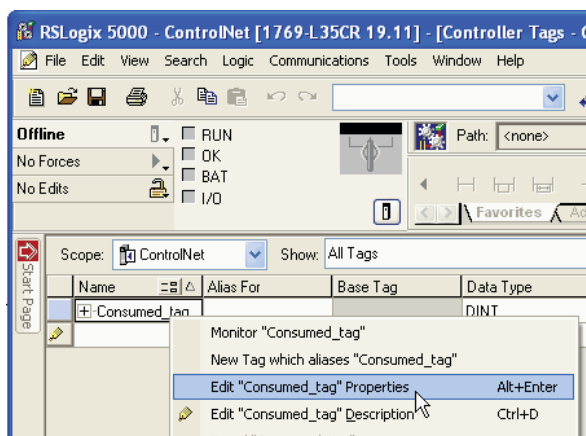
4. Within the Controller Organizer of RSLogix 5000 software, right-click Controller Tags and choose Edit Tags.



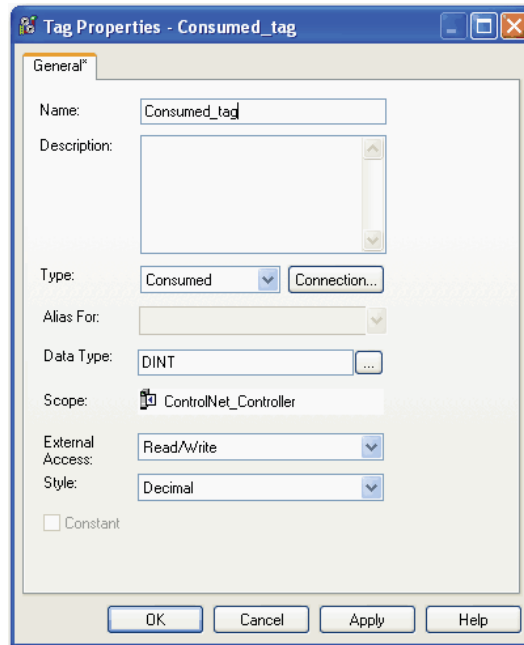
5. From the Controller Tags dialog box, type the name of the new tag in an available Tag Name field.



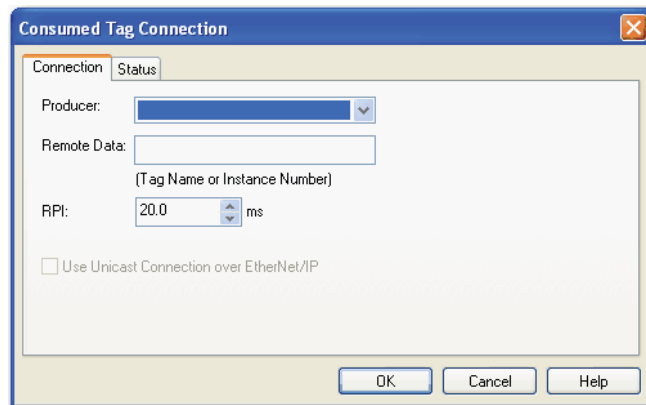
6. Right-click the new tag name and choose Edit Properties.



7. From the Tag Properties dialog box, complete these fields:
  - From the Type pull-down menu, choose Consumed.
  - In the Data Type field, type a data type that the controller can produce. A controller cannot produce a tag by using the MSG or INT data types.



8. Click Connection.
9. From the Consumed Tag Connection dialog box, complete these fields:
  - From the Producer pull-down menu, choose Producing\_controller. This menu contains all possible paths to previously configured controllers in the I/O tree.
  - In the Remote Data field, type the name of the produced tag in the producing controller.
  - In the RPI field, enter the rate at which the tag will be produced.



10. Click OK.
11. Use RSNetWorx for ControlNet software to schedule the network.

**Notes:**

## Messaging

This chapter explains how to use MSG instructions to send data to and receive data from other modules on a ControlNet network.

<b>Topic</b>	<b>Page</b>
Set Up the Hardware	84
Guidelines for MSG Instructions	85
Determine Connections for Messages	86
Enter Message Logic	86
Configure a Message Instruction	88
Stagger the Messages	90

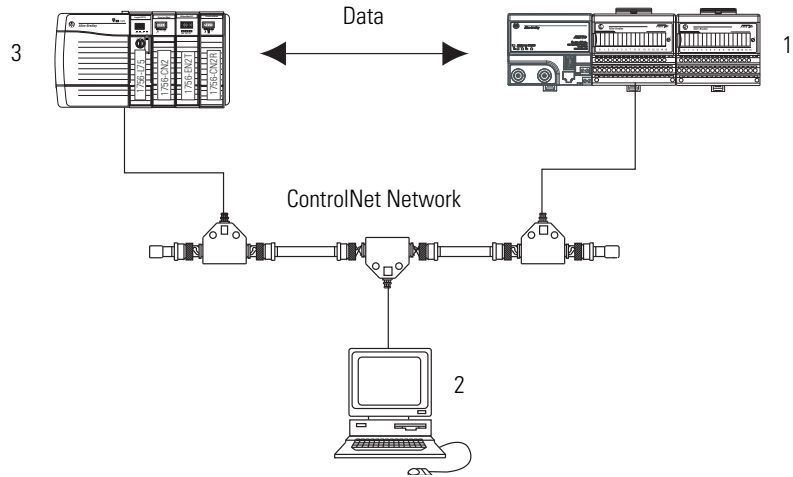
Use peer-to-peer messaging when these conditions apply:

- Data is sent when a specific condition occurs in your application.
- Data is sent at a slower rate than is required by produced and consumed tags.
- Data is sent to devices that communicate only with unscheduled data.

## Set Up the Hardware

In this example, the controller in the local chassis uses a MSG instruction to send a message to another module, which can be a controller, on the ControlNet network.

**Figure 12 - Peer-to-Peer Messaging Example**



Item	Description
1	Remote chassis with any of these configurations: <ul style="list-style-type: none"> <li>• PLCs, SLC, or Logix5000 controllers on a ControlNet or other network</li> <li>• I/O modules, such as ControlLogix analog module configuration data on a ControlNet or other network</li> <li>• 1771 block transfer modules</li> </ul>
2	Programming terminal
3	Local chassis with any of these combinations: <ul style="list-style-type: none"> <li>• 1756 ControlLogix controller with a 1756-CN2 or 1756-CN2R communication module in the chassis</li> <li>• 1756 ControlLogix controller with a 1756-CNB or 1756-CNBR communication module in the chassis</li> <li>• 1768-L43 CompactLogix controller with a 1768-CNB or 1768-CNBR communication module in the chassis</li> <li>• 1769-L32C or 1769-L35CR CompactLogix controller</li> <li>• 1789 SoftLogix controller with a 1784-PCICS communication card</li> <li>• PowerFlex 700S with DriveLogix controller and a 1788-CNx ControlNet communication card</li> <li>• Non-Logix5000 controller or other device connected to ControlNet via a ControlNet scanner card</li> </ul>

**IMPORTANT** The 1769-L32C and 1769-L35CR controllers can produce and consume tags over a ControlNet network to other Logix5000 controllers. However, Compact I/O modules that are local to the 1769-L32C and 1769-L35CR controllers are not accessible to other Logix5000 controllers.

Make sure of the following:

- The ControlNet modules are connected to a ControlNet network.
- All wiring and cabling are properly connected.
- The communication driver is configured for the programming workstation.

## Guidelines for MSG Instructions

Follow these guidelines as you work with message instructions.

**Table 21 - Guidelines for MSG Instructions**

Function	Guidelines
For each MSG instruction, create a control tag.	Each MSG instruction requires its own control tag. This tag contains control elements for messages, such as DN and EN, error codes, and information to execute the message, such as destination path and number of words to transfer: <ul style="list-style-type: none"> <li>• Data type = MESSAGE</li> <li>• Scope = controller</li> <li>• The tag cannot be part of an array or a user-defined data type</li> </ul>
Keep the source or destination data at the controller scope.	A MSG instruction can access only tags that are in the Controller Tags folder (controller scope).
If your MSG is to a module that uses 16-bit integers, use a buffer of INTs in the MSG and DINTs throughout the project.	If your message is to a module that uses 16-bit integers, such as an SLC 500 controller, and it transfers integers, not REALs, use a buffer of INTs in the message and DINTs throughout the project. This increases the efficiency of your project because Logix5000 controllers execute more efficiently and use less memory when working with 32-bit integers (DINTs).
If you want to enable more than 16 MSGs at one time, use some type of management strategy.	If you enable more than 16 MSGs at one time, some MSG instructions may experience delays in entering the queue. To guarantee the execution of each message, you can take these actions: <ul style="list-style-type: none"> <li>• Enable each message in sequence.</li> <li>• Enable the messages in smaller groups.</li> <li>• Program a message to communicate with multiple modules.</li> <li>• Program logic to coordinate the execution of messages.</li> </ul>
Cache connected MSGs that execute most frequently.	Cache the connection for those MSG instructions that execute most frequently, up to the maximum number permissible for your controller revision. This optimizes execution time because the controller does not have to open a connection each time the message executes.
Limit the number of unconnected and uncached MSGs to fewer than the number of unconnected buffers.	The controller can have 10...40 unconnected outgoing buffers: <ul style="list-style-type: none"> <li>• The default number is 10.</li> <li>• If all the unconnected buffers are in use when an instruction leaves the message queue, the instruction errors and does not transfer the data.</li> <li>• You can increase the number of unconnected buffers to a maximum of 40.</li> </ul>

For more information on programming MSG instructions, see the Logix5000 Controllers General Instructions Reference Manual, publication [1756-RM003](#). The individual system user manuals for Logix5000 controllers also provide MSG examples unique to specific controller platforms.

## Determine Connections for Messages

Messages transfer data to other modules, such as other controllers, I/O modules or operator interfaces. Each message uses one connection, regardless of how many modules are in the message path. To conserve connections, you can configure one message to read from or write to multiple modules. Also, you can configure multiple messages for the same path and use only one connection if only one message is active at a time; however, this requires that you write your ladder logic correctly to make sure that only one message is active at any time.

These connected messages can leave the connection open (cache) or close the connection when the message has finished transmitting.

**Table 22 - Message Connections and Communication Methods**

Message Type	Communication Method	Connection Required
CIP data table read or write	CIP	Yes
CIP generic	CIP	Optional <sup>(1)</sup>
Block-transfer read or write	Not applicable	Yes

(1) You can connect CIP generic messages, but for most applications we recommend you leave CIP generic messages unconnected.

## Guidelines for Caching Message Connections

Follow these guidelines to determine whether to cache a connection.

Message Execution	Appropriate Action
Repeated	Cache the connection. This keeps the connection open and optimizes message completion time. Opening a connection during the execution of each message increases execution time.
Infrequent	Do not cache the connection. This closes the connection upon completion of the message, freeing up that connection for other uses.

## Enter Message Logic

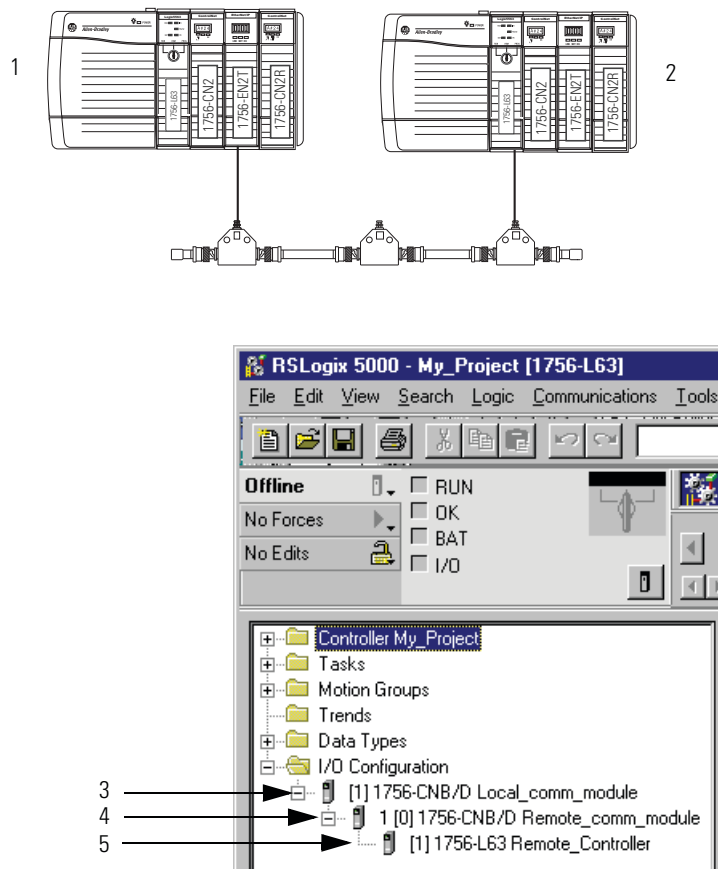
To send or receive data from a ControlNet module via a message, you must program a MSG instruction in the local controller's logic. If the target module is configured in the I/O Configuration folder of the controller, browse to choose the module or manually type the message path in the MSG instruction.



## Add the ControlNet Modules and Remote Devices to the Local Controller's I/O Configuration

Browse to choose the target device of a MSG instruction and add that remote device to the I/O configuration folder of the local controller. Within the I/O configuration folder, organize the local and remote devices into a hierarchy of tree/branch and parent/child.


Figure 13 - I/O Configuration Order for MSG Instruction



Item	Description
1	Local controller and communication module
2	Remote controller and communication modules
3	Local communication module for the local controller
4	Remote communication module for the remote controller
5	Remote controller

For more information on how to add ControlNet modules and remote devices to the local controller's I/O configuration, see [Chapter 4](#).

## Enter a Message

Use relay ladder logic to enter a MSG instruction. Click  to configure the MSG instruction, as shown in the example below.

**EXAMPLE** Enter a MSG instruction as shown below.


**ATTENTION:** If user\_bit and count\_messages.EN = 0 (MSG instruction is not already enabled), then execute a MSG instruction that sends data to another controller.



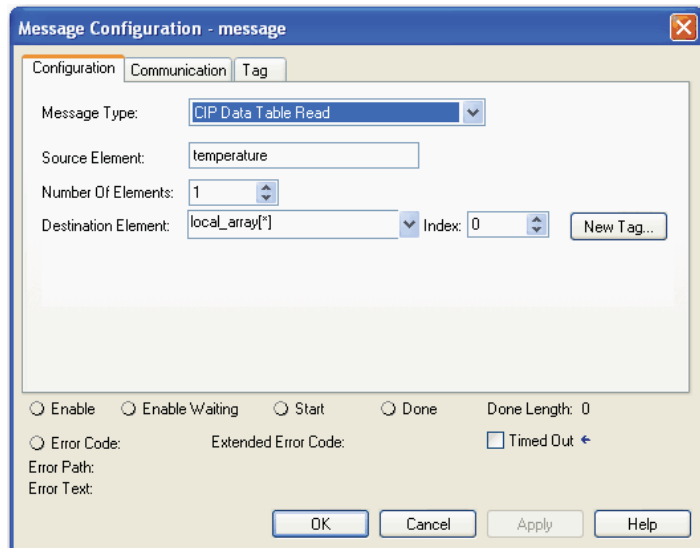
**TIP** We recommend an XIO of the MSG control block tag.en, such as the count\_messages.EN portion of this rung, as an in-series precondition for all message instructions.  
Do not manipulate the control bits of a message instruction.

## Configure a Message Instruction

To configure a MSG instruction, perform this procedure.

1. Click  in the MSG box.

The Module Configuration dialog box appears.



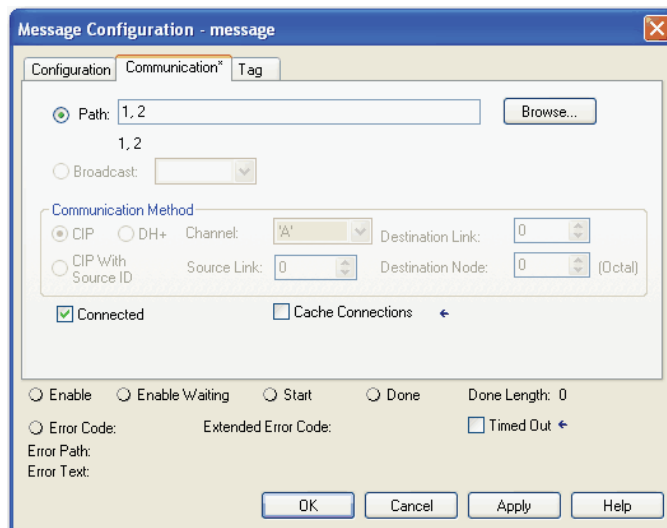
2. From the Message Type pull-down menu, choose a message type.
3. In the Source Element field, type the appropriate information.
4. In the Number of Elements field, enter the number of elements.

- From the Destination Element pull-down menu, choose the instruction's destination element.

The message instruction's destination determines how the message is configured.

Function	Configuration Box	Required Information
Read (receive) the data	Message Type	CIP Data Table Read
	Source Element	First element of the tag that contains data in the other controller
	Number of Elements	Number of elements to transfer
	Destination Tag	First element of the controller-scoped tag in this controller for the data
Write (send) the data	Message Type	CIP Data Table Write
	Source Tag	First element of the controller-scoped tag in this controller that contains the data
	Number of Elements	Number of elements to transfer
	Destination Element	First element of the tag for the data in the other controller

- Click the Communication tab.



- Specify the path of the module for which you sent the message instruction to the I/O configuration tree:
  - If the module has been added, click Browse to choose the path.
  - If the module has not been added, type the path in the Path field.
- Click OK.

## Stagger the Messages

As you add messages to your project, you may have to coordinate the execution of the messages. To avoid errors and assure that each message is processed, follow these rules.

<b>Rule 1</b>	Enable no more than 16 messages at one time, including block transfers.
<b>Rule 2</b>	Enable no more than 10 of these types of messages at one time: <ul style="list-style-type: none"><li>• CIP data table reads or writes that are <b>not</b> cached</li><li>• CIP generic</li><li>• PLC-2, PLC-3, PLC-5, or SLC (all types)</li><li>• Block transfer reads or writes that are <b>not</b> cached</li></ul>

If the number of messages in your application exceeds rules 1 and 2, then stagger the execution of your messages. Here are some options:

- Send each message in sequence.
- Send the messages in groups that are within the limits of rules 1 and 2.
- Program a message to communicate with multiple devices.

## Communicate with PanelView Terminals

This chapter explains how a controller uses a ControlNet communication module to communicate with PanelView software products over a ControlNet network.

<b>Topic</b>	<b>Page</b>
Set Up the Hardware	92
Determine Connections to PanelView Terminals	93
Add a PanelView Terminal	94
Organize Controller Data for a PanelView Terminal	96