Universal Device Driver

Table of Contents

Universal Device Driver

Help version 1.014

CONTENTS

Overview

The Universal Device Driver provides a reliable way for end users to connect to a wide variety of Ethernet protocols and devices using profiles developed with the Profile Library Plug-in. The Universal Device Driver provides an interface for mapping existing profiles to devices in the environment through communication channels. The Profile Library Plug-in allows users to create the JavaScript-based scripts that define the functions and parameters necessary for the driver to communicate with a device. In addition to the script, the profile contains a universally unique identifier (UUID) used to link the channel to the profile and standard server properties, such as the profile name and description.

For more information about profiles, consult the Profile Library Plug-in help file.

Architecture

The Universal Device Driver works in conjunction with the Profile Library Plug-in to enable users to create custom drivers that communicate with devices in the environment.

Service

Installing the Universal Device Driver adds a new server component, the Script Engine. The service is an isolated environment to run the V8 JavaScript engine. Using an external service for the JavaScript engine protects the server runtime from unexpected JavaScript failures or problems. The user does not interact with the service; all interaction is handled by the server. The service is started when the driver requires a user script to execute and stops five minutes after use.

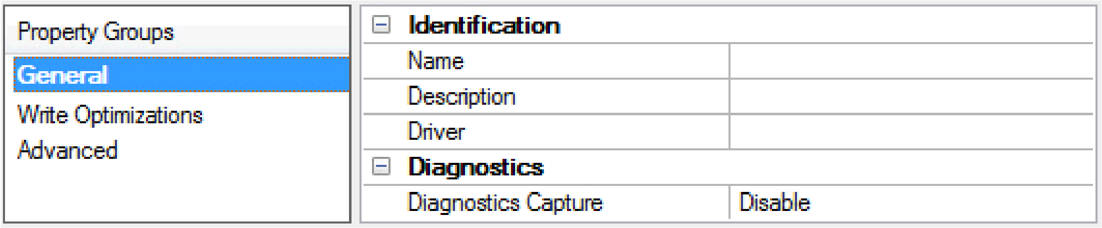
General Operation

A Universal Device Driver can be configured either via the Configuration API Service or through the server configuration user interface. In addition to many of the standard server properties, Universal Device Driver a contains a **Profile** section under channel properties.

When building a channel, the ID of the profile for communicating with a device must be specified. This requires that a profile exist before attempting to create a channel.

Channel Properties — General

This server supports the use of multiple simultaneous 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.



Property Groups
- General
- Write Optimizations
- Advanced

⊟ **Identification**	
Name	
Description	
Driver	
⊟ **Diagnostics**	
Diagnostics Capture	Disable

Identification

Name: Specify the 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. The property is required for creating a channel.
● *For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.*

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

Driver: Specify the 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. The property is required for creating a channel.
● **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. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.

Diagnostics

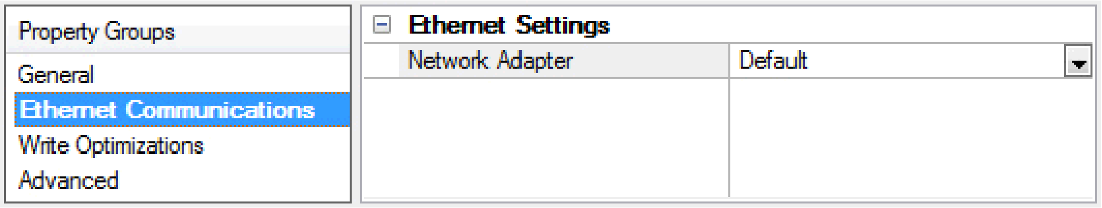
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications allows the usage of statistics tags that provide feedback to client applications regarding the operation of the channel. Because the server's diagnostic features require a minimal amount 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" and "Statistics Tags" in the server help.

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.



Ethernet Settings

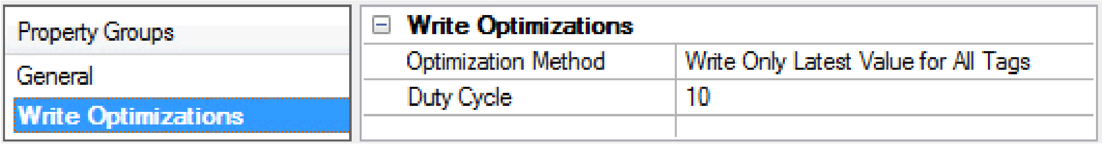
Network Adapter	Default

Ethernet Settings

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

Channel Properties — Write Optimizations

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



Write Optimizations

Optimization Method	Write Only Latest Value for All Tags
Duty Cycle	10

Write Optimizations

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

- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.

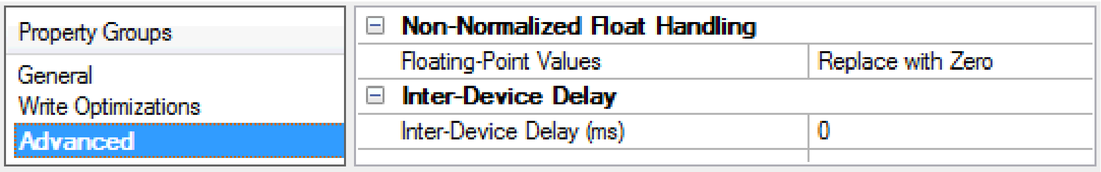
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

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

Channel Properties — Advanced

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



⊟ Non-Normalized Float Handling	
Floating-Point Values	Replace with Zero
⊟ **Inter-Device Delay**	
Inter-Device Delay (ms)	0

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

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

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

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

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

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

Channel Properties — Profile

Profile properties are specific to Universal Device Driver channels.

ID: Specify a unique identifier used to link device channels to profiles defined in the Profile Library. The ID must be in a GUID format. Changing the ID updates the associated profile in the Profile Library Plug-in. This property can be changed at any time, including while there are active client references on the channel, but bad quality data is reported while the channel is re-linking to the profile. Once the channel is successfully linked to the profile, the driver runs all the tag addresses through the profile's ValidateAddress function. Any that fail validation in the profile's ValidateAddress function continue to report bad quality data. All tags that pass validation and successfully parse a response frame from the device return good quality data when the profile re-linking is complete.

A profile must exist in the Profile Library Plug-in and the GUID generated there before it can be assigned to a channel.

Device Setup

A device can be configured through the Configuration API Service or through the server configuration user interface.

Supported transport types currently only include Ethernet solicited.

Supported properties for Ethernet solicited include host / port and IP protocol (TCP/IP) for connecting to the target device.

The Universal Device Driver allows a maximum of 1024 channels with one device per channel.

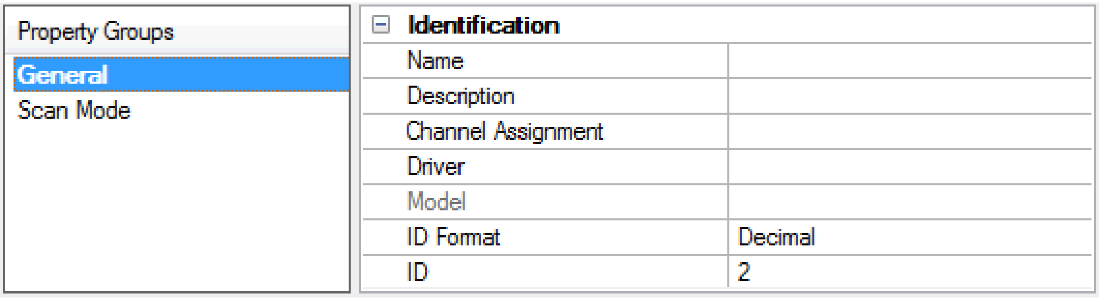
Channel-level settings apply to all devices configured on that channel.

Notes:

- A profile must exist before a channel can be created.
- In "Device Properties — General" below the device ID is set to 1 as this is an Ethernet driver with a limit of one device per channel.

Device Properties — General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.



Identification	
Name	
Description	
Channel Assignment	
Driver	
Model	
ID Format	Decimal
ID	2

Identification

Name: Specify 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.

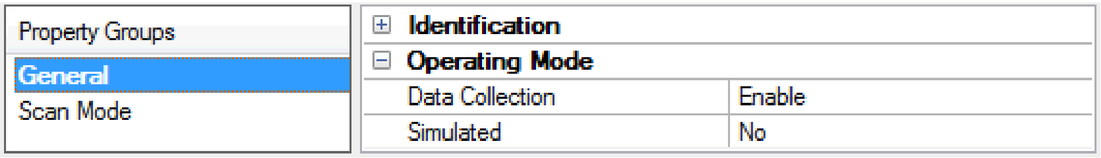
Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name 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: Specify the user-defined information about this device.
Many of these properties, including Description, have an associated system tag.

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

Driver: Selected protocol driver for this device.

Operating Mode



Data Collection	Enable
Simulated	No

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

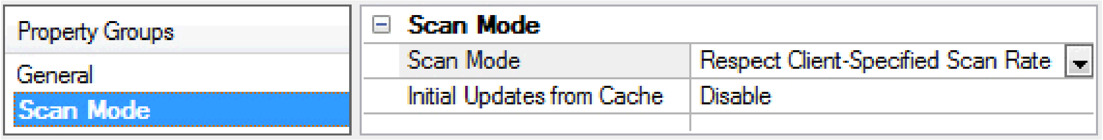
Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client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



Property Groups

- General
- Scan Mode

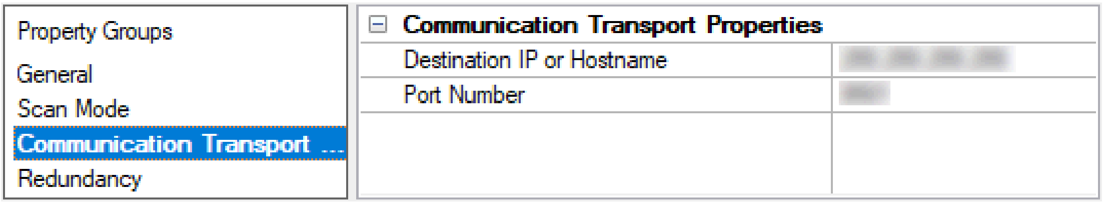
Scan Mode	
Scan Mode	Respect Client-Specified Scan Rate
Initial Updates from Cache	Disable

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

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



Property Groups

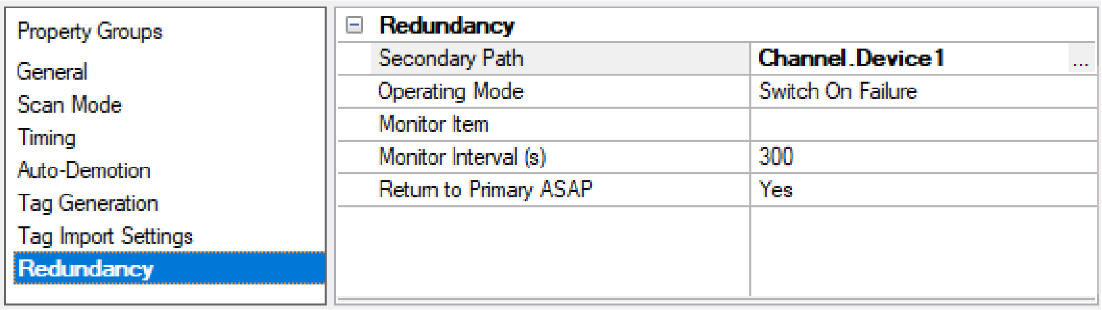
- General
- Scan Mode
- Communication Transport ...
- Redundancy

Communication Transport Properties	
Destination IP or Hostname	[redacted]
Port Number	[redacted]

Destination IP or Hostname: Specifies the destination IP or host name of the device that communicates with the driver.

Port Number: Identifies which port the driver uses to communicate with the device.

Device Properties — Redundancy



Redundancy	
Secondary Path	Channel.Device1
Operating Mode	Switch On Failure
Monitor Item	
Monitor Interval (s)	300
Return to Primary ASAP	Yes

Redundancy is available with the Media-Level Redundancy Plug-In.

*Consult the website, a sales representative, or the **user manual** for more information.*

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.

Invalid profile link. | Profile ID = '<id>'.

Error Type:
Error

Possible Cause:
There is no profile that matches the identifier currently assigned to the driver.

Connection to the service failed. | Channel = '<name>', host = '<host>', port = '<port>', service name = '<name>'.

Error Type:
Error

Possible Cause:
The service has been disabled or the port is in use.

Possible Solution:
Enable the service or reassign the port using the system administration utility.

Failed to start service. | Channel name = '<name>', service name = '<name>'.

Error Type:
Error

Possible Cause:
The service has been disabled or the binary has been tampered with.

Possible Solution:
Enable the service or reinstall the product.

Profile registration failed because the profile syntax is bad or is missing required functions. | Channel name '<name>'.

Error Type:
Error

Possible Cause:
The profile script syntax is bad or is missing required functions.

Possible Solution:

Correct the syntax errors or implement the missing functions.

Script failure. | Result = '<result value>'.

Error Type:

Error

Possible Cause:

1. Return type was invalid
2. Returned object was missing fields
3. Field types were invalid
4. Field is an enumeration, but the value does not match the enumerated type

Possible Solution:

Ensure that the script returns a JavaScript object with the required fields.

Failed to convert value from script to '<data type>' for address '<address>'.

Error Type:

Error

Possible Cause:

The value provided is not convertible to the requested data type

Possible Solution:

Verify that the data type is valid for this address.

Could not validate address because profile invocation failed. | Address = '<address>'

Error Type:

Error

Possible Cause:

The service is not running or processing the profile timed out.

Possible Solution:

Ensure that the service is enabled and that the profile execution time is not excessive.

State machine unknown (required script function GetDriverInfo is missing or returned a bad result). | Channel name '<name>'.

Error Type:

Error

Possible Cause:

1. The profile is missing required script function GetDriverInfo.
2. Required script function GetDriverInfo returned a bad result.

Possible Solution:
Correct the syntax errors or implement the missing functions.

Could not build a send frame because the script execution failed. | Address = '<address>', profile ID = '<id>'.

Error Type:
Error

Possible Cause:
The profile or script is invalid or took too long to process.

Possible Solution:
Ensure that the profile is valid, and that the script execution time is not excessive.

Failed to process received frame. | Channel = '<name>', host = '<host>', port = '<port>', service name = '<name>'.

Error Type:
Error

Possible Cause:
The ParseMessage script function failed to execute.

Possible Solution:
Re-examine the runtime logic of this function.

Failed to generate write request. | Channel = '<name>', host = '<host>', port = '<port>', service name = '<name>'.

Error Type:
Error

Possible Cause:
The BuildMessage script function failed to execute.

Possible Solution:
Re-examine the runtime logic of this function.

Failed to process write response. | Channel = '<name>', host = '<host>', port = '<port>', service name = '<name>'.

Error Type:
Error

Possible Cause:

The ParseMessage script function failed to execute.

Possible Solution:

Re-examine the runtime logic of this function.

ValidateAddress() returned an invalid result. | Result = '<result value>'.

Error Type:

Error

Possible Cause:

The profile script returned an incorrect value from the ValidateAddress() function.

Possible Solution:

Ensure that ValidateAddress() returns a JavaScript object with the required fields.

Could not validate address because state machine version is unknown. | Address = '<address>'.

Error Type:

Error

Possible Cause:

1. The profile is missing required script function GetDriverInfo.
2. Required script function GetDriverInfo returned a bad result.

Possible Solution:

Correct the syntax errors or implement the missing functions.

Failed to process response. | Type = '<read/write>', Channel = '<name>', host = '<host>', port = '<port>', service name = '<name>'.

Error Type:

Error

Possible Cause:

The ParseMessage script function failed to execute.

Possible Solution:

Re-examine the runtime logic of this function.

Profile link established. | Channel = '<name>', profile = '<name>'.

Error Type:

Informational

Profile link established. | Channel = '<name>', profile = '<name>', state machine version = '<version>'.

Error Type:

Informational

Index