EUROMAP 63 Driver

© 2018 PTC Inc. All Rights Reserved.

Table of Contents

EUROMAP 63 Driver

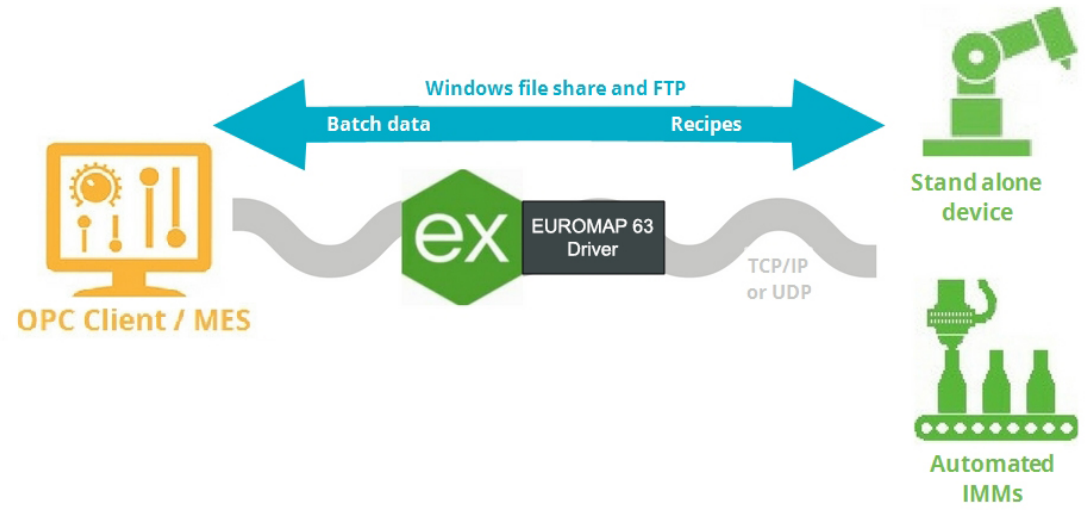
Help version 1.038

CONTENTS

Overview

The EUROMAP 63 Driver provides a reliable way to connect a variety of Injection Molding Machine (IMM) devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications.

The EUROMAP 63 Driver connects directly to EUROMAP devices with today's OPC communication technology, providing a secure channel for information from automated lines and stand-alone equipment, including files from decades-old IMMs. The EUROMAP 63 Driver allows users to monitor the IMM status in real-time, track historical data, and react to poor-quality indicators by extracting the file-based information and publishing it to the OPC layer. The EUROMAP 63 Driver utilizes Windows file share protocols to communicate with the IMM devices over TCP/IP and UDP transport layers.

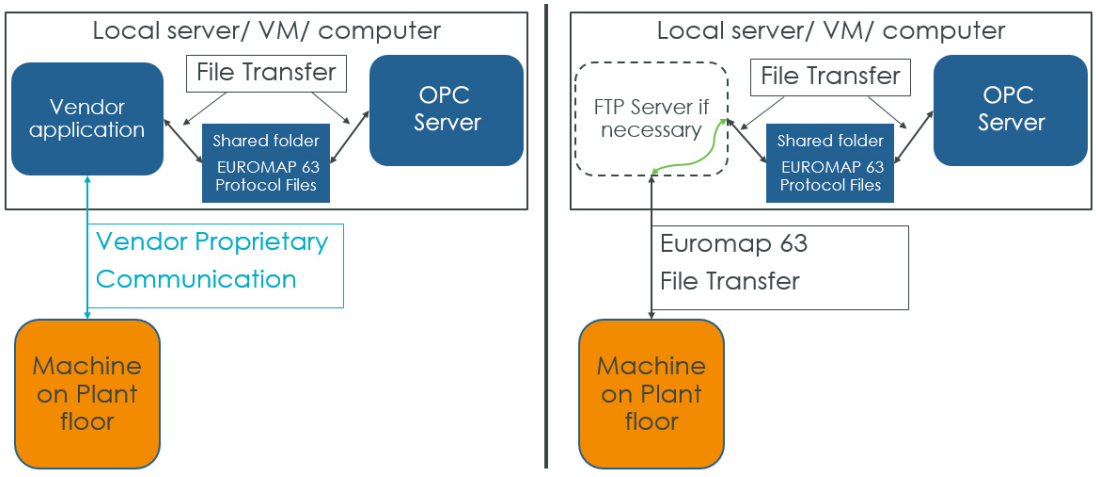


***See Also**: Architecture*

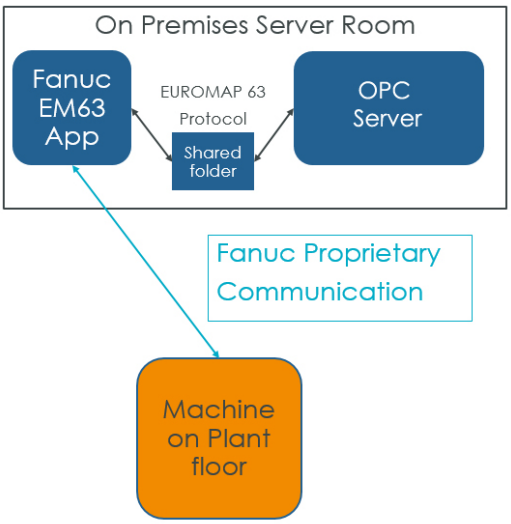
Architecture

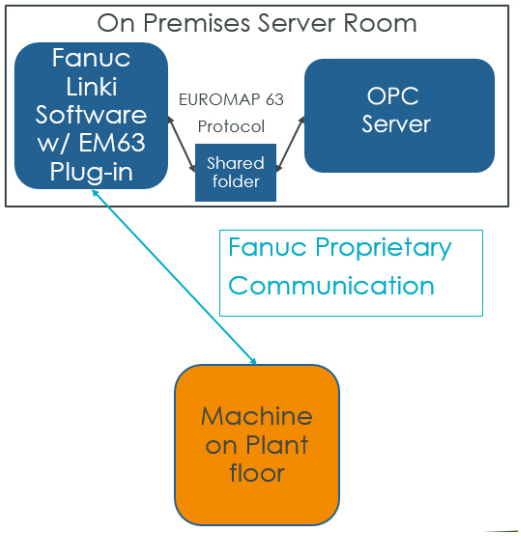
The EUROMAP 63 Driver connects to injection molding machines to monitor status based on the EUROMAP 63 specification. Some injection molding machine manufacturers retrofit these devices with the ability to enable EUROMAP 63, while others provide an application to receive EUROMAP 63 requests, communicate with the device using its proprietary protocol, and deliver EUROMAP 63 responses. For this reason, various configurations are provided below as examples.

Generalized Archetypes

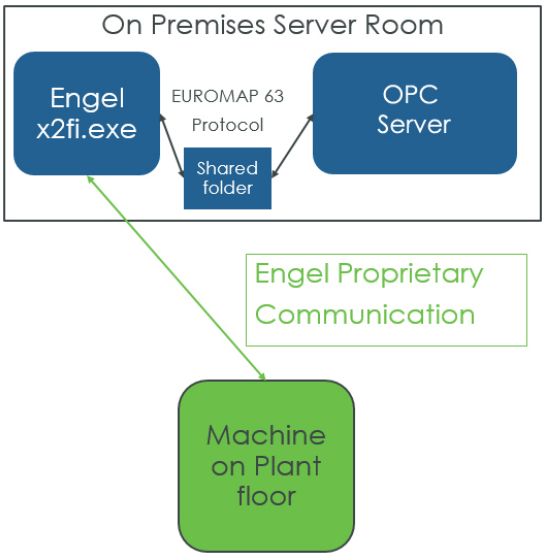


Cincinnati Milacron Roboshot (FANUC)

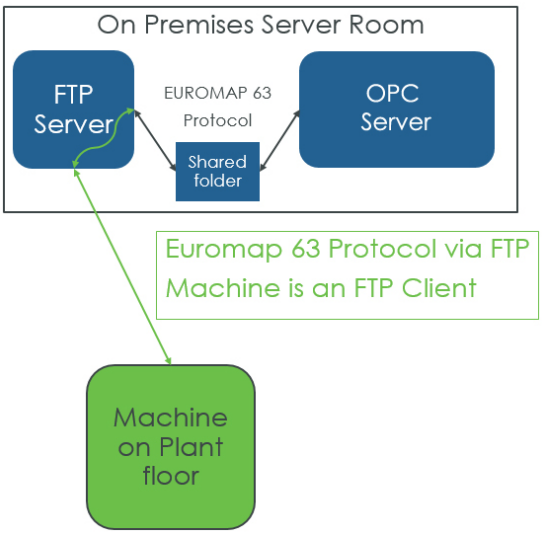




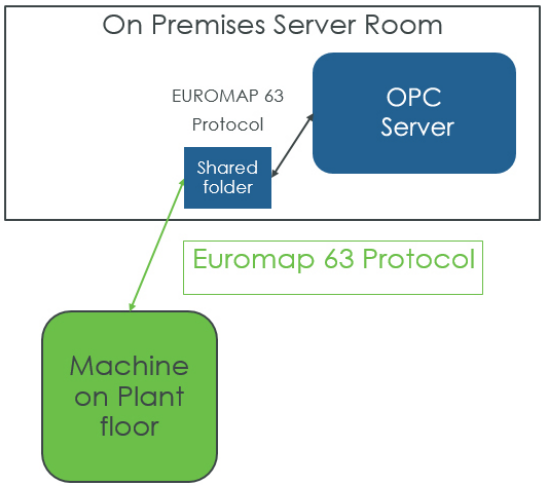
ENGEL



Sumitomo Demag



TOYO Machinery



***See Also**: Optimizing Communications, EUROMAP 63 Connectivity Guides*

Setup

Supported Devices

ENGEL, FANUC, KraussMaffei, Sumitomo Demag, and TOYO Injection Molding Machines

Communication Protocols

EUROMAP 63

Supported Communication Parameters

Session File Directory Path: Local Path, Windows File Share, Mapped FTP Drive

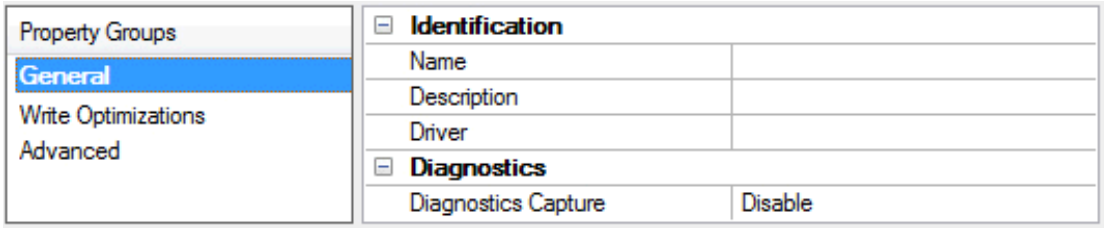
Maximum Number of Channels and Devices

The EUROMAP 63 Driver supports up to 1024 channels and 16 devices per channel. The recommended configuration is one device per channel if monitoring every transaction of an injection molding machine is required.

See Also: *Optimizing Communications*, *EUROMAP 63 Connectivity Guides*

Channel Properties — General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.



Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information.
For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

Description: User-defined information about this channel.
Many of these properties, including Description, have an associated system tag.

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties.
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-

acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics

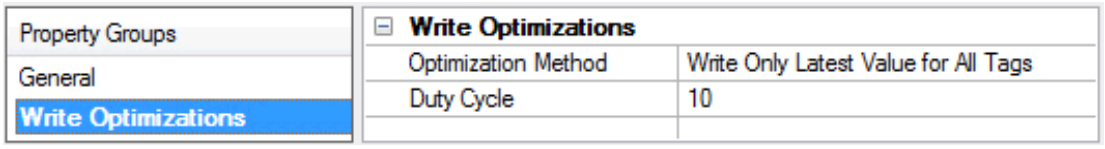
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

Note: This property is not available if the driver does not support diagnostics.

For more information, refer to "Communication Diagnostics" in the server help.

Channel Properties — Write Optimizations

As with any server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used 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.

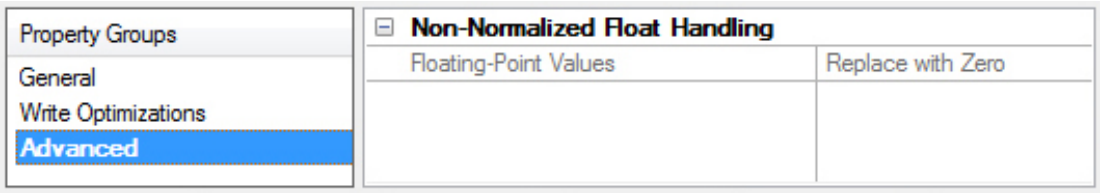
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second 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

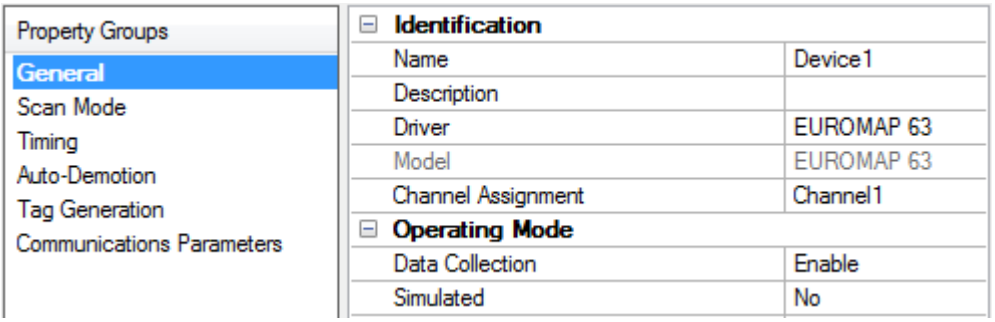
This group is used to specify advanced channel properties. Not all drivers support all properties.



Non-Normalized Float Handling: This property is disabled for this driver.
Note: The EUROMAP 63 Driver handles floats differently than other drivers because data is transferred through text. As a result when the text is converted to a floating point value some numbers may be rounded to the closest value that can be represented by a float. If the text value is not a number, such as infinity, the tag value for the float data types will become bad quality.
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

Device Properties — General

A device represents a single target on a communications channel.



Identification

Name: This property specifies the name of the device. It is a logical user-defined name that can be up to 256 characters long, and may be used on multiple channels.
Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".

For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

Description: User-defined information about this device.
Many of these properties, including Description, have an associated system tag.

Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device. This property specifies the driver selected during channel creation. It is disabled in the channel properties.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.
Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. *For more information, refer to the driver help documentation.*

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: This option places the device into 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.
Notes:

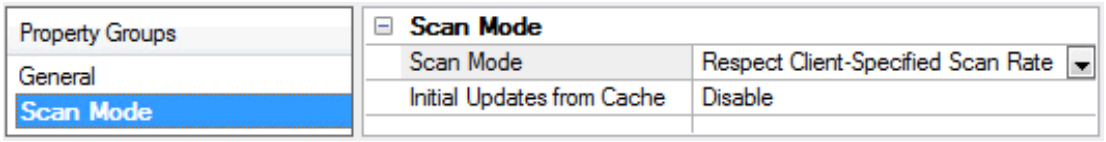
1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

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.



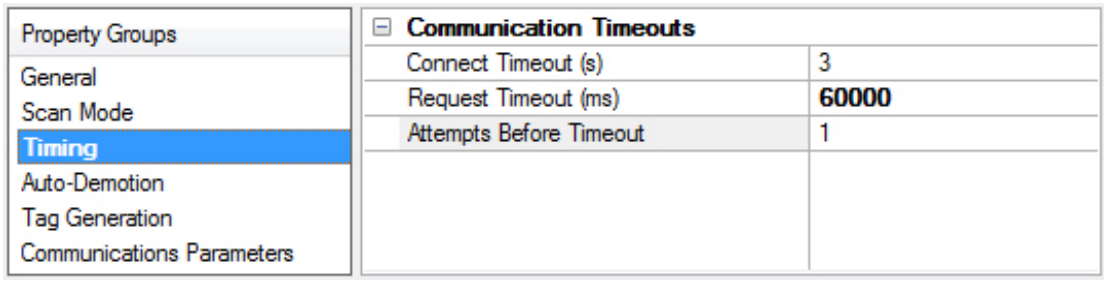
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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 maximum scan rate to be used. 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 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.

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 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

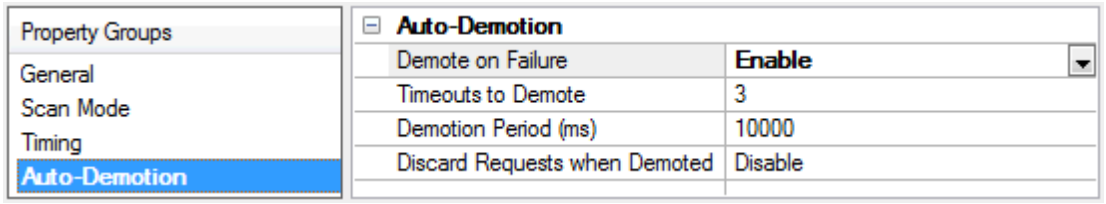
Connect Timeout: This property controls the amount of time allowed to establish a session with a device each time a request to read or write data is sent. Establishing a session includes validating the Session Directory exists and checking that there is an available session number. The valid range is 1 to 30 seconds. The default is 3 seconds.

Request Timeout: This property specifies an interval used to determine how long the driver waits for a response from the target device to complete. The valid range is 1000 to 9,000,000 milliseconds (150 minutes / 2.5 hours). The default value is 60,000 milliseconds.

Attempts Before Timeout: This property specifies how many times the driver retries 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 1. The number of retries configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

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.

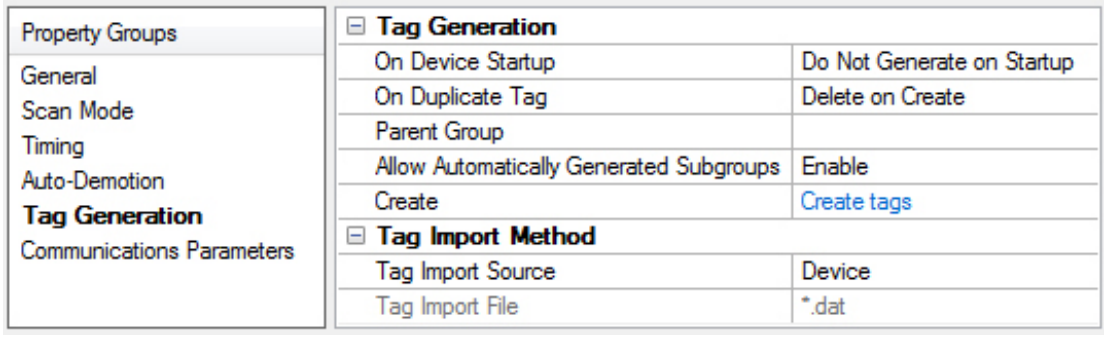
Device Properties — Tag Generation

The automatic tag database generation features make setting up an application a plug-and-play operation. The EUROMAP 63 Driver can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags can be browsed from the clients.

If the target device supports its own local tag database, the driver reads the device's tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.
2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

Note: Automatic tag database generation's mode of operation is completely configurable. *For more information, refer to the property descriptions below.*



Tag Generation

On Property Change: If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation.

On Device Startup: This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

- **Do Not Generate on Startup**: This option prevents the driver from adding any OPC tags to the tag space of the server. This is the default setting.
- **Always Generate on Startup**: This option causes the driver to evaluate the device for tag information. It also adds tags to the tag space of the server every time the server is launched.
- **Generate on First Startup**: This option causes the driver to evaluate the target device for tag information the first time the project is run. It also adds any OPC tags to the server tag space as needed.

Note: When the option to automatically generate OPC tags is selected, any tags that are added to the server's tag space must be saved with the project. Users can configure the project to automatically save from the **Tools** | **Options** menu.

On Duplicate Tag: When automatic tag database generation is enabled, the server needs to know what to do with the tags that it may have previously added or with tags that have been added or modified after the communications driver since their original creation. This setting controls how the server handles OPC tags that were automatically generated and currently exist in the project. It also prevents automatically generated tags from accumulating in the server.

For example, if a user changes the I/O modules in the rack with the server configured to **Always Generate on Startup**, new tags would be added to the server every time the communications driver detected a new I/O module. If the old tags were not removed, many unused tags could accumulate in the server's tag space. The options are:

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server's tag space.
- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.
- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

Note: Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

Parent Group: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically generated tags are added.

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

Create: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database creation.

Note: **Create tags** is disabled if the Configuration edits a project offline.

Tag Import Method

Tag Import Source: Specify the source from which tags will be imported. Select **Device** to import tags from a device using the EUROMAP 63 GETID request. Select File to import tags from a file of the EUROMAP 63 specified GETID response file format. The default source is **Device**.

Tag Import File: Specify or browse to (...) the path and file name of the EUROMAP 63 GETID response file from which to generate tags. This property is disabled if the selected **Tag Import Source** is set to **Device**.

Tip: Once the source is configured, start tag generation using the Create Tags command.

Automatic Tag Generation

The EUROMAP 63 Driver can be configured to automatically generate a list of server tags within the server that correspond to device-specific data. The automatically generated tags are based on the tokens defined in the EUROMAP 63 device and can be browsed from the OPC client. All tokens and tokens in arrays are imported as atomic tags meaning one tag is generated for each token and for each token in an array.

Preparing for Automatic Tag Generation

Online
In the server:

1. Open the device properties of the device for which tags will be generated.
2. Select **Tag Import Method** and select **Device** for **Tag Import Source**.
3. Select **Tag Generation** and, under **Create**, click the blue link to Create tags.

Offline
The EUROMAP 63 Driver uses a file from a EUROMAP 63 device to generate tags without being connected to a device. The file can also be created manually or requested manually and used for offline automatic tag generation.

In the OPC server:

1. Open the device properties of the device for which tags will be generated.
2. Select **Tag Import Method** and select **File** for **Tag Import Source**.
3. Select **Tag Import File** and enter or navigate to the file to be used for importing tags.
4. Select **Tag Generation** and, under **Create**, click the blue link to Create tags.

Tag Generation File

For offline Automatic Tag Generation a file may be specified. The specified file may be created manually or requested from the device manually. The data file imported is a text file. To create the data file, open a text editor and enter a line with the filled out parameters below for each token:

```
<Token Address>, <Token Type>, <Integer Digits>, <Fractional Digits>, <Write Permission>, <Units>, <Description>;
```

Token Address: The address of the token in the device
Token Type: A – Alphanumeric, N – Numeric, B - Boolean
Integer Digits: Number of digits before the decimal
Fractional Digits: Number of digits after the decimal
Write Permission: 0 – Read only, 1 – Read and Write
Units: String representing the units of the data
Description: String describing the measurement data

An example of a line in an offline automatic tag generation data file is:

```
ActCntCyc, N, 16, 0, 0, "Cycles", "Actual Cycle Count";
```

The data file can also be captured by the device by issuing a GETID request to the device. To do this a Session and Job file must be created and placed into the Session directory of the device. The Session file

should be properly named using the EUROMAP 63 naming specification (ex. SESS0000.REQ). The Session file is a text file and the contents of the file should be:

```
00000000 CONNECT;
00000001 EXECUTE "<Job_File_Name>.JOB";
```

The Job file should be properly named and the contents of the Job file should be

```
JOB GETIDJOB RESPONSE "<Job_File_Name>.RSP";
GETID "<Job_File_Name>.DAT";
```

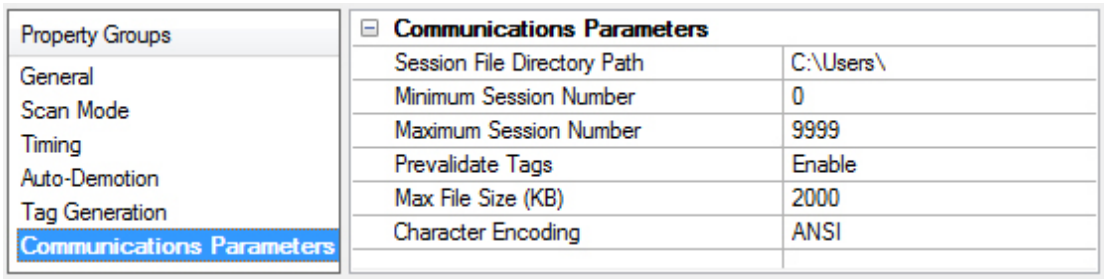
- *For more information on transaction file naming conventions, refer to **Data Exchange Interface**.*
- *For more information on file formats and character encoding, refer to **Communications Parameters**.*

Once both the Session and Job files have been created they should be moved into the Session directory at the same time. The device should respond with three files and it should delete the Session request file once it is done. If not all 3 files are created before the request file is deleted an error likely occurred. Inspection of the Session response and the Job response files should detail this issue. If all three files are created then copy the .DAT file to a location accessible by the server to be used for importing tags. The response files and Job request file should be removed manually after the transaction is complete.

Examples

Line in File	Tag Name and Address	Tag Data Type	Client Access	Tag Description
ActCntCyc,N,5,0,0,"Cycles","Actual Cycle Count";	ActCntCyc	DWord	Read	Actual Cycle Count; Unit: Cycles
ActStsCyc,A,20,0,0,"n/a","Actual Cycle Status";	ActStsCyc	String	Read	Actual Cycle Status; Unit: n/a
ActTmpOil,N,3,1,0,"Celsius","Oil Actual Temperature";	ActTmpOil	Float	Read	Oil Actual Temperature; Unit: Celsius
@VendorSpecificTrigger,B,1,0,1,"","Trigger flag";	@VendorSpecificTrigger	Boolean	Read / Write	Trigger flag; Unit: n/a

Device Properties — Communications Parameters



Session File Directory Path: Specify or browse (...) to the directory path where the session files for device communication are located. Ensure the path is read, write, and delete accessible to the server and the

injection molding machine to prevent communications interference. The session file directory path supports local Windows file share locations. The default value is empty. A device cannot be created without populating this property.

Minimum Session Number: Specify the minimum session number used for session requests. It must be less than or equal to the Maximum Session Number. The range of session numbers, as defined between the minimum and the maximum, should be less than or equal to the maximum number of simultaneous communication sessions that the device can support. The range should not overlap other active communications in the session directory. The value must be between 0 and 9999. The default value is 0.

Maximum Session Number: Specify the maximum session number used for session requests. It must be greater than or equal to the Minimum Session Number. The range of session numbers, as defined between the minimum and the maximum, should be less than or equal to the maximum number of simultaneous communication sessions that the device can support. The range should not overlap other active communications in the session directory. The value must be between 0 and 9999. The default value is 9999.

Prevalidate Tags: Select Enable to instruct the driver to obtain the list of supported parameters on first communication with the device. When enabled, tags that address unsupported parameters are deactivated to prevent failures while reading valid tags. The default is Enabled.
Note: If the **Prevalidate Tags** property is enabled and the attempt to build the parameter list from the device continuously fails, the tags have bad quality. To correct the problem, either disable this property or address the issue causing the GETID request to fail.

Max File Size: Specify the maximum file size that the EUROMAP 63 Driver will attempt to open. The value must be between 50 to 65535 KB. The default setting is 2000 KB.

Character Encoding: Specify the method of character encoding corresponding to the character definition code page defined by the device. Options include UTF-8 and ANSI. The default setting is ANSI. Select UTF-8 only if the response files from the device are encoded using 8-bit Unicode Transformation Format. This may be the case if unexpected or special characters appear in a response error description, the parameter units, the parameter description, the vendor specific parameter name, or the value of a string parameter.

Data Exchange Interface

The EUROMAP 63 Driver Protocol specifies a file-based (ASCII) communication interface that is organized based on the OSI 7 Layer Model. This data exchange interface relies on the implementation of all seven layers.

Refer to the EUROMAP 63 Data Exchange Interface document for the specific definitions of the session, presentation, and application layers that facilitate communication with IMM devices. The document does not specifically define the other four layers, but offers guidelines on the implementation of the underlying network system which provides services required for secure file access.

The session layer is responsible for initiating communication sessions. An IMM requires a unique exclusive session directory location for file exchange between the IMM and this OPC application. The file names used to initiate communications follow a specified format of *SESSxxxx.REQ* and *SESSxxxx.RSP*, where xxxx is a numeric text string that specifies the session number.

Concurrent use of this location by other applications is not recommended.

The presentation layer files generated by the driver have file names comprised of eight characters for the job name and four characters for the file extension. The job name consists of three characters for the Channel ID, one character for the Device ID, one character indicating the transaction type, and three-characters for the Transaction ID. The naming conventions for these files adhere to the following format:

- `<Channel ID><Device ID><Transaction Type><Transaction ID>.JOB` for presentation requests
- `<Channel ID><Device ID><Transaction Type><Transaction ID>.RSP` for presentation responses
- `<Channel ID><Device ID><Transaction Type><Transaction ID>.DAT` for application responses

When the other six layers are configured correctly to allow successful communications between the IMM and this application, the expected application layer files contain the data requested. They end with a .DAT extension.

Shared Session Folder

Device manufacturers vary in interpretation and implementation of the specification. The following table describes an example of the EUROMAP 63 communication flow between the EUROMAP 63 Driver and an injection molding machine (IMM) using the EUROMAP 63 protocol. However, some IMMs write only the session RSP response file before deleting the session REQ request file and then, once the machine's cycle has completed, write the presentation and application RSP response files.

Driver	Shared Session Directory	IMM
Writes	Presentation JOB request file	
Writes	Session REQ request file	Reads
	Presentation JOB request file	Reads
	Session RSP response file	Writes
	Presentation RSP response file	Writes
	Application DAT response file	Writes
	Session REQ request file	Deletes
Reads	Session RSP response file	
Reads	Presentation RSP response file	
Reads	Application DAT response file	

Optimizing Communications

Server Runtime Process Mode

The server runtime can operate as a system service or run interactively in a specific user session. The configured Session File Directory Path(s) on the device(s) in the project must be accessible by the user that is operating the server runtime process.

- If the process is running interactively, the user must have read / write / delete access to each **Session File Directory path**.
- If the process is running as a system service, the log on account for the service must have read / write / delete access to each Session File Directory Path.

***See Also**: The server help system section on Process Modes*

One Device per Channel

Communications protocols like EUROMAP 63 are referred to as channels. Each **channel** defined in the project represents a separate path of execution in the server. Each device represents a single target from which data can be collected.

The driver sequentially issues one request per channel for each active **device** under that channel. Typically, the injection molding machine must wait until a cycle completes before responding to a EUROMAP 63 request. No other requests to other devices on the same channel are issued until the current request completes.

If collecting data from every cycle of multiple injection molding machines is desired, configure only one device per channel. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests.

The driver allows up to 1024 channels, which means it can simultaneously communicate with up to 1024 machines. However, the system configuration and network environment must be considered and tested to verify efficient performance.

Consider the number and power of the processors and the RAM of the system running the runtime process, the location of the session directories, and the number of monitored parameters.

For example, a Windows 7 PC with two CPUs and three GBs of RAM allows the runtime process using 50 devices, each on their own channel and with their own session directory located on multiple networked PCs, to continuously monitor 500 parameters in 50 simulated machines.

Determining Device Request Timeout Property Value

When the driver issues a request to an injection molding machine, it expects the response to be completed before the request timeout has elapsed. A completed response is defined by the deletion of the session request file and the existence of the expected completed response file(s). Typically, the injection molding machine must wait until a cycle completes before responding to the EUROMAP 63 request. Therefore, the **request timeout** should be longer than the expected cycle time of the machine. Additional time may be required if a vendor application or FTP server is necessary for communications.

To determine the typical time the device needs to respond, perform the following steps:

1. Configure a project with a EUROMAP 63 channel with the default property values.
2. Add a device to the channel.
    a. Set the Session File Directory Path to the directory that the IMM using EUROMAP 63 will use and to which the runtime process user has read / write / delete access.
    b. Disable the Prevalidate Tags property.
3. Connect a client to the runtime process.
4. Add a DWord tag with address ActCntCyc.
5. Issue a read of that tag.
6. Watch the session directory for the time the driver creates the session request file (SESSnnnn.REQ) and for the time the device creates and completes the response files (SESSnnnn.RSP <job>.RSP and <job>.DAT).
7. Set the request timeout to allow the device to completely process the request and complete the response.
   Read requests for more parameters require more time.
8. Attempt an automatic tag generation. *See **Automatic Tag Generation** for details.*
9. Add all of the new tags to the client.
10. Issue a read of all of the tags.
11. Watch the session directory for the time the driver creates the session request file and for the time the device creates and completes the response files.
12. Determine if a longer request timeout is required to read all of the parameters.

Setting the timeout too long does not cause issue when the device is communicating. As soon as a request completes successfully, another request is sent, if required. However, when the device is not responding, the driver does not report any issues with communications or missing response data or start another request to this device until the request timeout has elapsed.

If the request timeout is too short and the response has not completed when the request timeout elapses, the driver reports the error, cleans up any request or response files related to the request, sets the tags to bad quality, and moves on to the next request. Response files for the previous request may arrive after the driver has moved on. If the session directory accumulates *.RSP and/or *.DAT files, the request timeout may be too short.

Minimum and Maximum Session Numbers

When adding a new device to a channel, the valid range of session numbers for the device is set to 0 – 9999 by default, as allowed by the EUROMAP 63 protocol. When the driver issues a request, it uses the lowest available session number within the configured range. If there are no session numbers available, the request is not sent and an error message is sent to the event log.

Change the minimum and maximum session number of a device when:

- The range exceeds the maximum number of communication sessions allowed by the machine.
    - Set the minimum and maximum session numbers to ensure the driver does not attempt to start a request when the machine already has the maximum communication sessions in progress.

    - For example, if the machine allows a maximum of ten EUROMAP 63 communication sessions active at one time and the minimum session number of Channel1.Device1 is zero, set the maximum session number to 9.
- Other devices in the project are configured with the same session directory.
    - Set the range of each device so that they do not overlap each other.
    - For example, if Channel1.Device1 uses session numbers between 0 and 9, the minimum session number for Channel2.Device1 should be 10 or greater.
- Other applications are communicating with the EUROMAP 63 protocol in the same session directory.
    - Set the range of each device so that they do not overlap the session range used by other applications.
    - For example, if another application communicates with the same machine in the same session directory using session numbers between 1000 and 1010, the maximum session number of the Channel1.Device1 should be 999 or lower.

Supported EUROMAP 63 Request Commands

CONNECT – Verifies connection between the network station and the session layer.

EXECUTE – Requests a network station to execute a command file.

JOB – Specifies the start of a presentation job and the file specification to write all job command responses.

REPORT – Specifies how the IMM is to generate an application data report. The driver updates tag values from the parameter values provided in this report.

GETID – Requests all available variables from an IMM. The driver uses the information returned in the application data file to automatically generate tags as well as to prevalidate tag addresses, when these options are configured.

SET – Sets the value of a parameter token in the IMM.

Data Types Description

Data Type	Description
Boolean	Single bit
Char	Signed 8-bit value
Byte	Unsigned 8-bit value
Short	Signed 16-bit value
Word	Unsigned 16-bit value
Long	Signed 32-bit value
DWord	Unsigned 32-bit value
Float	32-bit floating-point value
Double	64-bit floating-point value
String	Null-terminated character string

Address Descriptions

The EUROMAP 63 protocol provides standard tokens that can be used with all devices that support EUROMAP 63. Each of these token names can be used as a valid tag address. The following tables provide a list of these defined tokens. For multiple data types, the default is shown in **bold**.

The driver does not currently support Array data types. Standard EUROMAP 63 array element tags are available for use, but are treated the same as all other standard tokens and are requested individually. For example, if ActCfgBrl[1] exists in the project, the server only requests that particular array item index instead of requesting all elements of ActCfgBrl. If the user tries to create this tag with an Array type, such as Word Array, the server forces it to its native type (in this case, Word). The GETID command returns all entries for all dimensional tokens that are available in the IMM. All dimensions are numbered starting with 1 and there should be no spaces inside the square brackets that specify the array index.

Tip: To access vendor specific tokens, which are outside of those provided by the standard, prefix the tag address with the at symbol '@'. For example, @ActInjPrs.

Note: When the device Character Encoding is configured to ANSI, vendor-specific parameter names with any non-ANSI characters are invalid. Invalid ANSI characters cannot be represented in an ANSI file; for example, @MyTag我的标签. If the machine supports the parameter name and also supports reading UTF-8 encoded files, change the device Character Encoding to UTF-8.

Machine Status Tokens

Token / Tag Address	E63 Data Format	Data Type	Notes
SetDescMach	VSTRING(256)	String	Customer Device Description. Can be set to any text string to help identify the machine to the customer.
SetTimMach	CHAR(14)	String	Clock Synchronization – 14 character field formatted as follows: HHMMSSYYYYMMDD Where • HH: The hours value from 00 to 23 • MM: The minutes value from 00 to 59 • SS: The seconds value from 00 to 59 • YYYY: The year • MM: The month from 01 to 12 • DD: The date

Token / Tag Address	E63 Data Format	Data Type	Notes
			from 01 to 31
ActStsMach	CHAR(5)	String	Actual Machine Status. Each character field position is used as follows: • Pos. 1: Machine Status (0: Machine is running (powered on); 1: Machine is not running (powered off)) • Pos. 2: Machine Mode (A: Automatic mode selected; S: Semi-automatic mode is selected; M: Manual mode is selected; U: Setup mode is selected; H: Hold to Run mode is selected; C: Commissioning / Maintenance mode is selected; 0: Unknown state currently selected; I: Idle state currently selected) • Pos. 3: Assist Call (0: No assistance is required; 2: Assistance is required) • Pos. 4: Bad Part (cycle by cycle, is reexamined

Token / Tag Address	E63 Data Format	Data Type	Notes
			each cycle) (0: Last cycle not bad; 1: Last cycle bad) • Pos. 5: Active Alarm (0: No active alarms; 1: Alarms are active)
ActStsCyc	VSTRING(256)	String	Actual Cycle Status. Text string describes reason machine is not currently in cycle.
SetCntCyc	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Number of machine cycles requested for production run
ActCntCyc	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual cycle count
ActCntCycRej	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Production Rejects - Process Control
ActCntPrtRej	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Production Rejects Parts
SetDescJob	VSTRING(256)	String	Job Name / Description
SetDescOp	VSTRING(256)	String	Operator ID
SetDescPrt	VSTRING(256)	String	Part Name / Description
SetDescMld	VSTRING(256)	String	Mold or Tool Name / Description
SetDescMat [InjUnit,Material]	VSTRING(256)	String	Material Name - 1 or more entries for each injection unit. Dimensional limits set by machine.
SetDescMatLot [InjUnit,Material]	VSTRING(256)	String	Material Lot Number - 1 or more entries for each injection unit. Dimensional limits set by machine.
SetRecMld	VSTRING(256)	String	Mold Setup (Recipe) file name
SetCntMld	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Set Mold Count - Number of Cavities Run
ActCntMld	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Act Mold Count - Number of Cavities Run
SetRecMldNxt	VSTRING(256)	String	Setup (Recipe) File

Token / Tag Address	E63 Data Format	Data Type	Notes
			Name of Next Mold to be Run
SetCntPrtBox	VSTRING(256)	String	Part Box Count
SetCntPrt	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Piece Counter
ActCntPrt	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Piece Counter

Barrel Temperature Tokens

Token / Tag Address	E63 Data Format	Data Type	Notes
ActCfgBrl[InjUnit]	CHAR(1)	String	Barrel Configuration - Active Barrels. Single-character field for each injection unit. '0' indicates barrel is OFF, '1' indicates barrel is ON.
SetDescBrlZn[InjUnit, Zone]	VSTRING (256)	String	Barrel Zone Description / Name. Maximum number of injection units and zones specified by machine.
SetCfgBrlZn[InjUnit, Zone]	CHAR(1)	String	Barrel Temperature Zone Configuration. Single character field for each injection unit barrel zone as follows: O: Barrel Zone is OFF; 0: Barrel Zone is Not

Token / Tag Address	E63 Data Format	Data Type	Notes
			Supported; A: Barrel Zone is in AUTO mode; T: Barrel Zone is in TUNING mode; S: Barrel Zone is in STANDBY mode; M: Barrel Zone is in MANUAL mode
SetTmpBrlZn[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone Set Temperatures
ActTmpBrlZn[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone Actual Temperatures
SetTmpBrlZnStb[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone Standby Set Temperatures
SetTmpBrlZnHdev[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone High Deviation Set point; Deviation set points are relative to the SetTmpBrlZn or SetTmpBrlZnStb set points
SetTmpBrlZnLdev[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone Low Deviation Set point. Deviation set points are relative to the SetTmpBrlZn or SetTmpBrlZnStb set points
SetTmpBrlZnHlmt[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone High Limit

Token / Tag Address	E63 Data Format	Data Type	Notes
			Set point. Limit set points are absolute temperature values
SetTmpBrlZnLlmt[InjUnit, Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Barrel Temperature Zone Low Limit Set point. Limit set points are absolute temperature values

Mold Temperature Tokens

Token / Tag Address	E63 Data Format	Data Type	Notes
SetCfgMldZn[Zone]	CHAR(1)	String	Mold Temperature Zone Configuration. Single character field for each mold zone as follows: • O: Mold Zone is OFF; • 0: Mold Zone is Not Supported; • A: Mold Zone is in AUTO mode; • T: Mold Zone is in TUNING mode; • S: Mold Zone is in STANDBY mode;

Token / Tag Address	E63 Data Format	Data Type	Notes
			• M: Mold Zone is in MANUAL mode
SetTmpMldZn[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone Set Temperatures
ActTmpMldZn[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone Actual Temperatures
SetTmpMldZnStb[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone Standby Set Temperatures
SetTmpMldZnHdev[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone High Deviation Set point. Deviation set points are relative to the SetTmpMldZn or SetTmpMldZnStb set points.
SetTmpMldZnLdev[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone Low Deviation Set point. Deviation set points are relative to the SetTmpMldZn or SetTmpMldZnStb set points.
SetTmpMldZnHlmt[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone High Limit Set point. Limit set points are absolute temperature values.
SetTmpMldZnLlmt[Zone]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Mold Temperature Zone Low Limit Set point. Limit set points are absolute temperature values.

Miscellaneous Temperature Tokens

Token / Tag Address	E63 Data Format	Data Type	Notes
ActTmpOil	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Oil Actual Temperature
SetTmpOil	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Oil Set Temperature
ActTmpWtrIn	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Water Intake Actual Temperature
ActTmpWtrOut	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Water Outtake Actual Temperature
ActTmpCab	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Cabinet Actual Temperature
ActTmpMlt	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Melt Actual Temperature

Process Monitoring Parameter Tokens

Token / Tag Address	E63 Data Format	Data Type	Notes
ActTimFil[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual fill time for each injection unit
ActTimPlst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual plasticization time for each injection unit
SetTimCyc	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Overall cycle time set point
ActTimCyc	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual cycle time
ActStrCsh[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual stroke position at cushion for each injection unit
ActVolCsh[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual volume at cushion for each injection unit
ActStrPlst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual plasticization stroke for each injection unit

Token / Tag Address	E63 Data Format	Data Type	Notes
SetStrPlst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization stroke set point for each injection unit
ActVolPlst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual Plasticization Volume for each injection unit
SetVolPlst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Volume Set point for each injection unit
ActDiaScr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual Screw Diameter for each injection unit
SetDiaScr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Screw Diameter Set point for each injection unit
ActStrDcmpPre [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual Decompression Stroke Before Plasticization for each injection unit
SetStrDcmpPre [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Decompression Stroke Before Plasticization Set point for each injection unit
ActVolDcmpPre [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual Decompression Volume Before Plasticization for each injection unit
SetVolDcmpPre [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Decompression Volume Before Plasticization Set point for each injection unit
ActStrDcmpPst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord,	Actual

Token / Tag Address	E63 Data Format	Data Type	Notes
		Float, Double, String	Decompression Stroke After Plasticization for each injection unit
SetStrDcmpPst[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Decompression Stroke After Plasticization Set point for each injection unit
ActVolDcmpPst [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Actual Decompression Volume After Plasticization for each injection unit
SetVolDcmpPst [InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Decompression Volume After Plasticization Set point for each injection unit
ActStrXfr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Transfer Stroke Actual for each injection unit
SetStrXfr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Transfer Stroke Set point for each injection unit
ActVolXfr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Transfer Volume Actual for each injection unit
SetVolXfr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Transfer Volume Set point for each injection unit
ActPrsXfrHyd[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hydraulic Pressure at Transfer Actual for each injection unit
SetPrsXfrHyd[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hydraulic Pressure at Transfer Set

Token / Tag Address	E63 Data Format	Data Type	Notes
			point for each injection unit
ActPrsXfrCav[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Cavity Pressure at Transfer Actual for each injection unit
SetPrsXfrCav[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Cavity Pressure at Transfer Set point for each injection unit
ActPrsXfrSpec[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Specific Pressure at Transfer Actual for each injection unit
SetPrsXfrSpec[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Specific Pressure at Transfer Set point for each injection unit
ActTimXfr[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Time of Transfer Actual (relative from start of cycle)
ActPrsCavMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Cavity Maximum Pressure Actual
ActPrsMachHydMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Machine Hydraulic Pressure Actual Maximum during cycle
ActPrsMachSpecMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Machine Specific Pressure Actual Maximum during cycle
ActSpdPlstMax[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Speed Actual Maximum for each injection unit
ActSpdPlstAve[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Speed Actual Average for

Token / Tag Address	E63 Data Format	Data Type	Notes
			each injection unit
ActVelPlstMax[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Velocity Actual Maximum for each injection unit
ActVelPlstAve[InjUnit]	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Velocity Actual Average for each injection unit
ActFrcClp	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Clamp Force Actual
SetFrcClp	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Clamp Force Set point
ActPrsHldHydMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hold Hydraulic Pressure Actual Maximum
ActPrsHldHydAveMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hold Hydraulic Pressure Actual Average
ActPrsHldSpecMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hold Specific Pressure Actual Maximum
ActPrsHldSpecAveMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Hold Specific Pressure Actual Average
ActPrsPlstHydMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Hydraulic Pressure Actual Maximum
ActPrsPlstHydAveMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Hydraulic Pressure Actual Average
ActPrsPlstSpecMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Specific Pressure Actual Maximum
ActPrsPlstSpecAveMax	NUMERIC	Boolean, Byte, Char, Word, Short, **Long**, DWord, Float, Double, String	Plasticization Specific Pressure Actual Average

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the 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.

Expected response file is missing. | File = '<name>'.

Error Type:
Error

Possible Cause:

1. The device did not provide the response file or provided it with an unexpected name.
2. The file did not move successfully from the session directory to the working directory.
3. The configured Request Timeout is too short.

Possible Solution:

1. Verify the device follows EUROMAP 63 specification.
2. Verify the user has access to both the session and working directory paths.
3. Verify the machine cycle time and adjust the configured Request Timeout accordingly.

Failed to parse response file. | File = '<name>',

Error Type:
Error

Possible Cause:

1. The configured Request Timeout is too short.
2. The file is empty.
3. The file contains data that is not in the expected EUROMAP 63 format.
4. Missing end of command character.
5. Missing or invalid command identifier.
6. Missing or invalid keyword.
7. Malformed response information.
8. Missing or invalid error class.
9. Missing or invalid error code.
10. Malformed error description.

11. Missing or invalid date of occurrence. Should be in yyyymmdd format.
12. Missing or invalid time of occurrence. Should be in hh:mm:ss format.
13. Missing or invalid CSV header.
14. Missing or invalid line of CSV data.
15. Number of fields in CSV header do not match number of fields in CSV data.

Possible Solution:

1. Verify the machine cycle time and adjust the configured Request Timeout accordingly.
2. Verify the device follows EUROMAP 63 specification.
3. If the vendor software controlling the press generates a log file, refer to that file for more details on the problem.

Error opening file. | File = '<name>', OS Error = '<OS supplied message>'.

Error Type:
Error

Possible Cause:

1. The file is locked for access.
2. The path no longer exists.
3. The configured Request Timeout is too short.

Possible Solution:

1. Verify the device has closed the file before deleting the session request file.
2. Review the OS supplied message to diagnose the problem.
3. Verify the machine cycle time and adjust the configured Request Timeout accordingly.

Unable to read tag. | Address = '<address>',

Error Type:
Error

Possible Cause:

1. The device did not provide the data for the tag in the CSV file.
2. Converting the provided data to the tag data type failed.

Possible Solution:

1. Verify tag address matches a token name in the device.
2. Verify the tag data type matches the expected data type of the token in the device.

An error was returned from the device. | Command = '<command>', Class = '<class>', Code = '<code>', Description = '<description>'.

Error Type:
Error

Possible Cause:

1. Message was corrupted.
2. Device does not support the request type.
3. Unexpected EUROMAP 63 implementation.
4. Unable to create/open file.
5. Device does not support one or more parameters in the request.

Possible Solution:

1. No corrective actions may be needed if subsequent requests succeed.
2. Note the details in the error message and refer to the hardware vendor manual to diagnose the problem.

Unable to communicate with the device. Session directory does not exist. | Path = '<path>'.

Error Type:
Error

Possible Cause:

1. The user supplied an invalid session directory path.
2. The user does not have access to the path.

Possible Solution:

1. Verify the path is valid for the target device.
2. Verify the user has access to the path

Unable to communicate with the device. Exceeded the allowed range of session numbers for the device. | Minimum Session Number = '<min session number>', Maximum Session Number = '<max session number>'.

Error Type:

Error

Possible Cause:

1. Unexpected session request and session response files exist in the session directory.
2. Another application is creating session request files in the same session directory.
3. Configured range of session numbers for the device is too small.

Possible Solution:

1. Remove unexpected session request and session response files from the session directory.
2. Only allow one application to create session request files in the session directory.
3. Verify the minimum session number and maximum session number allowed for the device.

Failed to complete transaction due to internal driver error.

Error Type:
Error

Possible Cause:
An unknown error occurred.

Possible Solution:
Attempt the operation again or contact technical support.

Unable to read tag. Unknown parameter. Tag deactivated. | Address = '<address>'

Error Type:
Error

Possible Cause:

1. The Prevalidate Tags device property is enabled.
2. The parameter is not available in the device.

Possible Solution:
Verify tag address matches a token name in the device.

Response file byte count is too large. | File = '<name>', Size (KB) = '<size>'.

Error Type:
Error

Possible Cause:

1. The device has returned a response file that has an unreasonable number of bytes.
2. The Max File Size device property is too small.

Possible Solution:

1. Verify the device is returning the correct amount of data. Reduce the expected amount of data returned in the file.
2. Verify the expected maximum file size.

Unable to write tag. Unknown parameter. Tag deactivated. | Address = '<address>'

Error Type:
Error

Possible Cause:

1. The Prevalidate Tags device property is enabled.
2. The parameter is not available in the device.

Possible Solution:
Verify tag address matches a token name in the device.

Unable to write tag. An error was returned from the device. | Address = '<address>', Class = '<class>', Code = '<code>', Description = '<description>'.

Error Type:
Error

Possible Cause:

1. Message was corrupted.
2. Device does not support the request type.
3. Value out of range for the parameter.
4. Permission denied.
5. Device does not recognize the parameter.

Possible Solution:

1. No corrective actions may be needed if subsequent requests succeed.
2. Note the details in the error message and refer to the hardware vendor manual to diagnose the problem.

Unable to communicate with the device. Access to session directory is denied. | Path = '<path>'.

Error Type:
Error

Possible Cause:
The user does not have access to the session directory path.

Possible Solution:
Verify the user has access to the session directory path.

Unable to write tag. Value contains UTF-8 characters but device Character Encoding is ANSI. | Address = '<address>', Value = '<value>'.

Error Type:
Error

Possible Cause:
Value contains a UTF-8 character, but the device Character Encoding property is set to ANSI mode.

Possible Solution:
Configure the device Character Encoding property to UTF-8.

The transaction has been aborted.

Error Type:
Warning

Possible Cause:

1. The device has been deleted from the server while the transaction was in progress.
2. The server runtime shut down while a transaction was in progress.

Deleted remaining EUROMAP 63 files from the session directory on startup.

Error Type:
Informational

Possible Cause:

1. EUROMAP 63 specific files exist in the Session directory from a previous transaction that terminated abnormally.
2. EUROMAP 63 specific files exist in the Session directory that could not previously be deleted due to access rights.

Possible Solution:

1. Ensure proper shut down.
2. Ensure proper access rights.

Completed retrieval of parameters from device. | Count = '<count>'.

Error Type:
Informational

Possible Cause:

1. The Prevalidate Tags device property is enabled.
2. The parameter list is obtained when first communicate with the device to allow tag address validation.
3. Tag addresses that are not supported in the device will not be included in read requests.

Deleted remaining EUROMAP 63 files from the session directory on transaction reset.

Error Type:
Informational

Possible Cause:

1. The driver reached its maximum transaction number, reset it to 0, and deleted the stranded EUROMAP 63 specific files from the Session directory of this device.
2. The configured Request Timeout is too short. Response files from machine arrived after the driver completed a transaction.
3. EUROMAP 63 specific files exist in the Session directory from a previous transaction that terminated abnormally.
4. EUROMAP 63 specific files exist in the Session directory that could not previously be deleted due to access rights.

Possible Solution:

1. Verify the machine cycle time and adjust the configured Request Timeout accordingly.
2. Ensure proper shut down.
3. Ensure proper access rights.

Index

A

C

D

E

F

G

H

I

M

N

O

P

R

S

T

U