



PI Interface for OPC DA Version 2.5.1.x

User Guide

OSIsoft, LLC
777 Davis St., Suite 250
San Leandro, CA 94577 USA
Tel: (01) 510-297-5800
Fax: (01) 510-357-8136
Web: <http://www.osisoft.com>

PI Interface for OPC DA User Guide

© 1998-2014 by OSIsoft, LLC. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of OSIsoft, LLC.

OSIsoft, the OSIsoft logo and logotype, PI Analytics, PI ProcessBook, PI DataLink, ProcessPoint, PI Asset Framework (PI AF), IT Monitor, MCN Health Monitor, PI System, PI ActiveView, PI ACE, PI AlarmView, PI BatchView, PI Coresight, PI Data Services, PI Event Frames, PI Manual Logger, PI ProfileView, PI WebParts, ProTRAQ, RLINK, RtAnalytics, RtBaseline, RtPortal, RtPM, RtReports and RtWebParts are all trademarks of OSIsoft, LLC. All other trademarks or trade names used herein are the property of their respective owners.

U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the OSIsoft, LLC license agreement and as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12.212, FAR 52.227, as applicable. OSIsoft, LLC.

Version: 2.5.1.x

Published: March 2014

Contents

Introduction to the PI OPC DA interface.....	1
Related manuals.....	2
How the PI OPC DA interface works.....	3
Interface startup.....	4
Reading data from the OPC server.....	4
Time stamps.....	4
Logging.....	5
Buffering.....	5
Server connection management.....	5
Polled, advise, and event PI points.....	6
Output PI points and scan classes.....	7
Data type compatibility.....	7
Time stamps.....	9
Transformations and scaling.....	10
Data quality information.....	12
Failover.....	15
Plug-ins (post-processing DLLs).....	15
Installing and configuring the PI OPC DA interface.....	17
Installation directory and file locations.....	17
Installation prerequisites.....	18
Create trusts.....	18
Install PI OPC Interface.....	19
Create and configure the interface instance.....	21
Verify interface startup.....	22
Configure the Windows service.....	22
Manage the PI OPC interface service.....	23
Enable buffering.....	23
Configuring PI points for the PI OPC DA interface.....	25
Create PI points manually.....	26
Create PI points automatically.....	26
Tag (PI point name).....	27
PointSource (point source).....	27
PointType (data type).....	28
Location1 (interface instance).....	28
Location2 (data-type handling).....	28
Location3 (processing type of tag).....	29
Location4 (scan class).....	29
Location5 (OPC deadband).....	31
InstrumentTag (OPC ItemID).....	31
ExDesc (extended descriptor).....	31
SourceTag.....	32
TotalCode.....	32
SquareRoot.....	32
Convers.....	33
UserInt1 (OPC array index).....	33
UserInt2 (event group).....	33

Scan.....	33
Shutdown.....	34
Exception processing.....	34
Output points.....	34
Event points.....	35
Reading OPC array item PI points.....	36
Reading OPC arrays as event points.....	37
Reading OPC quality into a digital PI point.....	37
Configuring failover for the PI OPC DA interface.....	39
Unilnt failover.....	39
How Unilnt failover works.....	40
Hot, warm, and cold failover modes.....	41
Configure shared-file (Phase 2) failover.....	43
Test failover configuration.....	44
OPC server-level failover.....	45
Configuring server-level failover.....	45
Controlling failover timing.....	49
Configuring DCOM for the PI OPC DA interface.....	51
DCOM security levels.....	51
DCOM clients and servers.....	51
Windows domains and users.....	52
Determining the effective user.....	52
Firewalls and security.....	53
OPC server issues.....	55
Item browsing.....	55
Time stamps.....	55
False values.....	55
Access path.....	55
Problems with data returned by OPC server.....	56
Troubleshooting OPC server operation.....	57
OPC refreshes.....	57
Features supported by the PI OPC DA interface.....	61
Installation checklist for the PI OPC DA interface.....	63
Installation prerequisites.....	63
Install PI OPC DA interface on an interface node.....	64
PI ICU reference for the PI OPC DA interface.....	67
OPC Server settings.....	67
Advanced Options settings.....	68
Data Handling settings.....	69
DCOM Security settings.....	71
Failover settings.....	72
Plug-In settings.....	74
Miscellaneous settings.....	74
Debug settings.....	74
Command-line parameters for the PI OPC DA interface.....	77

Alphabetical list of parameters.....	77
Parameters by function.....	88
Error and informational messages for the PI OPC DA interface.....	91
Messages.....	91
System errors and PI System errors.....	99
Unint failover-specific messages.....	99
Technical support and other resources.....	103

Introduction to the PI OPC DA interface

The PI OPC interface is an OPC Data Access (DA) client application that communicates with an OPC server and sends data to the PI Server (and, optionally, receives data from the PI Server). The PI OPC interface supports versions 1.0a and 2.05 of the OPC Data Access standard. Because OPC depends on the Microsoft COM and DCOM technologies, the PI OPC interface is supported only on Windows platforms. For details, see the *UniInt Interface User Manual*.



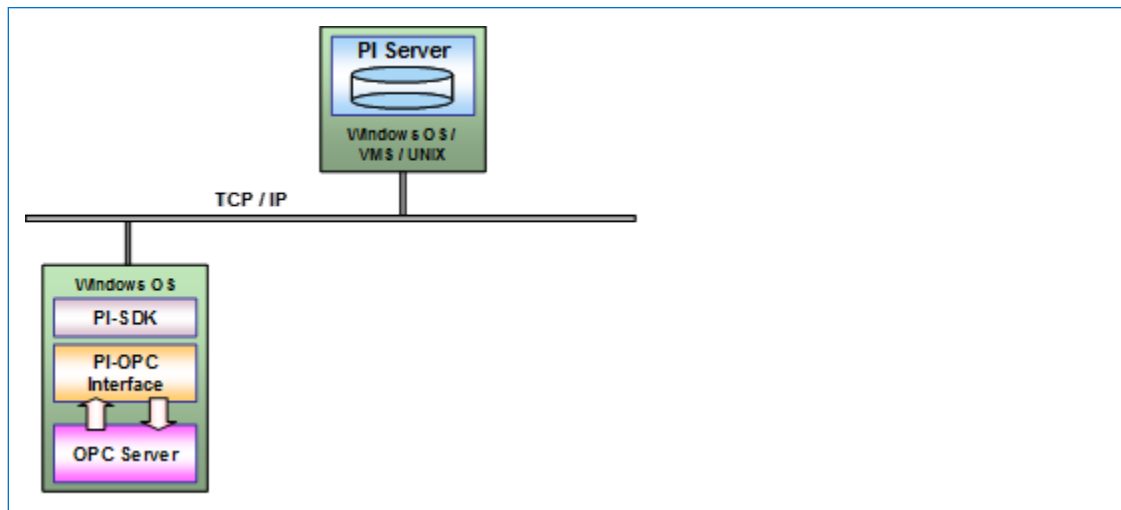
Note:

The OPC DA standard is designed for real-time data. The OPC HDA standard is designed for the retrieval of historical process data. If your goal is to retrieve high-performance, real-time data for process monitoring, use OPC DA with buffering. If you need to synchronize historical data between two different platforms, use the PI OPC HDA interface.

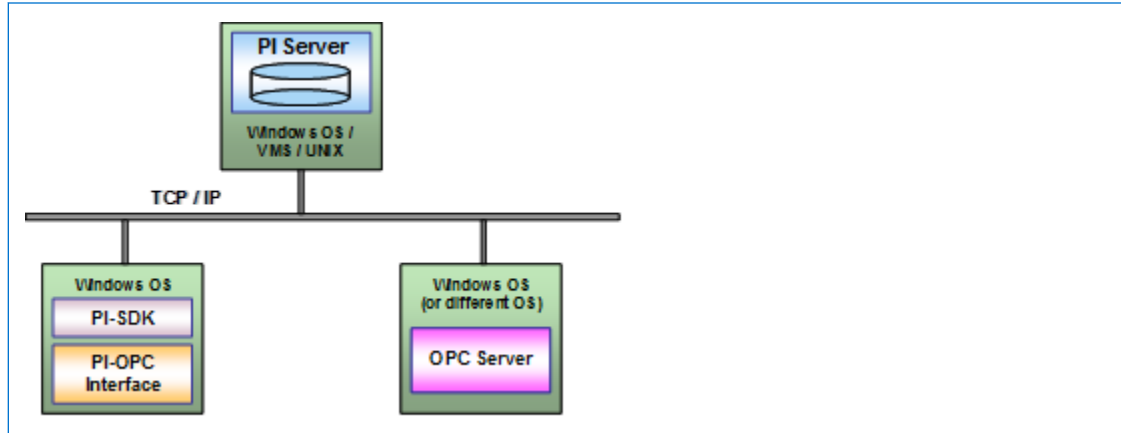
The PI OPC interface can be configured to run on the same system as the OPC server, the PI server, or on another system altogether. This section illustrates a few basic configurations. The configuration that you choose determines the specific system settings that are needed for the PI OPC interface to perform correctly.

To ensure that the PI OPC interface does not compete with the PI Server for system resources, install the interface on a dedicated computer, not on the computer where the PI Server is running. To ensure that the PI OPC interface restarts when the computer is restarted, install it as an automatic service. After the interface has been installed and tested, enable buffering.

The following example configuration is simple and permits data buffering on the interface node.



The following configuration places the OPC server on its own computer.



Related manuals

For details about related products and technologies, refer to the following OSIsoft manuals:

- PI Server manuals
- *PI API Installation Manual*
- *PI OPCClient User's Guide*
- *PI Interface Configuration Utility User Manual*
- *UniInt Interface User's Guide*
- *DCOM Security and Configuration Guide*

How the PI OPC DA interface works

The PI OPC DA interface is started using a Windows batch file that invokes the interface executable and specifies settings using command line parameters. To ensure a correctly-formatted batch file, do not edit the batch file manually: use PI ICU. For a complete list of UniInt startup parameters, refer to the *UniInt Interface User Manual*.

At startup, the PI OPC DA interface performs the following steps:

1. Connects to PI Server (unless disconnected startup is configured)
2. Gets PI points from PI Server
3. Connects to OPC server
4. Creates OPC groups
5. Adds items to groups
6. Activates groups (unless groups are created as active)
7. Begins data collection
 - Services scheduled input points and process each scan class in turn.
 - Services output points as events arrive.
 - Services triggered input points as events arrive.
 - Checks the PI point database for points that are added, edited, and deleted.

If the OPC interface detects new points or changes to existing points, it reloads those points from the PI Server. The interface processes 25 point updates at a time. If more than 25 points are added, edited, or deleted at one time, the interface processes the first 25 points, waits 30 seconds or the time specified by `UniInt /updateinterval` parameter, whichever is lower, processes the next 25 points, and so on. After all points have been processed, the OPC interface resumes checking for updates.



Note:

During startup, the OPC interface loads all the points that it maintains. After startup, the OPC interface processes subsequently-updated points periodically, in batches of 25. For efficiency, if you've changed a large number of PI points, stop and restart the interface.

Topics in this section

- [Interface startup](#)
- [Reading data from the OPC server](#)
- [Time stamps](#)
- [Logging](#)
- [Buffering](#)
- [Server connection management](#)
- [Polled, advise, and event PI points](#)
- [Output PI points and scan classes](#)
- [Data type compatibility](#)

- [Failover](#)
- [Plug-ins \(post-processing DLLs\)](#)

Interface startup

The OPC interface is started using a Windows batch file that invokes the OPC interface executable and specifies settings using command-line parameters. To ensure a correctly-formatted batch file, do not edit the batch file manually; use PI ICU. For a complete list of *UniInt* startup parameters, see the *UniInt Interface User Manual*.

Reading data from the OPC server

Data is read from OPC servers in groups, not as individual items. The PI OPC interface creates OPC groups for PI scan classes. (For advise PI points in scan class 1, multiple groups might be created.) The OPC server caches the most recent data. By default, the PI OPC interface reads from the cache of an OPC server. When the PI OPC interface creates a group, it specifies how often the cache values for the points in that group are to be updated. The requested update rate is usually the same as the scan rate for the points. The OPC server might decline the requested rate and return the update rate that it supports for that group. The update rate to which the OPC server agrees is recorded in the local PI message log file.

Time stamps

The PI OPC interface can use the time stamps provided by the OPC server or create its own time stamps at the time that the data is received. Time stamps coming from the OPC server are in Coordinated Universal Time (UTC), and are sent to the PI system in UTC as well.

If the OPC server provides time stamps, you can use PI ICU to configure the behavior of the PI OPC interface as follows:

Option	Description	Timestamp Offset Applied
Interface Provides Time stamp (/TS=N)	(Default) The PI OPC interface time stamps each value as it is received. Choose this option if the OPC server cannot provide time stamps or you do not want to use the time stamps returned by the OPC server.)	Difference between PI Server node and interface node.
OPC Server Provides Time stamp (/TS=Y)	The OPC interface uses the UTC time stamp provided by the OPC server.	Difference between PI Server node and OPC server node.
Time stamp for Advise Tags Only (/TS=A)	For polled reads, some OPC servers return the time that the value last changed rather than the time of the read. This option configures the OPC interface to use the advise time stamps but provides time stamps for the polled values. For more details about advise and polled points, see Polled, advise, and event PI points .	For advise data, difference between PI Server node and OPC server node. For all other data, difference between PI Server node and interface node.

Option	Description	Timestamp Offset Applied
OPC Server Provides Timestamp, no Offset (/TS=U)	The OPC interface uses the UTC timestamp provided by the OPC server and does not apply any offset to the timestamps. Use this setting with extreme caution, as data loss occurs if the OPC Server sends a value with a timestamp that is 10 minutes or more later than the PI Server's current time.	None.

For details about reading and writing time stamps from a PI point when the time stamp is the value of the point, see [Time stamps](#).

Logging

The PI OPC interface logs messages about its operation in the local PI message log file. The following information is logged:

- Startup and shutdown status messages
- The scan rate configured for each scan class and the actual update rate provided by the OPC server
- The number of PI points in each scan class, output points, and advise and event tags
- Mis-configured points
- Points rejected by the OPC server (and other error messages from the OPC server)
- OPC server connection attempts and results, including loss of connectivity

Buffering

Buffering is temporary storage of the data that the PI OPC interface collects and forwards to the PI Server. To ensure that you do not lose any data if the PI OPC interface cannot communicate with the PI Server, enable buffering. The PI SDK installation kit installs two buffering applications: the PI Buffer Subsystem (PIBufss) and the PI API Buffer Server (BufServ). PIBufss and BufServ are mutually exclusive; that is, on a particular computer, you can run only one at a time. For details about configuring buffering, refer to the *PI Buffering User Guide*.

To ensure data integrity, enable buffering even if the PI OPC interface runs on the PI server node, because the OPC server sometimes sends data in bursts, with all values arriving within the same millisecond. To ensure that the interface and buffering restart when the interface node is restarted, configure both as Windows services.

Server connection management

Each instance of the PI OPC interface connects to a single OPC server. To handle multiple OPC servers, run multiple instances of the PI OPC interface. Multiple instances of the interface can be configured to connect to the same OPC server. To enable the PI OPC interface to collect data

without a connection to the PI server, you can start the PI OPC interface in disconnected mode. Refer to the *UniInt Interface User Manual* for more details.

If the PI OPC interface cannot connect to the OPC server during startup, it logs the problem and retries the connection every five seconds until it reconnects. When the OPC server reports that it is running, the PI OPC interface connects to it and starts creating groups and adding items. If the PI OPC interface loses the connection to the OPC server after the initial connection, it tries to re-establish the connection. To ensure that no data from the OPC server is lost if the PI server is inaccessible, configure buffering on the PI OPC interface node.

Polled, advise, and event PI points

The PI OPC interface has three methods of getting data from data sources into PI points: polled points, advise points, and event points. All three types of points are received asynchronously by the PI OPC interface. To assign point type, set **Location3** as follows:

PI Point Type	Location3
Polled	0
Advise	1
Event	2

To assign the scan class for a point, set **Location4**. Do not assign the same scan class to both advise and polled points; use separate scan classes.

Polled points

Polled PI points are grouped by scan class, and, if possible, groups are read at the rate configured for scan class of the point. However, the OPC server determines its own update rate for scanning its data sources, and you can configure the update rate manually (using PI ICU). The PI OPC interface requests the OPC server to use an update rate identical to the scan class, but the OPC server does not guarantee that the rates match. The PI scan class offset has no effect on the OPC server, unless the interface is configured for staggered group activation and the OPC server uses the activation of the group to initiate the scanning cycle.

For details about polled points, see the *Data Access Custom Interface Standard v2.05a* from the OPC Foundation.

Advise points

Advise points are sent to the PI OPC interface by the OPC server only when a new value is read into the server's cache. Scan class 1 is reserved for advise points, and you can create additional scan classes for advise points as required. Be sure that the scan rate is fast enough to capture all the changes from the data source. The default maximum number of points in scan class 1 is 800. Up to 800 points with the same deadband can reside in the same group. If there are more than 800 points with the same deadband in scan class 1, the OPC interface creates as many groups as needed. (To ensure best performance, ensure that groups size does not exceed 800 items). To change the default limit, use PI ICU to set the **Number of Tags** in the **advise group** field on the **OPCInt > Data Handling** page. Your server might perform better with smaller group sizes; a limit of 200 points per group has proven effective with a number of OPC servers.

Event points

Event points are read by the PI OPC interface when it receives notification that a trigger point has a new event. The PI point that triggers the read is specified in the event point's **ExDesc** attribute. When a new event for a trigger point is sent to the PI Snapshot, the PI system notifies the PI OPC interface, which reads the values for all the associated event points from the OPC server. For v1.0a servers, an asynchronous read is sent to the server's cache. For v2.0 servers, the PI OPC interface performs an asynchronous read from the device.

To configure event points, specify \emptyset for the scan class. To assign event points to the same OPC event group (so they are read together), specify the same integer in each point's **UserInt2** attribute. Set each event point's **ExDesc** attribute to the name of the triggering point. For details about configuring event points, refer to the *UniInt Interface User Manual*.

Frequent device reads can impair the performance of the OPC server. For any asynchronous read, the OPC server is required to return all of the values together, which can delay the return of new values to the PI Server if the OPC server encounters a delay in reading the values. To improve performance in this case, group points according to the device where the data originates.

Output PI points and scan classes

When a value is written to the OPC server, the PI OPC interface waits for an acknowledgement (ACK) from the server. To speed up processing of outputs, you can configure multiple output groups, specifying the number of outstanding writes a group is permitted and the amount of data to be sent in each write. The OPC server is not required to accept more than one write at a time from any group, but many servers permit multiple writes to be sent without waiting for the first one to be acknowledged. Even if the server accepts only one write at a time, defining multiple output groups can improve throughput. If you specify more outstanding writes than the OPC server can accept, the PI OPC interface reduces its setting to the OPC server's maximum.

The interface monitors acknowledgements of writes, and you can specify how long to wait for the OPC server to acknowledge a write. If no acknowledgement is received in the specified period, the interface cancels the write and reissues it.

If your OPC server does not acknowledge writes, you can create an alert using the Device Status health point. Configure the alert to detect a desired number of queued writes. When the specified level is reached, the alert sets an alarm state and drops a specified number of values, oldest or newest.

To assign an output point to an output group, set its **Location4** attribute to the group number. For load balancing, output points with **Location4** set to \emptyset are distributed across output groups (including groups to which output points are explicitly assigned).

Data type compatibility

The data type of a PI point must be compatible with the data type of the corresponding OPC item. For example, if a string value from the OPC server is to be put into an Int32 PI point, the string must contain a number. If a 64-bit floating-point value is to be put into an Int16 point, its value must not overflow the target data type.

The PI OPC interface specifies the desired data type when requesting information from the OPC server, and the OPC server is responsible for delivering the requested data type if it can. The PI OPC interface normally requests values using the following default data types:

PI PointType	OPC Data Type
Digital	Two-byte Integer (VT_I2)
Int16	Two-byte Integer (VT_I2)
Int32	Four-byte Integer (VT_I4)
Float32	Four-byte Float (VT_R4)
Float64	Eight-byte Float (VT_R8)
String	String (VT_BSTR)

If your OPC server does not permit clients to specify a data type, set **Location2** to 8 for all your OPC PI points. Use with caution: The OPC interface might receive data for which no reasonable conversion is possible. Where possible, always specify the OPC data type that matches the PI point.

- **Reading numeric points as strings**

Some OPC servers return certain numeric data types only as strings. To interpret string-formatted Int16, Int32, Float32, and Float64 values, set **Location2** to 1. The PI OPC interface requests the data as a string (BSTR) and interprets the data as a number.

PI digital points contain integer values that correspond to specific strings in the digital state table in the point's digital set property. Some devices read and write the string value rather than the integer value. To read and write digital points as string points, set **Location2** to 1. Make sure that the strings used by the OPC server are identical to the strings in the digital set, including punctuation and spaces. For optimal performance, read digital points as numbers whenever possible.

- **Booleans**

Some OPC servers send Boolean values as 0 and -1 when read as integers. This approach creates a problem when reading that data into a PI digital point, because "-1" is not the value that must be stored. To handle the data from such servers, the OPC interface uses the absolute value of any integer or real values read for digital points. Because digital point values are actually offsets into the digital set for the point, and a negative offset has no functional meaning, this issue does not cause problems for properly-written servers.

The PI OPC interface can also request the item as a Boolean (VT_BOOL). This approach works only for items that have two possible states, because any non-zero value is interpreted as 1. To have points read and written as though they were Booleans, set **Location2** to 2.

- **Four-byte integers**

If your OPC server does not support the two-byte integer (VT_I2) data type, you can configure the PI OPC interface to request the data as a four-byte integer (VT_I4) by setting **Location2** to 3.

- **Float64 values**

To handle eight-byte floating-point numbers (VT_R8), set the **Location2** of the target point to 5. PI stores the value as a four-byte floating-point number, with possible loss of precision. If the number is too large to fit in the point, a status of **BAD INPUT** is stored.

Topics in this section

- [Time stamps](#)
- [Transformations and scaling](#)
- [Data quality information](#)

Time stamps

The PI OPC interface does not adjust the time stamps it receives, regardless of the time zone settings or **/TS** parameter specified on the command line. Any scaling or transformation is performed after the string has been translated into seconds, which enables a wide range of values to be handled.

- **Converting time stamps into seconds**

To store a timestamp string (VT_BSTR) as seconds, set Location2 to 6. If the PI point is an integer, the PI OPC interface attempts to translate the time stamp into whole seconds. (Because Int16 points can only hold numbers up to 32767, use Int32 points for time spans longer than nine hours.) If the PI point has a floating-point data type, the time stamp is translated into seconds and stored as a floating-point number.

- **Reading time stamps as VT_DATE data types**

The OPC standard allows the VT_DATE data type, which is an internal representation of a time stamp. To configure the PI OPC interface to use the VT_DATE data type for reading the value from the OPC server or for writing the value to output points, set Location2 to 7. The PI OPC interface translates between VT_DATE and integer, float, or string points. The PI OPC interface does not adjust the time stamps received, regardless of the time zone settings. For string points, the format of the string must be specified as above.

- **Time stamp strings**

To configure the format of the time stamp sent by the OPC server using PI ICU, go to the **OPCInt > Data Handling** page and specify the format in the **Format of Timestamp Strings** field using the following tokens:

Token	Description
<i>cc</i>	Two-digit century
<i>yy</i>	Two-digit year
<i>mn</i>	Two-digit month
<i>mon</i>	Three-character month (Jan Feb Mar, etc.)
<i>dd</i>	Two-digit day
<i>hh</i>	Two-digit hour from 0 to 23
<i>hr</i>	Two-digit hour from 0 to 12
<i>mm</i>	Two-digit minute
<i>ss</i>	Two-digit second
<i>24</i>	Three-digit milliseconds

Token	Description
<i>XM</i>	AM or PM

The position of the tokens and delimiters must specify the format of the time stamp string precisely. Examples:

Format String	Result
<i>ccyy/mn/dd hh:mm:ss.000</i>	1998/11/29 15:32:19.391
<i>dd mon, ccyy hr:mm:ss XM</i>	29 Nov, 1998 03:32:19 PM
<i>mn-dd-ccyy hh:mm:ss</i>	11-29-1998 15:32:19
<i>hh:mm:ss.000</i>	15:32:19.482

Only one format string can be specified for each instance of the PI OPC interface. If more than one format of time stamp needs to be processed, configure additional instances of the PI OPC interface with the required format string.

If you omit elements of the format strings, the defaults are as follows ("current" values are GMT):

Format String Element Omitted	Default
Day	Current day
Month	Current month
Year	Current year
Century	Current century



Note:

If you specify only hours, minutes and seconds, the date defaults to January 1, 1970. To ensure accurate timestamps, be sure to specify all the elements of the timestamp format. If the OPC server returns a zero value for day, month or year, the interface applies the defaults described above, regardless of the format string you specify.

Transformations and scaling

You can configure PI points so that the PI OPC interface performs transformations and scaling. Transformation and scaling are applied before the value is compared to the exception parameters for the point, to ensure that the exception parameters are applied to the value that is to be sent to PI Server rather than the raw value.

Scaling

To configure scaling for a PI OPC point, set the **TotalCode** and **SquareRoot** attributes of the point. The **Convers** attribute specifies the span of the device, and the **ExDesc** specifies the device zero (Dzero). Using these values, the PI OPC interface can translate a value from the scale of the device to the scale of the point. Scaling is only supported for numeric points.

For simple square/square root scaling, set **TotalCode** and **Convers** to zero. To configure how the value is stored, set **SquareRoot** as follows:

- To square a value before sending it to PI Server, set **SquareRoot** to 1. For output values, the square root is calculated before it is written to the device.
- To send the square root to PI Server and the square to the device, set **SquareRoot** to 2.

Transformation

To transform the value to another scale of measurement, to apply an offset or conversion factor, or to perform bit masking, configure the settings as shown in the following table. If **SquareRoot** is set to 1 or 2, the square root or square of the value is calculated first, then the formula is applied.

Conver s	TotalCo de	SquareRo ot	Dzero	Operation Input points	Operation Output points
0	0	1	No effect	$(\text{Value})^2$	$(\text{Value})^{0.5}$
		2	No effect	$(\text{Value})^{0.5}$	$(\text{Value})^2$

Convers	TotalCode	SquareRoot	Dzero	Operation Input points	Operation Output points
Non-zero	1	0	Defined	$[(\text{Value} - \text{Dzero}) / \text{Convers}] * \text{Span} + \text{Zero}$	$[(\text{Value} - \text{Zero}) / \text{Span}] * \text{Convers} + \text{Dzero}$
		1	Defined	$[((\text{Value})^2 - \text{Dzero}) / \text{Convers}] * \text{Span} + \text{Zero}$	$[((\text{Value})^{0.5} - \text{Zero}) / \text{Span}] * \text{Convers} + \text{Dzero}$
		2	Defined	$[((\text{Value})^{0.5} - \text{Dzero}) / \text{Convers}] * \text{Span} + \text{Zero}$	$[((\text{Value})^2 - \text{Zero}) / \text{Span}] * \text{Convers} + \text{Dzero}$
	2	0	No effect	$\text{Value} * \text{Convers}$	$\text{Value} / \text{Convers}$
		1	No effect	$(\text{Value})^2 * \text{Convers}$	$(\text{Value})^{0.5} / \text{Convers}$
		2	No effect	$(\text{Value})^{0.5} * \text{Convers}$	$(\text{Value})^2 / \text{Convers}$
	3	0	Defined	$(\text{Value} / \text{Convers}) - \text{Dzero}$	$(\text{Value} + \text{Dzero}) * \text{Convers}$
		1	Defined	$((\text{Value})^2 / \text{Convers}) - \text{Dzero}$	$((\text{Value})^{0.5} + \text{Dzero}) * \text{Convers}$
		2	Defined	$((\text{Value})^{0.5} / \text{Convers}) - \text{Dzero}$	$((\text{Value})^2 + \text{Dzero}) * \text{Convers}$
	4	0	Defined	$(\text{Value} - \text{Dzero}) / \text{Convers}$	$(\text{Value} * \text{Convers}) + \text{Dzero}$
		1	Defined	$((\text{Value})^2 - \text{Dzero}) / \text{Convers}$	$((\text{Value})^{0.5} * \text{Convers}) + \text{Dzero}$
		2	Defined	$((\text{Value})^{0.5} - \text{Dzero}) / \text{Convers}$	$((\text{Value})^2 * \text{Convers}) + \text{Dzero}$
	5	0	No effect	$\text{Value} + \text{Convers}$	$\text{Value} - \text{Convers}$
		1	No effect	$(\text{Value})^2 + \text{Convers}$	$(\text{Value})^{0.5} - \text{Convers}$
		2	No effect	$(\text{Value})^{0.5} + \text{Convers}$	$(\text{Value})^2 - \text{Convers}$
	6	No effect	No effect	Value AND Convers	Value AND Convers
	7	No effect	No effect	Value OR Convers	Value OR Convers
	8	No effect	No effect	$\text{Value} = \text{Value XOR Convers}$	$\text{Value} = \text{Value XOR Convers}$

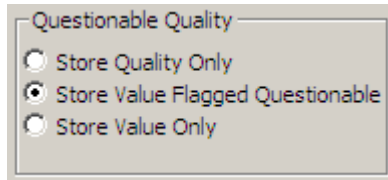
Data quality information

The OPC Data Access standard specifies a set of quality flags. The PI OPC interface translates the quality flags to the closest approximation in the PI system state table. The low eight bits of the quality flags are arranged into three fields, quality (QQ), sub-status (SSSS) and limit status (LL), arranged as follows: QQSSSSL.

Handling data of questionable quality

The PI archive stores either the quality or the value in a tag, whereas the OPC server returns value and quality in separate fields. If a value is good, it is stored in the tag. If a value is bad, a digital state that describes the quality is stored. For questionable-quality data, you can configure the OPC interface to treat the values as good and store them, or treat them as bad and store a digital state. You cannot configure the interface to store a bad-quality value.

To configure handling of questionable-quality data using PI ICU, go to the **OPCInt > OPC Server** page and enable the desired option, as shown in the following figure.



Storing both values and quality information

To record both the values reported and the quality information returned with the values, store the quality in a separate PI tag. To configure a tag to store the quality for the associated ItemID, set Location2 to 4. Because OPC qualities are unsigned 16-bit integers, the PI OPC interface requires an Int32 tag to store them. The values are stored in PI without any change, and their status is always GOOD. For details about OPC quality values, download the *OPC Data Access specification* from <http://www.opcfoundation.org> or consult the *OPCClient User's Guide*.

Quality states

Quality data is returned using a bit mask. The first number corresponds to a hexadecimal value between 0xC0 (11000000) and 0xFF (11111111). The following tables list the values that are returned.

Good quality

Quality	OPC Definition	PI Status
11SSSSL	Non-specific	Good
Except: 110110LL	Local Override	_SUBStituted

Not used by OPC

Quality	OPC Definition	PI Status
10SSSSL	Invalid	Bad Input

Questionable quality

Quality	OPC Definition	PI Status
010110 LL	Sub-Normal	Bad_Quality
010101LL	Engineering Units Exceeded	
LL=01	Low Limited	Under LCL
LL=10	High Limited	Over UCL
Otherwise		Inp OutRange

Quality	OPC Definition	PI Status
010100LL	Sensor Not Accurate	
LL=01	Low Limited	Under Range
LL=10	High Limited	Over Range
Otherwise	Out of calibration (if not under or over range)	Invalid Data
010011LL	Invalid	Bad Input
010010LL	Invalid	Bad Input
010001LL	Last Usable Value	No_Sample
010000LL	Non-specific	Doubtful

Bad quality (PI version 3.3 and higher)

Quality	OPC Definition	PI Status
000111LL	Out of Service	Out of Service
000110LL	Comm Failure	Comm Fail
000101LL	Last Known Value	Scan Timeout
000100LL	Sensor Failure	Equip Fail
000011LL	Device Failure	Unit Down
000010LL	Not Connected	Not Connected
000001LL	Configuration Error	Configure
000000LL	Non-specific	Bad

To replace the default PI digital states with custom states using PI ICU, go to the **OPCInt > Data Handling** page and set the **Alternate Digital State for Questionable/Bad Qualities** field. To override the default states, you must specify the full set of replacements, and the numeric values must be contiguous. The following table lists the digital states and PI statuses that you can override.

Custom digital states

Order After Marker State	Default PI status
1	Bad_Quality
2	Under LCL
3	Over UCL
4	Inp OutRange
5	Under Range
6	Over Range
7	Invalid Data
8	Bad Input
9	No_Sample
10	Doubtful
11	Out of Service
12	Comm Fail

Order After Marker State	Default PI status
13	Scan Timeout
14	Equip Fail
15	Unit Down
16	Not Connected
17	Configure
18	Bad

Failover

The PI OPC interface is designed to provide redundancy for both the OPC server and the PI OPC interface, as follows.

- **Server-level failover**

The PI OPC interface can be configured to change to another OPC server when a problem is detected.

- **Interface-level failover**

To ensure against data loss, you can run two instances of the PI OPC interface on different machines. If the primary instance fails, the backup instance can take over.

Plug-ins (post-processing DLLs)

The PI OPC interface can be configured to use plug-ins, which are DLLs that contain libraries of routines that perform application-specific data processing before the data is sent to the PI Server or OPC server. The DLLs and their accompanying files and documentation are included in the PI OPC interface installer and are installed into the `Plug-ins` sub-directory under the OPC interface directory. Each plug-in package contains user documentation, and you can download plug-in user guides from the OSIsoft Download Center.

Installing and configuring the PI OPC DA interface

To install and configure the PI OPC DA interface, you must:

- Run the installer
- Create any required trusts
- Create and configure an interface instance
- Configure the Windows service
- Configure buffering
- Create PI points

The following sections describe these tasks in detail. For details about features that are common to all UniInt interfaces, refer to the *UniInt Interface User Manual*.



Note:

Even if you are using the same node for both the OPC server and the OPC client, you must configure DCOM security settings. In this case, make sure that DCOM permissions have been granted to the accounts under which the OPC server and the PI OPC interface run. For details, see the *OSIsoft DCOM Configuration Guide*.

Topics in this section

- [Installation directory and file locations](#)
- [Installation prerequisites](#)
- [Create trusts](#)
- [Install PI OPC Interface](#)
- [Create and configure the interface instance](#)
- [Verify interface startup](#)
- [Configure the Windows service](#)
- [Enable buffering](#)

Installation directory and file locations

By default, the PI OPC interface is installed in the following location:

`\Interfaces\OPCInt\`

The `%PIHOME%` directory, which is the root directory under which OSIsoft products are installed, is defined by the `PIHOME` entry in the `pipc.ini` configuration file in the `%windir%` directory. To override the default locations, edit the `pipc.ini` configuration file.



Note:

Reserve the C: drive for the operating system and install the interface on another drive.

The PI OPC interface installation directory contains all the files required to configure and run the interface. The `OPCEnum` tool, which discovers OPC servers, is installed in the `...\%windir%\system32` directory, except on 64-bit systems, where it is installed in `%windir%\sysWOW64`.

The OPC libraries, provided by the OPC Foundation and installed with the PI OPC interface, are installed in the same directory as OPCEnum.

Installation prerequisites

Before installing and configuring, ensure that the following prerequisites are met:

- Verify that the PI Server is up and running.
- Using OPCClient, verify that the OPC server is up and running and populated with points.
- Verify that the PI OPC interface node time zone is set correctly.
- On the PI OPC interface node, install the following:
 - OSIssoft Prerequisites
 - PI Interface Configuration Tool (PI ICU)

Create trusts

When creating trusts, you have many options. Following is a simple and secure approach, creating a trust for the following applications:

- PI OPC interface
- PI Interface Configuration Utility (ICU)
- Buffering

To create each of these trusts using PI System Management Tools, connect to the PI Server and perform the following steps:

Procedure

1. Click **Security** and choose **Mappings & Trusts**.
2. On the **Trusts** tab, right-click and choose **New Trust**. The Add Trust wizard appears.
3. Specify a meaningful name and description for the trust.
4. Configure settings as follows:

Trust	Type	Application Name	Network Path	PI User
PI OPC interface	PI API application	OPCpE	Name of the interface node or IP address plus netmask 255.255.255.255	OPC tag data owner

Trust	Type	Application Name	Network Path	PI User
PI ICU	PI API application	PI-ICU.exe	Name of the interface node or IP address plus netmask 255.255.255.255	Dedicated PI identity with the precise permissions required (database read access, read ptsecurity and read-write permission for OPC points)
Buffering	PI API application	BufServ: APIBE PIBufss: Pibufss.exe	Name of the interface node or IP address, plus netmask 255.255.255.255	OPC tags' data owner

Install PI OPC Interface

This procedure summarizes the steps for installing and creating a basic configuration of the PI OPC interface on an OPC interface node. Many of the settings prescribed below are defaults that can be changed according to the requirements of more complex configurations.

Procedure

1. On the OPC interface node, run the PI OPC interface setup program.
For details on the file and directory structure, see [Installation directory and file locations](#).
2. On the OPC interface node, test the API connection to the PI Server: In the %PIPC%\bin directory, issue the `apisnap PISERVERNODE` command.
3. On the OPC interface node, test the SDK connection to the PI Server: Choose **Start > All Programs > PI System > About PI-SDK** and use **File > Connections** to connect to the PI Server.
4. Create an OPC data owner (that is, a PI user with read/write access to the PI points that this interface will be using).
5. On the server node, use PI SMT to create API trusts that permit the following applications to access the server node:
 - PI ICU
 - Buffering process
 - OPC interface (application name: OPCpE)

For details, see [Create trusts](#)

6. On the OPC interface node, use PI ICU to create a new OPC interface instance from the OPC interface batch file (`OPCint.bat_new`).

For details, see [Create and configure the interface instance](#).

7. Configure DCOM settings.

For details, see [Configuring DCOM for the PI OPC DA interface](#) and the *OSIsoft DCOM Security and Configuration Guide*.

8. Define the following settings for the OPC interface in PI ICU:

◦ **General**

Setting	Value
Point source	OPC (or an unused point source of your choice)
Scan classes	As desired. (Scan class 1 is reserved for advise points.)
Interface ID	1 or any unused numeric ID

◦ **OPCIntOPC Server tab**

Click the **List Available Servers** button, then select your server from the list. If the server is on another machine, specify that machine or IP address in the **Server Node** field, then click to display the list.

9. Using PI OPCClient, verify that the OPC server is up and running and populated with points.

10. Start the interface interactively and check the log to verify that it starts successfully.

For details, see [Verify interface startup](#).

11. If you intend to use digital points, define the appropriate digital state sets.

12. Use PI ICU to configure the PI OPC interface to run as a service.

For details, see [Configure the Windows service](#).

13. Stop the interface and configure and start buffering.

For details about verifying that buffering is working, see [Enable buffering](#). Additional information is available in the OSIsoft Knowledge Base.

14. Restart the OPC interface node and confirm that the PI OPC interface service and the buffering application restart.

15. Build input points and, if desired, output points for this PI OPC interface. Verify that data appears in PI Server as expected.

Point attribute	Description
PointSource	Identifies all points that belong to this instance of the PI OPC interface. Specify the same Point source entered on the PI ICU General tab .
Location1	Specifies the OPC interface instance ID, which is displayed on the PI ICU General tab.
Location2	To enable handling for OPC servers that do not return certain numeric types in their native format, set Location2 to 1. Numeric data is returned as a string.
Location3	Point type (0=polled, 1=advise, 2=output).
Location4	Specifies the scan class.
Location5	Optional deadband value for advise points.
ExDesc	Specifies event points, Long ItemID, Dzero for scaled points, or ItemID to get the time stamp for an output value.
Instrumentpoint	OPC ItemID that corresponds to the PI point you are defining. Case-sensitive. To display OPC server points, use PI OPCClient.

For details, see [Configuring PI points for the PI OPC DA interface](#).

16. Optional: The following procedures are useful but not required.
 - **Configure diagnostics**

Diagnostic points enable you to track the activity and performance of the OPC interface. Note that the OPC interface does not support scan class performance points. For details about diagnostic and health points, see the *UniInt Interface User Manual*.
 - **Configure disconnected startup**

Disconnected startup enables the OPC interface to start even if PI Server is not available. For details, see the *UniInt Interface User Manual*.
 - **Configure failover**

Failover enables the PI System to switch to another instance of the OPC interface if the currently-running instance fails. For details, see [Configuring failover for the PI OPC DA interface](#)
 - **Install the PI Interface Status Utility**

This utility enables you to monitor the state of the OPC interface. For details, see the *PI Interface Status Utility (ISU)*.

Create and configure the interface instance

For each OPC server intended to exchange data with the PI System, you must create at least one instance of the PI OPC interface. For each instance you create, settings are stored in a separate Windows command file (a .bat file) in the PI OPC interface installation directory.



Tip:

If you require multiple instances of the PI OPC interface, configure them with unique IDs and PointSources, to ensure that you know where the data written to the PI server originated, and to more easily trace any problems that arise.

Procedure

1. Launch PI ICU.
2. Choose **Interface > New from BAT file**
3. Browse to the directory where the PI OPC interface is installed (default is %PIPC%\Interfaces\OPCInt), select OPCInt.bat_new and click **Open**. The Select PI Host Server dialog box is displayed.
4. Specify the PI Server and click **OK**. PI ICU displays the settings of the new instance of the PI OPC interface.
5. Edit the basic settings as follows:
 - **General tab**
 - **Point source**

OPC or a point source not already in user Interface
 - **ID**

1 or a numeric ID not already in use by another instance of the interface

- **Scan Class**

Set to desired scan frequency. (Scan class 1 is reserved for advise tags.) Note that, when defining scan classes, you can spread the server workload using offsets.

- **OPCInt tab**

Click the **List Available Servers** button, then select your server from the drop-down list of servers. If the server resides on another machine, specify the node name or IP address in the **Server Node** field before listing the available servers. Please note that if the **Server Node** field is blank or set to LocalHost (case insensitive), the interface will treat the server as though it is local. If anything else is used in the **Server Node** field, the connection will be treated as remote. This can impact certain OPC Servers that may refuse “remote” connection attempts.

- **Security Parameters tab**

If your OPC server requires clients to use OPC security, enable OPC security (using PI ICU) and select **NT security** or **Private OPC security**, then enter the user ID and password. Before enabling OPC security, verify that your OPC server supports it; most OPC servers do not. Be advised that, when you enable **OPC Private security**, user ID and password are stored and transmitted in clear text. NT security encrypts this data and is therefore the recommended option if your server requires the use of OPC security.

Verify interface startup

You can use the message log to check for proper startup of the interface.

Procedure

1. To display the message log, launch PI System Management Tools and choose the **Operation > Message Logs** menu option.
2. To start the interface using PI ICU, choose **Interface > Start Interactive**.
PI ICU displays a command window and invokes the startup batch file, and you can observe progress as the interface attempts to initialize and run.
3. Watch log for messages indicating success or errors.
4. To stop the interface, close the command window.

Configure the Windows service


To ensure that the PI OPC interface restarts when the OPC interface node is restarted, configure it as a Windows service. To install the PI OPC interface as a service using PI ICU, perform the following steps:

Procedure

1. Launch PI ICU and click the **Service** tab in the PI ICU window.
2. Set the fields as described in the following table.

Field	Description
Service name	Descriptive name of the PI OPC interface service.


Field	Description
ID	Numeric ID of the PI OPC interface instance. Must be unique for each instance.
Display name	The service name displayed in the Windows Services control panel. The default display name is the service name with a PI- prefix. You can override the default. To ensure that OSIsoft-related services are sorted together in the Services control panel, retain the PI- prefix.
Log on as	The Windows account associated with the service. The user must have DCOM permissions configured on the OPC. Set password expiration to Never .
Password	Password, if any, for the preceding user.
Dependencies	Any services that the OPC interface depends on. The only dependency is the TCP/IP service, which is pre-configured. If buffering is enabled, you are prompted to create a dependency on the buffering service.
Startup type	Specifies whether the PI OPC interface service starts automatically when the interface node is restarted. Generally, PI OPC interface services are set to start automatically.

- To create the service, click **Create**.
- To start the service, click .

Manage the PI OPC interface service

After you install the PI OPC interface as a Windows service, use the procedures in this section to manage the service.

Procedure


- To verify that the service is running, use the Windows Services applet in Control Panel.
- To stop the service, click .
- To remove the service, stop it and click **Remove**.
- To start the service interactively, use Windows Explorer to browse to the batch file for the PI OPC interface instance, then double-click it.

Interactive startup is usually done only for debugging.

- If the PI OPC interface can connect to the PI server when run interactively but not when run as a service, check the DCOM permissions and consult the local PI message log file and Windows Event Viewer.

Enable buffering

Procedure

- In PI ICU, choose **Tools > Buffering**. The Buffering dialog box appears.
- Click **Enable buffering with PI Buffer Subsystem**.
- To start the buffering service, click **PI Buffer Subsystem Service**, then click .

4. To verify that buffering starts successfully, check the message log for messages that indicate that the buffering application is connected to PI Server.
5. To verify that the configuration is working as intended, reboot the interface node and confirm that the interface service and the buffering application restart.

Configuring PI points for the PI OPC DA interface

The PI point (also called a *PI tag*) is a time-stamped record of a single set of measurements (for example, tank temperature). If you mis-configure PI points, the OPC interface cannot correctly transmit the data from OPC server to PI Server. The following sections tell you how to configure PI points correctly.



Note:

To populate the PI Server with the points that are defined in your OPC server, use PI OPCClient or OPCtoCSV to export OPC items to an Excel file (.csv), then use PI System Management Tool's Tag Configurator feature to load the tags into the PI Server. For details, see [Create PI points manually](#).

To define PI OPC points, you provide, at a minimum, the following information:

- Tag name
- Point source
- Data type
- Interface instance
- Tag type
- Scan class
- Instrument tag

Depending on the type of point you are creating, a few other settings might be required. The following sections describe basic PI point settings in more detail.

Topics in this section

- [Create PI points manually](#)
- [Create PI points automatically](#)
- [Tag \(PI point name\)](#)
- [PointSource \(point source\)](#)
- [PointType \(data type\)](#)
- [Location1 \(interface instance\)](#)
- [Location2 \(data-type handling\)](#)
- [Location3 \(processing type of tag\)](#)
- [Location4 \(scan class\)](#)
- [Location5 \(OPC deadband\)](#)
- [InstrumentTag \(OPC ItemID\)](#)
- [ExDesc \(extended descriptor\)](#)
- [SourceTag](#)
- [TotalCode](#)
- [SquareRoot](#)

- [Convers](#)
- [UserInt1 \(OPC array index\)](#)
- [UserInt2 \(event group\)](#)
- [Scan](#)
- [Shutdown](#)
- [Exception processing](#)
- [Output points](#)
- [Event points](#)
- [Reading OPC array item PI points](#)
- [Reading OPC arrays as event points](#)
- [Reading OPC quality into a digital PI point](#)

Create PI points manually

To build individual PI points manually, use Point Builder in PI System Management Tools (SMT). To start Point Builder, choose **Points > Point Builder**

Create PI points automatically

To populate the PI Server with the points that are defined in your OPC server, use PI OPCClient or OPCtoCSV (installed in %PIPC%\PI OPC Tools\PI_OPCToCSV) to export OPC items to an Excel file (.csv), then use the Tag Configurator in PI System Management Tools (SMT) to load the points into the PI Server.



Note:

To permit PI Tag Configurator to create PI points, you must define a trust or configure permissions that enable Microsoft Excel to write to the PI Server.

To export OPC points and create the corresponding PI points, perform the following steps:

Procedure

1. Start PI OPCClient and connect to your OPC server.
2. To select the OPC points you want to export, create a group (click) and add the desired points to it.
3. Choose **File > Save As** and specify the name and location of the export file.
4. Click **Save**.
PI OPCClient creates a .csv file containing the OPC points you selected.
5. In PI SMT, start Microsoft Excel by choosing **Tools > Tag Configurator**.
6. In Microsoft Excel, open the .csv file that contains the exported OPC points.
7. Examine the generated entries to ensure that the desired points are listed. If any entries have Unknown in the **PointType** column, specify the desired data type for the point.
8. To generate the PI points, choose **PI SMT > Export Tags**. The Export PI Tags window appears

9. Choose the target PI Server and click **OK**.
10. Examine the list of results to verify that the PI points are created.

Tag (PI point name)

When assigning names to PI points, follow these rules:

- The point name must be unique.
- The first character must be alphanumeric, underscore (`_`), or percent sign (`%`).
- Control characters such as linefeeds or tabs are illegal, as are the following characters:
`* ' ? ; { } [] | \ ` ' "`

The following table indicates the maximum length of the length attribute, which depends on the combination of PI API and PI Server that you have installed.

PI API	PI Server	Maximum Length
1.6.0.2 or higher	3.4.370.x or higher	1023
1.6.0.2 or higher	Below 3.4.370.x	255
Below 1.6.0.2	3.4.370.x or higher	255
Below 1.6.0.2	Below 3.4.370.x	255

If your PI Server is earlier than version 3.4.370.x or the PI API version is earlier than 1.6.0.2 and you want to create points with names that exceed 255 characters, you must enable the PI SDK.



Note:

If the source point name length exceeds 80 characters, you must use the **UserInt1** attribute for source point mapping, due to a limitation of the PI API.

PointSource (point source)

PointSource is an identifier that associates a PI point with a PI interface instance, enabling the interface to query the PI Server for the points that it updates. This field is not case-sensitive. In the interface batch startup file, the point source is specified using the **/PS** command-line parameter.

The following point sources are reserved. Do not configure them for interface instances.

Point Source	Reserved By
T	Totalizer Subsystem
G and @	Alarm subsystem
R	Random interface
9	RampSoak interface
C	Performance equations subsystem

PointType (data type)

PointType specifies the data type of the point. Typically, OPC item data types do not need to match PI point data types exactly, but the data types must be compatible. For example, integer values from a device can be sent to floating-point or digital PI points. Similarly, a floating-point value from the device can be sent to integer or digital PI points, although the values might be truncated.

The PI OPC interface supports all PI point types except BLOB. However, some OPC servers lack support for the full range of PI point types. To determine which PI point types are supported by your OPC server, refer to the vendor-supplied documentation.

If the point type defined in PI does not match the canonical data type defined in the OPC server, the PI OPC interface attempts to translate the data. To determine whether the point can be read as the required type, use the PI OPCClient to try to read the point directly from the OPC server. For more information on data type compatibility, see [Data type compatibility](#).

Location1 (interface instance)

Location1 specifies the instance of the interface to which the PI point belongs. The value of this attribute must match the ID configured for the interface instance. This setting plus **PointSource** identify the interface instance that writes to a particular point. (**/ID**)

Location2 (data-type handling)

Location2 configures handling of data types.

Valid settings for **Location2** are as follows:

Value	Description
0	Normal processing; no special handling is used.
1	Read and write value as a string. For digital points, the strings read from the OPC server must match the strings in the digital state set used by the PI point. For integer and real points, the OPC interface requests a string value, and translates it to a number.
2	Read value as a Boolean. Booleans have only two possible values, zero and nonzero. For numeric points, any value but 0 (False) is set to -1 (True). Use this option to correctly convert an OPC server Boolean into the PI Digital State, to prevent the PI point from receiving Bad quality values for a Boolean when it is True.
3	Read value as a four-byte integer. This setting is provided to accommodate servers which cannot send the value as a two-byte integer, which is how Digital points are normally read.
4	Stores the quality of the item rather than the value.
5	Request real points as VT_R8 items (eight-byte real). By default, the PI OPC interface requests real points as VT_R4 items (four-byte real). For float32 points (including all PI2 Real points), values that cannot fit in a 32-bit floating-point number lose precision. This setting is included to support servers that do not translate VT_R8 data to VT_R4 data and to permit the use of float32 points where the benefit of greater precision is not worth the overhead of using float64 points.

Value	Description
6	Read time stamps from the OPC server as strings and transform them into seconds. The PI point can be an int or a float. The format of the time stamp string is specified in the startup file with the /TF parameter.
7	Read time stamps from the OPC server as VT_DATE variables. These values can be translated into time stamp strings or passed to PI as a number of seconds, suitable for use in computations. If the value is translated into a string, the time stamp format is used (/TF).
8	Directs the OPC server to send the canonical data type. The PI OPC interface tries to transform the value into the proper data type for the PI point. Use with caution, because the transformation can fail if the source data type is not compatible with the PI point data type, or if the value cannot be represented using the PI point data type.
>= 1024	When a post-processing DLL is used with the PI OPC interface, directs the data to be processed by the DLL. Adding any of the above settings (1-8) to 1024 enables the abovementioned functionalities to be used as well. For more information, see the <i>TimeArray Plug-in User Manual</i> .

Location3 (processing type of tag)

Location3 specifies whether this PI point is a polled, advise, event, or output point.

Location3	Description
0	Polled or event
1	Advise
2	Output
3	Polled watchdog used with server-level failover
4	Advise watchdog used with server-level failover

For an advise point, the PI OPC interface registers for updates with the OPC server, and the OPC server sends new values to the PI OPC interface (at a rate not exceeding the update rate for the group.)

Location4 (scan class)

Location4 configures the scan class for the PI point. The scan class determines the frequency at which input tags are scanned for new values. **Location4** must be a positive number. For trigger-based points, set **Location4** to zero. For output tags, **Location4** configures the output class. When necessary for load balancing, the interface distributes tags in scan class 1 across multiple OPC groups. Scan classes other than scan class 1 are assigned to separate groups for load balancing.

You can configure scan classes for the PI OPC interface as follows:

Tag	Maximum Number of Groups
Polled	200
Advise	600
Event	199

Tag	Maximum Number of Groups
Output	No maximum

Specify scan frequency and optional offset using the following format:

HH:MM:SS.##,HH:MM:SS.##

Examples:

/f=00:01:00,00:00:05 /f=00:00:07

or, equivalently:

/f=60,5 /f=7

If you omit HH and MM, the scan period is assumed to be in seconds. Sub-second scans are specified as hundredths of a second (.01 to .99).

To define a time of day at which a single scan is performed, append an L following the time:

HH:MM:SS.##L

The OPC standard does not guarantee that it can scan data at the rate that you specify for a scan class. If the OPC server does not support the requested scan frequency, the frequency assigned to the class is logged in the `pipc.log` file. If the interface workload is heavy, scans can occur late or be skipped. For more information on skipped scans, see the *UniInt Interface User Manual*.

Scanning offsets

To mitigate the interface and OPC server workload, you can use the offset to stagger scanning. If an offset is specified, scan time is calculated from midnight on the day that the interface was started, applying any offset specified. In the above example, if the interface was started at 05:06:06, the first scan occurs at 05:07:05, the second scan at 05:08:05, and so on. If offset is omitted, scanning is performed at the specified interval, regardless of clock time.

Offsets determine when the interface asks the OPC server for the current values for polled classes. They do not control the behavior of the OPC server, and have no effect on advise classes unless the `/GA` parameter is specified to stagger the activation of groups. In this case, the offsets are used to time the activation of all groups except for scan class 1 (which is reserved for advise tags).

Update rates

The OPC server reads data from the device according to the update rate for the group in which the item resides. By default, the update rate is the same as the scan rate. To override the default using PI ICU, browse to the **OPCInt > OPC Server > Advanced Options** window and enter the desired update rates in the **Update Rates** section. (`/UR`).

For polled groups, configuring an update rate that is shorter than the scan period can ensure that the interface is receiving current data. For example, if the scan period is five seconds but the update rate is two seconds, the data is no more than two seconds old when it is read. However, note that a faster update rate increases the OPC server workload.

For advise groups, assign identical update and scan rates, with one exception: if you are using UniInt Failover Phase 1, then to ensure that the interface sees new values for failover heartbeat tags as soon as possible, set the update rate to half the scan period. This configuration reduces the risk of thrashing, where control switches back and forth needlessly. Dedicate a scan class with faster update rate to the failover heartbeat tags. OSIsoft recommends using Phase 2 failover instead.

Location5 (OPC deadband)



Note:

Under the OPC standard, deadband processing is optional for servers. Before attempting to configure advise points, be sure that your OPC server supports deadband processing. If the OPC server does not support deadband processing, the PI point is updated for all value changes to the tag, depending on the exception parameters specified for the PI point.

For advise points, **Location5** specifies a deadband value for analog OPC items. Use deadband to reduce the amount of network traffic from the OPC server to the PI OPC interface. If the change between the last value read and the new value is less than the deadband, the OPC server does not send the value to the PI OPC interface. Note that OPC deadband processing is not the same as PI deadband (exception) processing.

The **EuMin** and **EuMax** attributes in the OPC item definition specify the value range for the tag. These attributes correspond to the **Zero** and **Span** attributes in the PI point definition. To configure the deadband, specify a percentage of the range multiplied by 100. For example, if the OPC server point is defined as analog with an **EuMin** of **-10** and an **EuMax** of **10**, and **Location5** contains **2500** (meaning 25%), data is sent to the PI OPC interface only when the difference between the new value and the old value is at least 5 (25% of 20 = 5). PI exception processing continues to be applied to the values received by the interface. The deadband only affects the values sent by the OPC server.

InstrumentTag (OPC ItemID)

The **InstrumentTag** attribute maps the PI point to an OPC item. This field must exactly match the item name as defined on the OPC server, including any punctuation, spaces, and case. To verify an ItemID, use OPCClient. If the OPC server supports browsing, choose **List Server's Tags** to see a list of defined ItemIDs. To display the full ItemID required for **InstrumentTag** field, double-click the ItemID in the list.

The maximum length of the **InstrumentTag** attribute depends on the versions of the PI Server and API in use. If PI API is version 1.6.0.2 or higher and PI Server is 3.4.370.x or higher, the maximum length is 1023. For all lower versions, the maximum is 32. If you are running lower versions and require more than 32 characters to specify the ItemID, you must enable the PI SDK or use the extended descriptor (**ExDesc** attribute) to specify the OPC ItemID.

ExDesc (extended descriptor)

The extended descriptor attribute is a multi-purpose field that is used as follows:

- **Event-based data collection**

To define an event tag, set this attribute to **event=tagname**. When the specified tag has an exception event, the tags for which it is the trigger are read from the OPC server.

- **Dzero for scaled tags**

When the device returns values that must be scaled to fit the range of values stored by the tag, store the device zero in **ExDesc**. To specify the device span, use the **Convers** attribute. The format for specifying the device zero is **Dzero=nnnnn.nnn**



Note:

If the ItemID for this point is longer than 32 characters and the PI SDK is disabled, the ItemID must specify the **ExDesc** as `instr=ItemID`. The ItemID must exactly match the ItemID defined on the OPC server. If the ItemID contains a comma or space, enclose it in double quotes.

OPC ItemIDs might have trailing spaces, which can be truncated if not using the PI SDK and specifying the ItemID in the **InstrumentTag** field. To include the trailing blanks, enclose the ItemID in double quotes.

- **Target OPC item for output tag time stamp**

To direct the time stamp of an output tag to an OPC item, specify the target ItemID in **ExDesc**. The format written depends on the data type of the ItemID that is to receive the time stamp, as follows:

```
Tim=ItemID
```

```
Dat=ItemID
```

- **Tim:** The time stamp is written as a string (VT_BSTR), formatted as configured for the PI OPC interface instance (/TF)
- **Dat:** The time stamp is written as a VT_DATE.

VT_DATE is a universal (UTC) format that does not depend on the time zone or daylight savings time setting. For VT_BSTR, the time stamp comes from the PI Server and is not adjusted for differences in time zone or daylight savings time setting. In error messages related to this time stamp ItemID, the PI OPC interface reports a generated tag name of the form TS:xxxxxx, where xxxxxx is the name of the PI output tag.

If you use this attribute to specify more than one setting, put a comma between the definitions. By default, leading and trailing spaces are stripped from entries in this attribute. To preserve leading and trailing spaces, enclose your entry in double quotes.

SourceTag

For output points (points that write data to the OPC server), this attribute specifies the PI point from which data is read. See [Output points](#) for more information.

TotalCode

This attribute contains a code that specifies how the value is to be scaled. **TotalCode** is used in conjunction with the **SquareRoot**, **Convers**, and **ExDesc** attributes. See [Transformations and scaling](#) for details.

SquareRoot

Specifies that the square or square root of the value is to be used. See [Transformations and scaling](#) for details.

Convers

For scaled tags, this attribute contains the device span. The device item can have a zero and a span, which define the actual span of values that the device sends. The PI OPC interface can use these two values to translate the units used by the device to the units defined for the PI point. The **Convers** attribute can also contain an offset or multiplier. See [Transformations and scaling](#) for details.

UserInt1 (OPC array index)

UserInt1 maps a PI point value to an element in an OPC array item. For PI points that are not mapped to an OPC array, set **UserInt1** to 0 (zero). Ensure that all PI points that are mapped to the same OPC array have identical settings for **InstrumentTag**, **ExDesc**, and all location attributes.

An OPC array contains multiple values plus a single time stamp and a quality field. These items can be identified by using the PI OPCClient tool to read the item and examining the data type returned by the OPC server. If it is an array item, the type of the value is VT_ARRAY | VT_ other, where VT_ other is a data type such as VT_R4 or VT_I2. The values in the array are sent as one data item and they all have the same data type.

PI Server does not support PI points with an array type, so values must be assigned to a number of individual PI points. The first value in the array maps to the PI point that has **UserInt1** set to 1, the second to the tag with **UserInt1** set to 2, and so on. If these values need to be processed as different data types, use the **Location2** attribute for the PI point with **UserInt1**=1 and the settings for scaling and transformation for each individual point to configure how the PI OPC interface handles the individual value. The PI OPC interface receives the data using the data type specified by the **Location2** value for the point with **UserInt1**=1, then processes the value according to how the individual point is configured. Note that some servers cannot provide array data using any data type other than the canonical data type (the one displayed in the PI OPCClient if you omit data type). For those servers, you must either use a PI tag with the correct data type, or set **Location2** to 8 to configure the interface to ask for the canonical data type. For maximum efficiency, always use the canonical data type.

UserInt2 (event group)

This attribute assigns an event group to an event point. For points that are not event points, set **UserInt2** to 0 (zero). See [Event points](#) for details.

Scan

This attribute enables or disables data collection for the PI point. By default, data collection is enabled (**Scan** is set to 1). To disable data collection, set **Scan** to 0. If the **Scan** attribute is 0 when the interface starts, the interface does not load or update the point. If you enable scanning while the interface is running, the time required for data collection to start depends on how many points you enable, because they are processed in batches. For efficiency, if you need to enable scanning for a large number of points, stop and restart the interface. If a point

that is loaded by the interface is subsequently edited so that the point is no longer valid, the point is removed from the interface and `SCAN OFF` is written to the point.

Shutdown

By default, the PI Shutdown subsystem writes the SHUTDOWN digital state to all PI points when PI Server is started. The time stamp that is used for the SHUTDOWN events is retrieved from a file that is updated by the snapshot subsystem. The time stamp is usually updated every 15 minutes, which means that the time stamp for the SHUTDOWN events is accurate to within 15 minutes in the event of a power failure. For additional information on shutdown events, refer to PI Server manuals.



Note:

The SHUTDOWN events that are written by the PI shutdown subsystem are independent of the SHUTDOWN events that are written by the interface.

To prevent SHUTDOWN events from being written when PI Server is restarted, set the **Shutdown** attribute to `0`. To configure the PI shutdown subsystem to write SHUTDOWN events only for PI points that have their **Shutdown** attribute set to `1`, edit the `\\PI\dat\Shutdown.dat` file, as described in PI buffering documentation.

Exception processing

The **ExcMax**, **ExcMin**, and **ExcDev** parameters control exception reporting in the PI OPC interface. To turn off exception reporting, set **ExcMax**, **ExcMin**, and **ExcDev** to `0`. See the *UniInt Interface User Manual* for more information about exception processing.

- **ExcMax**

This attribute configures the maximum time period allowed between sending values to PI Server. This setting applies to both advise and polled tags. For advise tags, if the PI OPC interface does not receive a value after the specified number of seconds and does not detect a dropped connection, it sends the last value received to the PI server with the time stamp set to the current time. For polled tags, the interface sends a value to PI Server if it has not sent one in the last **ExcMax** seconds, even if the new value does not pass **ExcDev** tests.

- **ExcMin**

This attribute configures the minimum time period between values sent to PI Server.

- **ExcDev**

This attribute configures the minimum change from the last value sent to PI Server required for the PI OPC interface to send a new value.

Output points

Output points send data from the PI server to the OPC server. Note that only good values can be sent. System digital states are not sent to OPC items. To configure an output point, edit the point using PI Point Builder and specify the following settings:

- Set **Location1** to the PI OPC interface instance. (**/ID**)
- Set **Location3** to 2.
- Specify the ItemID (the OPC item to be written).
- Optional: Specify the source point (the PI point that contains the value to be written to the OPC server). Not required if you intend to send the value directly to the output item without copying the values and time stamps to a PI point.
- Optional: Set **Location4** to the desired output group. Output points with **Location4** set to \emptyset are distributed across output groups for load balancing.

There are two mechanisms for triggering an output: configuring a separate source point, and writing new values to a snapshot, as described in the following sections.

Use a source point

To configure the output point using a source point, set the **SourceTag** attribute to the name of another PI point that will contain the values that you want written to the OPC item. When the source point is successfully updated, the new value is written to the target OPC item. If the PI OPC interface succeeds in updating the OPC item, it writes the value and time stamp to the output point. If the interface does not succeed in updating the OPC item, it writes a digital state that describes the error to the output point. For output points, a success status indicates that the OPC server item has been updated, but there is no guarantee that the corresponding data source has been updated. To verify that the data source has been updated, create a corresponding input point and add logic to ensure that the values of the input and output points match.

The **PointSource** of the output point must match the **PointSource** of the interface instance, but the source point can be associated with any point source. The data type of the source point must be compatible with that of the output point.

No source point

To use the same PI point as the source and the output point, leave the **SourceTag** attribute blank. Any means of updating the snapshot value of the output point is acceptable. To trigger the output to the target OPC item, the time stamp must be more recent than the previous time stamp, regardless of whether the value changes. The value that you enter into the output point's snapshot is written to the target item on the OPC server. Any new value is sent to the OPC item.

Event points

Event PI points are configured with a trigger PI point. When the trigger point receives a value, the event point is read. To create event points, set Location4 to \emptyset and specify the name of the trigger PI point in the ExDesc attribute using the following format:

```
TRIG='triggertagname' event_condition
```

Enclose the name of the trigger point in single quotes. To treat all changes as triggering events, omit *event_condition*. For more information about the *event_condition* argument, see the *UniInt Interface Users Manual*.

By default, the server is requested to update its cache every second for every event point defined. OPC v2.0 servers always read event points from the device, not the cache. To minimize

the overhead incurred when the OPC server updates the cache, set the event rate (**/ER**) to a high value such as eight hours. For v1.0a OPC servers, asynchronous reads come from the cache. The cache does not need to be updated frequently for all event points, so you can increase the event rate.

To define a set of event PI points that are read together in the same OPC event group, assign identical integer values to the **UserInt2** attribute of the PI points. (For example, a plug-in DLL that post-processes data might require the data to be sent in a single group.)

For efficiency with v1.0a servers, separate event points into groups based on the triggering event. For OPC v2.0 servers, separate event points according to the data source. The OPC v2.0 standard requires that all asynchronous reads originate from the device rather than from the server's cache, so set the cache update rate high and do not group values that come from different devices. The following example point definitions illustrate this approach:

Tag	ExDesc	InstrumentTag	Location1	Location2	Location3	Location4	Location5	UserInt1	UserInt2
PM1_Temp.PV	TRIG=PM1_Trigger	ItemID1	1	0	0	0	0	0	1
PM1_Rate.PV	TRIG=PM1_Trigger	ItemID2	1	0	0	0	0	0	1
PM2_Temp.PV	TRIG=PM2_Trigger	ItemID3	1	0	0	0	0	0	2

In the preceding example, **PM1_Trigger** and **PM2_Trigger** are points that are updated either by this PI OPC interface instance, another interface, or by manual entry. When **PM1_Trigger** gets a new event in the PI snapshot, the PI OPC interface sends the OPC server a **read** command that requests data for both **PM1_Temp.PV** and **PM1_Rate.PV**. Both values are returned in a single call. Likewise, when **PM2_Trigger** gets an event in the snapshot, the interface requests a value for **PM2_Temp.PV**.

Reading OPC array item PI points

OPC servers can contain arrays of data, which are composed of multiple values of the same data type plus a single quality and time stamp. Because the PI server does not support array data types, you must configure one PI point for each array element that you want to store in the PI System. (or use the TimeArray plug-in; for details, see the TimeArray plug-in user guide).

To define a PI point to contain an element from an OPC array, specify the **ItemID** of the array item in the **InstrumentTag** attribute and set **UserInt1** to the index number of the element in the array. You must define a PI point for the first array element (**UserInt1 = 1**), even if you do not require its data. However, you do not need to define points for all array elements, only for the first array element and any individual elements of interest.

All PI points configured for an OPC array must have identical settings for **PointSource**, **Location1**, and **Location4** (and **UserInt2**, if they are event points). For advise points, set **Location3** to 1. For polled points, set **Location3** to 0.

When configuring PI points to read OPC arrays, note the following:

- You must define a point that reads the first array element.
- Assign the points to the same scan class.

- To optimize CPU usage, do not use the same scan class to read more than one OPC array.
- If you need to read the same OPC array element into more than one point, you must assign the points to different scan classes.

Reading OPC arrays as event points

Multiple scan classes can have the same scan period, and event classes are a logical grouping of PI points. For efficiency, put event arrays into their own scan classes with any other points that need to be read with the array.

If an array tag from the OPC server is read into multiple PI points, each PI point receives the value of the array element indexed by the **UserInt1** setting and the same time stamp and quality, because an array contains multiple values but only one time stamp and quality. To read an array tag into a single PI point, you must use the TimeArray plug-in, which stores an array of values into a single PI point as successive data values, incrementing the time stamp that came with the array by a configured interval for each value. For details, refer to the TimeArray plug-in manual.

Configuring arrays that are read as event tags is complex: because only the first array item (with **UserInt1** = 1) causes a read, you must create a dummy trigger PI point to use with the rest of the array items. That PI point must have a **PointSource** that is either unused or used for manual entry points (lab data usually is entered manually, so L is often used as the **PointSource** for manual entry PI points). In the following example, the trigger PI point is called **TriggerTag** and the dummy trigger PI point is called **DummyTrigger**.

Tag	ExDesc	InstrumentTag	Location1	Location2	Location3	Location4	Location5	UserInt1	UserInt2
Array000 1.PV	TRIG=TriggerTag	Data.Array	1	0	0	0	0	1	1
Array000 2.PV	TRIG=DummyTrigger	Data.Array	1	0	0	0	0	2	1
Array000 3.PV	TRIG=DummyTrigger	Data.Array	1	0	0	0	0	3	1

Because all the tags in an array must belong to the same group, even if the OPC server is v2.0 and some part of the array data comes from a different device than the rest of the array data, all the array tags must be configured to be in the same event group.

Reading OPC quality into a digital PI point

To store OPC quality in a digital PI point, use transformations and scaling to translate quality to a digital state set of **Bad Value**, **Questionable Value**, **Invalid Value**, or **Good Value**. To define such a PI point, set **Location2** to 4 to read the quality of the item rather than its value, then define a mathematical transformation that translates the quality values to an integer from 0 to 3. Divide the quality number by a conversion factor that produces the proper number.

OPC quality is returned in ranges of values, as follows:

Range	Description
Less than 0x40	Bad Value
Greater than or equal to 0x40 and less than 0x80	Questionable Value
Greater than or equal to 0x80 and less than 0xc0	Not used by OPC
Greater than or equal to 0xc0	Good Value

Because each range has the same size (decimal 64), you can use a simple conversion to obtain the corresponding digital state, as follows:

Convers	TotalCode	SquareRoot	Dzero	Operation
Not 0	3	0	Defined	Input points: $Value = (Value / \mathbf{Convers}) - Dzero$ Output points: $Value = (Value + Dzero) * \mathbf{Convers}$

Define the point attributes as follows:

Attribute	Setting
Convers	64
TotalCode	3
SquareRoot	0
ExDesc	"Dzero=0"

Configuring failover for the PI OPC DA interface

The PI OPC DA interface provides two methods for configuring failover, to ensure that data collection continues if either the interface or the OPC Server fails.

- OPC Server-level failover ensures that, if the PI OPC interface stops receiving data from the currently connected OPC server, it can switch to another OPC server and resume data collection.
- UniInt failover ensures that, if one instance of the PI OPC interface fails, another instance can take over data collection.

If you are configuring both, configure and verify UniInt failover first. Disable UniInt failover and configure and test server-level failover separately, then re-enable UniInt failover.

Topics in this section

- [UniInt failover](#)
- [OPC server-level failover](#)

UniInt failover

UniInt failover ensures against data loss by enabling a backup PI OPC interface instance to take over data collection if a primary instance fails. There are two approaches to configuring failover: synchronization through the OPC server (phase 1 failover), and synchronization through a shared file (phase 2 failover). This guide tells you how to configure phase 2 failover.



Note:

Phase 1 failover is now deprecated and is not recommended. For details, contact OSISOFT Technical Support. For more details about UniInt failover, refer to the *UniInt Interface User Manual*.

Failover works as follows: you configure two identical instances of the PI OPC interface on two different computers. One instance functions as the primary instance and the other one as the backup, depending on which one is started first. If the primary fails, the backup becomes the primary and takes over transmitting data from the OPC server to the PI server. If that interface subsequently fails and the other interface has been restored, the other interface becomes primary and resumes transmitting data. (Note that “primary” and “backup” are terms used to clarify operation. Failover seeks to keep a running instance of the PI OPC interface connected with a functional OPC server, so, in action, either interface might be primary.)

If the PI OPC interface instances are configured to use disconnected startup, the interfaces can start and fail over even if the PI Server is unavailable, as long as they both have access to the shared file.

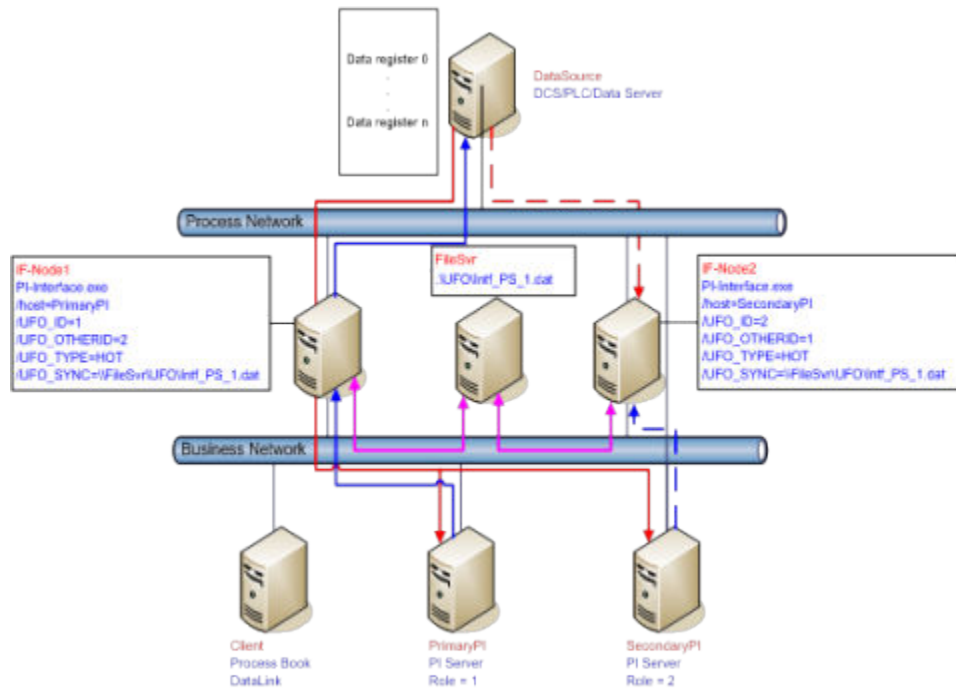
Topics in this section

- [How UniInt failover works](#)
- [Hot, warm, and cold failover modes](#)
- [Configure shared-file \(Phase 2\) failover](#)
- [Test failover configuration](#)

How UniInt failover works

Shared-file (Phase 2) UniInt failover uses PI points to control failover operation. Status information from the points is maintained in a shared file, removing the requirement for a PI server connection after the instances are running. If the shared file cannot be accessed, the interface instances use the PI server to exchange control data.

The following diagram shows a typical setup operating normally.



The solid magenta lines show the data path from the PI OPC interface nodes to the shared file. During normal operation, the primary PI OPC interface collects data from the OPC server and sends it to the PI server. The ActiveID point and its corresponding entry in the shared file are set to the failover ID of the primary instance. Both primary and backup instances regularly update their heartbeat value, monitor the heartbeat value and device status for the other instance, and check the active ID. Normal operation continues as long as the heartbeat value for the primary instance indicates that it is running, the ActiveID has not been manually changed, and the device status on the primary PI OPC interface is good.

Phase 2 failover tracks status using the following points.

- **ActiveID**

Tracks which PI OPC interface instance is currently forwarding data from the OPC server to the PI server. If the backup instance detects that the primary instance has failed, it sets ActiveID to its own failover ID and assumes responsibility for data collection (thereby becoming the primary).

- **Heartbeat Primary**

Enables the backup PI OPC interface instance to detect whether the primary instance is running.

- **Heartbeat Backup**

Enables the primary PI OPC interface instance to detect whether the backup instance is running.

- **Device Status Primary and Device Status Backup**

Interface node status.

The following table lists common values.

Device Status	Description
0	Interface working properly, reading/writing data
1	Starting PI OPC interface, not connected to device
2	Connected to device, not receiving data
3	Communication error with device
4	Shutting down the PI OPC interface
50	Attempting to force failover
10	Connected to device, not receiving data
90	Starting PI OPC interface, not connected to device
95	Communication error with device
99	Shutting down the PI OPC interface



Note:

Do not confuse the device status points with the UniInt health device status points. The information in the two points is similar, but the failover device status points are integer values, while the health device status points are string values.

To indicate that it is up and running, each PI OPC interface instance refreshes its heartbeat value by incrementing it at the rate specified by the failover update interval. The heartbeat value starts at one and is incremented until it reaches 15, at which point it is reset to one. If the instance loses its connection to the PI server, the value of the heartbeat cycles from 17 to 31. When the connection is restored, the heartbeat values revert back to the one-to-15 range. During a normal shutdown process, the heartbeat value is set to zero.

If the shared file cannot be accessed, the PI OPC interface instances attempt to use the PI Server to transmit failover status data to each other. If the target PI OPC interface also cannot be accessed through the PI server, it is assumed to have failed, and both interface instances collect data, to ensure no data loss. In a hot failover configuration, each PI OPC interface instance queues three failover intervals worth of data to prevent any data loss. When failover occurs, data for up to three intervals might overlap. The exact amount of overlap is determined by the timing and the cause of the failover. For example, if the update interval is five seconds, data can overlap between 0 and 15 seconds.

Hot, warm, and cold failover modes

The failover mode specifies how the backup PI OPC interface instance handles connecting to an OPC server, creating groups, and adding points when failover occurs. The faster the backup interface can take over data collection, the less data is lost. However, the ongoing activities required to maximize the readiness of the backup PI OPC interface incur a cost in OPC server load and system resources. To determine which mode to use, consider how long failover takes

and how much workload your system can handle. Be prepared to experiment, and consult your OPC server documentation and vendor as needed.

The PI OPC interface provides five levels of failover, from cold to hot. Higher (“hotter”) levels ensure that more data is preserved in the event of failover, but impose increasing workload on the system. (The highest level, hot failover, is lossless unless both the primary and backup PI OPC interface nodes fail together.) The following sections provide more details about each level.

Hot failover

Hot failover is the most resource-intensive mode. Both the primary and backup OPC interface instances are collecting data, possibly from the same OPC server. No data is lost during failover, but the OPC server carries a double workload, or, if two servers are used, the backend system must support both OPC servers.

Warm failover

There are three options for warm failover:

- **No groups on backup OPC server (Option 1)**

The backup instance connects to the OPC server every 30 seconds and checks its status, but does not create groups or add items. Because the OPC interface preloads point information from PI Server, this option is faster than cold failover, but when the backup becomes the primary instance, it must create groups, add items to them, activate them, and then advise them. Because of the time required, data can be lost during failover. This option is for OPC servers that cannot support groups when they are not the active OPC server.

- **Inactive groups on backup OPC server (Option 2)**

The backup instance connects to the OPC server, creates inactive groups, and adds items to the groups, but does not activate the groups. For most OPC servers, this approach reduces workload, because the server does not need to maintain the current values for inactive groups. When the OPC interface becomes primary, it activates the groups and then advises them. This approach is faster than option 1.

- **Active groups on backup OPC server (Option 3)**

The backup instance connects to the OPC server, creates groups, adds items, and activate the groups, but does not advise the groups. The OPC server must maintain its cache of current values for all the items, but does not send the values to the OPC interface. If both OPC interfaces are connected to the same server, and the server maintains one central cache for data, this approach might impose very little load on the server, because the cache must be updated for the primary OPC interface. For an OPC server that does not use a centralized cache or a configuration in which the OPC interface instances connect to different OPC servers, this approach can impose a considerable load on an OPC server or the data source system. When the backup OPC interface becomes primary, all it needs to do to start collecting data is to advise the groups, making this the fastest warm failover option.

Cold failover

Cold failover is desirable if an OPC server can support only one client, or if you are using redundant OPC servers and the backup OPC server cannot accept connections. The backup instance does not connect with the OPC server until it becomes primary. At this point, it must create groups, add items to groups, and advise the groups. This delay almost always causes some data loss, but imposes no load at all on the OPC server or data source system.

**Note:**

The OPC interface supports using *watchdog points* to control failover. Watchdog points enable the OPC interface to detect when its OPC server is unable to adequately serve data and failover to the other interface if the other interface is better able to collect data. This approach is intended for OPC servers that are data aggregators, collecting data from multiple PLCs. If one point on each PLC is designated as a watchdog point, the interface can be instructed to failover if less than a specified number of those points are readable. This approach enables the benefits of redundancy to be applied at the data collection level. For more on how to configure this option, see [Configure server-specific watchdog PI points for efficient failover](#).

Configure shared-file (Phase 2) failover

To configure failover, perform the following steps:

Procedure

1. Create identical PI OPC interface instances on the primary and backup nodes.

A simple way to ensure that the instances are identical is to use PI ICU to configure the primary instance correctly, then copy its batch file to the backup PI OPC interface node. On the backup node, create the instance by using PI ICU to import the batch file. Verify that the instances can collect data.

2. Configure buffering for each instance and verify that buffering is working.
3. To configure the location of the shared file, create a folder and set its sharing properties to grant read/write access for both PI OPC interface nodes to the user that runs the OPC interface instance. To ensure that the file remains accessible if either of the OPC interface nodes fails, put the folder on a machine other than the primary or backup OPC interface nodes.

Note that the shared file is a binary file, not text.

4. On both OPC interface nodes, use PI ICU to configure failover as follows:
 - a. Choose **Unilnt > Failover**. The Unilnt Failover page is displayed.
 - b. Check **Enable Unilnt Failover** and choose **Phase 2**.
 - c. In the **Synchronization File Path** field, specify the location of the shared file.
 - d. In the **UFO Type** field, choose the level of failover that you want to configure, ranging from **COLD** to **HOT**.
 - e. Specify a different failover ID number for the primary and backup instances, and configure the location of the primary and backup instances.

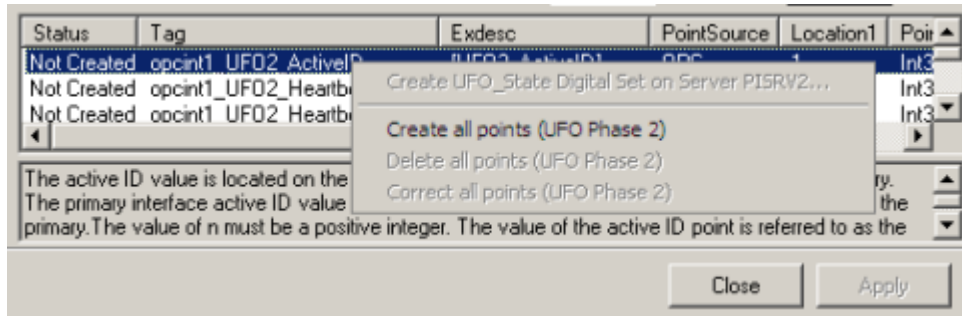
If you use a PI collective, point the primary and backup instances to different members of the collective. Go to the **General** tab and set the **SDK Member** field. (**/host**).

**Note:**

Make sure that the **UFO_ID** of one interface matches the **UFO_OtherID** of the other interface, and vice versa. If the PI Servers are a collective, set **Host** on the primary interface node (PI ICU **General** tab) to the Primary PI Server, and set **Host** on the backup interface node (PI ICU **General** tab) to the secondary PI Server.

- f. Click **Apply**.

- g. When creating the first instance, create the required PI points by right-clicking the list of failover points and choosing **Create all points (UFO Phase 2)**, as shown in the following figure.



- h. Click **Close** to save changes and update the PI OPC interface batch file.

Test failover configuration

To verify that failover is working, perform the following steps:

Procedure

1. Start the first PI OPC interface using PI ICU. Verify that the startup output indicates that failover is correctly configured:

```
OPCpi> 1 1> UniInt failover: Successfully Initialized:
          This Failover ID (/UFO_Id):      1
          Other Failover ID (/UFO_OtherId): 2
```

2. After the primary OPC interface has successfully started and is collecting data, start the other PI OPC interface instance. Again, verify that startup output indicates that failover is correctly configured.
3. To test failover, stop the primary OPC interface. Verify that the backup OPC interface has detected the absence of the primary instance and taken over data collection by examining its output for the following messages:


```
> UniInt failover: Interface is attempting to assume the "Primary" state.
Waiting 2 ufo intervals to confirm state of other copy.
Fri Jun 22 11:43:26 2012
> UniInt failover: Waited 2 ufo intervals, Other copy has not updated our
activeId, transition to primary.
Fri Jun 22 11:43:26 2012
> UniInt failover: Interface in the "Primary" state and actively sending data
to PI.
```
4. Check for data loss in PI Server (for example, using PI ProcessBook to display a data trend).
5. Test failover with different failure scenarios (for example, test loss of PI Server connection for a single PI OPC interface copy). Verify that no data is lost by checking the data in PI and on the data source.
6. Stop both copies of the PI OPC interface, start buffering, configure and start each interface instance as a service.

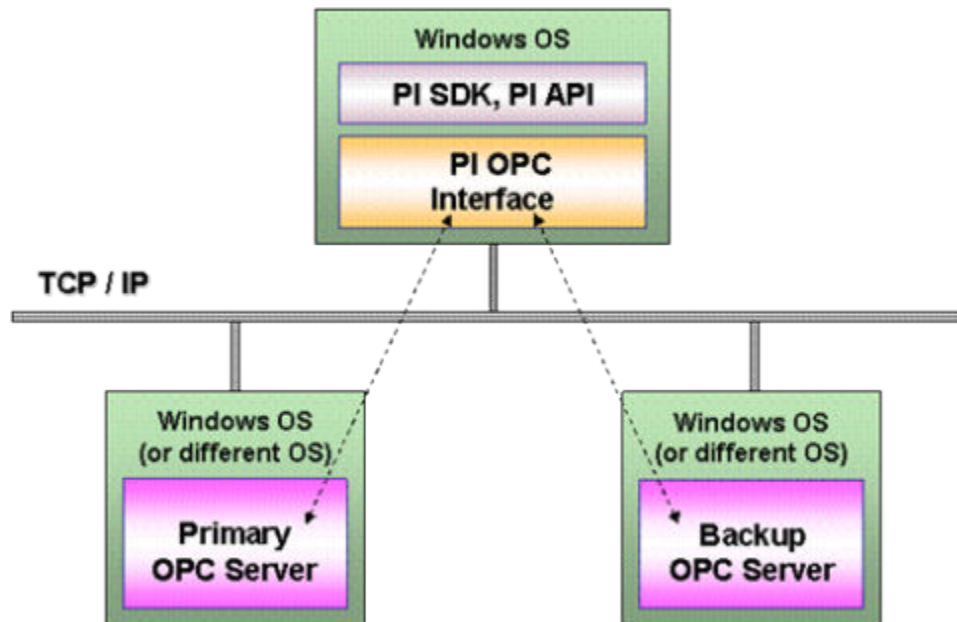
OPC server-level failover

To ensure that data continues to flow from the OPC server to the PI OPC interface, the interface can be configured to switch to another OPC server under the following conditions:

- Loss of connectivity to current OPC server
- Status of the OPC server status changes to a value other than "RUNNING".
- Specified OPC server items indicate that the OPC server is unavailable, either by a specific value or by the quality.

OPC servers can be fully redundant (multiple servers active at the same time) or configured for failover, where one server at a time is active. Note that OPC servers vary widely in their approach to tracking and reporting status, so consult your OPC server documentation to determine what options are supported.

The following diagram illustrates a basic OPC server-level failover configuration.



Topics in this section

- [Configuring server-level failover](#)
- [Controlling failover timing](#)

Configuring server-level failover

The following sections tell you how to configure various approaches to OPC server-level failover. For debugging and testing, you can create a PI string point to track the active OPC server. Assign the PI point to an unused point source. In PI ICU, go to the **OPCInt > Failover > Server Level** page and enter the PI point name in the **Current Active Server Tag** field. To display the value of the point, launch PI SMT and use its **Data > Current Values** feature. When failover occurs, the value of this point changes to the name of the currently connected OPC server. The archive of these changes enables you to view failover history.

Topics in this section

- [Test OPC server failover](#)
- [Failover on OPC server status change](#)
- [Monitoring server status using OPC item values](#)

Test OPC server failover

If the PI OPC interface cannot connect to the current server, use PI ICU to configure the interface to fail over to the other OPC server as follows:

Procedure

1. Go to the **OPCInt Failover > Server Level** pane and enter the node and name of the other OPC server.

This basic configuration triggers failover only when the PI OPC interface loses connectivity to the OPC server.

2. To verify that failover occurs when connectivity is lost:
 - a. Start both OPC servers, then start the PI OPC interface.
 - b. Use PI SMT or the pigetmsg utility to check the PI SDK log on the PI server node for messages that verify successful startup.
 - c. Stop the currently active OPC server and check the SDK log to confirm that the OPC interface has switched to the other OPC server.
 - d. To switch back to the first OPC server, restart it, stop the second server, and check the SDK log or the value of the PI active server point, if defined, to verify that the PI OPC interface has switched back to the first OPC server.

Failover on OPC server status change

On the PI ICU **OPCInt > Failover > Server Level** page, check **Failover if Server Leaves RUNNING State**.

To configure a waiting period, set the **Wait For RUNNING State** field to the desired number of seconds. If the OPC server has not entered the **RUNNING** state before the specified period expires, the PI OPC interface attempts to connect to the other server. If the other server has not entered the **RUNNING** state before the specified period expires, the interface retries the first server, alternating between the two until one is detected to be running again.

Monitoring server status using OPC item values

To enable the PI OPC interface to track the availability of OPC servers, you can configure watchdog PI points that are mapped to OPC items that reliably reflect the state of the OPC server. The OPC items must contain a positive integer if the OPC server is running or 0 if the OPC server is unavailable, and watchdog points must have an integer data type.

When choosing the OPC items that you map to the watchdog points, consider which ones are most reliable and representative of the state of the OPC server. For example, in a manufacturing context, an item that counts number of units manufactured might make sense.

To ensure that the values in watchdog points are valid (0 or positive integer), use PI scaling and transformation as required. To ensure that the primary and backup OPC servers report status consistently, choose an OPC item based on the data source itself (as opposed to an item that originates in the OPC server).

To reduce system workload, configure the watchdog points as advise points if the OPC server supports advise points. If not, assign them to a scan class with a short scan period.

If more than one instance of the PI OPC interface is running on the same node, you must create separate watchdog points for each instance of the interface, because each instance scans only those PI points that belong to its unique **PointSource**.

Topics in this section

- [Configure a single watchdog PI point](#)
- [Configure multiple watchdog PI points](#)
- [OPC item quality](#)
- [Configure server-specific watchdog PI points for efficient failover](#)

Configure a single watchdog PI point

If you can determine the availability of the OPC server based on a single OPC item, create a single watchdog PI point. When the value of the OPC item is 0, the PI OPC interface attempts to connect to the other server. If it cannot connect successfully to the other server within the specified connection timeout period, it attempts to reconnect to the first OPC server again.

To create and configure a watchdog point:

Procedure

1. Create a PI point. Map the point to an OPC item that you consider a reliable indicator of server status.

The OPC item to which the point is mapped must be defined identically in both the primary and backup OPC servers, while possibly having different values on the two servers.
2. Indicate the watchdog point for the primary and backup OPC servers: Using PI ICU, go to the **OPCInt > Failover > Server Level** page and configure the **Primary Server Watchdog Tag** and **Backup Server Watchdog Tag** fields.
3. Verify that the OPC item triggers failover:
 - a. Start the OPC servers and verify that the watchdog item is non-zero on at least one of the servers. Start the OPC interface.
 - b. Manually set the OPC item to 0 on the currently used server.
 - c. Examine the PI SDK log or check the Active Server point to determine whether the interface failed over to the other OPC server.

Configure multiple watchdog PI points

If you must assess several OPC items to determine availability (such as when the OPC server can report on the availability of the back end data sources), configure multiple watchdog PI points. The sum of the values of the watchdog points determines whether the server is considered active. The OPC interface initially assigns each watchdog a value of 1 and

recalculates the total whenever it receives a new value for one of the tags. If the sum goes below the specified minimum, failover is triggered.

Procedure

1. Create PI points and map them to the OPC items that you consider reliable indicators of OPC server status. For each point, set **Location3** to 3 for polled points or 4 for advise points.
2. Using PI ICU, go to the **OPCInt > Failover > Server Level** page and set the **Multiple Watchdog Tags Trigger Sum** field to the minimum acceptable total of the values of the watchdog points.
3. Verify that failover is triggered if the total of the values drops below the specified minimum:
 - a. Start the OPC servers and the OPC interface.
 - b. Manually set the values of the OPC items.
 - c. Examine the SDK log or check the Active Server point to determine whether the interface failed over to the backup OPC server.

OPC item quality

If you have multiple watchdog PI points, you can configure failover to occur when a specified number of watchdog points are read with BAD quality. To configure this setting, go to the **OPCInt > Failover > Server Level** page and enter the desired maximum in the **Maximum number of Watchdog Tags which can have Bad Quality or Any Error without triggering Failover** field.

Configure server-specific watchdog PI points for efficient failover

OPC servers track their own states (isolated mode). To enable the PI OPC interface to determine the state of an OPC server before attempting to failover to it, configure both OPC servers to track each other's state as well (server-specific mode). This configuration enables the OPC server to determine the state of both servers without the overhead of creating a second connection.



Note:

The method by which an OPC server tracks its state is highly vendor-dependent and implementations vary. For details, consult your OPC server documentation.

To configure server-specific mode:

Procedure

1. In both OPC servers, create identical items that track the status of each server.

If an OPC server is active, the OPC item must contain a positive value. If an OPC server is unable to serve data, the item value must be zero. Implement any logic required to ensure that both servers correctly detect and maintain the status of the other server and that, in both OPC servers, the values are identical.
2. Configure the OPC servers so that, during normal operation, one server sends data to the PI OPC interface and the other waits until the primary server fails.

Status for the primary server must be positive, and for the backup server, status can be zero. If failover occurs, the primary server status must be set to zero and the backup server status to a positive value.

3. In PI Server, create a watchdog PI point for each OPC server, mapped to the OPC items you created in step 1.
4. Using PI ICU, go to the **OPCInt > Failover > Server Level** page and set the **Primary Server Watchdog Tag** and **Backup Server Watchdog Tag** fields to the names of the watchdog PI points you created in the previous step.

In the interface batch startup file, these settings are specified by the `/WD1` and `/WD2` parameters.

Results

If both watchdog points are zero, data collection stops until one watchdog point becomes positive. If both watchdog points are positive, the OPC interface remains connected to the server that is currently serving data to it.

Controlling failover timing

When failover is triggered, the PI OPC interface must quickly recognize that the current OPC server is no longer available and determine whether the backup OPC server is available. To configure failover efficiency in the PI ICU, you can adjust the following settings on the **OPCInt > Failover > Server Level** page:

- **Switch to Backup Delay (/FT=#)**

Specifies in seconds how long the OPC interface tries to reconnect to the current server, before failing over to the other server and, if less than 30, how often the OPC interface checks the server state.

- **Number of Interfaces on this Node (/NI=#)**

Specifies the number of OPC interface instances running on this node. This value is used to stagger the startup of the OPC interfaces, to avoid the impact caused when multiple instances connect to the OPC server simultaneously. (To reduce the impact of restarting multiple instances, you can also use the Startup Delay setting.)

- **Wait for Running State (/SW=#)**

Specifies how many seconds the OPC interface waits for the server to enter the **RUNNING** state before failing over to the other server. By default, the OPC interface waits indefinitely for the server to enter the **RUNNING** state. Note that OPC servers vary significantly in the time required to enter the **RUNNING** state.

Configuring DCOM for the PI OPC DA interface

Classic OPC server and client applications are based on Microsoft's COM/DCOM communication model. COM (Component Object Model) provides a set of interfaces that enables software components to communicate on a single computer. DCOM (Distributed Component Object Model) extends COM to enable software components to communicate between network nodes. DCOM enables a process on one computer to request another computer to execute programs on its behalf, so permissions must be granted carefully to ensure security is maintained.

For detailed information, refer to the *DCOM Security and Configuration Guide*.

Topics in this section

- [DCOM security levels](#)
- [DCOM clients and servers](#)
- [Windows domains and users](#)
- [Determining the effective user](#)
- [Firewalls and security](#)

DCOM security levels

Access to a COM server is governed by Windows security and controlled by access control lists (ACLs), which grant specific users or groups permission to use that server. In addition, system-level policies and settings determine how users are authenticated and how permissions are granted.

DCOM security is implemented on several levels:

- System-level ACLs, settings and policies define the minimum level of security for all DCOM components (the **Edit Limits** ACLs in **dcomcnfg**).
- Default ACLs and security levels are used if a DCOM component does not explicitly set a security level (the **Edit Defaults** ACLs in **dcomcnfg**).
- Custom ACLs and security levels can be specified for individual DCOM servers using the Windows **dcomcnfg** utility.
- Custom security can be implemented in code by the DCOM server (**CoInitializeSecurity**).

DCOM clients and servers

OPC servers are DCOM servers, and OPC clients are DCOM clients, but the roles are not fixed. The PI OPC interface retrieves data from the OPC server using asynchronous callbacks. During a callback, the OPC interface acts as a DCOM server, while the OPC server acts as a DCOM client. For this reason, DCOM security on the OPC interface node must be configured to allow access by the account associated with the OPC server.

Windows domains and users

A Windows domain is a network of computers with a common security database. If the OPC interface node and the OPC server are members of the same domain (including domains with bi-directional trusts), domain users can be used in DCOM access control lists. If the interface and server reside in different domains, you must configure identical local accounts on both machines and use those accounts in ACLs. These accounts must have the same user name and password, and password expiration for these accounts must be disabled.

If the OPC interface and the OPC server are on the same computer, any account that is recognized by that computer can be used, including the SYSTEM built-in account. Some OPC servers implement custom security. In this case, the account must be granted access to the OPC server, apart from any DCOM permissions.

Computers in Windows workgroup configuration have default policies that interfere with OPC, requiring you to disable simple file sharing.

Determining the effective user

If the OPC server vendor has prescribed DCOM settings, do not change them unless it is not possible to communicate with the server using those settings. If you must change them, make a note of the original settings, in case they must be restored. It is generally best to adjust the client's settings to accommodate the server, rather than the other way around.

Access to DCOM servers is controlled by ACLs that specify user accounts and their associated permissions. When the client calls the server, the user account associated with the client process is authenticated, then the ACL is evaluated to determine if that account has permission to use the server.

For interactive clients such as the PI OPC Client or the OPC interface run interactively, the effective user is the account that was used to log on to the computer. An interactive process can be run under a different account by using the RunAs command.

For client programs running as Windows services, the user is the account specified in the **Logon** tab in the Services control panel.

For the OPC server, the user is the account specified in the **Identity** tab in the DCOM configuration for the server. Windows provides several options for the identity used by a DCOM server:

- **Interactive user**

The account that is logged on to the console of the computer where the server is running. This setting is problematic for OPC: if no one is logged on to the console or the user logged on does not have DCOM permissions, the client cannot connect to the OPC server.

- **Launching user**

The server process runs under the same account as the calling client. Do not use this setting if multiple clients running under different accounts need to access the same OPC server, because a new instance of the OPC server is launched for each user. Note that the calling client's user ID might not have permission to connect to the server, because many servers implement their own user authentication aside from DCOM permissions.

- **This user**

Recommended, unless the OPC server vendor specifies a different setting. Include the specified user in the default DCOM ACLs on the interface node. If the OPC server runs as a Windows service, use the same account as the logon account for the service.

- **System account (services only)**

Use only if the interface runs on the same computer as the OPC server.

Firewalls and security

DCOM relies on dynamically-assigned TCP ports. When an OPC client connects to an OPC server, it connects to port 135 (the RPC port mapper), which assigns one TCP port and one UDP port to the component. Communication between client and server is directed to those ports. Because of these limitations, it is difficult to configure DCOM to work through a conventional firewall .

Third-party vendors offer products that address these limitations. OPC tunnelers use a specialized OPC client that mirrors data to a specialized OPC server through an encrypted channel. OPC-aware firewalls enable secure communication with OPC servers with minimal configuration.

If third-party solutions are not desirable, secure OPC through configuration as follows:

- If the OPC server vendor supports it, install the PI OPC interface on the machine running the OPC server. The local COM connection permits you to disregard firewall issues between client and server.
- If a separate interface node is required, locate the interface on the OPC server's subnet. It is much easier to open a single firewall exception to port 5450 on the PI server than to configure DCOM to work through a conventional firewall.
- Configure DCOM permissions on a "least privilege" basis, by including only specific service accounts in DCOM ACLs.
- Use the built-in Windows firewall included in Windows XP SP2 and later versions of Windows.

OPC server issues

The OPC specification permits a great deal of flexibility in how OPC servers are designed and in what features they support. These variations in OPC servers can affect users of the PI OPC interface.

Topics in this section

- [Item browsing](#)
- [Time stamps](#)
- [False values](#)
- [Access path](#)
- [Problems with data returned by OPC server](#)
- [Troubleshooting OPC server operation](#)
- [OPC refreshes](#)

Item browsing

To be able to map PI points to OPC items, you must have access to OPC item names. However, OPC servers are not required to support item browsing. If browsing is supported, you can use the PI OPCClient to display the points that the OPC server recognizes.

Time stamps

Some OPC servers send the time stamp for the last time that the data value and quality were read from the device, which means that the time stamp changes even if the value does not. Others send the time stamp of the last change to value or quality, so if the data remains the same, the time stamp does not change. In this case, you must configure the time stamp setting using PI ICU. (*/TS*)

False values

Some OPC servers return a value when a client connects to a point, even if the server does not yet have a valid value for the point. Some servers send a false value with a status of **GOOD**, which results in the false value being sent to the PI archive. To screen out these false values, use PI ICU to enable the **Ignore First Value** option on the Data Handling page (*/IF=Y*).

Access path

In OPC items, the access path suggests how the server can access the data. The OPC standard states that it is valid for servers to require path information to access a value, but not for them to require that it be sent in the access path field. According to the standard, the OPC server can ignore it, but some non-compliant OPC servers require the access path. For example, RSLinx requires path information in the access path or as part of the ItemID, in the following format:

`[accesspath]itemid`

If your OPC server requires an access path, contact your OPC server vendor to determine how best to configure the server with the OPC interface.

Problems with data returned by OPC server

Unpack2 error messages in the local PI message log can indicate invalid data. In some cases, the OPC interface might be sending data to PI and also logging Unpack2 errors for the PI point, because the OPC server sends good values when it can and error codes when it cannot. Intermittent failures between the OPC server and the data source result in a combination of errors and values.

The following generic error messages are commonly logged:

In UnPack2 Tag MyPV.pv returns error 80020005: 80020005(Type mismatch)

In UnPack2 Tag MyPV2.pv returns error: The operation failed (80004005)

The following error messages indicate that the data received from the OPC server contained errors and the OPC server did not return a text explanation of the error:

In UnPack2 Tag MyPV3.pv returns error : Unknown error(800482d2)

In UnPack2 Tag MyPV4.pv returns error E004823E: Unknown error (e004823e).

In UnPack2 Tag MyPV5.pv returns error E241205C: Unknown error (e241205c)

In UnPack2 Tag MyPV6.pv returns error E2412029: Unknown error (e2412029)

To troubleshoot such data-related issues, consider the following causes and solutions:

- If you see Unknown errors, check with the OPC server vendor and have them look up the error code displayed in the error message. OPC servers can generate vendor-specific error codes, and only the OPC server vendor can explain what they mean.
- Restarting the OPC server might resolve the issue.
- **Type mismatch** errors indicate incompatible data types. Check for a mismatch between the PI Server data type and the OPC item type. Check **Location2** settings. To avoid cache issues after data types are changed, restart the OPC interface.
- Verify that the data type of the PI point can accommodate the range of values being sent by the OPC server. For example, if a PI point is defined as a two-byte integer and the OPC server sends values that are too large for it to accommodate, the point overflows.
- Make sure the data type of the OPC item and PI point are compatible.
- The data source might be sending corrupt data to the OPC server. Check for network issues that might corrupt the data packets.
- Check the size of the OPC server group. If the scan class contains more points than permitted in the OPC server group, Unpack2 errors might result. Consult the OPC server documentation for group size limits.
- If the point is digital, and the data can be read into a PI string point, and the underlying control system is Honeywell, the digital state strings in PI might need to exactly match the string reported by the DCS. To determine the digital states, go to Honeywell Universal Station or GUS to look at each controller block (data source).

Troubleshooting OPC server operation

The PI OPC interface can log OPC server interactions in the `opcresponse.log`, `opcscan.log`, and `opcrefresh.log` files. To interpret the information in these files, you must understand the basic OPC interface architecture. The OPC interface has two threads of operation:

- **PI thread**
Interacts with PI Server
- **COM thread**
Interacts with the OPC server

Polled PI points

For polled PI points, the OPC interface notifies the PI thread when it's time to scan. The PI thread starts the data collection process and logs the time, group number, and current flag value in `opcscan.log`, then sets the flag. (If the flag in `opcscan.log` is non-zero, the last call made to the server did not return before the OPC interface initiated another poll, and data might have been missed as a result.)

When the COM thread detects that the flag is set, it logs the time, group number and transaction ID in the `opcrefresh.log` file and makes a refresh call to the OPC server. When it receives the synchronous response from the OPC server, it clears the flag.

Now the OPC server can send data at any time, in an asynchronous manner. When the OPC server sends data to the OPC interface COM thread, the time, group number and transaction ID are logged in `opcresponse.log`.

Advise PI points

For advise PI points, the COM thread receives callbacks only when the data from OPC server changes value. Therefore, advise points do not generate entries in the `opcscan.log` or `opcrefresh.log` files, and only the data callbacks are logged in the `opcresponse.log` file. Advise points can be identified in the `opcresponse.log` file by group numbers that range from 200 to 800.

OPC refreshes

Logging refreshes

To log OPC refreshes, enable debug option 8, which causes the PI OPC interface to create three log files: `opcscan.log`, `opcrefresh.log`, and `opcresponse.log`. If the OPC interface is running as a service, the files are located in the `%windows%/system32` directory (`%windows%/syswow64` for 64-bit systems), otherwise the files reside in the directory where the OPC interface is running. When the OPC interface sets the flag for a scan, it logs the current time, the number of the scan class, and the current value of the scan flag in the `opcscan.log` file. The time stamp is in UTC (Greenwich time zone, daylight savings time is not observed), structured as a FILETIME structure written as an I64X field. The lower and upper halves of the number are transposed and the actual number is a count of the interval since January 1, 1601, measured in 10E-7 seconds.

After logging the data, the OPC interface sets the scan flag for the group, then the COM thread takes its turn. When the OPC interface cycles around to perform the poll, it logs the time, the scan class, and the TransID used in the `opcrefresh.log` file. For v1.0a server, the TransID logged is the TransID that was returned from the last poll of the group. For v2.0 servers, it is the actual TransID returned from the server.

When the OPC interface receives data from the OPC server, the OPC interface logs the time, the scan class, and the TransID received in the `opcresponse.log` file. For advise points, no entries are logged in the `opcrefresh.log` and `opcscan.log` files. Only the `opcresponse.log` file is updated.

Time stamps in OPC interface logs are stored in their native format, which is hard to read. To translate the time stamps to a readily readable format, use the following programs, which are installed into the `Tools` sub-directory below the OPC interface directory:

- `opcscan.exe`
- `opcrefresh.exe`
- `opcresponse.exe`

To run one of these programs from the command line, specify the input and output file names. Examples:

```
> opcscan.exe opcscan.log scan.log
> opcrefresh c:\pipc\Interfaces\OPCInt\opcrefresh.log c:\temp\refresh.log
> tools\opcresponse opcresponse.log response.log
```

The utilities display the UTC time stamp that came with the data, both raw and translated, the time stamp translated into local time, both raw and translated, and the PI time sent to the PI Server. For example:

```
response.log 126054824424850000 2000/06/14 18:54:02.485 126054680424850000
2000/06/14 14:54:02.485 960994309.485001 2 1db8
```

To check the time stamp returned from the OPC server, consult these log files. The time stamp is UTC time, which is based in 1600, so if you see a date around 1600, it indicates that the server is not sending valid time stamps. To configure the OPC interface to create time stamps when it gets the data, use PI ICU to enable the **Interface Provides Timestamp** option on the **OPCInt** tab (or edit the batch file and specify the `/TS=N` flag).

If the OPC interface is running with debugging options 32 or 64 enabled, the log file contains entries for individual data items that were received by the COM thread. For advise points, the group number in the `opcresponse.log` file might not be correct for entries generated by debugging options 32 or 64, although the shorter entries generated by debugging option 8 correspond to the correct group number.

By looking at the log files, you can see when the OPC interface decided to poll, when it made the call, and when the data came in. If the flag in `opcscan.log` is non-zero, the last call made to the server did not return by the time the OPC interface started another poll. If you find non-zero flags in the log file, contact your server vendor and have them contact OSIsoft.

No OPC server response to refresh calls

To determine whether the OPC server is responding to the refresh calls made by the OPC interface, check the local PI message log file for the following message:

The OPC server did not respond to the last refresh call for scanclass 2, and has not has not responded to the previous 100 refresh call(s).

This message indicates that the OPC server failed to respond to a refresh call. This problem occurs when the OPC server cannot keep up with the update rates or has suspended operation due to a bug. The message is repeated for each additional 100 refresh calls that receive responses from the OPC server for each scan class. If these messages appear in your local PI message log, data loss might be occurring. Contact your OPC server vendor immediately, and consider the following adjustments to reduce load on the OPC Server:

- Move points into the Advise scan class (#1).
- Reduce the total number of scan classes for the interface.

Features supported by the PI OPC DA interface

PI OPC DA interface part number: PI-IN-OS-OPC-NTI

Supported Operating Systems

Platform	32-bit application	64-bit application
Windows XP 32-bit OS	Yes	No
Windows XP 64-bit OS	Yes (emulation mode)	No
Windows 2003 Server 32-bit OS	Yes	No
Windows 2003 Server 64-bit OS	Yes (emulation mode)	No
Windows Vista 32-bit OS	Yes	No
Windows Vista 64-bit OS	Yes (emulation mode)	No
Windows 2008 32-bit OS	Yes	No
Windows 2008 R2 64-bit OS	Yes (emulation mode)	No
Windows 7 32-bit OS	Yes	No
Windows 7 64-bit OS	Yes (emulation mode)	No
Windows 8	Yes (emulation mode)	No
Server 2012	Yes (emulation mode)	No

No 64-bit builds of the PI OPC DA interface are available.

Feature	Support
OPC Data Access Standard	1.0a / 2.0 / 2.05
Auto creates PI points	APS Connector
Point Builder utility	No
ICU control	Yes
PI point types	Int16 Int32 Float16 Float32 Float64 Digital String Timestamp
Sub-second time stamps	Yes
Sub-second scan classes	Yes
Automatically incorporates PI point attribute changes	Yes
Exception reporting	Interface: PI exceptions OPC server: Deadband
Outputs from PI Server	Yes
Inputs to PI Server	Scan-based unsolicited event tags
Supports questionable bit	Yes
Supports multi-character PointSource	Yes
Maximum point count	Unlimited
Uses PI SDK	Yes
PINet string support	N/A

Features supported by the PI OPC DA interface

Feature	Support
Source of time stamps	Interface or OPC server (configurable)
History recovery	No
Disconnected startup	Yes
SetDeviceStatus	Yes
Failover options	OPC server-level failover and UniInt Phase 2 Interface-level failover(Phase 1 deprecated)
Vendor software required on PI interface node	No
Vendor software required on DCS system	Yes
Vendor hardware required	No
Additional PI software included with interface	Yes
Device point types	VT_I2 VT_I4 VT_R4 VT_R8 VT_BSTR VT_DATE
Serial-based interface	No

Installation checklist for the PI OPC DA interface

This section summarizes the steps for installing and creating a basic configuration of the PI OPC DA interface on an OPC interface node. Many of the settings prescribed below are defaults that can be changed according to the requirements of more complex configurations. For step-by-step instructions, see [Installing and configuring the PI OPC DA interface](#).

Each PI OPC interface instance requires its own startup batch file. The batch file contains commands and flags that configure the required and optional settings for the interface, and this guide describes the flags associated with those settings. The batch files reside in the PI OPC interface installation directory, which also includes a template batch file (`opcint.bat_new`) that you can use to create new instances of the OPC interface.

To ensure a correctly-formatted batch file, use the PI Interface Configuration Utility (ICU) graphical tool to configure and troubleshoot the OPC interface (as opposed to manually editing the startup batch file). This guide tells you how to configure the OPC interface using PI ICU, and also notes the command-line parameters associated with settings, to help you debug configuration issues by reading the generated batch file. For a complete list of flags, open a command window, navigate to the OPC interface installation directory, and issue the following command:

```
opcint -help
```

To display the contents of your OPC server, OSIsoft provides PI OPCClient, another graphical tool. To launch PI OPCClient, double-click the `OPCClient.exe` executable file or choose **Start menu > All Programs > PI System > PI OPCClient**. You can use PI OPCClient to connect to the OPC server and test data exchange procedures such as `sync read`, `refresh`, `advise`, and `outputs`.

Topics in this section

- [Installation prerequisites](#)
- [Install PI OPC DA interface on an interface node](#)

Installation prerequisites

Before installing and configuring, ensure that the following prerequisites are met:

- Verify that the PI Server is up and running.
- Using OPCClient, verify that the OPC server is up and running and populated with points.
- Verify that the PI OPC interface node time zone is set correctly.
- On the PI OPC interface node, install the following:
 - OSIsoft Prerequisites
 - PI Interface Configuration Tool (PI ICU)

Install PI OPC DA interface on an interface node

Procedure

1. On the OPC interface node, run the PI OPC interface setup program.
2. On the OPC interface node, test the API connection to the PI Server: In the %PIPC%\bin directory, issue the `apisnap PISERVERNODE` command.
3. On the OPC interface node, test the SDK connection to the PI Server: Choose **Start > All Programs > PI System > About PI-SDK** and use **File > Connections** to connect to the PI Server.
4. Create an OPC data owner (that is, a PI user with read/write access to the PI points that this interface will be using).
5. On the server node, use PI SMT to create API trusts that permit the following applications to access the server node:
 - PI ICU
 - Buffering process
 - OPC interface (application name: OPCpE)
6. On the OPC interface node, use PI ICU to create a new OPC interface instance from the OPC interface batch file (`OPCint.bat_new`).
7. Configure DCOM settings. For details, see the *OSIsoft DCOM Security and Configuration Guide*.
8. Define the following settings for the OPC interface in PI ICU:

- **General**

Setting	Value
Point source	OPC (or an unused point source of your choice)
Scan classes	As desired. (Scan class 1 is reserved for advise points.)
Interface ID	1 or any unused numeric ID

- **OPCIntOPC Server tab**

Click the **List Available Servers** button, then select your server from the list. If the server is on another machine, specify that machine or IP address in the **Server Node** field, then click to display the list.

9. Using PI OPCClient, verify that the OPC server is up and running and populated with points.
10. Start the interface interactively and check the log to verify that it starts successfully.
11. If you intend to use digital points, define the appropriate digital state sets.
12. Use PI ICU to configure the PI OPC interface to run as a service.
13. Stop the interface and configure and start buffering. For details about verifying that buffering is working, consult the OSIsoft Knowledge Base.

14. Restart the OPC interface node and confirm that the PI OPC interface service and the buffering application restart.
15. Build input points and, if desired, output points for this PI OPC interface. Verify that data appears in PI as expected. For detailed information about OPC points, refer to [Configuring PI points for the PI OPC DA interface](#).

Tag Attribute	Description
PointSource	Identifies all points that belong to this instance of the PI OPC interface. Specify the same Point source entered on the PI ICU General tab .
Location1	Specifies the OPC interface instance ID, which is displayed on the PI ICU General tab.
Location2	To enable handling for OPC servers that do not return certain numeric types in their native format, set Location2 to 1. For details, see Location2 (data-type handling)
Location3	Point type (0=poll, 1=advise, 2=output).
Location4	Specifies the scan class.
Location5	Optional deadband value for advise points.
ExDesc	Specifies event points, Long ItemID, Dzero for scaled points, or ItemID to get the time stamp for an output value.
InstrumentTag	OPC ItemID that corresponds to the PI point you are defining. Case-sensitive. To display OPC server points, use PI OPCClient.

16. Optional: The following procedures are useful but not required.
 - Configure diagnostics
 - Configure disconnected startup
 - Configure failover
 - Install the PI Interface Status Utility

PI ICU reference for the PI OPC DA interface

The PI Interface Configuration Utility (ICU) is a graphical user interface for configuring PI interfaces. PI ICU ensures that the configuration information stored in the PI OPCDA interface startup batch file and the PI Module Database is generated and updated correctly, eliminating the need to edit it manually. PI ICU requires PI 3.3 or greater. For more detailed information, refer to the *PI Interface Configuration Utility User Manual*. The next sections describe the fields and settings that you configure on the OPC interface pages.

Topics in this section

- [OPC Server settings](#)
- [Advanced Options settings](#)
- [Data Handling settings](#)
- [DCOM Security settings](#)
- [Failover settings](#)
- [Plug-In settings](#)
- [Miscellaneous settings](#)
- [Debug settings](#)

OPC Server settings

OPC Server Node Name

The name or IP address of the OPC server node (`/SERVER=node::name`). Leave blank or set to "localhost" (case insensitive) if the OPC interface and OPC server are on same node. If anything else is used for the Node Name, the interface will treat the OPC Server as remote. This can impact certain OPC Servers that may refuse "remote" connection attempts.

OPC Server Name

Registered name of the OPC server on the OPC server node.

Force v1.0a Protocol

By default, the OPC interface tries to connect to the OPC server using the v2.0 OPC Data Access specification (`/VN=2`). If the attempt fails, it uses the v1.0a protocol. To force the OPC interface to use only the v1.0a OPC Data Access specification, enable this option.

Timestamps

Interface Provides Timestamp: The OPC interface provides a time stamp when the data is received (`/TS=N`).

OPC server Provides Timestamp: The OPC interface uses the data time stamps provided by the OPC server, and accounts for the offset between the OPC server and the PI server (`/TS=Y`).

Timestamp for Advise Tags Only: The OPC server provides time stamps only for advise tags, and the OPC interface accounts for the offset between the OPC server and the PI server. For all other tags, the OPC interface provides a time stamp when the data is received (**/TS=A**).

OPC Server Provides Timestamp (no offset): The OPC Server provides the timestamps for all data, and the interface will not apply any time offset to these values. Data loss will occur if a value is received from OPC with timestamp 10 minutes or more past the PI Server's current time. (**/TS=U**)

Questionable Quality

Store Quality Only: If data has other than GOOD quality, store the quality information rather than the value (**/SQ=Y**).

Store Value Only: The OPC interface treats “questionable” quality as “good” (**/SQ=I**). Bad quality data is stored as a system digital state.

Advanced Options settings

Time delay before reading OPC Tags (sec)

Specify a delay (in seconds) before reading from or writing to the OPC server. If this parameter is configured, the OPC interface connects to an OPC server and waits the specified amount of time before attempting to read data. (**/SD=#**).

Event Tags Source

For v1.0a OPC servers, specifies whether event tags are read from the OPC server's cache or directly from the device. For v2.0 servers, it has no effect, because all event tag reads are from the device. (**/ES=CACHE** or **DEVICE**).

Advise Groups on Creation

Some OPC servers do not return an initial value when a PI advise tag is created. The resulting symptom is that, for a value that does not change often, the PI OPC interface does not write a value to PI when the OPC interface starts. To determine whether your OPC server has this problem, use PI OPCClient to create a group, add tags, and then advise the group. If a value is not immediately returned for your tags, but adding a tag to the scan class causes a value to be returned, enable this setting. (**/AF=Y**)

Disable Mass Tag Adding

To direct the OPC interface to add tags to an OPC group one at a time, enable this option. By default, mass adds are disabled (unless you configure the interface using PI ICU, which enables the option) and multiple tags are added to a group at once, and some OPC servers reject the entire group if one tag is invalid. Note that disabling can delay interface startup significantly. (**/MA=N**)

GlobalLocking Not Valid

If you see **OnDataChange: Invalid group ID** messages in the local PI message log file, enable this option. If the problem is resolved, the OPC server does not follow the OPC specifications. In this case, e-mail details (including OPC server vendor and version) to techsupport@osisoft.com. This flag is only meaningful for version 1.0a OPC DA. (**/GL=N**).

Ignore Group Status

If you see `OnDataChange: Header status:` in the local PI message log file, the group status sent by the server is invalid. To ignore the group status, enable this option. This flag is only meaningful for version 1.0a of OPC DA. (**/GS=N**)

Ignore Server Status

If the OPC server does not go to `OPC_STATUS_RUNNING` state when it is ready to send data, enable this option to direct the OPC interface to attempt to communicate with it anyway (**/IS=Y**).

Ignore OPC Server Access Rights

If you see “Invalid read/write mode requested” messages in the local PI message log file, enable this option. (**/AR=N**).


Use Honeywell Plantscape Failover Error Codes

Enable checking for error codes that are specific to the Honeywell Plantscape system for server-level failover. Configures the OPC interface to fail over if it receives an error code of `0xE00483FD` or `0xE00483FC` on any tag. This is obsolete because Honeywell stopped using these codes after only one release. (**/HWPS**)

Reconnect to Server Delay (sec)

Specifies how long to wait, in seconds, before attempting to reconnect to the OPC server if the connection is broken. (**/RD=#**).

Update Rates

Specifies the requested update rate, if different from the scan period. Select a scan class from the dropdown box, enter the desired rate in the box to the right of the scan class, and click .

The scan class, scan rate, and update rate appear in the box below the period. Only scan classes that have update rates are listed.

This option is useful when the server must have a recent value for the items but the OPC interface does not read it very often, for example, if the PI OPC interface polls for the value every 30 minutes, but the value itself must be no more than one minute old. This situation imposes more load on the OPC server than if the update rate and the scan period are the same, but it can reduce latency of values for items that need to be read less frequently. (**/UR=period**).

Data Handling settings

Staggered Group Activation

This option directs the OPC interface to inactivate all groups on startup and to stagger the activation of the groups based upon the offsets specified for the group’s scan period. This feature does not affect the operation of all OPC servers. It is intended to help level the workload by spreading out updates for groups that have the same scan period. (**/GA**)

Inactivate Groups on Startup

Inactivate all groups on startup. After the groups are built, they are activated. This option helps reduce the load on the OPC server during startup. (**/GI**)

Update Snapshot

If the current snapshot is a system digital state (such as I/O timeout, Shutdown, and so forth) and the OPC interface reads in a new value that is older than the snapshot, the OPC interface sends the new value one second after the snapshot time stamp of the system digital state. This check is omitted if the current snapshot is a good value. This is useful for setpoints that rarely change. (**/US**).

Ignore First Value

If the OPC server sends data before it reads its data source, it is likely to transmit zeros or erroneous values. This parameter directs the OPC interface to ignore the first value that it receives after startup for each tag. (**/IF=Y**)

Ignore Subsecond Timestamps

If the millisecond portion of the time stamp is not required, it can be truncated, which can speed up processing on the PI server (**/IT=Y**).

No Timeout

Direct the OPC interface never to write I/O timeout errors, even if the OPC interface loses its connection with the OPC server. Set this when configuring failover. (**/NT=Y**)

Disable Callbacks

Reduce the load on the OPC server by disabling call backs for polled groups. By default, polled groups have call backs enabled, but these call backs are not used by the PI OPC interface. This option has no effect on advise groups. (**/DC**)

Write Status to Tags on Shutdown

This parameter specifies the digital state to be written to all PI tags when the OPC interface is shut down (**/OPCSTOPSTAT=state**).

Alternate Digital State for Questionable/Bad Qualities

Assign alternate digital states for questionable and bad qualities. To use this option, create a contiguous set of digital states in the system digital state set that corresponds to the set of states listed in the manual, then assign the first digital state of the set to the command line option. (**/AS=system digital**). To view digital states using PI System Management Tools, go to **Points > Digital States**

Format of Timestamp Strings

Sets the format for time stamp strings read from or written to the OPC server. (**/TF=format**).

Number of Tags in Advise Group

Configure the maximum number of tags for each advise group created with scan class 1. The recommended maximum is 800 tags per group, which is the default. Adjust this number according to the OPC server requirements. (**/AM=#**)

Time Offset

If the OPC server node is set to a time zone other than the local time zone, this option directs the PI OPC interface to adjust all the time stamps by the specified amount. To specify the offset, use the format [-]HH:MM:SS. (**/TO=offset**)

Event Update Rate

Specify the requested update rate for the event class group. All event-based tags belong to the same group and the default update rate for the group is one second. If the OPC server's data cache for event-based tags does not need to be updated every second, you can reduce load on the OPC server by setting this parameter to a higher value (that is, a lower rate). For v2.0 servers, all events are read from the device, so this value can be set quite high unless there is some other reason to update the cache.

Trend Advise

For advise tags, send the value from the preceding scan if the new value's timestamp is greater than the number of scan periods (configured by the **/TA** flag). Enabling this setting causes advise tags to behave as if the **Step** attribute is enabled.

DCOM Security settings

For detailed information about configuring DCOM security, refer to the OSIsoft *DCOM Security and Configuration Guide*.

Default Authentication Level

Set the DCOM security authentication level (**/DA**) to one of the following:

- DEFAULT
- NONE
- CONNECT (default)
- CALL
- PKT
- PKT_INTEGRITY
- PKT_PRIVACY

Default Impersonation Level

Set the DCOM security Impersonation level (**/DI**) to one of the following:

- ANONYMOUS
- IDENTIFY (default)
- IMPERSONATE
- DELEGATE

Failover settings

UniInt-Interface Level Failover

The following three options are enabled only if warm failover is enabled on the **UniInt > Failover** page:

- **Warm_1:** Do not create groups on the server (/FM=1)
- **Warm_2:** Create inactive groups and add PI points (/FM=2)
- **Warm_3:** Create active groups; do not use advise groups (default)

- **Percent of tags accepted by OPC Server as valid**

Specify the percentage of points that are required to be accepted by the OPC server as valid. If less than this percentage is accepted, the PI OPC interface sets its device status to Connected/No Data, which triggers failover if UniInt failover is configured. (/RP).

- **Maximum number of Watchdog Tags which can have Bad Quality or Any Error without triggering Failover**

Specify the maximum number of watchdog PI points that can have an error or bad quality before failover is triggered failover. You can configure watchdog PI points to control failover when the OPC interface is unable to read some or all of the points, or when the points have bad quality. This feature enables you to trigger failover when a data source loses the connection to one OPC server, but is able to serve data to the other. To configure watchdog PI points, set **Location3**. For a watchdog point that is in an advise group, set **Location3** to 4. For a watchdog point that is in a polled group, set **Location3** to 3. (/UWQ).

Cluster Interface Failover

To make selections in this option, first enable it by selecting the **Enable Cluster Interface Failover** check box. Note this tab is only available when UniInt Failover is not selected.

Setting	Description
This node is the	Specify whether this node is Primary (/PR=1) or Backup (/PR=2).
Failover Mode	<p>Chilly: Do not create groups on the server (/FM=1).</p> <p>Cool: Create inactive groups , and add points (/FM=2).</p> <p>Warm: Create active groups, do not advise groups (default) (/FM=3).</p>
Cluster Mode	<p>Configure behavior of the backup PI OPC interface.</p> <p>Primary Bias: This node is the preferred primary. (/CM=0).</p> <p>No Bias: No node is preferred. The active PI OPC interface stays active until the cluster resource fails over, either as the result of a failure or through human intervention. (/CM=1)</p>

Setting	Description
Resource Number for APIONline	Identify the apionline instance that goes with this PI OPC interface instance. For example, to configure the OPC interface to depend on an instance named apionline2, set this field to 2. To configure the OPC interface to depend on an instance named apionline (no resource number), set this field to -1. (/RN=#)
Active Interface Node Tag	Specify the string point that contains the name of the currently active OPC interface node. (/CN).
Health Tag ID	This parameter is used to filter UniInt health points by Location3 . The parameter must be unique for each PI OPC interface – failover member parameter. If this parameter has an invalid value or is not set, the default value of 0 is used for the Location3 PI attribute when creating UniInt health points. (/UHT_ID)

Server Level Failover

Setting	Description
Backup OPC Server Node Name	The name or IP address of the backup OPC server node (/BACKUP).
Backup OPC Server Name	The registered name of the backup OPC server (/BACKUP).
Number of Interfaces on this Node	The number of instances of the OPC interface that are running on this node (/NI=#).
Switch to Backup Delay (sec)	The number of seconds to try to connect before switching to the backup server (/FT=#).
Wait for RUNNING State (sec)	The number of seconds to wait for RUNNING status before switching to the backup server (/SW=#).
Current Active Server Tag	(Optional) PI string point that contains the name of the currently active server. If set, the OPC interface writes the name of the OPC server to this point whenever it connects. Useful for debugging server-level failover. (/CS=tag).
Primary Server Watchdog Tag	Watchdog point for the primary server (/WD1=tag).
Backup Server Watchdog Tag	Watchdog point for the backup server (/WD2=tag).
Multiple Watchdog Tag Trigger Sum	The minimum total value of the watchdog points. Failover is triggered if the sum of the value of these points drops below the specified value. (/WD=#)
Maximum number of Watchdog Tags which can have Bad Quality or Any Error without triggering Failover	Default=0 if only one watchdog point. Cannot exceed the number of watchdog points defined. (/WQ=#)
Failover if Server Leaves RUNNING State	Triggers failover if the server state changes to anything other than RUNNING.(/WS=1)

Plug-In settings

- **Post Processing DLL**

Enter the DLL name and path to the post-processing DLL, for example, /DLL="Interfaces\OPCInt\plug-ins\mydll.dll"

- **Plug-In Configuration File**

Enter the name of the post-processing DLL configuration file. This text box is displayed only if the post-processing DLL requires a configuration file.

Miscellaneous settings



Caution:

Do not modify these settings unless directed to do so by OSIsoft Technical Support.

- **OPC Server Status Tag**

Specify a PI point to store the status of the OPC server when the status changes. (/ST)

Debug settings

To enable debugging options using PI ICU, go to the **UniInt > Debug** tab. In general, enable debug options for a short period of time, as they can bloat log files and reduce performance. For options marked "Technical Support only," enable only at the direction of OSIsoft Technical Support. For details about other command-line parameters, refer to the *UniInt Interface Users Manual*.

Option	Description	Value
Internal Testing Only	For OSIsoft internal testing only.	/DB=1
Log of Startup	Logs startup information for each PI point, including InstrumentTag and ExDesc	/DB=2
Log Write Op's and Acks for Tag	Logs OPC interface writes, ACKs from the OPC server, and writes queued to the "pending write" queue. Can be configured to log values sent for a specific point if the Debug Tag field is specified.	/DB=4
Log Timestamps of refresh	For use by OSIsoft Technical Support only.	/DB=8
Log Information for ExcMax	Log information about exception reporting	/DB=16
Log Timestamp and Data (All Tags)	For each data value that the OPC interface receives, logs the time stamp with the data, the adjusted time stamp, the PI time, the scan class, and transaction ID.	/DB=32

Option	Description	Value
Log Timestamp and Data for Tag	For use by OSIsoft Technical Support only.	/DB=64 /DT=tagname
Logging of Event Tags	Logs the name of each PI point into the local PI message log file as it receives data for the point.	/DB=256
Logging of Array Tags	Logs information about the array PI points	/DB=512
Logging of OPC List Pointers	For OSIsoft internal testing only.	/DB=1024
Log TS, Data and Quality for Tag	For the point that is specified in the Debug Tag field, logs time stamps, values, and qualities in the local PI message log. If there is no point specified, the first point for which a value is received is logged. This option is verbose and can bloat the log file.	/DB=4096
Log debugging info for /US command	Provide debugging information for the Update Snapshot (/US) option, if enabled.	/DB=8192
Log client, server, and group handles	Log the addresses used by the interface and server for each tag added to the interface. Enable only under direction of OSIsoft Technical Support.	/DB=16384

Enter any additional parameters that are not available through PI ICU, (for example, **/dbUniInt=0x0400**). Separate parameters with one or more spaces. If a parameter argument contains embedded spaces, enclose the argument in double quotes.

Command-line parameters for the PI OPC DA interface

There are two tables that list the command-line parameters used in the interface startup batch file to configure settings. One is organized alphabetically and the other is organized by functionality. These parameters are provided for debugging purposes only, to help you read the file. To ensure a correctly-formatted file, use the PI Interface Configuration Utility to configure the interface.

Alphabetical list of parameters

Alphabetical List

Parameter	Description
/AF=Y or N Default: N	(Optional) Configures data handing for advise points. Enable if advise points do not receive a current value on startup. Do not enable this setting if you are using Windows cluster failover, because it causes OPC groups to be advised as soon as they are created.
/AM=# Default: 800	(Optional) Specifies the number of points in each OPC advise group created for scan class 1. Intended for managing OPC server workload.
/AR=Y or N Default: Y	(Optional) Enable/disable use of access rights property on items added to a group. If the interface logs the error “Invalid read/write mode requested”; try disabling access rights by setting /AR=N
/AS=system_digital	(Optional) Assign alternate digital states for questionable and bad qualities. To use this option, it is necessary to create a digital state in the system digital state set that corresponds to the <i>system_digital</i> option of the command line option. Then, any digital states that follow the <i>system_digital</i> argument are used to map digital states to PI points when data with questionable or bad qualities are received from the OPC server, overriding the default digital states. For more information, see Data quality information .
/AT=# Default: 2000 msec (two seconds)	How long to wait for write acknowledgement from OPC server (milliseconds). When this time has elapsed, the interface cancels the write and resends it. Minimum is 200 msec.

Parameter	Description
<code>/BACKUP=hostname::OPCservername</code>	For server-level failover, specifies the name of the backup OPC server. If the OPC server is on the local machine, omit <i>hostname</i> . If your server name has embedded spaces, enclose the name in double quotes.
<code>/CACHEMODE</code>	Enable disconnected startup.
<code>/CACHEPATH=path</code>	(Optional) Specifies the directory where point caching files are created for disconnected startup. The directory must already exist on the target machine. By default, the files are created in the same location as the interface executable. If the path contains any spaces, enclose the path in quotes. Examples: <code>/CachePath=D:\PIPC\Interfaces\CacheFiles</code> <code>/CachePath="D:\Program Files\PIPC\MyFiles"</code>
<code>/CACHESYNC=#</code> Default: 250 ms	(Optional) Specifies the time slice period in milliseconds (ms) allocated for synchronizing the interface point cache file with the PI Server. By default, the interface synchronizes the point cache if running in the disconnected startup mode. To disable synchronization of the point cache file, specify <code>/CACHESYNC=0</code> . The minimum is 50ms and the maximum 3000ms (3s). Values less than the minimum or greater than the maximum are adjusted accordingly. This value must be less than the smallest scan class period. If the value is greater than the smallest scan class value, input scans are missed while the point cache file is being synchronized.
<code>/CM= 0 or 1</code> Default: 0	(Optional) Configure cluster-related behavior for interface-level failover running on a cluster. Mode 0 (<code>/CM=0</code>) is preferred-primary mode. Mode 1 (<code>/CM=1</code>) is a non-preferential mode, where whichever interface instance is active stays active until the cluster resource fails over.
<code>/CN=tag_name</code>	(Optional) For interface-level failover running on a cluster, this parameter specifies a PI string point that receives the node name of the interface instance that is currently gathering data. This feature enables you to track the cluster node that is the active interface node. Ensure that the point source of the specified point is not in use by any interface.
<code>/CS=tag_name</code>	(Optional) PI point that tracks the currently-active server, for failover. Ensure that the point source of the specified point is not in use by any interface.

Parameter	Description
<p><i>/DA=option</i></p> <p>Default: CONNECT</p>	<p>(Optional) Configures default authentication level, part of DCOM security settings for the interface. This parameter sets the interface-specific authentication level required to verify the identity of the OPC server during calls. Valid values are as follows:</p> <ul style="list-style-type: none"> • DEFAULT • NONE • CONNECT (default) • CALL • PKT • PKT_INTEGRITY • PKT_PRIVACY <p>Use this setting with the /DI parameter. If you set /DI and omit /DA, CONNECT is used. If neither /DA nor /DI is configured, the interface uses the default permissions on the client machine.</p>
<p><i>/DB=#</i></p>	<p>(Optional) Set level of debugging output to be logged. By default, debug logging is disabled. For valid settings, see Debug settings.</p>
<p><i>/DC</i></p>	<p>(Optional) Disable callbacks for polled groups, to reduce OPC server workload. No effect on advise groups. By default, callbacks are enabled.</p>
<p><i>/DF=tag_name</i></p>	<p>(Optional) Configure a PI point that contains the debug level, to enable you to change debug level while the interface is running. Configure an Int32 output point for the interface, and set its value to 0, then configure the point using the /DF parameter. After starting the interface, you can change debug level by setting the point to the desired level. For valid settings, see Debug settings.</p> <p>For InstrumentTag, you are required to enter a value, but the value is ignored and need not be a valid OPC ItemID.</p>
<p><i>/DI=level</i></p> <p>Default:</p>	<p>(Optional) Sets the interface-specific impersonation level to be granted to an OPC server to perform processing tasks on behalf of the interface. Default Impersonation level is one of the DCOM security settings for the interface. Valid authority levels are:</p> <ul style="list-style-type: none"> • ANONYMOUS • IDENTIFY (default) • IMPERSONATE • DELEGATE <p>Use with the /DA parameter. If you specify the /DA parameter and omit /DI, the default IDENTIFY is used. If neither parameter is set, the interface uses the computer's default DCOM security settings.</p>

Command-line parameters for the PI OPC DA interface

Parameter	Description
<i>/DLL=postproc.dll</i>	(Optional) Configure a post-processing DLL: /DLL=drive:\path\filename.dll The default path is the PlugIns sub-directory of the interface installation directory. You cannot configure more than one plug-in.
<i>/DLLCONFIG=config_file_name</i> or <i>/DLL_INI=config_file_name</i>	(Optional) Specifies the directory path and file name of the configuration file for a post-processing DLL. (Some post-processing DLLs do not require a configuration file.)
<i>/DT=tag_name</i>	(Optional) Specify the point for which detailed information is to be logged when using the verbose debug level. (/DB=64). If you omit this setting, the interface uses the first point for which it receives a value.
/EC=# Default: /EC=1	(Optional) Specify a counter number for an I/O rate point. If you require multiple interface instances with event counters, specify a different counter number for each instance. Must correspond to a PI point in the <i>iorates.dat</i> file. If you specify /EC and omit the argument, the default counter is 1.
<i>/ER=hh:mm:ss</i> Default: 00:00:01	(Optional) Specifies the requested update rate for the event scan class group. (All event-based points belong to the same group.) The default update rate for the group is one second. If the OPC server's data cache for event-based tags does not need to be updated that frequently, reduce workload by specifying a lower rate. For v2.0 OPC servers, all event reads are done from the device, so set this rate high (/ER=24:00:00) unless you require more frequent updates to the cache for other reasons.
<i>/ES=option</i> Default: CACHE	(Optional) For V1.0a OPC server, specifies whether event tag reads come from the cache (CACHE) or the device (DEVICE). Device reads can adversely affect the performance of the OPC server. For details, refer to Event points .
<i>/F=frequency[,offset]</i>	Define a scan class, specify how often the data in the class is scanned. Specify scan frequency and optional offset using the following format: HH:MM:SS.##,HH:MM:SS.## . For details, see Location4 (scan class) .

Parameter	Description
<p>/FM=#</p> <p>Default: 3</p>	<p>(Optional) Configure type of interface-level failover. Valid options are:</p> <ul style="list-style-type: none"> • 1: (Chilly) Do not create groups on the server • 2: (Cool) Create inactive groups and add tags • 3: Warm Create active groups , but do not advise groups <p>For details, see Configuring failover for the PI OPC DA interface.</p>
<p>/FT=#</p> <p>Default: 60</p>	<p>(Optional) Specifies (in seconds) how long the interface tries to connect to the current server before failing over to the server specified by the /BACKUP parameter. If the specified value is less than 30, also sets how often the interface checks server status. By default (and, at a minimum), the interface checks server status every 30 seconds.</p>
<p>/GA</p>	<p>(Optional) Staggers group activation to reduce OPC server workload. Use in conjunction with scan class offsets.</p>
<p>/GI</p>	<p>(Optional) To reduce load on the OPC server during startup, inactivates groups until they are built.</p>
<p>/GL=Y or N</p> <p>Default: Y</p>	<p>(Optional) Fix for some early v1.0a servers. If the log contains the error message OnDataChange: Invalid group ID, try setting /GL to N.</p>
<p>/GS=Y or N</p>	<p>(Optional) Fix for older, non-compliant OPC servers that do not provide a valid GroupStatus on asynchronous reads. If the log file contains a OnDataChange: Header status error, try setting /GS=N to tell the interface to ignore the group status parameters.</p>
<p>/HOST=host:5450</p>	<p>(Required) Specifies the host and port of the PI Server to which the interface sends data. Host is the node name of the host node.</p>
<p>/HS=Y or N</p>	<p>(Obsolete) Request a cache update rate of one half of the scan rate for the scan class. Use /UR instead.</p>
<p>/HWPS</p>	<p>(Optional) Check for Plantscape-specific Item error codes 0xE00483FD or 0xE00483FC and, if found, failover to alternate OPC server.</p>
<p>/ID=#</p>	<p>(Optional) Specifies the ID of the interface instance. Maximum nine digits. Optional, but highly recommended.</p>

Command-line parameters for the PI OPC DA interface

Parameter	Description
/IF=Y or N Default: N	(Optional) Ignore the first value sent for each point. For use with OPC servers that send a response when the interface connects to a point, regardless of whether they have a valid value.
/IS=Y or N Default: N	(Optional) Ignore the status returned by the OPC server. Some OPC servers do not return OPC_STATUS_RUNNING when ready. If the OPC interface hangs on startup and PI OPCClient displays an OPC server status other than RUNNING, set /IS=Y and report the issue to your OPC server vendor.
/IT=Y or N Default: N	(Optional) To truncate the sub-second portion of the time stamps being passed to PI and only send whole integer seconds, set to Y. Reduces CPU and disk consumption.
/MA=Y or N Default: N	(Optional) By default, the OPC interface adds items to groups one at a time, because some OPC servers reject an entire group if one item is invalid. To add all the items in a class at the same time, set to Y. Recommended for efficiency if supported by your OPC server.
/MAXSTOPTIME=# Default: 120 seconds	(Optional) Specifies how many seconds are allocated for the interface to close its connections and exit cleanly.
/NI=#	(Optional) Specifies the number of instances of the interface running on the node. Used in conjunction with /FT parameter to determine how long to wait before initiating server-level failover.
/NT=Y or N Default: N	(Optional) Write/do not write I/O Timeout to PI tags when the connection to the OPC server is lost. Set to Y to disable writing.
/OC=# Default: 1	Maximum number of outstanding outputs per group. After issuing the specified number of writes, the interface waits for one or more to be acknowledged before issuing any additional writes.
/OD=#	(Optional) Level at which to start dropping outputs. When the output queue for a group contains the specified number of outputs, the interface drops the oldest or newest output. To drop the newest output, specify a positive value. To drop the oldest output, specify a negative value.

Parameter	Description
/OG=# Default: 1	Number of output groups. Each group has its own queue.
/OPCSTOPSTAT=system_digital_state Default: "Intf Shut"	(Optional) To indicate that data collection was stopped when the interface is shut down, writes digital state to each input tag. If digital state is omitted, "I/O Timeout" is written. If digital state is specified, it must be a valid state from the System State Set. <div style="background-color: #e0f0ff; padding: 5px;"> <p>! Caution: Do not use the UniInt /STOPSTAT parameter with the OPC interface. /STOPSTAT can cause invalid values to be stored in PI points.</p> </div>
/OT=# Default: 36	Maximum number of point values to write at one time.
/OUTPUTSNAPTIME	(Optional). For output tags, use the timestamp from the original event that triggered the output. By default, the interface uses the time it receives a new value as the timestamp for storing the output after it has completed.
/OUTPUTACKTIME	(Optional) For output tag timestamps, use "acknowledge" time rather than the timestamp from the event that triggered the output. Uses the timestamp from the OPC server's acknowledgement of the write. Overrides /OUTPUTSNAPTIME, if enabled.
/OW=#	(Optional) Number of pending outputs at which to set Device Status to warn that outputs are arriving faster than the OPC server can process them. If /OW and /OD are both specified, /OW must be less than /OD .
/PISDK=#	(Optional) Enable (1) or disable (0) the PI SDK. If required by an OSIsoft interface, the PI SDK is enabled and cannot be disabled using this setting.
/PISDKCONTIMEOUT=# Default: 15	(Optional) Set the number of seconds to wait before timing out on PI SDK calls.
/PR=# Default: 0	Configure cluster failover for the interface instance: <ul style="list-style-type: none"> • 0: No cluster failover (default) • 1: Primary interface instance • 2: Backup interface instance

Command-line parameters for the PI OPC DA interface

Parameter	Description
<i>/PS=point_source</i>	(Required) Specifies the point source for the interface instance. Not case sensitive. The interface instance uses this setting to determine which PI points to load and update.
<i>/PW=password</i>	(Optional) The password for the user ID specified with /SEC . Case sensitive.
<i>/RD=#</i>	(Optional) How many seconds to wait before trying to reconnect to the OPC server.
<i>/RN=#</i>	(Optional) Specifies the resource number of the API service that the interface depends on. For example, /RN=1 configures the interface to depend on <code>apionline1</code> . Required if there are multiple instances of the OPC interface running with different service names on the same machine. For configuring cluster failover.
<i>/RP=#</i> Default: 80	Specifies the minimum percentage of points required to be accepted by the OPC server as valid. If less than the specified percentage is accepted, the OPC interface sets its device status to <code>Connected/No Data</code> , which triggers <code>UniInt</code> failover if configured.
<i>/RT=#</i> Default: 10	Polled scan classes only: The maximum number of scans that can return no data before the group is assumed to be nonresponsive and the interface sets the <code>DeviceStatus</code> of the interface to warn of the problem. Minimum is 2 scan periods.
<i>/SD=#</i> Default: 0	(Optional) Specifies how many seconds to wait after connecting before reading OPC points. By default, there is no delay after connecting.
<i>/SEC</i> or <i>/SEC=userid</i>	(Optional) To enable the NT security option of the OPC standard, specify /SEC (omit user ID). If you are using the OPC private security option, specify the user ID using this parameter and the password using the /PW parameter. Requires an OPC server that supports OPC security.
<i>/SERVER=host::name</i>	(Required) Configures the target OPC server. If the OPC server runs on the same machine as the interface, omit host name and colon. If your server name has embedded spaces, enclose the name in double quotes.

Parameter	Description
/SG[= S]	<p>(Optional) Send only GOOD quality data. Questionable quality data and BAD quality data are ignored. To ignore substatus for values that have GOOD status, specify /SG=S.</p> <p>To treat OPC_QUALITY_LOCAL_OVERRIDE as SUBSTITUTED, specify /SG. To treat OPC_QUALITY_LOCAL_OVERRIDE as GOOD, specify /SG=S.</p> <p>If the /SQ=I or /SQ=Y parameter is also set, questionable quality data is sent to PI. BAD quality data is ignored. Quality information continues to be sent to points that are configured to store quality instead of values.</p>
/SIN=node	(Obsolete) Specifies the name of the secondary interface's node for cluster failover.
/SQ=Y or I Default: N	(Optional) By default, the OPC interface stores and flags uncertain-quality values. To store quality data instead of value for data that is not GOOD, specify /SQ=Y . To store questionable data, specify /SQ=I . For BAD quality data, the interface sends a digital state code to the archive.
/ST=tag_name	<p>(Optional) Configure a PI digital point to store the status of the OPC server when it changes. Ensure there is a digital state set with the following states:</p> <ol style="list-style-type: none"> 1. OPC_STATUS_RUNNING 2. OPC_STATUS_FAILED 3. OPC_STATUS_NOCONFIG 4. OPC_STATUS_SUSPENDED 5. OPC_STATUS_TEST <p>If the server returns anything other than one of the preceding states, 0 is stored. Configure a zero state for this state set that reflects the non-standard server status.</p>
/STARTUP_DELAY=# Default: 30	(Optional) Configures a post-startup delay. The OPC interface waits for the specified period before beginning operation. Intended for use if you have configured the OPC interface for autostart and the network layer needs time to complete startup before it becomes available. If you specify /STARTUP_DELAY and omit the delay, a thirty-second delay is configured.
/SW=#	(Optional) Specifies how long (in seconds) the interface waits for the OPC server to enter the OPC_STATUS_RUNNING state. If the specified period elapses and the OPC server is not running, the interface fails over to the alternate OPC server.

Parameter	Description
/TA= #.#	(Optional) For advise tags, send the value from the preceding scan if the new value's timestamp is greater than the specified number of scan periods. Enabling this setting causes advise tags to behave as if the Step attribute is enabled.
/TF=<i>format</i>	(Optional) Specifies the format of time stamp strings. Used for points with Location2 = 6 or 7, where the ItemID is either a string that contains a time stamp or a VT_DATE value. Also used for writing output time stamps using /TIM= in the ExDesc field. Valid tokens are: cc yy mn mon dd hh hr mm ss 000 XM For details, see Time stamps .
/TO=HH:MM:SS	(Optional) Applies an offset to all time stamps coming from the server. Provided to deal with servers and installations that do not follow the OPC specifications (for example, where the time zone must be UTC regardless of the location of the server). The format is the same as scan period parameters (/F). For negative offsets, precede the format with a minus sign.
/TS=Y or N or A or U Default: N	(Optional) Specifies whether time stamps come from the OPC server or are applied by the interface when the data arrives. By default, the interface provides time stamps (/TS=N). If the OPC server can provide valid time stamps, specify /TS=Y . If the OPC server can provide valid time stamps only for advised points, specify /TS=A . To only use timestamps provided by the OPC Server and apply no time offset, specify /TS=U . For details, see Time stamps .
/UFO_ID=#	(Required for UniInt interface level failover phase 1 or 2) Specifies failover ID. Must be a unique, positive integer.
/UFO_INTERVAL=# Default for Phase 1 failover: 1000 Default for Phase 2 failover: 5000	(Optional) Specifies in milliseconds how often the failover heartbeat points are updated and interface status is checked. Must be the same on both interface nodes. Minimum: 50 Maximum: 600000 milliseconds (10 minutes)
/UFO_OTHERID=#	Failover ID of the other OPC interface instance. Required for phase 1 or 2 interface level failover.

Parameter	Description
<code>/UFO_SYNC=path[/file_name]</code>	(Required for phase 2 interface level failover) Path and, optionally, name of the shared file containing the failover data. The <i>path</i> can be a fully qualified node name and directory, a mapped drive letter, or a local path if the shared file is on an interface node. The <i>path</i> must be terminated by a slash or backslash character. The default filename is: <i>executable_name_pointsource_interfaceID.dat</i> If there are any spaces in the <i>path</i> or <i>filename</i> , the entire path and filename must be enclosed in quotes. If you enclose the path in double quotes, the final backslash must be a double backslash (\ \).
<code>/UFO_TYPE=type</code>	(Required for phase 2 interface level failover) Specifies the type of failover configuration: HOT, WARM, or COLD.
<code>/UHT_ID=#</code>	(Optional) Specifies a unique ID for interface instances that are run in a redundant mode without using the UniInt failover mechanism. If the ID is specified, only health points with the specified value in Location3 are loaded.
<code>/UR=HH:MM:SS.000</code>	(Optional) Specifies the requested update rate for the group. By default, the update rate requested for a scan class is the same as the rate. The update rate is applied to the scan period that it follows. For example: <ul style="list-style-type: none"> <code>/f=00:00:02</code> - Update rate two seconds <code>/f=00:00:03 /ur=00:00:00.5</code> - Update rate 0.5 seconds <code>/f=00:00:01</code> - Update rate one second
<code>/US</code>	(Optional) If the current snapshot is a system digital state and the new value is older than the snapshot, the interface sends the new value to the PI Server one second after the snapshot time stamp of the system digital state. This check is not done if the current snapshot is a good value.
<code>/UWQ=#</code>	(Optional) Directs the interface instance to fail over if the specified number of watchdog points do not have GOOD quality and, for v2.0 OPC servers, if there is an error reading watchdog points.
<code>/VN=1 or 2</code> Default: 2	(Optional) Specifies the OPC server version (v1.0a or V2.0).

Parameter	Description
<code>/WD=#</code>	(Optional) For configuring failover using multiple watchdog points, trigger failover if the sum of the values of the points drops below the specified value.
<code>/WD1</code> and <code>/WD2</code>	(Optional) Configure watchdog points for failover. For details, see Configuring failover for the PI OPC DA interface .
<code>/WQ=#</code>	(Optional) For configuring failover using multiple watchdog points. Directs the interface to fail over if the number of watchdog points with quality other than GOOD exceeds the specified value (and, for v2.0 servers, if there is an error reading the item).
<code>/WS=1</code> or <code>0</code> Default: <code>0</code>	(Optional) For failover, set to <code>1</code> to configure the interface instance to disconnect from the server if the server leaves the <code>OPC_STATUS_RUNNING</code> state. By default, the interface stays connected.

Parameters by function

The parameters are grouped by how they are used. They are specific to the PI OPC interface, except for the UniInt parameters that are common to all OSIsoft UniInt-based interfaces.

Common Unint Parameters	Advanced Parameters	Data Handling
/CACHEMODE	/AM	/AF
/CACHESYNC	/AR	/AS
/EC	/AT	/ER
/F	/DC	/HOST
/ID	/ES	/IF
/MAXSTOPTIME	/GA	/IT
/PISDK	/GI	/NT
/PISDKCONTIMEOUT	/GL	/OPCSTOPSTAT
/PS	/GS	/SG
/STARTUP_DELAY	/HWPS	/SQ
	/IS	/TA
	/MA	/TF
	/OC	/TO
	/OD	/UR
	/OG	/US
	/OT	
	/OUTPUTACKTIME	
	/OUTPUTSNAPTIME	
	/OW	
	/RD	
	/SD	

DCOM Security	Plug-ins (Post-processing DLLs)	Debugging
/DA	/DLL	/DB
/DI	/DLLCONFIG	/DF
		/DT

Command-line parameters for the PI OPC DA interface

Server-level Failover	Unitnt Interface-level Failover	Interface-level Failover
/BACKUP	/UFO_ID	/CM
/CS	/UFO_INTERVAL	/CN
/FT	/UFO_OTHERID	/FM
/NI	/UFO_SYNC	/PR
/SW	/UFO_TYPE	/RN
/WS	/UWQ	/UHT_ID
/WD	/RP	
/WD1	/RT	
/WD2		
/WQ		
/WS		

OPC Server	Miscellaneous
/Server	/PW
/TS	/SEC
/VN	/ST

Error and informational messages for the PI OPC DA interface

The location of the message log depends upon the platform on which the interface is running. See the *UniInt Interface User Manual* for more information. Messages are logged as follows:

- During startup: Messages include the version of the interface, the version of UniInt, the command line parameters used, and the number of points.
- During point retrieval: Messages are sent to the log if there are problems with the configuration of the points.
- If debugging is enabled.

Messages

The log contains messages from the OPC interface, the UniInt framework, and the PI API. This list describes only messages from the interface. If any error message has a PI point number as well as a point name, use the point number to identify the problem point, because long point names are truncated to 12 characters.

Informational

Message	No ConnectionPoint for OPCShutdown Shutdown Advise Failed
Meaning	The OPC server does not implement the Shutdown interface or does not implement it properly. Does not prevent proper operation of the interface.

Message	QueryInterface:IID_IconnectionPointContainer failed, using v1.0a protocol
Meaning	The OPC server does not support OPC DA v2.0.

Message	GetStatus: Server has no current time.
Meaning	Indicates a non-OPC-standard server that does not send the time of day. The OPC specifications state that the server is supposed to include current time when it sends its status. The interface guesses time stamps, but its accuracy is likely to be questionable.

Message	Cleaning up connections Cleaned up connections
Meaning	Indicates that the interface is disconnecting and exiting.

Message	Server sent shutdown notice.
---------	------------------------------

Error and informational messages for the PI OPC DA interface

Meaning	Interface received a shutdown notification from the OPC server. The interface periodically tries to reconnect to the server , until it is shut down or succeeds in connecting.
Message	Got %d and cleared it ClearWrite: dwTransID mismatch, have %d, got %d Stashing transid %d Sending transid %d” Writing transid %d
Meaning	Debug level 4 messages indicating that the server acknowledged the specified write and the interface is clear to send another write value.
Message	Can't find status tag, ignoring Can't find queue tag, ignoring Status tag is not Digital tag, ignoring Queue tag is not Integer tag, ignoring
Meaning	Status/queue point does not exist or its data type is incorrect.
Message	Can't connect to OPC Server, going into slow cycle wait
Meaning	Interface attempted to connect to OPC server. Check for other messages that contain details about the reason for the failed attempt. Interface periodically tries to reconnect.
Message	AddItems failed, server not in RUNNING state, will try later
Meaning	The interface is waiting for the OPC server to enter RUNNING mode. You can use the PI OPCClient to see the state of the server (use the Get Status button). If the server does not enter the RUNNING mode, investigate the cause.

Errors

Message	Out of Memory. Cannot allocate a list; fails. Unable to add tag.
Cause	The system has run out of resources.

Resolution	Use the Windows Task Manager to check the resources being used: press the Control, Shift, and Escape keys all together to get to the Task Manager. If you see high memory usage for <code>opcint.exe</code> , there might be a bottleneck between the interface and your PI system. Look for related messages in the log (see also <code>Running low on memory, dropping data</code>).
------------	---

Message	CLSIDFromProgID
Cause	PI Server's Registry entries are not valid.
Resolution	Check your server installation instructions.

Message	CoCreateInstanceEx
Cause	Indicates a problem with your DCOM configuration.
Resolution	Check your DCOM settings.

Message	IOPCServer
Cause	The proxy stub files are not registered.
Resolution	To register the <code>opcproxy.dll</code> and <code>opccomm_ps.dll</code> files, open a command prompt window, change to the directory where the interface is installed, and issue the following commands: <pre>>regsvr32 opcproxy.dll >regsvr32 opccomm_ps.dll</pre>

Message	AddRef
Cause	Indicates that the OPC server does not let the interface perform the simplest function.
Resolution	If you can read and write points using PI OPCClient but this error is logged, check your DCOM settings, check what user account the interface is running under, and try running the interface interactively.

Message	Advise returns error: 80040202(Unable to open the access token of the current thread)
Cause	If you see this error after connecting to the server successfully, the wrong <code>opcproxy.dll</code> might be loaded.

Error and informational messages for the PI OPC DA interface

Resolution	<p>If you have multiple copies of <code>opcproxy.dll</code> on your PI Interface node (possibly because you have more than one OPC server on your machine), make sure that they are the same version. It is safest to have only one version around on your system (in the <code>\%windows%\system32</code> or <code>\%windows%\syswow64</code> directory). Check whether directories containing older versions are specified in the system path.</p>
Message	<p>AddGroup failed for scanclass %d AddItem failed for %s AddItems failed for tag %s Advise Group failed for %s Advise returns E_OUTOFMEMORY Advise returns E_UNEXPECTED Advise returns error No ConnectionPoint for scanclass %d QueryInterface:IID_IdataObject failed for scanclass %d QueryInterface:IID_IOPCAsyncIO2 failed for scanclass %d Read: (string) Refresh: (string) Unable to add to group Unable to add to OPC group. Unable to advise event group Unable to advise group Unable to advise output group Unable to create group Write error %X for tag Write failed</p>
Cause	<p>These are fatal errors returned by the OPC server.</p>

Resolution	<p>Try the same operation using PI OPCClient. If successful, run the interface interactively to see if the same error occurs. If successful, check your DCOM configuration to make sure that you have granted required permissions to the INTERACTIVE account. c0040004: Indicates that the requested data type cannot be returned for this item. Use PI OPCClient to add that Item to a group, omitting data type. The server will send you values using the data type that it uses internally for that item. c0040007: Returned from AddItem, indicates that the server does not have the specified item. Verify that the InstrumentTag field of the PI point matches the OPC item name exactly. Using PI OPCClient, try to add the Item to a group, or if your OPC server supports browsing, browse to the item and double-click on it to display its full name.</p>
Message	Invalid read/write mode requested for tag %s
Cause	The server is returning invalid information about read/write access.
Resolution	To tell the interface to ignore this information, specify the /AR=N parameter.
Message	<p>RemoveItem failed for tag %s dev_remove_tag: Unable to unadvise %s dev_remove_tag: Unable to remove group %s</p>
Cause	The server won't remove a point from a group or stop collecting data for a group.
Resolution	Not a serious problem unless accompanied by lots of other messages, but indicates some problem with the OPC server.
Message	<p>Write unable to get values: Getsnapshotx error %d</p>
Cause	The interface was unable to read a value from PI to write to the OPC server.
Resolution	To verify that the PI Server is running, use apisnap (in the API directory). Verify that the source and target data types and values are compatible.

Error and informational messages for the PI OPC DA interface

Message	<p>Event Point has invalid scan class (!= 0)</p> <p>No Item name - instrumenttag and exdesc both empty</p> <p>Nonzero Totalcode requires nonzero Convers</p> <p>Point cannot be write and Read On Change</p> <p>Point has invalid scan class</p> <p>Point has invalid scan class (= = 0)</p> <p>ROC Point has invalid scan class (= = 0)</p> <p>Square root must be 0, 1 or 2</p> <p>This Totalcode requires Dzero to be specified.</p> <p>Total must be 0,1,2,3,4, or 5</p> <p>Unable to get point type</p> <p>Unable to get source point type.</p> <p>Unable to get square root</p> <p>Unable to get total specs</p>
Cause	Indicates that a PI point is improperly configured.
Resolution	Check the point configuration, especially the attribute indicated.

Message	GetStatus
Cause	The OPC server did not respond to a status query. It might be down or disconnected.
Resolution	Use PI OPCClient to check the status.

Message	Interface failed to write some %s states
Cause	When the OPC server shuts down, the interface sends a shutdown status to each point, if configured to do so (/OPCSTOPSTAT). This error message indicates that the interface was unable to send some or all of them, owing to lack of connectivity to the PI Server or lack of buffering.
Resolution	You may want to manually enter the digital state for the affected points to indicate that the interface shutdown.

Message	<p>OnDataChange:Invalid group ID < 0</p> <p>OnDataChange:Invalid advise group ID:</p> <p>OnDataChange:Invalid group ID > 999</p> <p>OnDataChange: Header status:</p> <p>OnDataChange has format not Hglobal</p> <p>OnDataChange:Invalid group ID for write completion</p> <p>Unknown access type for group %s</p>
Cause	Indicates that the server is sending meaningless data.
Resolution	To ignore the header status field in incoming data, set /GS=N. Contact your OPC server vendor. Use PI OPCClient to create groups and try AsyncReads and Advises. Check whether incoming data is valid: the meaningless data might not interfere with data collection.

Message	OnDataChange: Bad Timestamp
Cause	The interface received an invalid time stamp with data from the OPC server.
Resolution	Check your OPC server, and use PI OPCClient to display time stamps.

Message	Invalid timestamp for tag: %s, %d and %.36f
Cause	The interface received an invalid time stamp with data from the OPC server. Indicates an issue in the OPC server.
Resolution	Use PI OPCClient to display the item in question. Use Refresh or Advise or AsyncRead to see a time stamp.

Message	<p>Putsnap system error %d, %d</p> <p>Putsnap no longer in system error %d, %d</p>
Cause	System error indicating a problem sending data to PI Server.
Resolution	No action required unless the error persists.

Message	<p>Putsnapsx not implemented %d</p> <p>Getsnapshotx not implemented</p>
Cause	Indicates an outdated version of the PI API.
Resolution	Update your version of the PI API.

Message	Unable to translate string.
---------	-----------------------------

Error and informational messages for the PI OPC DA interface

Cause	Attempt to translate an ASCII string value from a PI point to Unicode failed.
Resolution	Check the value of the point.

Message	Unable to initialize server object
Cause	Privileges are not granting access.
Resolution	Verify that account has sufficient privileges.

Message	No OPC Server specified
Cause	The parameter /SERVER specifies a non-existent server or the invocation is on multiple lines.
Resolution	Verify that the OPC interface batch startup file specifies valid /SERVER parameter and that the invocation is a single line.

Message	Unable to create clock drift timer
Cause	Interface cannot create a timer to track drift.
Resolution	Check system resources.

Message	Running low on memory, dropping data Memory load within acceptable limits, resuming data collection
Cause	If the PI Server cannot accept data as quickly as the interface can send it, the interface buffers data in memory. To avoid consuming all available memory, the interface starts dropping data as the limits configured by /HQ and /LQ are approached.
Resolution	Consider giving the PI system more resources, changing the exception parameters of your points, or changing the scan period of your points.

Message	Failed to open cluster: error ####. Intf-failover will not be supported. Failed to open cluster resource: error ####. Intf-failover will not be supported.
Cause	Win32 error code indicating that an attempt to open the cluster service or resource failed.
Resolution	Check the cluster settings.

Critical errors

Message	Error from CoInitialize: Error from CoInitializeSecurity:
Cause	COM might not be properly installed on your system.

Resolution	If true, this is a serious problem. First check your COM setup. You may also need to contact OSIsoft Technical Support.
Message	Cannot get PI Server time.
Cause	Cannot connect to PI Server.
Resolution	For newly-installed systems, reboot and verify that you can connect to the PI Server. To verify connectivity, ping the PI server machine. To verify that the PI Server is running, use <code>apisnap</code> . For existing installations, contact OSIsoft Technical Support.
Message	OnDataChange: VariantCopy
Cause	Indicates that the OPC server sent meaningless data. Interface rejects the data and writes BADSTAT to the point. Time stamp is good.
Resolution	Check the data with PI OPCClient.

System errors and PI System errors

System error numbers are positive. PI error numbers are negative. To display descriptions of system and PI errors, use the `pidiag` utility:

```
\PI\adm\pidiag -e error_number
```

Unint failover-specific messages

Informational

Message	16-May-06 10:38:00 OPCInt 1> UniInt failover: Interface in the "Backup" state.
Meaning	Upon system startup, the initial transition is made to the backup state. In this state the interface monitors the status of the other interface participating in failover. When configured for hot failover, data received from the data source is queued and not sent to the PI Server while in this state. The amount of data queued while in this state is determined by the failover update interval. In any case, there will be typically no more than two update intervals of data in the queue at any given time. Some transition chains can cause the queue to hold up to five failover update intervals worth of data.
Message	16-May-06 10:38:05 OPCInt 1> UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface not available.
Meaning	In primary state, the interface sends data to the PI Server as it is received. This message also indicates that there is not a backup interface participating in failover.

Error and informational messages for the PI OPC DA interface

Message	16-May-06 16:37:21 OPCInt 1> UniInt failover: Interface in the “Primary” state and actively sending data to PI. Backup interface available.
Meaning	While in this state, the interface sends data to the PI Server as it is received. This message also states that the other copy of the interface appears to be ready to take over the role of primary.

Error messages (Phase 1 & 2)

Message	16-May-06 17:29:06 OPCInt 1> One of the required Failover Synchronization points was not loaded. Error = 0: The Active ID synchronization point was not loaded. The input PI tag was not loaded.
Cause	The Active ID tag is not configured properly.
Resolution	Check validity of point attributes. For example, make sure Location1 attribute is valid for the interface. All failover tags must have the same PointSource and Location1 attributes. Modify point attributes as necessary and restart the interface.

Message	16-May-06 17:38:06 OPCInt 1> One of the required Failover Synchronization points was not loaded. Error = 0: The Heartbeat point for this copy of the interface was not loaded. The input PI tag was not loaded.
Cause	The heartbeat tag is not configured properly.
Resolution	Check validity of point attributes. Make sure the Location1 attribute is valid for the interface. All failover tags must have the same PointSource and Location1 attributes. Modify point attributes as necessary and restart the interface.

Message	17-May-06 09:06:03 OPCint > The UniInt FailOver ID (/UFO_ID) must be a positive integer.
Cause	The UFO_ID parameter has not been assigned a positive integer value.
Resolution	Change and verify the parameter to a positive integer and restart the interface.

Message	17-May-06 09:06:03 OPCInt 1> The Failover ID parameter (/UFO_ID) was found but the ID for the redundant copy was not found
Cause	The /UFO_OtherID parameter is not defined or has not been assigned a positive integer value.
Resolution	Change and verify the /UFO_OtherID parameter to a positive integer and restart the interface.

Errors (Phase 1)

Message	17-May-06 09:06:03 OPCInt 1> UniInt failover: Interface in an “Error” state. Could not read failover control points.
Cause	The failover control points on the data source are returning an erroneous value to the interface. This error can be caused by creating a non-initialized control point on the data source. This message will only be received if the interface is configured to be synchronized through the data source (Phase 1).
Resolution	Check validity of the value of the control points on the data source.

Message	16-May-06 17:29:06 OPCInt 1> Loading Failover Synchronization tag failed Error Number = 0: Description = [FailOver] or HeartBeat:n] was found in the exdesc for Tag Active_IN but the tag was not loaded by the interface. Failover will not be initialized unless another Active ID tag is successfully loaded by the interface.
Cause	The Active ID or heartbeat tag is not configured properly. This message is received only if the interface is configured to be synchronized through the data source (Phase 1).
Resolution	Check validity of point attributes. For example, make sure Location1 attribute is valid for the interface. All failover tags must have the same PointSource and Location1 attributes. Modify point attributes as necessary and restart the interface.

Message	17-May-06 09:05:39 OPCInt 1> Error reading Active ID point from Data source Active_IN (Point 29600) status = -255
Cause	The Active ID point value on the data source caused an error when read by the interface. The value read from the data source must be valid. Upon receiving this error, the interface enters the "Backup in Error state."
Resolution	Check validity of the value of the Active ID point on the data source.

Message	17-May-06 09:06:03 OPCInt 1> Error reading the value for the other copy's Heartbeat point from Data source HB2_IN (Point 29604) status = -255
Cause	The heartbeat point value on the data source caused an error when read by the interface. The value read from the data source must be valid. Upon receiving this error, the interface enters the "Backup in Error state."
Resolution	Check validity of the value of the heartbeat point on the data source

Error Messages (Phase 2)

Message	27-Jun-08 17:27:17 PI Eight Track 1 1> Error 5: Unable to create file '\\georgiaking\GeorgiaKingStorage\UnIntFailover\PIEightTrack_eight_1.dat' Verify that interface has read/write/create access on file server machine. Initializing uniint library failed Stopping Interface
Cause	The interface is unable to create a new failover synchronization file at startup. The creation of the file takes place the first time either copy of the interface is started and the file does not exist. The error number most commonly seen is error number 5. Error number 5 is an "access denied" error and is probably the result of a permissions problem.
Resolution	Ensure the account that the interface is running under has read and write permissions for the directory. Set the "log on as" property of the Windows service to an account that has permissions for the directory.

Message	Sun Jun 29 17:18:51 2008 PI Eight Track 1 2> WARNING> Failover Warning: Error = 64 Unable to open Failover Control File '\\georgiaking\GeorgiaKingStorage\Eight\PIEightTrack_eight_1.dat' The interface will not be able to change state if PI is not available
---------	---

Error and informational messages for the PI OPC DA interface

Cause	The interface is unable to open the failover synchronization file. The interface failover continues to operate correctly if communication to the PI Server is not interrupted. If communication to PI is interrupted while one or both interfaces cannot access the synchronization file, the interfaces remain in the state they were in at the time of the second failure: the primary interface remains primary and the backup interface remains backup.
Resolution	Ensure the account that the interface is running under has read and write permissions for the directory. Set the "log on as" property of the Windows service to an account that has permissions for the directory.

Technical support and other resources

For technical assistance, contact OSIsoft Technical Support at +1 510-297-5828 or <http://techsupport.osisoft.com>. The OSIsoft Technical Support website offers additional contact options for customers outside of the United States.

When you contact OSIsoft Technical Support, be prepared to provide this information:

- Product name, version, and build numbers
- Details about your computer platform (CPU type, operating system, and version number)
- Time that the difficulty started
- Log files at that time
- Details of any environment changes prior to the start of the issue
- Summary of the issue, including any relevant log files during the time the issue occurred

The [OSIsoft Virtual Campus \(vCampus\) website](http://vcampus.osisoft.com) (<http://vcampus.osisoft.com>) has subscription-based resources to help you with the programming and integration of OSIsoft products.

