



PI Interface for IEEE C37.118

Version 1.1.0.x

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>

OSIsoft Australia • Perth, Australia
OSIsoft Europe GmbH • Frankfurt, Germany
OSIsoft Asia Pte Ltd. • Singapore
OSIsoft Canada ULC • Montreal & Calgary, Canada
OSIsoft, LLC Representative Office • Shanghai, People's Republic of China
OSIsoft Japan KK • Tokyo, Japan
OSIsoft Mexico S. De R.L. De C.V. • Mexico City, Mexico
OSIsoft do Brasil Sistemas Ltda. • Sao Paulo, Brazil
OSIsoft France EURL • Paris, France

PI Interface for IEEE C37.118

Copyright: © 2007-2015 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.

Published: 08/2015

Table of Contents

Chapter 1. Introduction	1
Reference Manuals	4
Supported Operating Systems	4
Supported Features.....	5
Diagram of Hardware Connection	8
Chapter 2. Principles of Operation	9
Startup Process.....	9
LocalEndPoint.....	10
RemoteEndPoint.....	10
PMU	10
Loading PI Points	10
Data Processing	11
Data Flow to the PI System	12
Multi-Threading	12
Communications Plug-Ins	13
PIC37118_Comm001	13
PIC37118_Comm002	13
PIC37118_Comm003	13
Data Quality Indications	13
Interface Robustness	14
PDC / PMU Data Source Level Failover	14
Failover.....	15
Communication Failure.....	15
Detection of Stale Data.....	15
Chapter 3. Installation Checklist.....	17
Data Collection Steps.....	17
Interface Diagnostics.....	18
Advanced Interface Features	19
Chapter 4. Interface Installation.....	21
Naming Conventions and Requirements	21
Interface Directories	22
PIHOME Directory Tree	22
Interface Installation Directory	22
Interface Installation Procedure	22
Silent Installation Procedure	22
PI Trust for Interface Authentication.....	22
Installing Interface as a Windows Service.....	23
Installing Interface Service with PI Interface Configuration Utility.....	23
Service Configuration	24
Installing Interface Service Manually.....	29

Chapter 5. Digital States.....	31
Chapter 6. PointSource	33
Chapter 7. PI Point Configuration.....	35
Point Attributes	35
Tag	35
PointSource	36
PointType	36
Location1	36
Location2	36
Location3	37
Location4	37
Location5	37
InstrumentTag.....	37
ExDesc.....	40
Scan	40
Shutdown	40
DataSecurity	41
PtSecurity.....	41
Chapter 8. Startup Command File	43
Configuring the Interface with PI ICU.....	43
C37118 Interface page	46
Sample PIC37118 XML File	63
Command-line Parameters	64
Sample PIC37118.bat File	71
Chapter 9. Unilnt Failover Configuration	73
Introduction.....	73
Quick Overview	74
Synchronization through a Shared File (Phase 2)	75
Configuring Synchronization through a Shared File (Phase 2).....	76
Configuring Unilnt Failover through a Shared File (Phase 2)	79
Start-Up Parameters	79
Failover Control Points	81
PI Points.....	82
Detailed Explanation of Synchronization through a Shared File (Phase 2)	86
Steady State Operation	87
Failover Configuration Using PI ICU	89
Create the Interface Instance with PI ICU	89
Configuring the Unilnt Failover Startup Parameters with PI ICU	90
Creating the Failover State Digital State Set	91
Using the PI ICU Utility to create Digital State Set	91
Using the PI SMT 3 Utility to create Digital State Set.....	92
Creating the Unilnt Failover Control and Failover State Points (Phase 2).....	95
Chapter 10. Interface Node Clock.....	97
Chapter 11. Security	99
Windows	99

Authentication.....	99
Authorization	100
Chapter 12. Starting / Stopping the Interface	103
Starting Interface as a Service	103
Stopping Interface Running as a Service.....	103
Chapter 13. Buffering for PI Interfaces.....	105
Buffering services.....	105
Buffering and collectives	105
Buffering configuration	105
Chapter 14. Interface Diagnostics Configuration	107
Scan Class Performance Points	107
Performance Counters Points	110
Performance Counters.....	112
Performance Counters for both (_Total) and (Scan Class x)	112
Performance Counters for (_Total) only	113
Performance Counters for (Scan Class x) only	116
Interface Health Monitoring Points	117
I/O Rate Point	122
Appendix A. Error and Informational Messages.....	125
Message Logs	125
Messages	126
Interface Configuration Error Messages	126
Point Loading Error Messages	128
Communications Related Error Messages	129
System Errors and PI Errors	131
UniInt Failover Specific Error Messages	132
Informational	132
Errors (Phase 1 & 2)	133
Errors (Phase 2).....	134
Appendix B. PI SDK Options.....	135
Appendix C. Communication Plug-In Selection	137
Appendix D. Terminology	139
Interface Specific Terms	139
General Terms	139
Appendix E. Technical Support and Resources.....	143
Appendix F. Revision History	145

Chapter 1. Introduction

This document describes the use of the PI Interface for IEEE C37.118, from here on referred to as the PI C37.118 interface. The PI C37.118 interface supports the IEEE C37.118/D7.3.5 Standard for Synchrophasors for Power Systems.

The PI C37.118 interface operates with any C37.118 compliant device via an Ethernet or serial interface. When connected to the C37.118 or Phasor Measurement Unit (PMU) or Phasor Data Concentrator (PDC) the interface provides highly accurate measurement of electrical power quality. Each instance of the interface can connect to multiple PDCs or PMUs and provides electrical measurements to the PI System.

The IEEE C37.118 specification does not specify what electrical measurements the device must provide; rather, it specifies only how the protocol-specific data (if provided) must be formatted. Therefore, each C37.118 device may provide its own unique set of Phasor measurements. The PI C37.118 interface is capable of handling specific Phasor configuration sets that are provided by the C37.118 compliant device.

The following electrical measurements and flags are typically provided by C37.118 compliant devices:

- C37.118 compliant Phasor data calculated at a rate defined by the PDC or PMU.
 - System Frequency
 - Frequency Deviation
 - Frequency Rate of Change
 - A, C, B magnitude and phase angle for Voltage and Current given in 16-bit (scaled) Polar format
 - A, C, B real and imaginary parts of complex number for Voltage and Current given in 16-bit (scaled) Rectangular format
 - A, C, B magnitude and phase angle for Voltage and Current given in 32-bit IEEE floating point Polar format
 - A, C, B real and imaginary parts of complex number for Voltage and Current given in 32-bit IEEE floating point format
 - Analog channel data as a 16-bit integer (scaled) or 32-bit IEEE floating point. The values and ranges defined by user.
 - Digital channel data as a 16-bit word with values and ranges defined by the user.

- Quality information reflecting the accuracy of the C37.118 data is available via a series of optional quality tags.
 - Data Valid quality returned in the Data Valid flag (Bit 15) of the C37.118 STAT word indicates problems with the C37.118 data block. Some data in the block may be invalid or filled in.
 - PMU Error quality returned in the PMU Error flag (Bit 14) of the C37.118 STAT word indicates internal errors with the PMU such as A/D calibration errors, memory errors, etc.
 - PMU Sync Error quality returned in the PMU Sync flag (Bit 13) of the C37.118 word indicates a loss of external time synchronization such as the loss of satellite tracking or IRIG-B input failure.
 - Data Sorting information used by PDC's to indicate the method by which the PMU data is inserted into the data frame. This information is extracted from the Data Sorting flag (Bit 12) of the C37.118 STAT word.
 - Time Quality indicator codes extracted from the Time Quality field (Bits 3-0) of the C37.118 FRACSEC word can be used with the PMU Sync Error quality and indicates the accuracy of the timestamp provided.
 - Time Lock quality can be used in conjunction with the PMU Sync Error quality to determine the length of time that time sync has been lost. This information is extracted from the Unlocked Time flags (Bits 4 and 5) of the C37.118 STAT word.
 - Leap Second status can be extracted from the Leap Second flags (Bits 6-4) of the C37.118 FRACSEC word indicates if a leap second is pending or occurring and indicates the direction of the leap.
- Non C37.118 measurement information can also be stored in the PI Data Archive.
 - Interface diagnostic information
 - Longitude and Latitude of the PMU.

Measured data from the PMU is time-stamped by the PMU in accordance with the C37.118 standard for C37.118 data. Timestamp accuracy is achieved through an internal Global Positioning System (GPS) satellite receiver that synchronizes the PMU to within 1 μ s of Coordinated Universal Time (UTC).

If a PDC or PMU indicates a data quality or time synchronization issue by setting any of the Data Valid, PMU Error, PMU Sync Error or Data Sorting Type flags in the STAT word, the interface can do one of the following (this behavior is controlled via the XML configuration file):

- Store normally.
- Mark all PMU measurements as Questionable.
- Apply a System Digital State corresponding to the flag(s) that are set. If multiple flags are set, the most critical applies.
- Discard the measurement values.

Multiple instances of the interface can be run on a single PI interface node. However, when running multiple instances of the interface on a single node, CPU and memory allocation should be monitored to ensure the system remains in a stable state of operation. The exact number of instances that may be run on a single PI interface node will vary according to hardware and network specifications.

Both PDC / PMU Data Source Level Failover and UniInt Phase 2 interface level failover are supported. When running in PDC / PMU Data Source Level Failover mode, the interface obtains data from one of two available sources PDC / PMU. The source PDC / PMU must have identical phasors, analog and digital channels and data streams for each interface. This requirement ensures that the interface obtains the same data regardless of which source is active. When running in UniInt Failover mode, two copies of the interface are connected to the PMU/PDC at the same time. When the primary interface stops collecting data, the backup interface assumes the primary role and continues data collection. PMU/PDC Data Source Level Failover and UniInt Phase 2 interface level failover modes can be run simultaneously. Failover maximizes interface data availability on the receiving PI Data Archive(s).

Note: It should be noted that the C37.118 specification is very vague with respect to communications configuration, and the various PDC / PMU vendors have implemented devices in a variety of ways. As such, the design of the C37118 interface uses an XML configuration file to provide the information needed to establish communications with the various C37.118 devices. **Check with OSIsoft to verify that your make and model of PDC or PMU has been validated for use with the C37118 interface.**

Note: The value of the [PIHOME] variable for the 32-bit interface will depend on whether the interface is being installed on a 32-bit operating system (C:\Program Files\PIPC) or a 64-bit operating system (C:\Program Files (x86)\PIPC).

The value of the [PIHOME64] variable for a 64-bit interface will be:
C:\Program Files\PIPC on a 64-bit operating system.

In this documentation [PIHOME] is used to represent the value for either [PIHOME] or [PIHOME64]. The value of [PIHOME] is the directory in which PI client applications are stored.

Note: This interface has been built against a UniInt version (4.5.0.59 and later) which now writes all its messages to the local PI Message log.

Please see the document *UniInt Interface Message Logging.docx* in the %PIHOME%\Interfaces\UniInt directory for more details about how to access these messages.

Note: OSIsoft is revising product documentation and other literature to reflect the evolution of the PI Server from a single server to a multi-server architecture. Specifically, the original historian core of the PI Server is now referred to as the PI Data Archive.

Originally, the PI Server was a single server that contained the PI Data Archive and other subsystems. To add features and improve scalability, the PI Server has evolved from a single server to multiple servers. While the PI Data Archive remains a core server of the PI Server product, the product name “PI Server” now refers to much more than the PI Data Archive. OSIsoft documentation, including this user manual, is changing to use “PI Server” in this broader sense and “PI Data Archive” to refer to the historian core.

Reference Manuals

OSIsoft

- PI Data Archive manuals
- *PI API Installation Instructions* manual
(%PIHOME%\bin\API_install.doc)
- *PI Universal Interface (UniInt) User Guide*
(%PIHOME%\Interfaces\UniInt\PI Universal Interface (UniInt) User Guide.pdf)
- *UniInt Interface Message Logging for UniInt 4.5.0.x and later Interfaces*
(%PIHOME%\Interfaces\UniInt\UniInt Interface Message Logging.pdf)
- *PI Interface Configuration Utility User Guide*
(%PIHOME%\ICU\PI Interface Configuration Utility.pdf)

Vendor


- *IEEE C37.118/D7.3.5 Standard for Synchrophasors for Power Systems*

Supported Operating Systems

Platforms		32-bit application	64-bit application
Windows Vista	32-bit OS	Yes	No
	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
	64-bit OS	Yes (Emulation Mode)	No
Windows 8	32-bit OS	Yes	No
	64-bit OS	Yes (Emulation Mode)	No
Windows 2012 Server	64-bit OS	Yes (Emulation Mode)	No

The interface is designed to run on the above mentioned Microsoft Windows operating systems and their associated service packs.

Please contact OSIsoft Technical Support for more information.

 **Security Note:** OSIsoft recommends installing all available updates from the Windows Update service. OSIsoft also recommends using the newest versions of Windows to obtain the latest security features.

Supported Features

Feature	Support
Interface Part Number	PI-IN-OS-C37.118-NTI
Auto Creates PI Points	No
Point Builder Utility	No
ICU Control	Yes
PI Point Types	Digital / Int16 / Int32 / Float16 / Float32
Sub-second Timestamps	Yes
Sub-second Scan Classes	No (Unsolicited Only)
Automatically Incorporates PI Point Attribute Changes	Yes
Exception Reporting	No
Outputs from PI	No
Inputs to PI:	Unsolicited
Supports Questionable Bit	Yes
Supports Multi-character PointSource	Yes
Maximum Point Count	No
* Uses PI SDK	No
PINet String Support	No
* Source of Timestamps	C37.118 Datastream
History Recovery	No
* UInt-based	Yes
* Disconnected Startup	Yes
* SetDeviceStatus	Yes
* Failover	Data Source Level Failover UInt Failover (Phase 2 Cold and Hot)
* Vendor Software Required on Interface Node / PINet Node	No
Vendor Software Required on Foreign Device	No
Vendor Hardware Required	Yes
Additional PI Software Included with interface	No
Device Point Types	Digital, Float, Integer (including Real and Imaginary)
Serial-Based interface	Yes

** See paragraphs below for further explanation.*

Uses PI SDK

The PI SDK and the PI API are bundled together and must be installed on each interface node. This interface does not specifically make PI SDK calls.

Source of Timestamps

Timestamps are acquired from the C37.118 data stream.

UniInt-based

UniInt stands for Universal Interface. UniInt is not a separate product or file; it is an OSIsoft-developed template used by developers and is integrated into many interfaces, including this interface. The purpose of UniInt is to keep a consistent feature set and behavior across as many OSIsoft interfaces as possible. It also allows for the very rapid development of new interfaces. In any UniInt-based interface, the interface uses some of the UniInt-supplied configuration parameters and some interface-specific parameters. UniInt is constantly being upgraded with new options and features.

The *PI Universal Interface (UniInt) User Guide* is a supplement to this manual.

Disconnected Start-Up

The PI IEEE C37.118 interface is built with a version of UniInt that supports disconnected start-up. Disconnected start-up is the ability to start the interface without a connection to the PI Data Archive. This functionality is enabled by adding **/cachemode** to the list of start-up parameters or by enabling disconnected startup using the ICU. Refer to the *PI Universal Interface (UniInt) User Guide* for more details about UniInt Disconnect startup.

SetDeviceStatus

The health point, having the attribute Exdesc = [UI_DEVSTAT], is used to represent the status of the source device. The following events can be written into this point:

- “1 | Starting | UI #.#.#.#” – The interface is currently starting up.
- “Good” – The interface is properly communicating and reading data from all configured PMUs. A value of “Good” does not mean that all points are receiving good values, but it is a good indication that there are no hardware or network problems.
- “3 | 1 device(s) in error | REP: 1 LEP 1” – The connection to one or more PMUs has failed.
- “4 | Intf Shutdown” – The interface has been shutdown.

Refer to the *PI Universal Interface (UniInt) User Guide* for more information on how to configure health points.

Failover

- PDC / PMU Data Source Level Failover
 - PDC / PMU Data Source Level Failover maximizes interface data availability on the receiving PI Data Archive(s). It enables the interface to obtain data from one of two PMU/PDC data sources. Each PMU/PDC data source must have identical tag definitions and data streams for interface source points.
 - The interface initiates failover if the active source data becomes stale or is not available due to network issues.
- UniInt Failover Support

UniInt Phase 2 Failover provides support for cold, warm or hot failover configurations. The Phase 2 hot failover results in a *no data loss* solution for bi-directional data transfer between the PI Data Archive and the Data Source given a single point of failure in the system architecture similar to Phase 1. However, in warm and cold failover configurations, you can expect a small period of data loss during a single point of failure transition. This failover solution requires that two copies of the interface be installed on different interface nodes, both of which are collecting data simultaneously from a single data source. Phase 2 Failover requires that each interface have access to a shared data file. Failover operation is automatic and operates with no user interaction. Each interface participating in failover has the ability to monitor and determine liveliness and failover status. To assist in administering system operations, the ability to manually trigger failover to a desired interface is also supported by the failover scheme.

The failover scheme is described in detail in the *PI Universal Interface (UniInt) User Guide*, which is a supplement to this manual. Details for configuring this interface to use failover are described in the [UniInt Failover Configuration](#) section of this manual.

This interface supports UniInt Failover (Phase 2 Cold and Hot).

Vendor Hardware Required

A C37.118-compliant Phasor Measurement Unit or Phasor Data Concentrator is required for proper communication with this interface.

Device Point Types

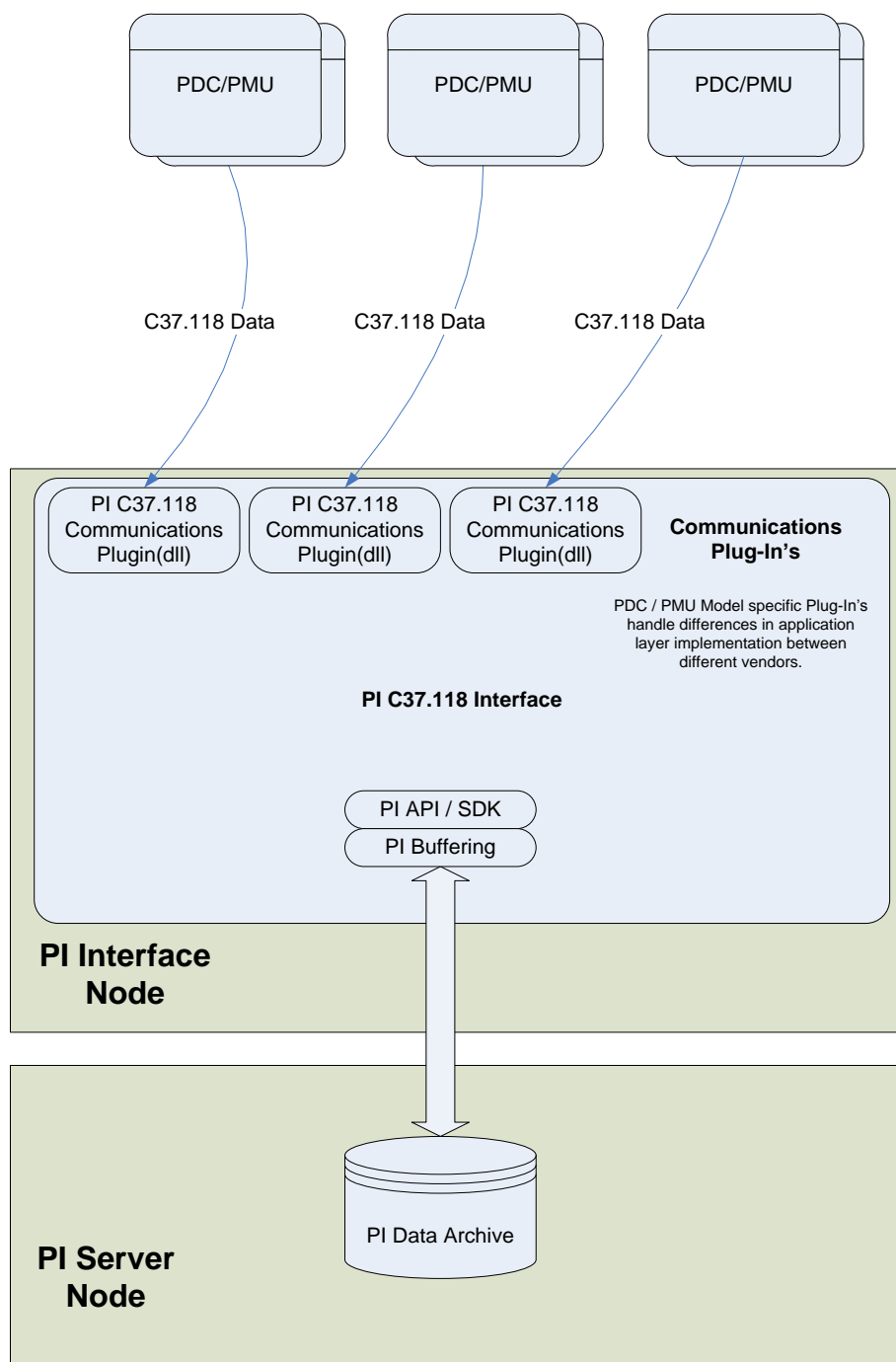
The interface supports all Phasor, Analog and Digital channels plus frequency and frequency rate of change. Frequency error is also computed by the interface.

Serial-Based Interface

This interface can run either with a serial connection or Ethernet network connection.

Server class machines often have low-quality serial ports. Server class machines are not required for most interfaces and should not be used, especially when serial port connections are required.

Diagram of Hardware Connection



Chapter 2. Principles of Operation

The PI C37.118 interface is designed to request and process data from C37.118 compliant devices. The PI C37.118 interface is a real-time based interface. The interface operations can be categorized into the following list of sub-operations:

- Startup Process
- Loading PI Points
- Data Processing
- Data Flow to the PI System
- Data Quality Indicators
- Interface Robustness

A more detailed explanation of the list of sub-operations for this interface is provided in this section.

Startup Process

The PI C37.118 interface must be configured in accordance with the Startup Command File (*.bat* file) and the Interface Configuration File (XML configuration file). Each instance of the C37.118 interface is assigned a specific identification number and/or point source, which are correlated with the interface-specific startup file parameters **/id**, **/ps** and **/configfile**.

The C37.118 interface establishes the initial connection to PI and reconnects to PI in the event that the connection is lost. If the interface is started while the PI Data Archive is down, the interface periodically attempt to establish a connection until the PI Data Archive is up and running.

The interface is designed to make use of a configuration file to set up the internal operation of the interface. The configuration file is written in XML format. A PI Interface Configuration Utility (ICU) plug-in is available for this interface to assist in the configuration of both the startup batch file and the interface (XML) configuration file. The use of this ICU Control (which is highly recommended) is described in the [Startup Command File](#) section of this document.

When the interface starts, it parses the command line parameters and searches for the XML configuration file that is specified in the **/configfile** command line parameter. If the **/configfile=<UNCpath>** is not found, a default file name (*pic37118.xml*), is assumed. The configuration file is then parsed and the interface instantiates the internal structures that are required for communication with the C37.118 PDC and / or PMUs.

A description of the internal structures and the terms used by the interface are described below.

LocalEndPoint

The LocalEndPoint represents the local (interface side) of a physical network or serial connection to a C37.118 device. Each LocalEndPoint should contain the various LocalEndPoint attributes as well as a single RemoteEndPoint leaf node which contains the RemoteEndPoint attributes.

RemoteEndPoint

The RemoteEndPoint object represents the remote (C37.118 device) side of a physical connection. Currently there can be only one RemoteEndPoint configured for each LocalEndPoint.

PMU

The PMU object represents a C37.118 Phasor Data Concentrator (PDC) or Phasor Measurement Unit (PMU). If the RemoteEndPoint is a PDC, there can be multiple PMU objects associated with the RemoteEndPoint.

Loading PI Points

When the PI C37.118 interface starts, it searches the PI Point Database for points that belong to the interface and creates a point list for the interface. The primary method of establishing a mapping between a PI point and a C37.118 device is through configuration of the `Location2`, `Location3`, `Location5`, and `InstrumentTag` PI point attributes. The correlation between the above attributes and the various C37.118 measurements are listed in the [PI Point Configuration](#) section of this document.

PI points for this interface have a one-to-one correspondence with electrical measurements that are available from the C37.118 device. There can be only one PI point for receiving a specific measurement from a given C37.118 device.

For example: if a PI point is configured to read the magnitude from a Phasor channel named “Phasor Volt CH-A” (`InstrumentTag=PhasorMag\Phasor Volt CH-A`) from a specific LocalEndPoint, RemoteEndPoint and PMU combination (`Location2`, `Location3` and `Location5`) there can be no other PI points specified with identical attributes.

After the interface loads a PI point, any additional PI points that have the same attributes as a previously loaded PI point are rejected by the interface and the interface logs an error message. Refer to [Appendix A. Error and Informational Messages](#) for more details about messages that are associated with loading PI points.

Data Processing

The PI C37.118 interface processes C37.118 formatted data messages as defined in the IEEE C37.118 specification. The input data is parsed to provide the following:

- Real and Imaginary part of the complex number for each Phasor channel if the PDC / PMU is configured to send Phasor measurements in rectangular format.
- Magnitude and Phase Angle for each Phasor channel for each Phasor channel if the PDC / PMU is configured to send Phasor measurements in polar format.
- Analog channel data
- Digital channel data
- System Frequency
- Frequency deviation (calculated by interface)
- Frequency Rate of Change

Note: If the PMU is configured to send data in rectangular format, the interface computes the polar coordinates; if the PMU is configured to send data in polar format the interface computes the rectangular coordinates.

In addition to the data parsed from the C37.118 data, the interface also provides for input of C37.118 configuration data from the CONFIG2 data block. The configuration data is parsed to provide the following:

- Configuration change count from CFGCNT
- Data transmit rate from DATA_RATE
- Nominal frequency (50Hz / 60Hz)

Optional data quality tags can be configured to receive information related to the quality of the C37.118 data. This quality information is extracted from the time quality, leap second status, and STAT fields of the C37.118 data stream. These include the following:

- Data Valid
- PMU Error
- PMU Sync
- Data Sorting
- Time Unlocked Time
- Trigger Reason

The interface can also store geographical information regarding the location of the PMU if defined in the Interface Configuration File. The Latitude and Longitude of the PMU can be defined in decimal form in the configuration file and stored in the PI Data Archive.

Data Flow to the PI System

Data flow from the C37.118 device to the PI System is unsolicited. The following is an explanation of the sequence of events that take place in order to transfer data from the C37.118 device interface into the PI System.

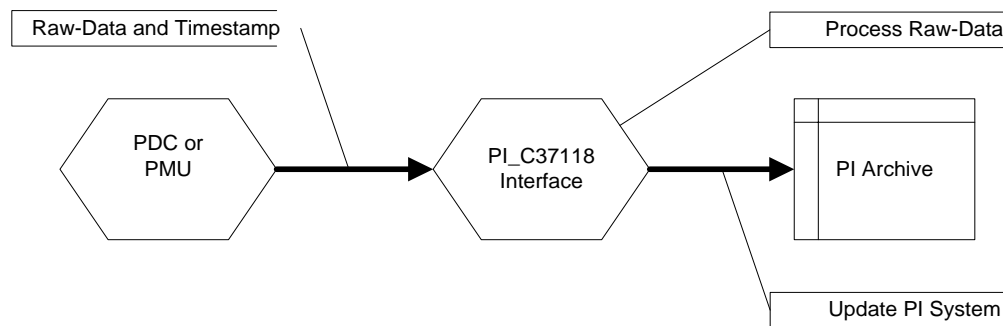
Phasor data from C37.118 is sent to the interface at a rate which is configured in the C37.118 device and is typically 20, 30, 60 or 120 times per second for 60Hz systems and 25, 50, and 100 times per seconds for 50Hz systems.

When the interface communications plug-in issues the start data command (which is issued at startup and after any communications failure), the C37.118 device starts sending data at the configured rate.

The communications plug-in then processes the incoming data stream. Each C37.118 data frame is processed and the size and CRC verified. The C37.118 frame is then queued to the main interface thread.

The main interface thread of the interface parses and processes C37.118 data frames in the incoming data queue and sends the results to the PI System on a real-time basis. This means that device data, after processing, is sent immediately and is available in the PI System.

The illustration below shows a graphical representation of data flow from the C37.118 PDC / PMU device to the PI System.



Multi-Threading

The PI C37.118 interface is multithreaded. Communications with the PDC / PMU is handled in separate threads from the main interface body. Depending on the Communications Plug-In being used, there will be either 1 or 2 communications threads created. If commands and command responses operate on the same communications port as the C37.118 data stream, a single communications thread will handle all communications with the PDC / PMU. If commands and command responses operate on independent ports then 2 communications threads are created. One thread handles commands and command responses and the second is dedicated to receiving the C37.118 data stream. Data from each communications thread is buffered and then serialized through the main interface thread. This must be done because the PI API is not currently thread safe.

Communications Plug-Ins

The PI C37.118 interface uses “Communications Plug-Ins” to handle differences in the application layer communications for various PDCs and PMUs. This is necessary because the IEEE C37.118 specification does not dictate the implementation.

The following describes the plug-ins distributed with the PI C37.118 interface.

PIC37118_Comm001

This plug-in is used for devices that use a single socket via either UDP or TCP for commands, command responses, and measurement data. This plug-in is also used for serial connections.

PIC37118_Comm002

This plug-in is used for devices that use two sockets. Commands and command responses are performed via one socket and measurement data is returned via a separate socket. Typically the device uses TCP for the commands and UDP for the measurement data.

PIC37118_Comm003

This plug-in is used for devices that use a single socket via either UDP or TCP but do not support output commands. Once the device starts it sends data automatically to a predetermined address and port. This is referred to as “Autonomous Mode” on most devices. It expects that the CONFIG2 block is sent periodically along with the measurement data.

Data Quality Indications

The PI C37.118 interface provides multiple methods to monitor the quality of the data being provided by the C37.118 PDC / PMU.

The following types of data quality indications are available. All of these are optional and can be used alone or in conjunction with other data quality indicators.

- **Quality Tags** – Quality tags are PI tags that are separate from data tags.
- **Conditional Data Tag Status** – The interface can be configured to apply one of 4 different rules in response to one or more of the data quality flags being set.

Interface Robustness

Where data acquisition is concerned, the C37.118 interface is designed to be as robust as possible. If the interface determines that Phasor data has not been received within the time specified by the **DataTimeout** parameter in the configuration file, an entry is made to the *pipc.log* file and the interface attempts to restart the C37.118 phasor data stream. Upon receipt of a successful response from the C37.118 device, the interface begins processing data and logs a message to the *pipc.log* file. The interface will retry failed connection attempts until data flow is restored but will not log further connection failure messages until data flow is restored. Refer to [Appendix A. Error and Informational Messages](#) for more details about messages that are associated with transmission loss.

The interface is configured to make use of the **WriteIOTimeout** parameter in the interface configuration file. This parameter instructs the interface about whether to write an error state to PI points when data from the device is not received after the period specified by the **DataTimeout** parameters. Setting **WriteIOTimeout=1** (the default), causes the system digital state of “I/O Timeout” to be written to PI points. Alternately, setting **WriteIOTimeout=0** instructs the interface not to update PI points during transmission loss of C37.118 data. The default behavior is to write “I/O Timeout” for all measured values.

When the interface starts, the interface searches the PI Point Database for points that belong to the interface and a point list is created for the interface.

When startup is complete, the interface enters the processing loop, which includes the following:

- Servicing scheduled input points. Each Scan Class is processed in turn.
- Servicing triggered input points as events arrive.
- The PI Point Database is checked every 2 minutes for points that are added, edited, and deleted. If point updates are detected, the points are loaded (or reloaded) by the interface as appropriate. The 2-minute update interval can be adjusted with the **/updateinterval** command-line parameter described in the *PI Universal Interface (UniInt) User Guide*. The interface will only process 25 point updates at a time. If more than 25 points are added, edited, or deleted at one time, the interface will process the first 25 points, wait 30 seconds (or by the time specified by the **/updateinterval** parameter, whichever is lower), process the next 25 points, and so on. After all points have been processed, the interface resumes checking for updates every 2 minutes (or by the time specified by the **/updateinterval** parameter). The interface writes the digital state **SCAN OFF** to any points that are removed from the interface while it is running.

PDC / PMU Data Source Level Failover

PDC / PMU Data Source Level Failover maximizes interface data availability on the receiving C37.118 data stream. It requires that two PDC / PMU data sources are available that have identical Phasor, analog and digital channels and data streams for interface points. This requirement ensures that the interface obtains the same data regardless of which data source is active.

Failover is enabled by specifying a pair of primary and secondary Local End Points by specifying Local End Point and Failover Local End Point IDs in the xml configuration file. On startup, the interface attempts to establish a connection to the primary data source, then load and initialize its tag list. Data collection begins from the primary PDC / PMU data source making it the active node while the secondary PDC / PMU data source assumes the standby (backup) role.

To set up PDC / PMU Data Source Level Failover you must create at least two Local End Points. You should bind the end points together to create a redundant pair. To do so, use ID and Failover ID fields. Failover ID should be equal to the ID of the redundant Local End Point and vice versa. Only one Local End Point in the redundant pair should be specified as a primary. Use the Primary field to specify which Local End Point is a primary. Another one will be a secondary Local End Point.

To control delay between switches from primary to backup PDC /PMU use the Switch Backup Delay parameter. Default value is 60 seconds.

You can have multiple redundant pairs configured for the interface, or you can set up redundant Local End Points and non-redundant (Failover ID field not assigned) together for the same interface.

Failover

There are two conditions that cause the interface to initiate failover: a communication failure or detection of stale data.

Communication Failure

The first failover condition occurs when the interface loses communication to the active PDC / PMU data source. This might be the result of a temporary network outage, shutdown of the active data source, hardware failure, and so on. The interface cannot identify the cause of the disconnect failure. A disconnection means that the interface did not receive a response from the active source within the timeout period. The interface may initiate a short wait and then attempt to reconnect to the active source before attempting failover. The data timeout, reconnection rate, and reconnection timeout are all user configurable parameters.

The default reconnection timeout period is 60 seconds, which is set as an interface configuration parameter. If the interface is unable to reconnect to the active PDC / PMU during the reconnection timeout period, the interface attempts to connect to the alternate (primary or backup) PDC / PMU.

Detection of Stale Data

The second failover condition is caused by stale source data. The data received by the PDC / PMU is high frequency. Depending on the brand and configuration of each individual PMU data might arrive at a rate of 10, 20, 30 or 60 Hz. If the interface does not receive data during the data timeout interval (which defaults to 1000 milliseconds) the data is marked as timed out and becomes stale. The interface attempts to reconnect with the active PDC / PMU at the period specified by the reconnection rate interval. After the reconnection timeout interval is expired, the interface attempts to connect to the alternate (backup or primary) PDC /PMU. After each reconnection timeout interval the interface attempts to switch back and forth

between the primary and the backup PDC /PMU until it receives data from the first available PDC / PMU.

UniInt Failover

This interface supports UniInt failover. Refer to the [UniInt Failover Configuration](#) section of this document for configuring the interface for failover.

Chapter 3. Installation Checklist

If you are familiar with running PI data collection interface programs, this checklist helps you get the interface running. If you are not familiar with PI interfaces, return to this section after reading the rest of the manual in detail.

This checklist summarizes the steps for installing this interface. You need not perform a given task if you have already done so as part of the installation of another interface. For example, you only have to configure one instance of Buffering for every interface node regardless of how many interfaces run on that node.

The Data Collection Steps below are required. Interface Diagnostics and Advanced Interface Features are optional.

Data Collection Steps

1. Confirm that you are able to use PI SMT to configure the PI Data Archive. You need not run PI SMT on the same computer on which you run this interface.
2. If you are running the interface on an interface node, edit the PI Data Archive's Trust Table to allow the interface to read attributes and point data. If a buffering application is not running on the interface node, the PI trust must allow the interface to write data.
3. Run the installation kit for the PI Interface Configuration Utility (ICU) on the interface node if the ICU will be used to configure the interface. This kit runs the PI SDK installation kit, which installs both the PI API and the PI SDK.
4. Run the installation kit for this interface. This kit also runs the PI SDK installation kit which installs both the PI API and the PI SDK if necessary.
5. If you are running the interface on an interface node, check the computer's time zone properties. An improper time zone configuration can cause the PI Data Archive to reject the data that this interface writes.
6. Run the ICU and configure a new instance of this interface. Essential startup parameters for this interface are:

Point Source (/PS=x)

Interface ID (/ID=#)

PI Data Archive (/Host=host:port)

7. If you will use digital points, define the appropriate digital state sets.

8. Build input tags for this interface. Important point attributes and their purposes are:

Location1 specifies the interface instance ID.

Location2 specifies the LocalEndPoint ID.

Location3 specifies the RemoteEndPoint ID. This attribute always reflects the IDCODE of PDC / PMU contained in word 3 of the C37.118 data stream. If the connected device is a PMU (versus a PDC), Location3 and Location5 will be the same.

Location4 is not used.

Location5 specifies the PMU ID. This attribute will always match the IDCODE of the PMU that is the source of the data.

ExDesc is not used.

InstrumentTag specifies the specific measurement and type.

PtSecurity must permit read access for the PI identity, group, or user configured in the PI trust that is used by the interface.

DataSecurity must permit read access (buffering enabled) or read/write access (unbuffered) for the PI identity, group, or user configured in the PI trust that is used by the interface.



Security Note: When buffering is configured, the DataSecurity attribute must permit write access for the *buffering application's* PI trust or mapping. DataSecurity write permission for the interface's PI trust is required only when buffering is not configured.

9. Start the interface interactively and confirm its successful connection to the PI Data Archive without buffering. (The DataSecurity attribute for interface points must permit write access for the interface's PI trust.)
10. Confirm that the interface collects data successfully.
11. Stop the interface and configure a buffering application (either Bufserv or PIBufss).
12. Start the buffering application and the interface. Confirm that the interface works together with the buffering application by physically removing the connection between the interface node and the PI Data Archive node. (The DataSecurity attribute for interface points must permit write access for the *buffering application's* PI trust or mapping. The interface's PI trust does not require DataSecurity write permission.)
13. Configure the interface to run as a Windows service. Confirm that the interface runs properly as a service.
14. Restart the interface node and confirm that the interface and the buffering application restart.

Interface Diagnostics

1. Configure scan class performance points.
2. Install the PI Interface for Performance Monitor on the interface node.
3. Configure performance counter points.
4. Configure UniInt health monitoring points
5. Configure the I/O Rate point.

Advanced Interface Features

1. Configure the interface for disconnected startup. Refer to the *PI Universal Interface (UniInt) User Guide* for more details on UniInt disconnected startup.
2. Configure UniInt failover; see the [UniInt Failover Configuration](#) chapter in this document for details related to configuring the interface for failover.

Chapter 4. Interface Installation

OSIsoft recommends that interfaces be installed on PI interface nodes instead of directly on the PI Data Archive node. A PI interface node is any node other than the PI Data Archive node where the PI Application Programming Interface (PI API) is installed (see the PI API manual). With this approach, the PI Data Archive need not compete with interfaces for the machine's resources. The primary function of the PI Data Archive is to archive data and to service clients that request data.

After the interface has been installed and tested, Buffering should be enabled on the PI interface node. Buffering refers to either PI API Buffer Server (Bufserv) or the PI Buffer Subsystem (PIBufss). For more information about Buffering see the [Buffering for PI Interfaces](#) chapter of this manual.

In most cases, interfaces on PI interface nodes should be installed as automatic services. Services keep running after the user logs off. Automatic services automatically restart when the computer is restarted, which is useful in the event of a power failure.

The guidelines are different if an interface is installed on the PI Data Archive node. In this case, the typical procedure is to install the PI Data Archive as an automatic service and install the interface as an automatic service that depends on the PI Update Manager and PI Network Manager services. This typical scenario assumes that Buffering is not enabled on the PI Data Archive node. Bufserv can be enabled on the PI Data Archive node so that interfaces on the PI Data Archive node do not need to be started and stopped in conjunction with PI, but it is not standard practice to enable buffering on the PI Data Archive node. The PI Buffer Subsystem can also be installed on the PI Data Archive.

Naming Conventions and Requirements

In the installation procedure below, it is assumed that the name of the interface executable is `PIC37118.exe` and that the startup command file is called `PIC37118.bat`.

When Configuring the Interface Manually

It is customary for the user to rename the executable and the startup command file when multiple copies of the interface are run. For example, `PIC371181.exe` and `PIC371181.bat` would typically be used for interface number 1, `PIC371182.exe` and `PIC371182.bat` for interface number 2, and so on. When an interface is run as a service, the executable and the command file must have the same root name because the service looks for its command-line parameters in a file that has the same root name.

Interface Directories

PIHOME Directory Tree

The [PIHOME] directory tree is defined by the PIHOME entry in the `pipc.ini` configuration file. This `pipc.ini` file is an ASCII text file, which is located in the `%windir%` directory.

For 32-bit operating systems, a typical `pipc.ini` file contains the following lines:

```
[PIPC]
PIHOME=C:\Program Files\PIPC
```

For 64-bit operating systems, a typical `pipc.ini` file contains the following lines:

```
[PIPC]
PIHOME=C:\Program Files (X86)\PIPC
```

The above lines define the root of the PIHOME directory on the C: drive. The PIHOME directory does not need to be on the C: drive. OSIsoft recommends using the paths shown above as the root PIHOME directory name.

Interface Installation Directory

The interface install kit will automatically install the interface to:

```
PIHOME\Interfaces\C37118\
```

PIHOME is defined in the `pipc.ini` file.

Interface Installation Procedure

The C37118 interface setup program uses the services of the Microsoft Windows Installer. Windows Installer is a standard part of Windows operating systems. To install, run the appropriate installation kit.

```
C37118_#.###.###.exe
```

Silent Installation Procedure

To launch a silent installation, type:

```
Setup.exe -f silent.ini
```

The `silent.ini` file is included in the interface installation kit. You can make site-specific alterations to the file as needed. See the `silent.ini` file for further information and descriptions of available arguments.


PI Trust for Interface Authentication

A PI Interface usually runs on an interface node as a Windows service, which is a non-interactive environment. In order for an interface to authenticate itself to a PI Data Archive and obtain the access permissions for proper operation, the PI Data Archive must have a PI trust that matches the connection credentials of the interface. Determine if a suitable

PI trust for the interface exists on the PI Data Archive. If a suitable PI trust does not exist, see the [Security](#) chapter for instructions on creating a new PI trust.

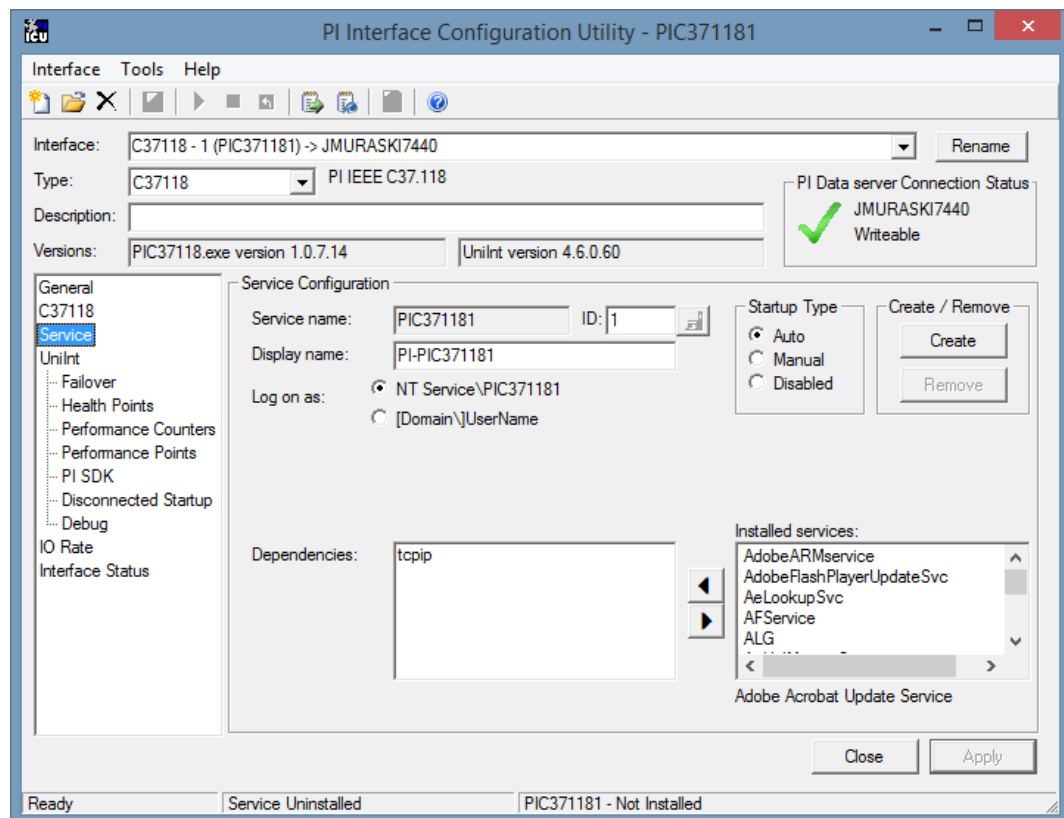
Installing Interface as a Windows Service

The preferred way to install the C37118 interface service is to use the PI Interface Configuration Utility; however, you can also create it manually.

 **Security Note:** For improved security, OSIsoft recommends running the interface service under a non-administrative account, such as a Windows built-in service virtual account, the built-in Network Service account, or a non-administrative account that you create.

Installing Interface Service with PI Interface Configuration Utility

The PI Interface Configuration Utility provides a user interface for creating, editing, and deleting the interface service:



Service Configuration

Service name

The *Service name* box shows the name of the current interface service. This service name is obtained from the interface executable.

ID

This is the service ID that is used to distinguish among multiple instances of the same interface using the same executable.

Display name

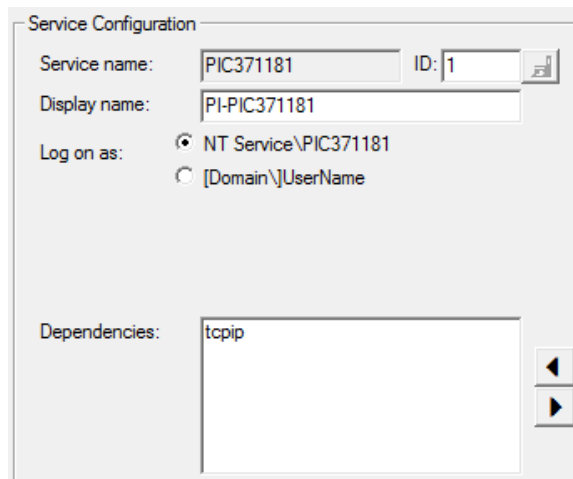
The *Display name* text box shows the current Display Name of the interface service. If there is currently no service for the selected interface, the default Display Name is the service name with a “PI-” prefix. Users may specify a different Display Name. OSIsoft suggests that the prefix “PI-” be appended to the beginning of the interface name to indicate that the service is part of the OSIsoft suite of products.

Log on as

The *Log on as:* options allow the interface service to be configured to use a *default Windows Service* account, a *user-specified Windows* account or the *Local System* account.

Default Windows Service account

Select the *NT Service\PIC371181* option to configure the internet service to use a *default Windows Service* account.



The screenshot shows a 'Service Configuration' dialog box. It has several fields: 'Service name' with the value 'PIC371181', 'ID' with the value '1', and 'Display name' with the value 'PI-PIC371181'. Below these is the 'Log on as' section with two radio button options: 'NT Service\PIC371181' (which is selected) and '[Domain]\UserName'. At the bottom is a 'Dependencies' list box containing the text 'tcpip'.

This figure shows the option to select to configure the interface service to use a *default Windows Service* account.

User-specified Windows account

Select the *[Domain]\UserName* option to add a *user-specified Windows* account which the interface service will use. Please note that this account differs from the *default Windows Service* account.

Service Configuration

Service name: PIC371181 ID: 1

Display name: PI-PIC371181

Log on as:
☐ NT Service\PIC371181
☒ [Domain]\UserName

UserName:
 Password:
 Confirm password:
 Dependencies: tcpip

This figure shows the option to select to configure the interface service to use a *user-specified Windows* account.

When the *[Domain]\UserName* option is selected, three additional boxes are displayed. These additional boxes are used to enter information about the *user-defined Windows* account.

Service Configuration

Service name: PIC371181 ID: 1

Display name: PI-PIC371181

Log on as:
☐ NT Service\PIC371181
☒ [Domain]\UserName

UserName: user_domain\user_name
 Password:
 Confirm password:
 Dependencies: tcpip

This figure shows an example of how to populate the configuration boxes to allow the interface service to use a *user-specified Windows* account.

- **UserName**

The *UserName:* box allows the entry of the Windows User account to be used by the interface service.

- **Password**

If a Windows User account is entered into the *UserName:* box, the password for the entered account must be provided in the *Password:* box, unless the account requires no password.

- **Confirm password**

If a password is entered into the *Password:* box, it must be confirmed in the *Confirm Password:* box.

Local System account

Select the *[Domain]\UserName* option and populate the *UserName* box with the value *LocalSystem* to configure the interface service as a *Local System* account. Leave the *Password:* and *Confirm password:* boxes empty.

The 'Service Configuration' dialog box is shown with the following fields and values:

- Service name: PIC371181
- ID: 1
- Display name: PI-PIC371181
- Log on as: ☐ NT Service\PIC371181, ☒ [Domain]\UserName
- UserName: LocalSystem
- Password: (empty)
- Confirm password: (empty)
- Dependencies: tcpip

This figure shows an example of how to populate the configuration boxes to configure the interface service to run as a *Local System* account.

After the interface service has been created to use the *LocalSystem* account, the *Service Configuration / Log on as:* area is redisplayed with a different dialog view.

The 'Service Configuration' dialog box is shown with the following fields and values:

- Service name: PIC371181
- ID: 1
- Display name: PI-PIC371181
- Log on as: ☒ LocalSystem, ☐ [Domain]\UserName
- Dependencies: tcpip

This figure shows how the *Service Configuration / Log on as:* area is redrawn when the interface service is configured to use the *Local System* account.

The screenshot shows the Windows Services console with the following table of services:

Name	Description	Status	Startup Type	Log On As
PI-PIC371181	OSIsoft PIC371181 Interface Service	Started	Automatic	Local System
Plug and Play	Enables a computer to recognize and ...	Started	Automatic	Local System

This figure shows the interface after it has been created to use the *Local System* account.




Security Note: For best security, we recommend running this interface service under an account with minimum privileges, such as a Windows built-in service virtual account, the built-in Network Service account, or a non administrative account that you create.


PI ICU versions earlier than 1.4.14.x cannot create a service that runs as a Windows built-in service virtual account or the built-in Network Service or Local Service accounts. After ICU creates the interface service, you can change the account with a Windows administrative tool, such as *Services on the Control Panel* or the *sc* command-line utility.

Dependencies

The *Installed services* list is a list of the services currently installed on this machine.

Services upon which this interface is dependent should be moved into the *Dependencies:* list

using the **Add** button, . For example, if API Buffering is enabled, then *Bufserv* or *PIBufss* should be selected from the list at the right and added to the list on the left. To

remove a service from the list of dependencies, use the **Remove** button, , and the service name will be removed from the *Dependencies:* list.

When the interface is started (as a service), the services listed in the dependency list will be verified as running (or an attempt will be made to start them). If the dependent service(s) cannot be started for any reason, then the interface service will not run.

Note: Please see the PI Log and Windows Event Logger for messages that may indicate the cause for any service not running as expected.



- Add Button

To add a dependency from the list of *Installed services:*, select the dependency name and click the *Add* button.



- Remove Button

To remove a dependency from the list of *Dependencies:*, highlight the service name in the *Dependencies* list and click the *Remove* button.

The full name of the service selected in the *Installed services:* list is displayed below the *Installed services:* list box.

Startup Type

The *Startup Type* indicates whether the interface service will start automatically or must be started manually upon reboot.

- If the *Auto* option is selected, the service will be installed to start automatically when the machine reboots.
- If the *Manual* option is selected, the interface service will not start on reboot, but will require manual intervention to start the service.

- If the *Disabled* option is selected, the service will not start at all.

Generally, interface services are set to start automatically.



Create

The *Create* button adds the displayed service with the specified *Dependencies* and with the specified *Startup Type*.

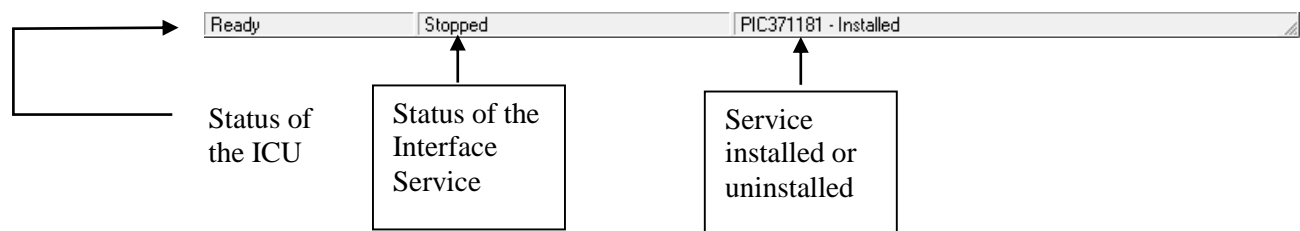
Remove

The *Remove* button removes the displayed service. If the service is not currently installed, or if the service is currently running, this button will be grayed out.

Start or Stop Service

The toolbar contains a *Start* button  and a *Stop* button . If this interface service is not currently installed, these buttons will remain grayed out until the service is added. If this interface service is running, the *Stop* button is available. If this service is not running, the *Start* button is available.

The status of the interface service is indicated in the lower portion of the PI ICU dialog.



Installing Interface Service Manually

Help for installing the interface as a service is available at any time with the command:

```
PIC37118.exe /help
```

Open a Windows command prompt window and change to the directory where the `PIC371181.exe` executable is located. Then, consult the following table to determine the appropriate service installation command.

Note: In the following Windows Service Installation Commands you may use either a slash (/) or dash (-) as the delimiter.

Windows Service Installation Commands on an Interface Node or a PI Data Archive Node with Bufserv implemented	
Manual service	PIC37118.exe /install /depend "tcpip bufserv"
Automatic service	PIC37118.exe /install /auto /depend "tcpip bufserv"
*Automatic service with service ID	PIC37118.exe /serviceid X /install /auto /depend "tcpip bufserv"
Windows Service Installation Commands on an Interface Node or a PI Data Archive Node without Bufserv implemented	
Manual service	PIC37118.exe /install /depend tcpip
Automatic service	PIC37118.exe /install /auto /depend tcpip
*Automatic service with service ID	PIC37118.exe /serviceid X /install /auto /depend tcpip

*When specifying service ID, the user must include an ID number. It is suggested that this number correspond to the interface ID (/id) parameter found in the interface .bat file.

Check the Microsoft Windows Services control panel to verify that the service was added successfully. The services control panel can be used at any time to change the interface from an automatic service to a manual service or vice versa.

The service installation commands in this section create an interface service that runs under a low-privilege built-in account. On Windows 7 and Server 2012 and later, the service logs on as the service virtual account. For earlier versions of Windows, the service logs on as Network Service.



Security Note: For best security, we recommend running this interface service under an account with minimum privileges, such as a Windows service virtual account, the built-in Network Service account, or a non administrative account that you create.

As discussed earlier, following the recommendation to run the interface service under a low-privilege account may affect performance counters.

The services control panel can change the account that the interface service runs under. Changing the account while the interface service is running does not take effect until the interface service is restarted.

Chapter 5. Digital States

For more information regarding Digital States, refer to the PI Data Archive documentation.

Digital State Sets

PI digital states are discrete values represented by strings. These strings are organized in the PI Data Archive as digital state sets. Each digital state set is a user-defined list of strings, enumerated from 0 to n to represent different values of discrete data. For more information about PI digital tags and editing digital state sets, see the PI Data Archive manuals.

An interface point that contains discrete data can be stored in the PI Data Archive as a digital point. A digital point associates discrete data with a digital state set, as specified by the user.

Interface Specific Digital Sets

If quality tags are being used, quality point specific digital sets must be created. Example structure files are provided to assist in the configuration. Configuration files for the following digital state sets are provided.

- PIC37118_CompositeQual
- PIC37118_ConfigChange
- PIC37118_ConnectState
- PIC37118_DataSorting
- PIC37118_DataVaild
- PIC37118_LeapSecond
- PIC37118_NominalFreq
- PIC37118_PMUError
- PIC37118_SyncError
- PIC37118_Timelock
- PIC37118_TimeQuality
- PIC37118_Trigger

The example structure files provided can be used to import the sets into PI using SMT. They are located in the `Configuration Files` subdirectory of the specified installation location.

System Digital State Set

Similar to digital state sets is the system digital state set. This set is used for all points, regardless of type, to indicate the state of a point at a particular time. For example, if the interface receives bad data from the data source, it writes the system digital state `Bad Input` to the PI point instead of a value. The system digital state set has many unused states that can be used by the interface and other PI clients. Digital States 193-320 are reserved for OSIsoft applications.

Chapter 6. PointSource

The PointSource is a unique, single or multi-character string that is used to identify the PI point as a point that belongs to a particular interface. For example, the string *Boiler1* may be used to identify points that belong to the *MyInt* interface. To implement this, the PointSource attribute would be set to Boiler1 for every PI point that is configured for the *MyInt* interface. Then, if **/ps=Boiler1** is used on the startup command-line of the *MyInt* interface, the interface will search the PI Point Database upon startup for every PI point that is configured with a PointSource of Boiler1. Before an interface loads a point, the interface usually performs further checks by examining additional PI point attributes to determine whether a particular point is valid for the interface. For additional information, see the **/ps** parameter. If the PI API version being used is earlier than 1.6.x or the PI Data Archive version is earlier than 3.4.370.x, the PointSource is limited to a single character unless the SDK is being used.

Case-sensitivity for PointSource Attribute

The PointSource character that is supplied with the **/ps** command-line parameter is not case sensitive. That is, **/ps=P** and **/ps=p** are equivalent.

Reserved Point Sources

Several subsystems and applications that ship with the PI System are associated with default PointSource characters. The Totalizer Subsystem uses the PointSource character **T**, the Alarm Subsystem uses **@** for Alarm points, **G** for Group Alarms and **Q** for SQC Alarm points, Random uses **R**, RampSoak uses **9**, and the Performance Equations Subsystem uses **C**. Do not use these PointSource characters or change the default point source characters for these applications. Also, if a PointSource character is not explicitly defined when creating a PI point; the point is assigned a default PointSource character of **Lab** (PI 3). Therefore, it would be confusing to use **Lab** as the PointSource character for an interface.

Note: Do not use a point source character that is already associated with another interface program. However it is acceptable to use the same point source for multiple instances of an interface.

Chapter 7. PI Point Configuration

The PI point is the basic building block for controlling data flow to and from the PI Data Archive. A single point is configured for each measurement value that needs to be archived.

Point Attributes

Use the point attributes below to define the PI point configuration for the interface, including specifically what data to transfer.

This document does not discuss the attributes that configure UniInt or PI Data Archive processing for a PI point. Specifically, UniInt provides exception reporting and the PI Data Archive or PIBufss provides data compression. Exception reporting and compression are very important aspects of data collection and archiving, which are not discussed in this document.

Note: See the *PI Universal Interface (UniInt) User Guide* and PI Data Archive documentation for information on other attributes that are significant to PI point data collection and archiving.

Tag

The Tag attribute (or tag name) is the name for a point. There is a one-to-one correspondence between the name of a point and the point itself. Because of this relationship, PI System documentation uses the terms “tag” and “point” interchangeably.

Follow these rules for naming PI points:

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

Length

Depending on the version of the PI API and the PI Data Archive, this interface supports Tag attributes whose length is at most 255 or 1023 characters. The following table indicates the maximum length of this attribute for all the different combinations of PI API and PI Data Archive versions.

PI API	PI Data Archive	Maximum Length
1.6.0.2 or later	3.4.370.x or later	1023
1.6.0.2 or later	Earlier than 3.4.370.x	255
Earlier than 1.6.0.2	3.4.370.x or later	255
Earlier than 1.6.0.2	Earlier than 3.4.370.x	255

If the PI Data Archive version is earlier than 3.4.370.x or the PI API version is earlier than 1.6.0.2, and you want to use a maximum Tag attribute length of 1023, you need to enable the PI SDK. See [Appendix B. PI SDK Options](#) for information.

PointSource

The PointSource attribute contains a unique, single or multi-character string that is used to identify the PI point as a point that belongs to a particular interface. For additional information, see the `/ps` command-line parameter and the [PointSource](#) chapter.

PointType

Typically, device point types do not need to correspond to PI point types. For example, integer values from a device can be sent to floating point or digital PI tags. Similarly, a floating-point value from the device can be sent to integer or digital PI tags, although the values will be truncated.

Float16, float32, float 64, int16, int32, digital, string, and blob point types are supported. For more information on the individual point types, see the PI Data Archive manuals.

Location1

Location1 indicates to which copy of the interface the point belongs. The value of this attribute must match the `/id` command-line parameter.

Location2

Location2 specifies the ID of the LocalEndPoint. This corresponds to the `ID` parameter of the corresponding LocalEndPoint section in the interfaces XML configuration file. This attribute is used by the interface to uniquely identify the LocalEndPoint object and does not correspond to any C37.118 attributes.

Note: The ID Must Be Unique among all LocalEndPoint sections in the configuration file.

Location3

Location3 specifies the ID of the RemoteEndPoint. This corresponds to the IdCode attribute of the corresponding RemoteEndPoint section in the interfaces XML configuration file and must match the configured IDCODE on the PDC / PMU hardware. If the C37.118 device is a PDC there will most likely be multiple PMUs. This attribute should match the IDCODE field for the PDC and not the PMU.

Location4

The C37.118 interface is exception based only. Location 4 should be set to zero for all points.

Location5

Location5 specifies the PMU ID. This attribute must match the IDCODE field for the PMU in the C37.118 CONFIG2 message as well as the IdCode attribute in the PMU section of the XML configuration file. The value for this will be determined when configuring PDC or PMU. If the C37.118 device is a PMU, versus a PDC, this value will typically match the IDCODE for the RemoteEndPoint specified in location3.

If the device is a PDC, each PMU being collected by the PDC must have a unique IDCODE field configured.

InstrumentTag

The InstrumentTag attribute specifies the specific C37.118 measurement to be stored in the PI Data Archive. The attribute contains two elements. The first is the Measurement Type and the second is the specific channel name, REP or PMU measurement and is in the format of **MeasurementType\Measurement**.

Length

Depending on the version of the PI API and the PI Data Archive, this interface supports an InstrumentTag attribute whose length is at most 32 or 1023 characters. The following table indicates the maximum length of this attribute for all the different combinations of PI API and PI Data Archive versions.

PI API	PI Data Archive	Maximum Length
1.6.0.2 or later	3.4.370.x or later	1023
1.6.0.2 or later	Earlier than 3.4.370.x	32
Earlier than 1.6.0.2	3.4.370.x or later	32
Earlier than 1.6.0.2	Earlier than 3.4.370.x	32

If the PI Data Archive version is earlier than 3.4.370.x or the PI API version is earlier than 1.6.0.2, and you want to use a maximum InstrumentTag length of 1023, you need to enable the PI SDK. See [Appendix B. PI SDK Options](#) for information.

Measurement Types

There are five primary Measurement Types that the interface supports. They are:

- Phasor – The interface will read one of the C37.118 Phasor channels. There are four subtypes for Phasor measurements.
- Analog – The interface will read one of the C37.118 Analog channels
- Digital – The interface will read one of the C37.118 Digital channels.
- PMU – PMU measurement types are those which are not Phasor, Analog or Digital and are related to the PMU. These include PMU values such Frequency (FREQ) and Frequency rate of Change (DFREQ), Configuration data such as the data scheduling time base (TIME_BASE) field from the C37.118 CONFIG message and interface configuration data from the XML configuration file such as the PMU Latitude and Longitude.
- REP – REP measurement types are those related to the RemoteEndPoint. These include connection status as well as time quality.

For Phasor measurement types, the interface can return four different measurements: Real, Imaginary, Magnitude and Phase Angle.

The following table illustrates the valid MeasurementType keywords for use in the InstrumentTag attribute. The keywords are not case sensitive.

Keyword	Description
PhasorReal	Returns the Real measurement for the specified Phasor Channel.
PhasorImag	Returns the Imaginary measurement for the specified Phasor Channel.
PhasorMag	Returns the Magnitude measurement for the specified Phasor Channel.
PhasorAngle	Returns the Phase Angle measurement for the specified Phasor Channel
Analog	Returns the measurement from the specified Analog Channel.
Digital	Returns the Digital measurement from the specified Digital Channel.
PMU	The interface returns one of the PMU measurements.
REP	Remote End Point related tag

Measurements

For Phasor, Analog and Digital Measurement types, the user must specify the C37.118 channel name from which the interface will collect data. Channel names vary by vendor.

For the PMU and REP Measurement types, the user must enter one of the PMU or REP Measurement keyword described below.

The C37.118 specification defines the channel names to be a 16-byte ASCII string that can have embedded as well as trailing spaces. The interface strips any trailing spaces but leaves any embedded spaces. The interface also treats the strings as case insensitive. If for example a PMU has a channel name of 'Watts CH-A ', the user would specify 'Watts CH-A' for the Measurement element of the InstrumentTag ('watts ch-a' would also be valid in this example).

For PMU MeasurementTypes tags, the interface will store various data from the C37.118 data stream, CONFIG1 and CONFIG2 blocks and configuration data from the XML configuration file. The follow table shows the valid values for the Measurement field when the MeasurmentType field is PMU.

Keyword	Description
FREQ	System frequency (FREQ) from C37.118 data stream
DFREQ	Frequency Rate of Change (DFREQ) from C37.118 data stream.
FREQERR	Frequency error calculated from FREQ – Nominal Frequency as defined by FNOM (50 or 60).
CFGCNT	Configuration change count (CFGCNT) from last CONFIG block
CONFIGFLAG	Digital tag that is set to 1 when the configuration change flag (bit 10) in the STAT word is true.
FNOM	Nominal line frequency (FNOM) from last CONFIG block
DATA_RATE	Date rate (DATA_RATE) from last CONFIG block
LATITUDE	PMU Latitude (-90.0-90.0) from XML configuration file
LONGITUDE	PMU Longitude (-180.0-180.0) from XML configuration file
TIMELOCK	Digital tag receives the Unlocked Time quality bits (5-4) from the STAT word. A sample digital set definition file PIC37118_Timelock.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
TRIGGER	Digital tag receives the Trigger Reason bits (3-0) from the STAT word. A sample digital set definition file PIC37118_Trigger.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
DATAVALID	Digital tag receives the Data Validity bit (15) from the STAT word. A sample digital set definition file PIC37118_DataValid.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
PMUERROR	Digital tag receives the PMU Error bit (14) from the STAT word. A sample digital set definition file PIC37118_PMUError.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
SYNCERROR	Digital tag receives the PMU Sync Error bit (13) from the STAT word. A sample digital set definition file PIC37118_SyncError.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
DATASORTING	Digital tag receives the Data Sorting bit (12) from the STAT word. A sample digital set definition file PIC37118_DataSorting.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
COMPOSITEQUAL	Digital tag receives a composite status from the Data Valid, PMU Error, Sync Error and Data Sorting flags. If any of these flags are set, the value of this tag will be set to 1. Otherwise it will be zero. This tag is intended to be used to enable the user to make a single check for any quality related issues and then do further checking only if needed. A sample digital set definition file PIC37118_CompositeQuality.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.

The following table lists measurement values for PMU measurement type tags.

Keyword	Description
TIMEQUAL	Digital tag receives the Qime Quality bits (3-0) from the time quality flags. A sample digital set definition file PIC37118_TimeQuality.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
LEAPSEC	Digital tag receives the Leap Second quality bits (6-4) from the time quality flags. A sample digital set definition file PIC37118_LeapSecond.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
CONNECTSTATE	Digital tag receives the current connection state. A sample digital set definition file PIC37118_ConnectState.csv is provided with the interface to assist in creating the required digital state set for this tag. This file can be used to import the state set into PI using the Digital States Plug-In in SMT.
EVENTSEQ	Integer tag intended for diagnostic purposes to verify that the interface is not losing messages. It records the sequence number of the message within a second. For example: If the update rate for the PMU is set to 30hz, the expected values would cycle from 0 to 29 without missing any values.

ExDesc

This attribute is not used by the PI C37.118 interface.

Scan

By default, the Scan attribute has a value of 1, which means that scanning is turned on for the point. Setting the scan attribute to 0 turns scanning off. If the scan attribute is 0 when the interface starts, a message is written to the log and the point is not loaded by the interface. There is one exception to the previous statement:

If any PI point is removed from the interface while the interface is running (including setting the scan attribute to 0), `SCAN OFF` will be written to the PI point regardless of the value of the Scan attribute. Two examples of actions that would remove a PI point from an interface are to change the point source or set the scan attribute to 0. If an interface-specific attribute is changed that causes the point to be rejected by the interface, `SCAN OFF` will be written to the PI point.

Shutdown

The Shutdown attribute is 1 (true) by default. The default behavior of the PI Shutdown subsystem is to write the `SHUTDOWN` digital state to all PI points when PI is started. The timestamp that is used for the `SHUTDOWN` events is retrieved from a file that is updated by the Snapshot Subsystem. The timestamp is usually updated every 15 minutes, which means that the timestamp for the `SHUTDOWN` events will be accurate to within 15 minutes in the event of a power failure. For additional information on shutdown events, refer to PI Data Archive 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 when the `/stopstat=Shutdown` command-line parameter is specified.

`SHUTDOWN` events can be disabled from being written to PI points when the PI Data Archive is restarted by setting the Shutdown attribute to 0 for each point. Alternatively, the default behavior of the PI Shutdown Subsystem can be changed to write `SHUTDOWN` events only for PI points that have their Shutdown attribute set to 0. To change the default behavior, edit the `\PI\dat\Shutdown.dat` file, as discussed in PI Data Archive manuals.

Bufserv and PIBufss

It is not recommended to write shutdown events when buffering is being used. Bufserv and PIBufss are utility programs that provide the capability to store and forward events to a PI Data Archive, allowing continuous data collection when the PI Data Archive is down for maintenance, upgrades, backups, or unexpected failures. That is, when the PI Data Archive is shut down, Bufserv or PIBufss will continue to collect data for the interface, making it undesirable to write `SHUTDOWN` events to the PI points for this interface. Disabling Shutdown is recommended when sending data to a Highly Available PI Data Archive Collective. Refer to the Bufserv or PIBufss manuals for additional information.

DataSecurity

The PI identity in the PI trust that authenticates the interface must be granted read access by the DataSecurity attribute of every PI point that the interface services. If the interface is used *without* a buffering application, write access also must be granted. (If the interface is used with a buffering application, the buffering application requires write access but the interface does not.)

PtSecurity

The PI identity in the PI trust that authenticates the interface must be granted read access by the PtSecurity attribute of every PI point that the interface services.

Chapter 8. Startup Command File

Command-line parameters can begin with a / or with a -. For example, the `/ps=M` and `-ps=M` command-line parameters are equivalent.

Command file names have a .bat extension. The Windows continuation character (^) allows for the use of multiple lines for the startup command. The maximum length of each line is 1024 characters (1 kilobyte). The number of parameters is unlimited, and the maximum length of each parameter is 1024 characters.

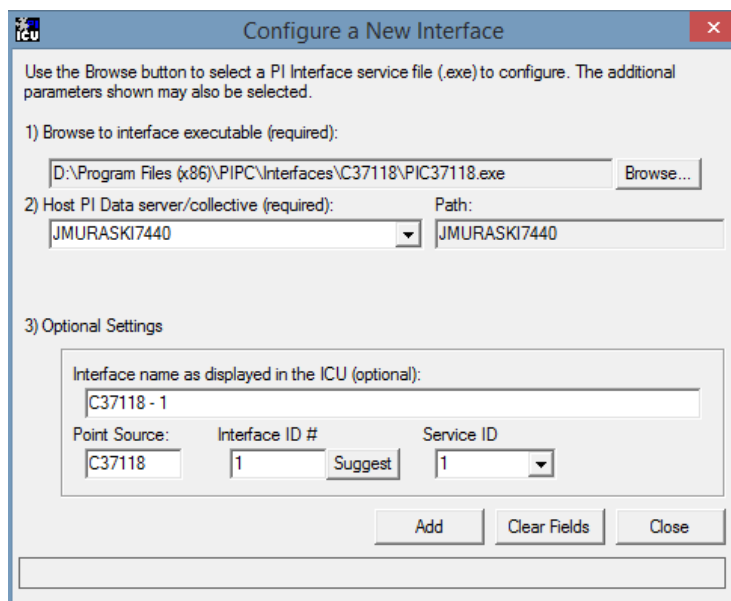
The PI Interface Configuration Utility (PI ICU) provides a tool for configuring the interface startup command file.

Configuring the Interface with PI ICU

Note: PI ICU requires PI 3.3 or later.

The PI Interface Configuration Utility provides a graphical user interface for configuring PI interfaces. If the interface is configured by the PI ICU, the batch file of the interface (`PI_C37118.bat`) will be maintained by the PI ICU and all configuration changes will be kept in that file and the module database. The procedure below describes the necessary steps for using PI ICU to configure the C37118 interface.

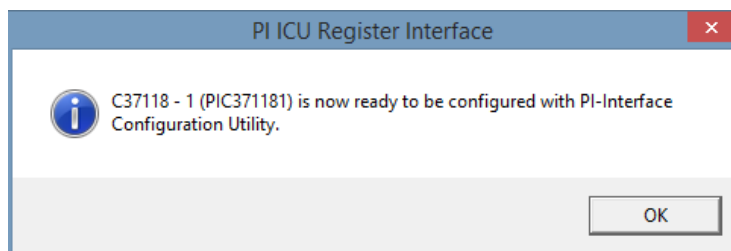
From the PI ICU menu, select *Interface*, then *NewWindows Interface Instance from EXE...*, and then *Browse* to the `PI_C37118.exe` executable file. Then, enter values for *Host PI System*, *Point Source* and *Interface ID#*. A window such as the following results:



Interface name as displayed in the ICU (optional) will have PI- pre-pended to this name and it will be the display name in the services menu.

Click *Add*.

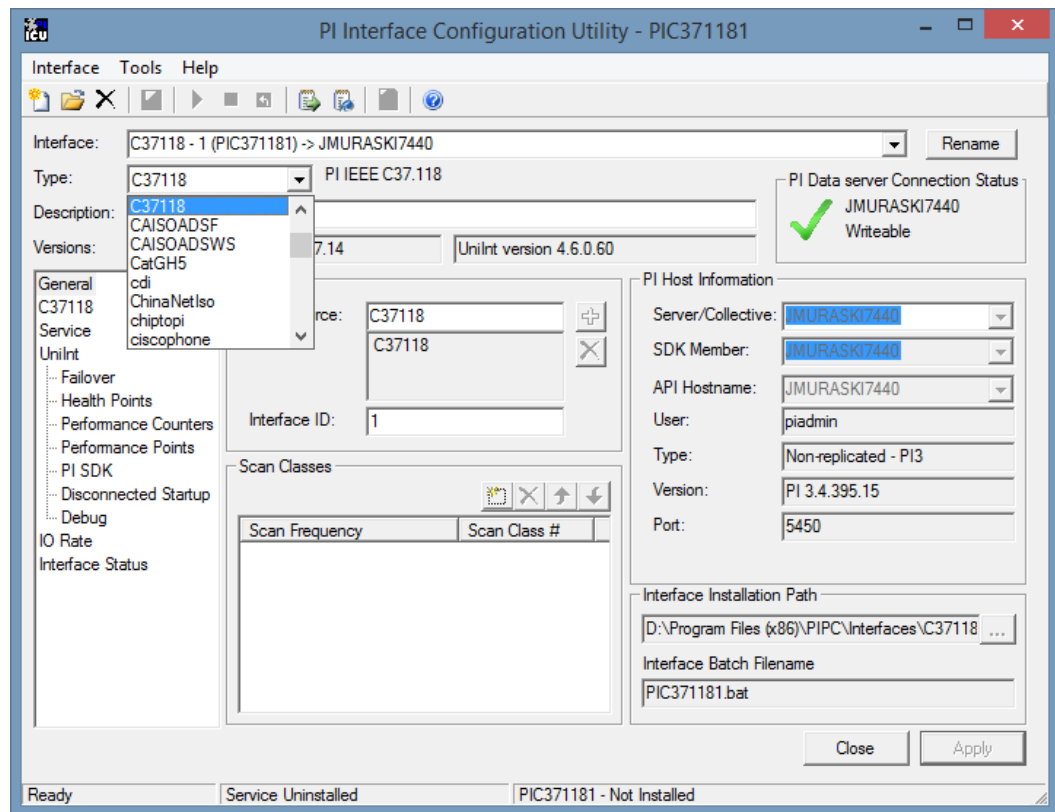
The following message displays:



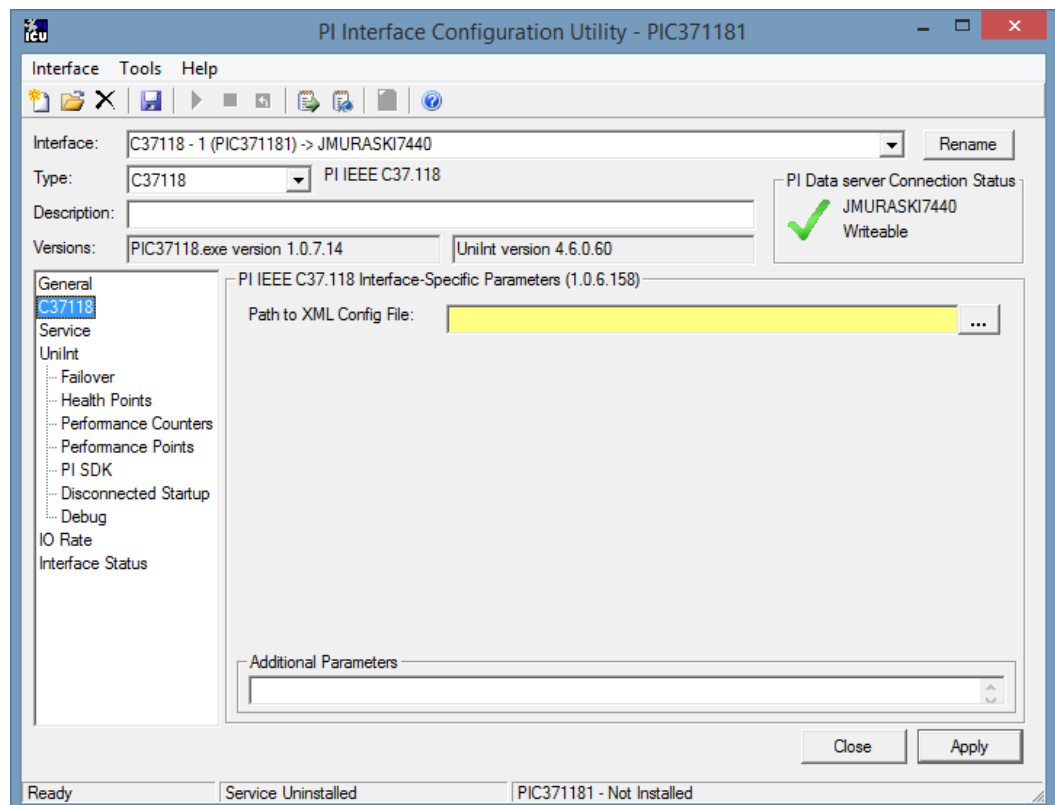
Note that in this example the host PI Data server is JMURASKI7440. To configure the interface to communicate with a remote PI Data server, select *Connections...* on the PI ICU *Interface* menu and select the default server. If the remote node is not present in the list of servers, it can be added.

After the interface is added to PI ICU, near the top of the main PI ICU screen, the *Interface Type* should be C37118. If not, use the drop-down box to change the *Interface Type* to be C37118.”

Click *Apply* to enable the PI ICU to manage this copy of the C37118 interface.



The next step is to make selections in the interface-specific tab (i.e. “C37118”) that allow the user to enter values for the startup parameters that are particular to the C37118 interface.



Since the C37118 interface is a UniInt-based interface, in some cases the user will need to make appropriate selections in the *UniInt* page. This page allows the user to access UniInt features through the PI ICU and to make changes to the behavior of the interface.

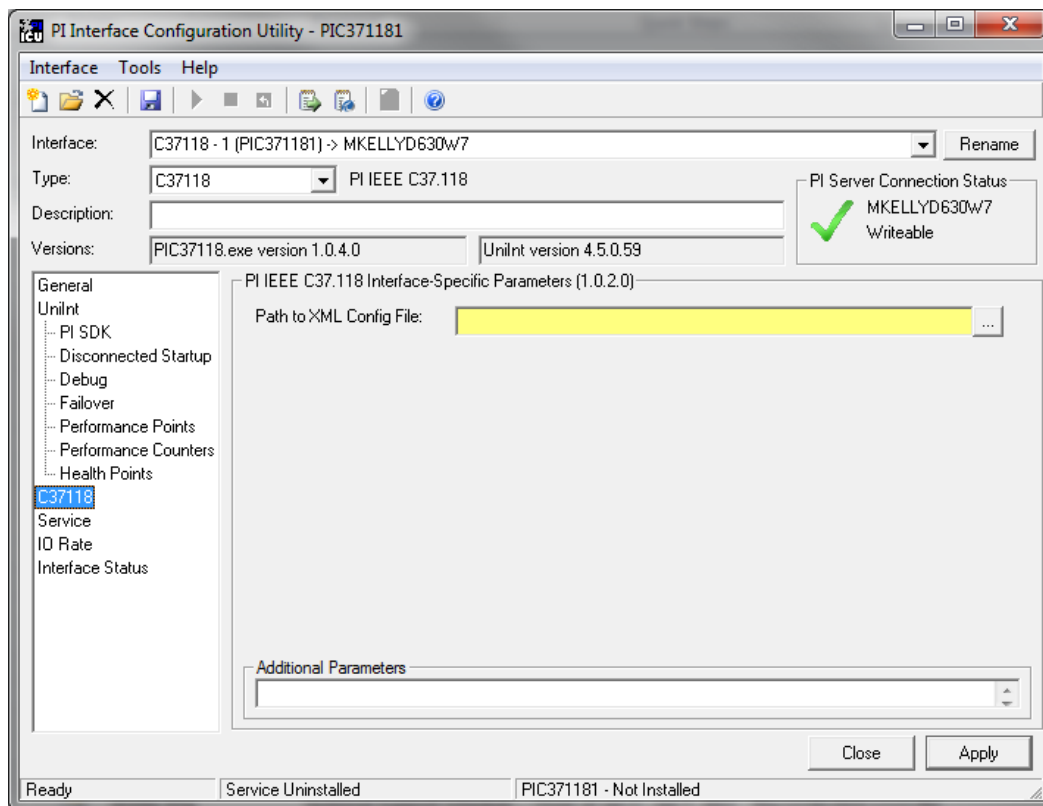
To set up the interface as a Windows Service, use the *Service* page. This page allows configuration of the interface to run as a service as well as to starting and stopping of the interface service. The interface can also be run interactively from the PI ICU. To do that, select *Start Interactive* on the *Interface* menu.

For more detailed information on how to use the above-mentioned and other PI ICU pages and selections, please refer to the *PI Interface Configuration Utility* user guide. The next section describes the selections that are available from the *C37118* page. Once selections have been made on the PI ICU GUI, press the *Apply* button in order for PI ICU to make these changes to the interface's startup file.

C37118 Interface page

Since the startup file of the C37118 interface is maintained automatically by the PI ICU, use the *C37118* page to configure the startup parameters; you should avoid making changes to the file manually. The following is the description of interface configuration parameters that are used in the PI ICU Control and the corresponding manual parameters.

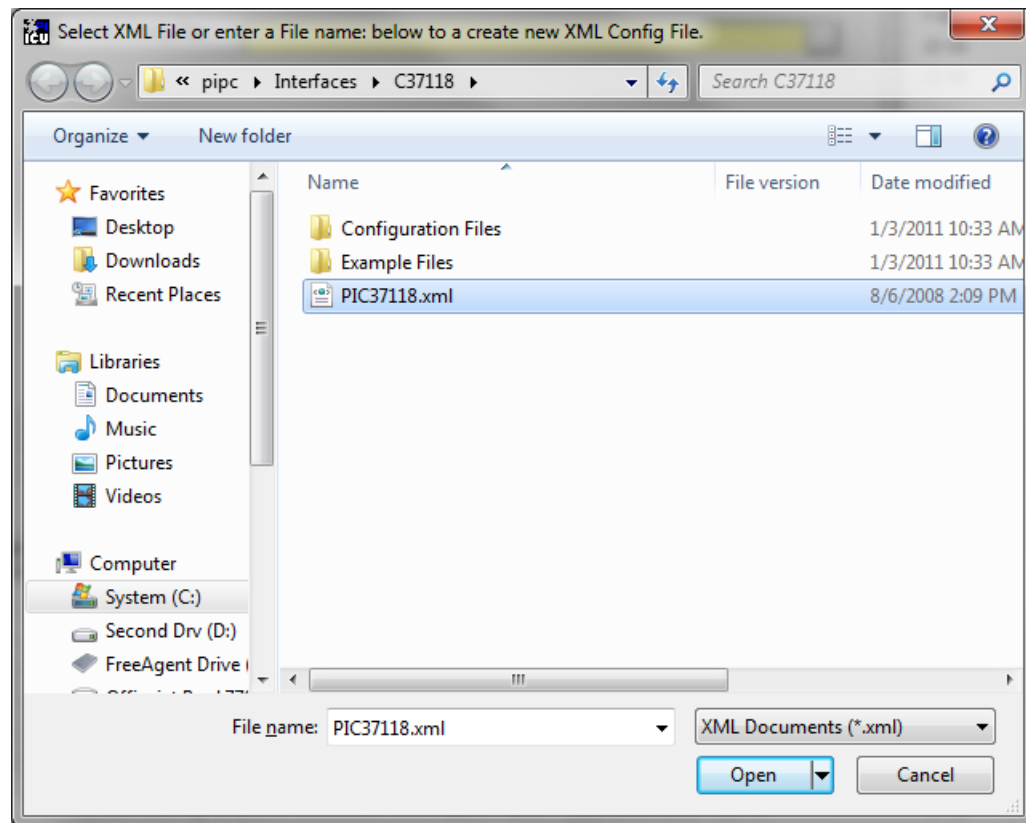
C37118



The PI Interface for IEEE C37.118 - ICU has one tab. A yellow text box indicates that an invalid value has been entered, or that a required value has not been entered.

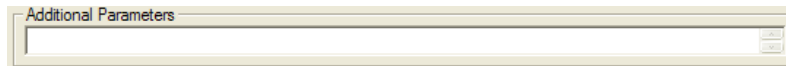
Path to XML Config File

This box is used to specify the path and filename of the XML Configuration file to be read by the interface. If the path contains spaces, then the path and filename should be enclosed in quotation marks. To enter a <UNC Path> click the Browse button and choose an existing file from the resulting dialog box (shown below) or enter a new filename in the File name: text box at the bottom of the screen (**/CONFIGFILE=<UNC Path>**)

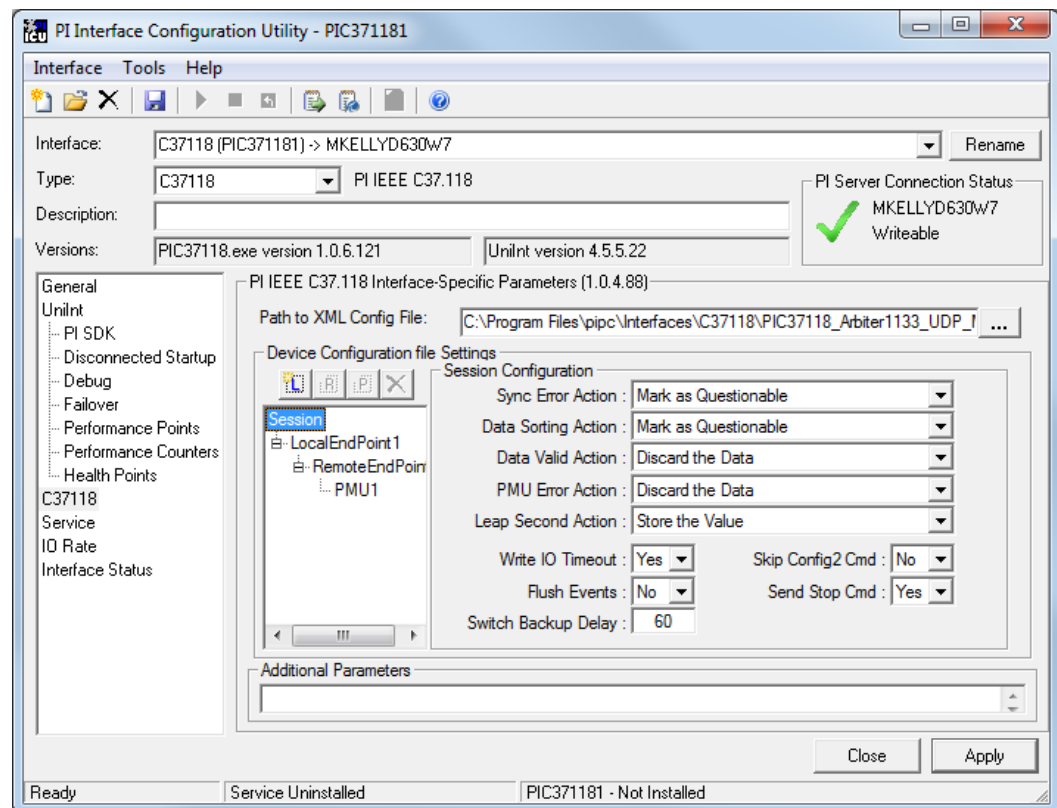


Additional Parameters

This section is provided for any additional parameters that the current ICU Control does not support.



XML Parameters



Session

The session section contains configuration parameters that control the behavior of the interface when various error conditions occur.

The following figure illustrates the Session section of the C37.118 ICU.

The screenshot shows a 'Session Configuration' window with the following settings:

Sync Error Action :	Store the Value
Data Sorting Action :	Mark as Questionable
Data Valid Action :	Store Corresponding System Digital State
PMU Error Action :	Discard the Data
Leap Second Action :	Mark as Questionable
Write IO Timeout :	Yes
Flush Events :	Yes
Switch Backup Delay :	125
Skip Config2 Cmd :	No
Send Stop Cmd :	No

- Sync Error Action – Controls the behavior of the interface when the SYNC flag is set in the C37.118 data stream. (SyncErrorAction=#, Default: 1 (Mark as Questionable), Range 0-3)
- Data Sorting Action – Controls the behavior of the interface when the Data Sorting flag is set in the C37.118 data stream. This flag is set on in a PDC data stream when a PMU loses time synchronization making it impossible for the PDC to time correlate the Phasor data. (DataSortingAction=#, Default: 1 (Mark as Questionable), Range 0-3)
- Data Valid Action – Controls the behavior of the interface when the Data Valid flag is set in the C37.118 data stream. This flag is set in a PDC data stream when the PDC loses connection to a PMU. When this happens, the PDC typically zero fills the data. (DataValidAction=#, Default: 3 (Discard the Data), Range 0-3)
- PMU Error Action – Controls the behavior of the interface when the PMU Error flag is set in the C37.118 data stream. This flag is set when an error occurs on a PMU and indicates a possible error in the data. (PMUErrorAction=#, Default: 3 (Discard the Data), Range 0-3)
- Leap Second Action – Controls the behavior of the interface when the Leap Second flag is set in the C37.118 data stream. This flag indicates that a leap second has occurred. (LeapSecondAction=#, Default: 0 (Store the Value), Range 0-3)
- Write IO Timeout – If set to yes, the interface will write the system digital state of I/O Timeout when no C37.118 data messages are received within the timeframe specified in the LocalEndPoint section. (WriteIOTimeout=#, Default: 1 (Yes), Range:0-1)
- Skip CONFIG2 Command – If set to yes, the interface will send CONFIG2 command to PDC /PMU only during startup. Otherwise, the interface will send CONFIG2 request each time when it is trying to reconnect to PDC / PMU. (SkipConfig2Cmd=#, Default: 0 (No), Range:0-1)
- Flush Events – If set to yes, the interface will flush all data to the PI Data Archive after each C37.118 data messages received from PDC /PMU. Otherwise, it will flush

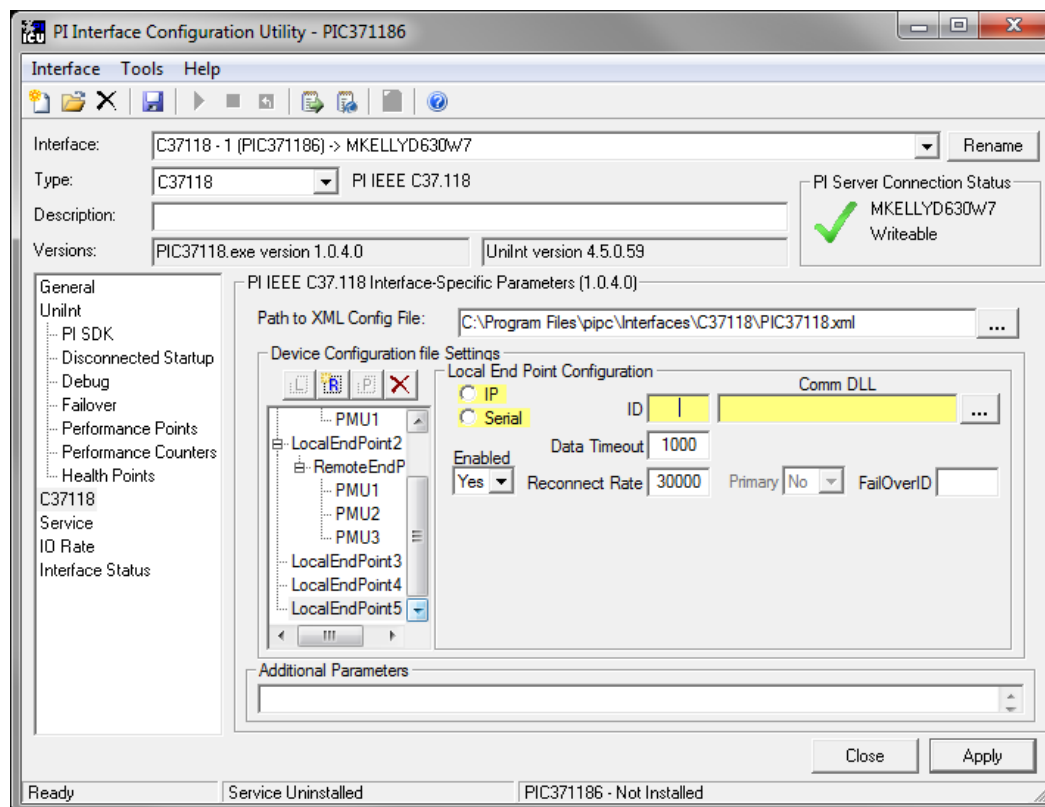
data when buffer gets full (192 records) (FlushEvents=#, Default: 0 (No), Range:0-1).

- Send Stop Data Command – If set to yes, the interface will send DATA_STOPCOMMAND to the PDC / PMU when user shutdown the interface (SendStopCmd=#, Default: 0 (NO), Range:0-1)
- Switch To Backup Delay – Controls the behavior of the interface for the PDC / PMU Data Source Level Failover. If the interface is not able to set connection and receive data from primary PDC / PMU during this period of time it will switch and try to connect to the backup PDC / PMU. If it loses connection to backup and is not able to reconnect to backup PDC / PMU again during this period of time it will try to reconnect to the primary PDC / PMU again. (SwitchToBackupDelay=#, Default: 60 sec (Store the Value), Range 30-600)

The available selections for Sync Error, Data Sorting, Data Valid, PMU Error and Leap Second Actions above are as follows.

- 0 (Store the Value) – The interface will store the value normally.
- 1 (Mark as Questionable) – The interface will store the value but mark it as questionable.
- 2 (Store the Corresponding System Digital State) – The interface will store the corresponding system state.
- 3 (Discard the Data) – The interface will not store the value in the PI Data Archive.

LocalEndPoint



The LocalEndPoint section defines parameters specific to the interface side of a PDC / PMU connection.

The following illustrates the format of the LocalEndPoint section of the C37.118 ICU.

Local End Point Configuration

☒ IP ☐ Serial

ID: 1

Comm DLL: [Yellow field with browse button ...]

Enabled: Yes

Data Timeout: 1000

Reconnect Rate: 30000

Primary: No

FailOverID: [Empty field]

- IP – Identifies this LocalEndPoint as using a TCP connection (Type=<string>, Range: IP or Serial)
- Serial – Identifies this LocalEndPoint as using a Serial connection (Type=<string>, Range: IP or Serial)
- ID – Specifies the numeric ID of the LocalEndPoint. The ID uniquely identifies the LocalEndPoint when configuring tags for an interface. The ID must be unique among all LocalEndpoints. (Id=#, Default:None, Range: 0-65535, Required)
- Comm Dll – Defines the Communications Plug-In to be used to communicate with the PDC / PMU. Due to the variation in application level functionality from one device vendor to another, the plug-in approach allows for new devices to be easily supported. Refer to [Appendix C. Communication Plug-In Selection](#) in this manual or to the OSIsoft support site for details on available plug-in's.
- Data Timeout – Controls the length of time to wait, in milliseconds, for data responses. The recommended value is 5 times the configured PMU data rate. (DataTimeout=#, Default:1000, Range: 10-65,535)
- Enabled – Allows the LocalEndPoint and everything associated with it to be disabled and not loaded by the interface. It can then be re-enabled with having to reconfigure. (Enabled=#, Default: 1 (Yes), Range: 0-1)
- Reconnect Rate – Controls the rate in which the interface attempts to re-establish communications with the PDC / PMU. (ReconnectRate=#, Default:30000, Range: 10-65,535)
- Primary – Allows to specify which LocalEndPoint in the redundant pair becomes a primary (value should be 1). For another LocalEndPoint this field should have opposite value (0) to become a backup. Use this field to setup Data Source Level Failover. (Primary=#, Default: 0 (No), Range: 0-1)
- Failover ID – Specifies the numeric Failover ID of the LocalEndPoint. The Failover ID allows creating redundant pair of primary and secondary LocalEndpoints for Data Source Level Failover. For example, if ID for LocalEndPoint is 1and Failover ID is 2 then another for LocalEndPoint with ID 2 and Failover ID 1 has to be created. In this case LocalEndPoint 1 and 2 becomes redundant pair. You should specify which

LocalEndPoint is primary using Primary field. Only one LocalEndPoint in the redundant pair could be a primary. Another one becomes a secondary. If no need to setup Data Source Level Failover for this LocalEndPoint leave this field blank. Use this field only in case to setup Data Source Level Failover. (FailOverId=#, Default:None, Range: 0-65535, Optional)

IP Configuration

The following controls are available when Type="IP" is selected.

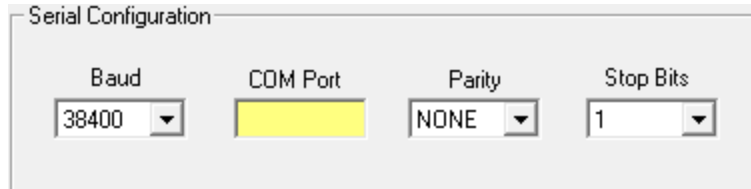
The screenshot shows a dialog box titled "IP Configuration". It has two rows of controls. The first row has a "Cmd" label, a dropdown menu set to "TCP", and a text box containing "0". To the right of this row are two checkboxes: "Use Local IP Address" and "Use Multicast IP Addr", both of which are unchecked. The second row has a "Data" label, a dropdown menu set to "UDP", and a text box containing "0". To the right of this row is a checkbox labeled "Src Specific MultiCast", which is also unchecked. Below the "Data" dropdown is a label "Close Command Socket" followed by a dropdown menu set to "Yes".

- Command Protocol – Specifies the protocol to use for C37.118 Command and Configuration messages. (CommandProtocol=<type>, Default: TCP, Range: TCP or UDP)
- Local Command Port – Specifies the local port to listen on for configuration packets. If not specified the interface will not specify a specific port (system assigned). This parameter will typically be used when the command protocol is UDP in order to facilitate configuration with firewalls. If this value is specified it must be different than the Local Data Port value. (LocalCommandPort=#, Default: 0 (System Assigned), Range: 1024-65,535)
- Data Protocol – Specifies the protocol to use for C37.118 data messages. (DataProtocol=<type>, Default: UDP, Range: TCP or UDP)
- Local Data Port – Specifies the local port to listen on for data packets. If not specified the interface will not specify a specific port (system assigned). This parameter will typically be used when the data protocol is UDP in order to facilitate configuration with firewalls. For some PDC / PMU configurations in which the device is sends to a predefined data port, this value will also need to be specified. It is also required if Multicast being used. (LocalDataPort=#, Default: 0 (System Assigned), Range: 1024-65,535)
- Close Command Socket – If Yes is specified, the interface will close the command socket (if different than the data socket) once a command is sent and the expected response is successfully received. Some devices, such as the Arbiter 1133, expect that the command socket be closed after use. Others, such as SEL devices require the socket remain open. (CloseCommandSocket=#, Default: 1 (Yes), Range: 0 (No) – 1 (Yes))
- Use Local IP Address – This parameter could be used if interface node has multiple IP addresses. When specified, the interface will be binding with that address. Otherwise the interface will be using default IP address. (LocalIPAddress=<ipaddr>, Default: None)
- Use Multicast IP Addr – This parameter is valid only when the Local Data Protocol is specified as UDP. When specified, the interface will be configured to listen for C37.118 data messages via Multicast. (MulticastIPAddress=<ipaddr>, Default: None)

- Src Specific Multicast – This parameter is valid only if the Local IP Address and Multicast IP Address fields have been populated. When specified, the interface will subscribe for multicast data from the source that has specified IP address.
(**MulticastSourceIPAddress**=<ipaddr>, Default: None)

Serial Configuration

The following controls are available when Type="Serial" is selected.



The image shows a 'Serial Configuration' dialog box with four settings: Baud (38400), COM Port (highlighted in yellow), Parity (NONE), and Stop Bits (1). Each setting is in a dropdown menu.

- Baud – Specifies speed at which to communicate with the PDC / PMU. Refer to your PDC / PMU hardware users guide for supported baud rates. (**Baud**=#, Default: 38400, Range: 300-115200)
- COM Port – The numeric value of the serial COM port to use. For example if using COM1 set this value to 1. (**Port**=#, Range: 1-Max Ports)
- Parity – Specifies the error checking parity to use. (**Parity**=#, Default: 0 (NONE), Values: 0 (NONE), 1 (ODD), 2 (EVEN), 3 (MARK), and 4 (SPACE))
- Stop Bits – Specifies the number of stop bits following a message. (**StopBits**=#, Default: 0 (1 bit), Values: 0 (1 bit), 1 (2 bits), 2 (1.5 bits))

RemoteEndPoint

The RemoteEndPoint represents the remote (PDC / PMU side) of a physical connection. Each RemoteEndPoint node will have one or more PMU child nodes. Each of the PMU nodes will represent a PMU that the PI C37.118 interface will communicate with through the physical port described by the LocalEndPoint / RemoteEndPoint child node pair.

RemoteEndPoint Child Nodes that apply to all connection types.

- ID Code – Specifies the numeric ID of the PDC or PMU. This must match the IDCODE field sent by the C37.118 device. If the device is a PDC it is important to note that this must match the IDCODE of the PDC and not the PMU(s) being serviced by the PDC. (**IdCode**=#, Range: 0-32,767)
- Description – User defined description of the RemoteEndPoint. Ie. "Substation xyz". This attribute is used by the interface when logging messages regarding the RemoteEndPoint. (**Description**=<string>)

The following illustrates the RemoteEndPoint section for an IP LocalEndPoint.

The dialog box is titled "Remote End Point Configuration". It contains the following fields and controls:

- Id Code:** A yellow rectangular field.
- Remote Data Port:** A text box containing the value "4712".
- Remote Cmd Port:** A text box containing the value "4713".
- Description:** A large empty text box.
- Host Name:** A radio button that is selected, followed by a yellow rectangular field.
- IP Address:** A radio button that is unselected, followed by a dotted decimal notation field (e.g., ". . .").

- Remote Data Port – Specifies the remote (PDC) port to be used for data packets. If not specified the interface will assume a specific port (system assigned). If this field is specified, the data being received from the PDC / PMU must originate from this port. Most PDCs and PMUs use a random system assigned port. (RemoteDataPort=#, Default: 4712, Range: 1024-65,535)
- Remote Command Port – Specifies the remote port to use for command and configuration packets. The interface will send CONFIG1, CONFIG2 and HEADER request commands to this port. (RemoteCommandPort=#, Default: 4713, Range: 1024-65,535)
- Host Name – If this control is selected the PDC / PMU addressing information is passed by host name.(HostName=<nodename> or <IPAddr>)
- IP Address – If this control is selected the PDC / PMU addressing information is passed as an IP address in dotted decimal notation. (HostName=<nodename> or <IPAddr>)

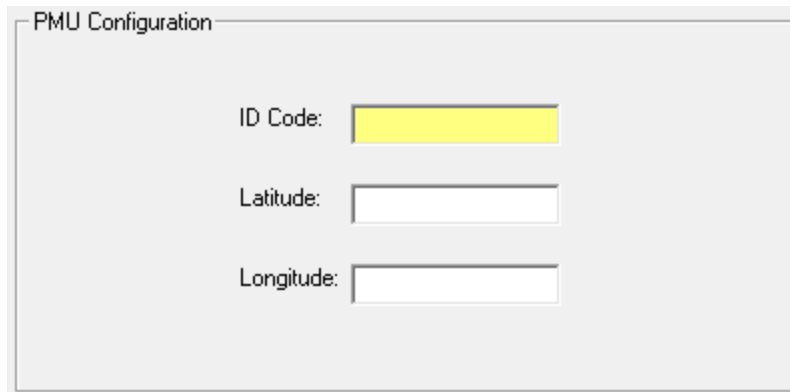
The following illustrates the RemoteEndPoint section for a Serial LocalEndPoint.

The dialog box is titled "Remote End Point Configuration". It contains the following fields and controls:

- Id Code:** A yellow rectangular field.
- Description:** A large empty text box.

PMU

The PMU section describes the characteristics of each PMU. If the RemoteEndPoint is a PDC then there may be multiple PMUs, otherwise there should be only one PMU child node per RemoteEndPoint.

A screenshot of a 'PMU Configuration' dialog box. It contains three input fields: 'ID Code' with a yellow background, 'Latitude', and 'Longitude', both with white backgrounds. The fields are arranged vertically and are currently empty.

- ID Code – Specifies the numeric ID of the PMU. This must match the IDCODE field sent by the C37.118 device. If there are multiple PMU nodes associated with a RemoteEndPoint (RemoteEndPoint is a PDC) with it this ID must be unique within the RemoteEndPoint collection and should match the PMUs IDCODE and not the IDCODE of the PDC itself. (IdCode=#, Range: 0-32,767)
- Latitude – Specifies the geographic latitude of the PMU. This field is optional. However, if the latitude is specified, then the longitude must also be specified, otherwise the interface will log an error and exit. If no sign is specified then + is assumed. (Latitude=+/-#. #, Range: -90.0 to +90.0)
- Longitude – Specifies the geographic longitude of the PMU. This field is optional. However, if the longitude is specified, then the latitude must also be specified, otherwise the interface will log an error and exit. If no sign is specified then + is assumed. (Longitude=+/-#. #, Range: -180.0 to +180.0)

PIC37118 XML Nodes, Elements, and Attributes e

Since the startup file of the C37118 interface is maintained automatically by the PI ICU, use the C37118 page to configure the startup parameters and do not make changes in the file manually. The following is the description of interface configuration parameters used in the PI ICU Control and corresponding manual parameters.

Root Node

The root node must be named PI_C37_118.Session and contains one or more LocalEndPoint leaf nodes.

PI_C37118.Session Child Nodes	Description
Example: <PI_C37_118.Session">	
WriteIOTimeout Optional XMLType: Attribute Default: 1	By default, the interface will write the digital state I/O Timeout to all measurement type tags when a communications timeout condition is encountered. Setting the WriteIOTimeout attribute to a value of 0 will cause the interface to suppress writing I/O Timeout to the measurement tags. Valid values for this attribute are 0 (No) to not write I/O Timeout or 1 (Yes) to write I/O Timeout.
PMUErrorAction Optional XMLType: Attribute Default: 3	The PMUErrorAction attribute controls the behavior of the interface when the PMU Error flag is detected in the STAT word of the message. This flag indicates that there is a problem with the PMU hardware and that the data may not be valid. The default behavior of the interface is to discard the data. Valid selections for this attribute are: 0 – Store the value normally (Not recommended) 1 – Mark the data as questionable. 2 – Stores the PMU Error system digital state for all measurement values. 3 – Discards the data.
DataValidAction Optional XMLType: Attribute Default: 3	The DataValidAction attribute controls the behavior of the interface when the Data Invalid flag is detected in the STAT word of the message. This flag indicates that there is a problem with the data for the PMU may not be valid. This flag is typically set when the data stream is coming from a PDC versus a single PMU and is usually due to the PDC not receiving a data frame from the PMU in a reasonable time. The PDC typically blank fills the data. The default behavior of the interface is to discard the data. Valid selections for this attribute are: 0 – Store the value normally (Not recommended) 1 – Mark the data as questionable. 2 – Stores the PMU Error system digital state for all measurement values 3 – Discards the data.
SyncErrorAction Optional XMLType: Attribute Default: 1	The SyncErrorAction attribute controls the behavior of the interface when the PMU Sync Error flag is detected in the STAT word of the message. This flag indicates that there is a problem with the time synchronization of the PMU and that the quality of the timestamp is other than normal. The default behavior of the interface is to mark the data as questionable. Valid selections for this attribute are: 0 – Store the value normally (Not recommended) 1 – Mark the data as questionable. 2 – Stores the PMU Error system digital state for all measurement values. 3 – Discards the data.

PI_C37118.Session Child Nodes	Description
Example: <PI_C37_118.Session">	
DataSortingAction Optional XMLType: Attribute Default: 1	<p>The DataSortingAction attribute controls the behavior of the interface when the Data Sorting flag is detected in the STAT word of the message. This error should only occur if the data stream is coming from a PDC. This flag indicates that the data for the PMU is being integrated into the data frame by arrival and not timestamp.</p> <p>The default behavior of the interface is to mark the data as questionable. Valid selections for this attribute are:</p> <ul style="list-style-type: none"> 0 – Store the value normally (Not recommended) 1 – Mark the data as questionable. 2 – Stores the PMU Error system digital state for all measurement values. 3 – Discards the data.
LeapSecondAction Optional XMLType: Attribute Default: 0	<p>The LeapSecondAction attribute controls the behavior of the interface when one of the leap second flags is detected in the STAT word of the message. This occurs when a leapsecond is occurring.</p> <p>The default behavior of the interface is to store the data normally. Valid selections for this attribute are:</p> <ul style="list-style-type: none"> 0 – Store the value normally 1 – Mark the data as questionable. 2 – Stores the PMU Error system digital state for all measurement values. 3 – Discards the data.
SkipConfig2Cmd XML Type: Attribute Default: 0	<p>The SkipConfig2Cmd attribute controls the behavior of interface to send CONFIG2 command to PDC / PMU each time when it is trying to reconnect to PDC / PMU. If set to yes, the interface will send CONFIG2 command to PDC / PMU only during startup.</p> <p>Valid values for this attribute are 0 (No) to not skip sending CONFIG2 request or 1 (Yes) to skip sending CONFIG2 request.</p>
SendStopCmd XML Type: Attribute Default: 0	<p>The SendStopCmd attribute controls the behavior of interface to send STOP_DATA command to the PDC / PMU when user shutdown the interface. It may affect other clients that connected to that PMU / PDC.</p> <p>Valid values for this attribute are 0 (No) to not send STOP_DATA command or 1 (Yes) to send STOP_DATA command when shutting down the interface.</p>
FlushEvents XML Type: Attribute Default: 0	<p>The FlushEvents attribute controls the behavior of interface to flush all data to the PI Data Archive after each C37.118 data messages received from PDC / PMU. If set to 0 (default value) it will flush data when buffer gets full (192 records).</p> <p>Valid values for this attribute are 0 (No) to not flush data or 1 (Yes) to flush all data to PI Data Archive after each data message received from PMU / PDC.</p>
SwitchToBackupDelay XML Type: Attribute Default: 60	<p>The SwitchToBackupDelay controls the behavior of the interface for the PDC / PMU Data Source Level Failover. If the interface is not able to set connection and receive data from primary PDC / PMU during this period of time it will switch and try to connect to the backup PDC / PMU. If it loses connection to backup and is not able to reconnect to backup PDC / PMU again during this period of time it will try to reconnect to the primary PDC / PMU again.</p> <p>Valid ranges: {30-600}</p>

LocalEndPoint Nodes

Child nodes of the root node are LocalEndPoint nodes. The LocalEndPoint nodes are used to specify the properties of the local (interface side) of a physical connection such as a communication port, or TCP port. The **Type** attribute must be either `Serial` or `IP`. The child nodes will depend on the value of the **Type** attribute. A serial communications port will have different child nodes from an IP port. There can only be one RemoteEndPoint Node for each LocalEndPoint Node.

LocalEndPoint Child Nodes that apply to all connection types.	Description
Example: <RemoteEndPoint>	
Type Required XML Type: Element Default: None	Specifies the communications type. Values are "IP" or "Serial"
Id Required XMLType: Attribute Default: None	Specifies the numeric ID of the LocalEndPoint. The ID uniquely identifies the LocalEndPoint when configuring tags for an interface. The ID must be unique among all LocalEndpoints for an interface. Valid ranges: {0-65535 }
Enabled Optional XML Type: Attribute Default: 1	Enabled – Allows the LocalEndPoint and everything associated with it to be disabled and not loaded by the interface. It can then be re-enabled with having to reconfigure. A value of 0 (No) indicates that the LocalEndPoint is disabled. A value of 1 (Yes) indicates it is enabled.
CommDll Required XMLType: Element Default: None	Defines the Communications Plug-In to be used to communicate with the PDC / PMU. Due to the variation in application level functionality from one device vendor to another, the plug-in approach allows for new devices to be easily supported. Refer to Appendix C. Communication Plug-In Selection in this manual or to the OSIsoft support site for details on available plug-in's.
DataTimeout Optional XMLType: Attribute Default: 1000 (1 second)	Controls the length of time to wait, in milliseconds, for data responses. Valid range: { 10-65535}
ReconnectRate Optional XMLType: Attribute Default: 30000 (30 seconds)	Controls the interval, in milliseconds, to be used when attempting to restart communications with a C37.118 device. Valid Range: {10-65535}
FailOverId Optional XML Type: Attribute Default: None	Specifies the numeric Failover ID of the LocalEndPoint. The Failover ID allows creating redundant pair of primary and secondary LocalEndpoints for Data Source Level Failover. Valid Range: {0-65535 or empty}
Primary Optional XML Type: Attribute Default: None	Controls which LocalEndPoint in the redundant pair becomes a primary (value should be 1). For another LocalEndPoint this field should have opposite value (0) to become a backup. Use this field to setup Data Source Level Failover. This field is disabled on ICU control if FailOverID is empty. Valid Range {0-1 or disabled}

LocalEndPoint Child Nodes that apply to all connection types.	Description
Example: <RemoteEndPoint>	
Type Required XML Type: Element Default: None	Specifies the communications type. Values are "IP" or "Serial"
LocalEndPoint Child Nodes for Type="Serial"	Description
Example: <LocalEndPoint Type="Serial">	
Port Required XMLType: Attribute Default: None	The numeric value of the serial COM port to use. For example if using COM1 set this value to 1. Valid range : {1 – Max Ports}
Baud Optional XMLType: Attribute Default: 38400	The speed at which to communicate. Valid range: {Refer to your PDC / PMU hardware users guide for supported baud rates.}
Parity Optional XMLType: Attribute Default: None	The error checking parity to use. This can be 0 for NONE, 1 for ODD, 2 for EVEN, 3 for MARK, and 4 for SPACE. Valid range: {0,1,2,3,4}
StopBits Optional XMLType: Attribute Default: None	The number of stop bits following a message. This can be 0 for one, 1 for two, or 2 for one and a half stop bits. Valid range: {0,1,2}

LocalEndPoint Child Nodes for Type="IP"	Description
Example: <LocalEndPoint Type="IP">	
DataProtocol Optional XMLType: Element Default: UDP	Specifies the protocol to use for C37.118 data messages. Valid Values: {UDP or TCP}
LocalDataPort Required if Multicast being used, otherwise optional XMLType: Attribute Default: 0 (System Assigned)	The local port to bind to for data packets. If not specified or zero the interface will not specify a specific port (system assigned). This parameter will typically be used when the data protocol is UDP in order to facilitate configuration with firewalls. If this value is specified it must be different than the LocalCommandPort value or zero. Valid range: {1024-65535}
MulticastIPAddress Optional XMLType: Element Default: None	This parameter is valid only when the DataProtocol is specified as UDP. When specified, the interface will be configured to listen for C37.118 data messages via Multicast. Valid values: {224.0.0.1-224.255.255.254}

LocalEndPoint Child Nodes for Type="IP"	Description
Example: <LocalEndPoint Type="IP">	
CommandProtocol Optional XMLType: Element Default: TCP	Specifies the protocol to use for C37.118 Command and Configuration messages. If this node is not specified, the interface will use the protocol specified by the DataProtocol node or the default DataProtocol if it is not specified. Valid Values: {UDP or TCP}
LocalCommandPort Optional XMLType: Attribute Default: 0 (System Assigned)	The local port to bind to for configuration packets. If not specified or zero the interface will not specify a specific port (system assigned). This parameter will typically be used when the data protocol is UDP in order to facilitate configuration with firewalls. If this value is specified it must be different than the LocalDataPort value or zero. Valid range: {1024-65535}
CloseCommandSocket Optional XML Type: Attribute Default Value: 1	Some PMU / PDC models require the socket used for sending commands and receiving configuration and header responses be closed after use, while others require that they remain open. The Arbiter 1133 for example, requires that the socket be closed. All SEL PMU and PDC models require the socket remain open. This option only effects configurations in which the command/command response is on a separate protocol and/or port from the C37.118 data stream. Valid Values: 0 (Leave Open) or 1 (Close After Use)
LocalIPAddress Optional XMLType: Element Default: None	This parameter could be used if interface node has multiple IP addresses. When specified, the interface will bind all ports with that address. Otherwise the interface will be using default IP address. Valid Values: Any IP address that could be used by the interface node or empty.
MulticastSourceIPAddr Optional XML Type: Element Default Value: None	This parameter is valid only if the Local IP Address and Multicast IP Address fields have been populated. When specified, the interface will subscribe for multicast data coming from the source that has specified IP address.

RemoteEndPoint Nodes

The RemoteEndPoint node is child node of the LocalEndPoint node(s). The RemoteEndPoint node represents the remote (PDC / PMU side) of a physical connection. The RemoteEndPoint node will have one or more PMU child nodes. Each of the PMU nodes will represent a PMU that the PI C37.118 interface will communicate with through the physical port described by the LocalEndPoint / RemoteEndPoint child node pair.

RemoteEndPoint Child Nodes that apply to all connection types.	Description
Example: <RemoteEndPoint>	
IdCode Required XMLType: Attribute Default: None	Specifies the numeric ID of the PDC or PMU. This must match the IDCODE field sent by the C37.118 device. If a LocalEndPoint has multiple RemoteEndPoint nodes associated with it this ID must be unique within the LocalEndPoint collection. Valid ranges: {0-65535 }

RemoteEndPoint Child Nodes that apply to all connection types.	Description
Description Optional XML Type: Element Default: None	User defined description of the RemoteEndPoint. I.e. "Substation xyz". This attribute is used by the interface when logging messages regarding the RemoteEndPoint.

RemoteEndPoint Child Nodes when LocalEndPoint is "Serial"	Description
Example: <RemoteEndPoint>	
	There are no Serial specific RemoteEndPoint child nodes

RemoteEndPoint Child Nodes when LocalEndPoint is "IP"	Description
Example: <RemoteEndPoint>	
HostName Required XMLType: Element Default: None	The Host name or IP address of the PDC or PMU. If using the PIC37118_Comm003.DLL, the supplied name should be set to 127.0.0.1 to indicate the interface will listen on the specified LocalDataPort for incoming C37.118 messages. Valid range: {Any valid address or name} Example: <Hostname> 192.164.55.128</Hostname> or <Hostname> myhost </Hostname>
RemoteDataPort Optional XMLType: Attribute Default: 4712	The remote port to be used for data packets. If not specified the interface will not specify a specific port (system assigned). Valid range: {1024-65535} or 0 (System Assigned)
RemoteCommandPort Optional XMLType: Attribute Default: 4713	The remote port to use for command configuration packets. Valid range: {1024-65535}

PMU Child Nodes

PMU nodes are child nodes of the RemoteEndPoint nodes. The PMU node describes the characteristics of the PMU. If the RemoteEndPoint is a PDC then there may be multiple PMUs, otherwise there should be only one PMU child node per RemoteEndPoint.

PMU Child Nodes	Description
Example: <PMU>	
IdCode Required XMLType: Attribute Default: None	Specifies the numeric ID of the PMU. This must match the IDCODE field sent by the C37.118 device. If there are multiple PMU nodes associated with a RemoteEndPoint (RemoteEndPoint is a PDC) with it this ID must be unique within the RemoteEndPoint collection and should match the PMUs IDCODE and not the PDCs. Valid ranges: {0-65535 }
Latitude Optional XMLType: Attribute Default: None	Specifies the geographic latitude of the PMU. If the latitude is specified, then the longitude must also be specified Otherwise the interface will log an error and exit. Valid range: {-90.0 to +90.0}
Longitude Optional XMLType: Attribute Default: None	Specifies the geographic longitude of the PMU. If the latitude is specified, then the longitude must also be specified Otherwise the interface will log an error and exit. Valid range: {-180.0 to +180.0}

Sample PIC37118 XML File

The following is an example file:

```
<?xml version="1.0"?>
<PI_C37_118.Session WriteIOTimeout="1" DataSortingAction="1"
PMUErrorAction="3" SyncErrorAction="1" DataValidAction="3"
LeapSecondAction="0">
  <LocalEndPoint Id="1" LocalDataPort="1024" LocalCommandPort="1025"
DataTimeout="1000">
    <Type>IP</Type>
    <CommDLL>PIC37118_Comm001.dll</CommDLL>
    <CommandProtocol>TCP</CommandProtocol>
    <DataProtocol>UDP</DataProtocol>
    <MulticastIPAddress>239.254.1.2</MulticastIPAddress>
    <RemoteEndPoint IdCode="1">
      <Description>Arbiter in Savannah</Description>
      <HostName>gap001</HostName>
      <PMU IdCode="1" Latitude="37.127652"
Longitude="-122.876365"/>
    </RemoteEndPoint>
  </LocalEndPoint>
</PI_C37_118.Session>
```

Command-line Parameters

Note: The *PI Universal Interface (Unifnt) User Guide* includes details about other command-line parameters, which may be useful.

Parameter	Description
/CacheMode Required when using disconnected startup Default: Not Defined	Required for disconnected startup operation. If defined, the /CacheMode startup parameter indicates that the interface will be configured to utilize the disconnected startup feature.
/CachePath=path Optional Default: Not Defined	Used to specify a directory in which to create the point caching files. The directory specified 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: /CachePath=D:\PIPC\Interfaces\CacheFiles /CachePath=D:/PIPC/Interfaces/CacheFiles /CachePath=D:/PIPC/Interfaces/CacheFiles/ Examples with space in path name: /CachePath="D:\Program Files\PIPC\MyFiles" /CachePath="D:/Program Files/PIPC/MyFiles" /CachePath="D:/Program Files/PIPC/MyFiles/"

<p>/CacheSynch=# Optional Default: 250 ms</p>	<p>NOTE: Care must be taken when modifying this parameter. This value must be less than the smallest scan class period defined with the /f parameter. If the value of the /CacheSynch parameter is greater than the scan class value, input scans will be missed while the point cache file is being synchronized.</p> <p>The optional /CacheSynch=# startup parameter specifies the time slice period in milliseconds (ms) allocated by Unilnt for synchronizing the interface point cache file with the PI Data Archive. By default, the interface will synchronize the point cache if running in the disconnected startup mode. Unilnt allocates a maximum of # ms each pass through the control loop synchronizing the interface point cache until the file is completely synchronized.</p> <p>Synchronization of the point cache file can be disabled by setting the value /CacheSynch=0. The minimum synchronization period when cache synchronization is enabled is 50ms Whereas, the maximum synchronization period is 3000ms (3s). Period values of 1 to 49 will be changed by the interface to the minimum of 50ms and values greater than 3000 will be set to the maximum interval value of 3000ms.</p> <p>Default: 250 ms Range: {0, 50 – 3000} time in milliseconds Example: /CacheSynch=50 (use a 50ms interval) /CacheSynch=3000 (use a 3s interval) /CacheSynch=0 (do not synchronize the cache)</p>
<p>/ec=# Optional</p>	<p>The first instance of the /ec parameter on the command-line is used to specify a counter number, #, for an I/O Rate point. If the # is not specified, then the default event counter is 1. Also, if the /ec parameter is not specified at all, there is still a default event counter of 1 associated with the interface. If there is an I/O Rate point that is associated with an event counter of 1, each copy of the interface that is running without /ec=# explicitly defined will write to the same I/O Rate point. This means either explicitly defining an event counter other than 1 for each copy of the interface or not associating any I/O Rate points with event counter 1. Configuration of I/O Rate points is discussed in the section called I/O Rate Point.</p> <p>For interfaces that run on Windows nodes, subsequent instances of the /ec parameter may be used by specific interfaces to keep track of various input operations. Subsequent instances of the /ec parameter can be of the form /ec*, where * is any ASCII character sequence. For example, /ecinput=10 and /ec=11 are legitimate choices for the second and third event counter strings.</p>
<p>/f=SS.## or /f=SS.##,ss.## or /f=HH:MM:SS.## or /f=HH:MM:SS.##,hh:mm:ss.##</p> <p>Required for reading scan-based inputs</p>	<p>The /f parameter defines the time period between scans in terms of hours (HH), minutes (MM), seconds (SS) and sub-seconds (##). The scans can be scheduled to occur at discrete moments in time with an optional time offset specified in terms of hours (hh), minutes (mm), seconds (ss) and sub-seconds (##). If HH and MM are omitted, then the time period that is specified is assumed to be in seconds.</p> <p>Each instance of the /f parameter on the command-line defines a scan class for the interface. There is no limit to the number of scan classes that can be defined. The first occurrence of the /f parameter on the command-line defines the first scan class of the interface; the second occurrence defines the second scan class, and so on. PI points are associated with a particular scan class via the Location4 PI point attribute. For example, all PI points that have Location4 set to 1 will receive input values at the frequency defined by the first scan class. Similarly, all points that have Location4 set</p>

	<p>to 2 will receive input values at the frequency specified by the second scan class, and so on.</p> <p>Two scan classes are defined in the following example: <code>/f=00:01:00,00:00:05 /f=00:00:07</code> or, equivalently: <code>/f=60,5 /f=7</code></p> <p>The first scan class has a scanning frequency of 1 minute with an offset of 5 seconds, and the second scan class has a scanning frequency of 7 seconds. When an offset is specified, the scans occur at discrete moments in time according to the formula: $\text{scan times} = (\text{reference time}) + n(\text{frequency}) + \text{offset}$ where n is an integer and the reference time is midnight on the day that the interface was started. In the above example, frequency is 60 seconds and offset is 5 seconds for the first scan class. This means that if the interface was started at 05:06:06, the first scan would be at 05:07:05, the second scan would be at 05:08:05, and so on. Since no offset is specified for the second scan class, the absolute scan times are undefined.</p> <p>The definition of a scan class does not guarantee that the associated points will be scanned at the given frequency. If the interface is under a large load, then some scans may occur late or be skipped entirely. See the section "Logging hit, missed, and skipped scans" in the <i>PI Universal Interface (Unilnt) User Guide</i> for more information on skipped or missed scans.</p> <p>Sub-second Scan Classes</p> <p>Sub-second scan classes can be defined on the command-line, such as <code>/f=0.5 /f=00:00:00.1</code> where the scanning frequency associated with the first scan class is 0.5 seconds and the scanning frequency associated with the second scan class is 0.1 of a second.</p> <p>Similarly, sub-second scan classes with sub-second offsets can be defined, such as <code>/f=0.5,0.2 /f=1,0</code></p> <p>Wall Clock Scheduling</p> <p>Scan classes that strictly adhere to wall clock scheduling are now possible. This feature is available for interfaces that run on Windows and/or UNIX. Previously, wall clock scheduling was possible, but not across daylight saving time. For example, <code>/f=24:00:00,08:00:00</code> corresponds to 1 scan a day starting at 8 AM. However, after a Daylight Saving Time change, the scan would occur either at 7 AM or 9 AM, depending upon the direction of the time shift. To schedule a scan once a day at 8 AM (even across daylight saving time), use <code>/f=24:00:00,00:08:00,L</code>. The <code>,L</code> at the end of the scan class tells Unilnt to use the new wall clock scheduling algorithm.</p>
--	---

<p><code>/host=host:port</code> Required</p>	<p>The <code>/host</code> parameter specifies the PI Data Archive node.</p> <p><i>Host</i> is the IP address or the domain name of the PI Data Archive node.</p> <p><i>Port</i> is the port number for TCP/IP communication. The <i>port</i> is always 5450. It is recommended to explicitly define the host and port on the command-line with the <code>/host</code> parameter.</p> <p>Nevertheless, if either the host or port is not specified, the interface will attempt to use defaults.</p> <p>Examples:</p> <p>The interface is running on an interface node, the domain name of the PI Data Archive node is Marvin, and the IP address of Marvin is 206.79.198.30. Valid <code>/host</code> parameters would be:</p> <pre> /host=marvin /host=marvin:5450 /host=206.79.198.30 /host=206.79.198.30:5450 </pre>
<p><code>/id=x</code> Highly Recommended</p>	<p>The <code>/id</code> parameter is used to specify the interface identifier.</p> <p>The interface identifier is a string that is no longer than 9 characters in length. Unilnt concatenates this string to the header that is used to identify error messages as belonging to a particular interface. See the Appendix A. Error and Informational Messages for more information.</p> <p>Unilnt always uses the <code>/id</code> parameter in the fashion described above. This interface also uses the <code>/id</code> parameter to identify a particular interface copy number that corresponds to an integer value that is assigned to one of the Location code point attributes, most frequently Location1. For this interface, use only numeric characters in the identifier. For example,</p> <pre> /id=1 </pre>
<p><code>/ps=x</code> Required</p>	<p>The <code>/ps</code> parameter specifies the point source for the interface. <code>X</code> is not case sensitive and can be any single or multiple character string. For example, <code>/ps=P</code> and <code>/ps=p</code> are equivalent. The length of <code>X</code> is limited to 100 characters by Unilnt. <code>X</code> can contain any character except '*' and '?'. </p> <p>The point source that is assigned with the <code>/ps</code> parameter corresponds to the PointSource attribute of individual PI points. The interface will attempt to load only those PI points with the appropriate point source.</p> <p>If the PI API version being used is prior to 1.6.x or the PI Data Archive version is prior to 3.4.370.x, the <code>PointSource</code> is limited to a single character unless the SDK is being used.</p>

<p><code>/stopstat=digstate</code> or <code>/stopstat</code></p> <p><code>/stopstat</code> only is equivalent to <code>/stopstat="ntf Shut"</code></p> <p>Optional Default = no digital state written at shutdown.</p>	<p>If <code>/stopstat=digstate</code> is present on the command line, then the digital state, <code>digstate</code>, will be written to each PI point when the interface is stopped. For a PI Data Archive, <code>digstate</code> must be in the system digital state table. . Unilnt will use the first occurrence of <code>digstate</code> found in the table.</p> <p>If the <code>/stopstat</code> parameter is present on the startup command line, then the digital state "Intf Shut" will be written to each PI point when the interface is stopped.</p> <p>If neither <code>/stopstat</code> nor <code>/stopstat=digstate</code> is specified on the command line, then no digital states will be written when the interface is shut down.</p> <hr/> <p>Note: The <code>/stopstat</code> parameter is disabled If the interface is running in a Unilnt failover configuration as defined in the Unilnt Failover Configuration section of this manual. Therefore, the digital state, <code>digstate</code>, will not be written to each PI point when the interface is stopped. This prevents the digital state being written to PI points while a redundant system is also writing data to the same PI points. The <code>/stopstat</code> parameter is disabled even if there is only one interface active in the failover configuration.</p> <hr/> <p>Examples: <code>/stopstat=shutdown</code> <code>/stopstat="Intf Shut"</code></p> <p>The entire <code>digstate</code> value should be enclosed within double quotes when there is a space in <code>digstate</code>.</p>
<p><code>/UFO_ID=#</code></p> <p>Required for Unilnt failover phase 1 or 2</p>	<p>Failover ID. This value must be different from the failover ID of the other interface in the failover pair. It can be any positive, non-zero integer.</p>
<p><code>/UFO_Interval=#</code></p> <p>Optional Default: 1000 for phase 1 failover Default: 5000 for phase 2 failover</p> <p>Valid values are 50-20000.</p>	<p>Failover Update Interval Specifies the heartbeat update interval in milliseconds and must be the same on both interface computers.</p> <p>This is the rate at which Unilnt updates the failover heartbeat points as well as how often Unilnt checks on the status of the other copy of the interface.</p>
<p><code>/UFO_OtherID=#</code></p> <p>Required for Unilnt failover phase 1 or 2</p>	<p>Other Failover ID. This value must be equal to the failover ID configured for the other interface in the failover pair.</p>

<p>/UFO_Sync=<i>path</i>/<i>[filename]</i></p> <p>Required for Unilnt failover phase 2 synchronization.</p> <p>Any valid pathname / any valid filename</p> <p>The default filename is generated as <i>executablename_pointsource_interfaceID.dat</i></p>	<p>The failover synchronization file <i>path</i> and optional <i>filename</i> specify the path to the shared file used for failover synchronization and an optional filename used to specify a user defined filename in lieu of the default filename.</p> <p>The <i>path</i> to the shared file directory can be a fully qualified machine name and directory, a mapped drive letter, or a local path if the shared file is on one of the interface nodes. The <i>path</i> must be terminated by a slash (/) or backslash (\) character. If no d terminating slash is found, in the /UFO_Sync parameter, the interface interprets the final character string as an optional <i>filename</i>.</p> <p>The optional <i>filename</i> can be any valid filename. If the file does not exist, the first interface to start attempts to create the file.</p> <hr/> <p>Note: If using the optional filename, do not supply a terminating slash or backslash character.</p> <hr/> <p>If there are any spaces in the <i>path</i> or <i>filename</i>, the entire path and filename must be enclosed in quotes.</p> <hr/> <p>Note: If you use the backslash and path separators and enclose the path in double quotes, the final backslash must be a double backslash (\\). Otherwise the closing double quote becomes part of the parameter instead of a parameter separator.</p> <p>Each node in the failover configuration must specify the same path and filename and must have read, write, and file creation rights to the shared directory specified by the <i>path</i> parameter.</p> <hr/> <p>The service that the interface runs against must specify a valid logon user account under the “Log On” tab for the service properties.</p>
<p>/UFO_Type=<i>type</i></p> <p>Required for Unilnt failover phase 2.</p>	<p>The failover Type indicates which type of failover configuration the interface will run. The valid types for failover are HOT, WARM, and COLD configurations.</p> <p>If an interface does not supported the requested type of failover, the interface will shut down and log an error to the log file stating the requested failover type is not supported. This interface supports failover configurations COLD and HOT.</p>

/configfile =<UNC Path> Optional Default: PIC37118.xml	This parameter specifies the interface configuration file to use to define the C37.118 device that the interface will communicate with.
/TagConfig =<UNC Path> Optional Default: None	<p>The <code>/TagConfig</code> parameter is used to assist in creating PI tags. Because the C37.118 specification does not dictate the names to use for Phasor, Analog, and Digital channels, different PMU vendors use different names.</p> <p>When the interface is configured initially, the command-line parameter can be specified temporarily in the Additional Parameters test box in the ICU. When the interface is started it reads the CONFIG2 block from the PDC/PMU and generates a comma-delimited file that contains the various PI point attributes. The file can then be edited as required and then imported into PI using SMT.</p> <p>Because the interface exits after the file is written, it is important to remove the command-line parameter for normal operation.</p> <p>If a path is not specified on the command line, the generated file is saved in the interface folder. In versions of the interface earlier than 1.0.4.88, the file was saved in the system drive folder: <code>Windows\System32</code>. Optionally, you can specify the fully qualified path of where to save the taglist.csv file; for example: <code>/TagConfig= C:\Taglist.csv</code>. If the path or filename contains spaces, you must enclose the path or filename within quotation marks.</p>

Sample PIC37118.bat File

The following is an example file:

```
REM=====
REM
REMPIC37118.bat
REM
REM Sample startup file for the PI Interface for IEEE C37.118
REM
REM=====
REM
REM OSIssoft strongly recommends using PI ICU to modify startup files.
REM
REM Sample command line
REM
    .\PIC37118.exe ^
    /configfile="E:\Program Files\PIPC\Interfaces\C37118\PIC37118.xml" ^
    /ps=C37118 ^
    /ID=1 ^
    /host=XXXXXX:5450
REM
REM End of pic37118.bat File
```


Chapter 9. UniInt Failover Configuration

Introduction

To minimize data loss during a single point of failure within a system, UniInt provides two failover schemes: (1) synchronization through the data source and (2) synchronization through a shared file. Synchronization through the data source is *Phase 1*, and synchronization through a shared file is *Phase 2*.

Phase 1 UniInt Failover is not supported by this interface.

Phase 2 UniInt Failover uses a shared file to synchronize failover operations and provides for *hot, warm, or cold failover*. The Phase 2 hot failover configuration provides a *no data loss* solution for a single point of failure similar to Phase 1. However, in warm and cold failover configurations, you can expect a small period of data loss during a single point of failure transition.

Note: This interface supports only Phase 2 failover, COLD and HOT configurations.

You can also configure UniInt failover to send data to a High Availability (HA) PI Data Archive collective. The collective provides redundant PI Data Archives to allow for the uninterrupted collection and presentation of PI time series data. In an HA configuration, PI Data Archives can be taken down for maintenance or repair. The HA PI Data Archive collective is described in the *High Availability Administrator Guide*.

When configured for UniInt failover, the interface routes all PI data through a state machine. The state machine determines whether to queue data or send it directly to PI depending on the current state of the interface. When the interface is in the active state, data sent through the interface gets routed directly to a PI point. In the backup state, data from the interface gets queued for a short period. Queued data in the backup interface ensures a *no-data loss* failover under normal circumstances for Phase 1 and for the hot failover configuration of Phase 2.

Quick Overview

The Quick Overview below may be used to configure this interface for failover. The failover configuration requires the two copies of the interface participating in failover be installed on different nodes. Users should verify non-failover interface operation as discussed in the [Installation Checklist](#) chapter of this manual prior to configuring the interface for failover operations. If you are not familiar with UniInt failover configuration, return to this section after reading the rest of the [UniInt Failover Configuration](#) chapter in detail. If a failure occurs at any step below, correct the error and start again at the beginning of step 6 Test in the table below. For the discussion below, the first copy of the interface configured and tested will be considered the primary interface and the second copy of the interface configured will be the backup interface.

Configuration

- One Data Source
- Two interfaces

Prerequisites

- Interface 1 is the Primary interface for collection of PI data from the data source.
- Interface 2 is the Backup interface for collection of PI data from the data source.
- You must set up a shared file if using Phase 2 failover.
- Phase 2: The shared file must store data for five failover tags:
 - (1) Active ID.
 - (2) Heartbeat 1.
 - (3) Heartbeat 2.
 - (4) Device Status 1.
 - (5) Device Status 2.
- Each interface must be configured with two required failover command line parameters: (1) its FailoverID number (**/UFO_ID**); (2) the FailoverID number of its Backup interface (**/UFO_OtherID**). You must also specify the name of the PI Data Archive host for exceptions and PI point updates.
- All other configuration parameters for the two interfaces must be identical.

Synchronization through a Shared File (Phase 2)

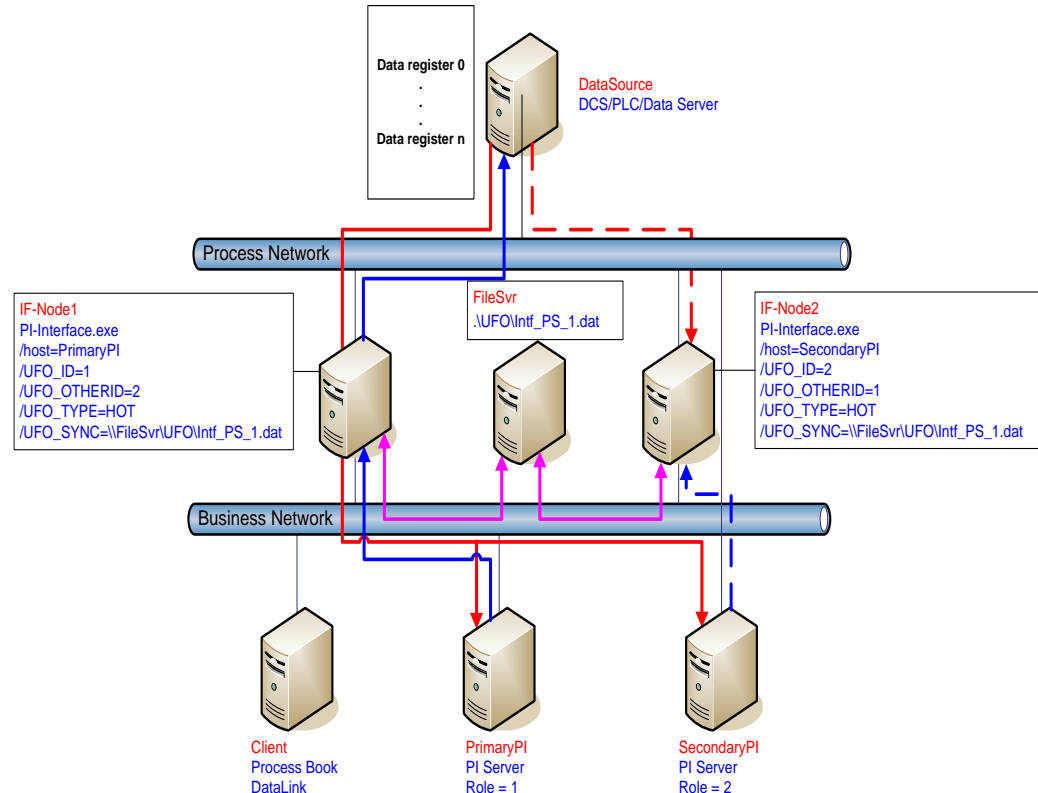


Figure 1: Synchronization through a Shared File (Phase 2) Failover Architecture

The Phase 2 failover architecture is shown in Figure 2 which depicts a typical network setup including the path to the synchronization file located on a File Server (FileSvr). Other configurations may be supported and this figure is used only as an example for the following discussion.

For a more detailed explanation of this synchronization method, see [Detailed Explanation of Synchronization through a Shared File \(Phase 2\)](#)

Configuring Synchronization through a Shared File (Phase 2)

Step	Description																									
1.	Verify non-failover interface operation as described in the Installation Checklist section of this manual																									
2.	Configure the Shared File Choose a location for the shared file. The file can reside on one of the interface nodes or on a separate node from the interfaces; however OSIsoft strongly recommends that you put the file on a Windows Server platform that has the “File Server” role configured. . Setup a file share and make sure to assign the permissions so that both primary and backup interfaces have read/write access to the file.																									
3.	Configure the interface parameters Use the Failover section of the interface Configuration Utility (ICU) to enable failover and create two parameters for each interface: (1) a Failover ID number for the interface; and (2) the Failover ID number for its backup interface. The Failover ID for each interface must be unique and each interface must know the Failover ID of its backup interface. If the interface can perform using either Phase 1 or Phase 2 pick the Phase 2 radio button in the ICU. Select the synchronization File Path and File to use for Failover. Select the type of failover required (Cold, Warm, Hot). The choice depends on what types of failover the interface supports. Ensure that the user name assigned in the “Log on as:” parameter in the Service section of the ICU is a user that has read/write access to the folder where the shared file will reside. All other command line parameters for the primary and secondary interfaces must be identical. If you use a PI Data Archive Collective, you must point the primary and secondary interfaces to different members of the collective by setting the SDK Member under the PI Host Information section of the ICU. [Option] Set the update rate for the heartbeat point if you need a value other than the default of 5000 milliseconds.																									
4.	Configure the PI points Configure five PI points for the interface: the active ID, heartbeat 1, heartbeat 2, device status 1 and device status 2. You can also configure two state points for monitoring the status of the interfaces. Do not confuse the failover device status points with the Unilnt health device status points. The information in the two points is similar, but the failover device status points are integer values and the health device status points are string values. <table><tr><th>Tag</th><th>ExDesc</th><th>digitalset</th><td rowspan="8">Unilnt does not examine the remaining attributes, but the PointSource and Location1 must match.</td></tr><tr><td>ActiveID</td><td>[UFO2_ACTIVEID]</td><td></td></tr><tr><td>IF1_Heartbeat (IF-Node1)</td><td>[UFO2_HEARTBEAT: #]</td><td></td></tr><tr><td>IF2_Heartbeat (IF-Node2)</td><td>[UFO2_HEARTBEAT: #]</td><td></td></tr><tr><td>IF1_DeviceStatus (IF-Node1)</td><td>[UFO2_DEVICESTAT: #]</td><td></td></tr><tr><td>IF2_DeviceStatus (IF-Node2)</td><td>[UFO2_DEVICESTAT: #]</td><td></td></tr><tr><td>IF1_State (IF-Node1)</td><td>[UFO2_STATE: #]</td><td>IF_State</td></tr><tr><td>IF2_State (IF-Node2)</td><td>[UFO2_STATE: #]</td><td>IF_State</td></tr></table>	Tag	ExDesc	digitalset	Unilnt does not examine the remaining attributes, but the PointSource and Location1 must match.	ActiveID	[UFO2_ACTIVEID]		IF1_Heartbeat (IF-Node1)	[UFO2_HEARTBEAT: #]		IF2_Heartbeat (IF-Node2)	[UFO2_HEARTBEAT: #]		IF1_DeviceStatus (IF-Node1)	[UFO2_DEVICESTAT: #]		IF2_DeviceStatus (IF-Node2)	[UFO2_DEVICESTAT: #]		IF1_State (IF-Node1)	[UFO2_STATE: #]	IF_State	IF2_State (IF-Node2)	[UFO2_STATE: #]	IF_State
Tag	ExDesc	digitalset	Unilnt does not examine the remaining attributes, but the PointSource and Location1 must match.																							
ActiveID	[UFO2_ACTIVEID]																									
IF1_Heartbeat (IF-Node1)	[UFO2_HEARTBEAT: #]																									
IF2_Heartbeat (IF-Node2)	[UFO2_HEARTBEAT: #]																									
IF1_DeviceStatus (IF-Node1)	[UFO2_DEVICESTAT: #]																									
IF2_DeviceStatus (IF-Node2)	[UFO2_DEVICESTAT: #]																									
IF1_State (IF-Node1)	[UFO2_STATE: #]	IF_State																								
IF2_State (IF-Node2)	[UFO2_STATE: #]	IF_State																								
5.	Test the configuration.																									

Step	Description
	<p>After configuring the shared file and the interface and PI points, the interface should be ready to run.</p> <p>For help resolving failover file issues, see OSIsoft KB article - KB00889 - Troubleshooting Failover Synchronization File Errors on the OSIsoft technical support web site.</p> <p>See Troubleshooting UniInt Failover for help resolving Failover issues.</p> <ol style="list-style-type: none"> 1. Start the primary interface interactively without buffering. 2. Verify a successful interface start by reviewing the log file. The log file will contain messages that indicate the failover state of the interface. A successful start with only a single interface copy running will be indicated by an informational message stating "UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface not available." If the interface has failed to start, an error message will appear in the log file. For details relating to informational and error messages, refer to Appendix A. Error and Informational Messages. 3. Verify data on the PI Data Archive using available PI tools. <ul style="list-style-type: none"> • The active ID control point in the PI Data Archive must be set to the value of the running copy of the interface as defined by the <code>/UFO_ID</code> startup command-line parameter. • The heartbeat control point in the PI Data Archive must be changing values at a rate specified by the <code>/UFO_Interval</code> startup command-line parameter. 4. Stop the primary interface. 5. Start the backup interface interactively without buffering. Notice that this copy will become the primary because the other copy is stopped. 6. Repeat steps 2, 3, and 4. 7. Stop the backup interface. 8. Start buffering. 9. Start the primary interface interactively. 10. Once the primary interface has successfully started and is collecting data, start the backup interface interactively. 11. Verify that both copies of the interface are running in a failover configuration. <ul style="list-style-type: none"> • Review the log file for the copy of the interface that was started first. The log file will contain messages that indicate the failover state of the interface. The state of this interface must have changed as indicated with an informational message stating "UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface available." If the interface has not changed to this state, browse the log file for error messages. For details relating to informational and error messages, refer to Appendix A. Error and Informational Messages. • Review the log file for the copy of the interface that was started last. The log file will contain messages that indicate the failover state of the interface. A successful start of the interface will be indicated by an informational message stating "UniInt failover: Interface in the "Backup" state." If the interface has failed to start, an error message will appear in the log file. For details relating to informational and error messages, refer to the Appendix A. Error and Informational Messages section below. 12. Verify data in the PI Data Archive using available PI tools.

Step	Description
	<ul style="list-style-type: none"> The active ID control point in the PI Data Archive must be set to the value of the running copy of the interface that was started first as defined by the <code>/UFO_ID</code> startup command-line parameter. The heartbeat control points for both copies of the interface in the PI Data Archive must be changing values at a rate specified by the <code>/UFO_Interval</code> startup command-line parameter or the scan class which the points have been built against.
13.	Test Failover by stopping the primary interface.
14.	Verify the backup interface has assumed the role of primary by searching the <code>pipc.log</code> file for a message indicating the backup interface has changed to the "UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface not available." The backup interface is now considered primary and the previous primary interface is now backup.
15.	Verify data in the PI Data Archive. For hot failover, there may be an overlap of data due to the queuing of data, but there must be no data loss. For warm or cold failover, short gaps in archived data are expected.
16.	Start the backup interface. Once the primary interface detects a backup interface, the primary interface will now change state indicating "UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface available." In the <code>pipc.log</code> file.
17.	Verify the backup interface starts and assumes the role of backup. A successful start of the backup interface will be indicated by an informational message stating "UniInt failover: Interface in "Backup state." Since this is the initial state of the interface, the informational message will be near the beginning of the start sequence of the <code>pipc.log</code> file.
18.	Test failover with different failure scenarios (for example, loss of the PI Data Archive connection for a single interface copy). UniInt hot failover guarantees no data loss with a single point of failure; verify no data loss by checking the data in the PI Data Archive and on the data source. For warm or cold failover, short gaps in archived data can occur.
19.	Stop both copies of the interface, start buffering, start each interface as a service.
20.	Verify data as stated above.
21.	To designate a specific interface as primary, set the active ID point to the ID of the desired primary interface as defined by the <code>/UFO_ID</code> startup command-line parameter.

Configuring UniInt Failover through a Shared File (Phase 2)

Start-Up Parameters

Note: The `/stopstat` parameter is disabled if the interface is running in a UniInt failover configuration. Therefore, the digital state, `digstate`, will not be written to each PI point when the interface is stopped. This prevents the digital state being written to PI points while a redundant system is also writing data to the same PI points. The `/stopstat` parameter is disabled even if there is only one interface active in the failover configuration.

The following table lists the start-up parameters used by UniInt Failover Phase 2. All of the parameters are required except the `/UFO_Interval` startup parameter. See the table below for further explanation.

Parameter	Required/ Optional	Description	Value/Default
<code>/UFO_ID=#</code>	Required	Failover ID for IF-Node1 This value must be different from the failover ID of IF-Node2.	Any positive, non-zero integer / 1
	Required	Failover ID for IF-Node2 This value must be different from the failover ID of IF-Node1.	Any positive, non-zero integer / 2
<code>/UFO_DNFBPI</code>	Optional	If this parameter is enabled and both the primary and backup instances lose connection to the PI Data Archive UniInt will not failover and incoming data will be buffered by a single interface instance to prevent the buffer from being split.	Undefined
<code>/UFO_OtherID=#</code>	Required	Other Failover ID for IF-Node1 The value must be equal to the Failover ID configured for the interface on IF-Node2.	Same value as Failover ID for IF-Node2 / 2
	Required	Other Failover ID for IF-Node2 The value must be equal to the Failover ID configured for the interface on IF-Node1.	Same value as Failover ID for IF-Node1 / 1
<code>/UFO_Sync= path/[filename]</code>	Required for Phase 2 synchronization	The Failover File Synchronization file <i>path</i> and optional <i>filename</i> specify the path to the shared file used for failover synchronization and an optional filename used to specify a user defined filename in lieu of the default filename. The <i>path</i> to the shared file directory can be a fully qualified machine name and directory, a mapped drive letter, or a local path if the shared file is on one of the interface nodes. The <i>path</i> must be terminated by a slash (/) or backslash (\) character.	Any valid pathname / any valid filename The default filename is generated as <code>executablename_pointsource_interfaceID.dat</code>

Parameter	Required/ Optional	Description	Value/Default
		<p>If no terminating slash is found, in the /UFO_Sync parameter, the interface interprets the final character string as an optional <i>filename</i>.</p> <p>The optional <i>filename</i> can be any valid filename. If the file does not exist, the first interface to start attempts to create the file.</p> <p>Note: If using the optional filename, do not supply a terminating slash or backslash character.</p> <p>If there are any spaces in the <i>path</i> or <i>filename</i>, the entire path and filename must be enclosed in quotes.</p> <p>Note: If you use the backslash and path separators and enclose the path in double quotes, the final backslash must be a double backslash (\\).</p> <p>Otherwise the closing double quote becomes part of the parameter instead of a parameter separator.</p> <p>Each node in the failover configuration must specify the same path and filename and must have read, write, and file creation rights to the shared directory specified by the <i>path</i> parameter.</p> <p>The service that the interface runs against must specify a valid logon user account under the "Log On" tab for the service properties.</p>	
/UFO_Type= <i>type</i>	Required	<p>The Failover Type indicates which type of failover configuration the interface will run. The valid types for failover are HOT, WARM, and COLD configurations.</p> <p>If an interface does not supported the requested type of failover, the interface will shutdown and log an error to the <code>pipc.log</code> file stating the requested failover type is not supported.</p>	COLD WARM HOT / COLD

Parameter	Required/ Optional	Description	Value/Default
<code>/UFO_Interval=#</code>	Optional	Failover Update Interval Specifies the heartbeat Update Interval in milliseconds and must be the same on both interface computers. This is the rate at which Unilnt updates the failover heartbeat points as well as how often Unilnt checks on the status of the other copy of the interface.	50 – 20000 / 5000
<code>/Host=server</code>	Required	Host PI Data Archive for exceptions and PI point updates The value of the <code>/Host</code> startup parameter depends on the PI Data Archive configuration. If the PI Data Archive is not part of a collective, the value of <code>/Host</code> must be identical on both interface computers. If the redundant interfaces are being configured to send data to a PI Data Archive collective, set the value of the <code>/Host</code> parameters on the different interface nodes to different members of the collective.	For IF-Node1 PrimaryPI / None For IF-Node2 SecondaryPI / None

Failover Control Points

The following table describes the points that are required to manage failover. In Phase 2 Failover, these points are located in a data file shared by the primary and backup interfaces.

OSIsoft recommends that you locate the shared file on a dedicated server that has no other role in data collection. This avoids potential resource contention and processing degradation if your system monitors a large number of data points at a high frequency.

Point	Description	Value / Default
ActiveID	Monitored by the interfaces to determine which interface is currently sending data to the PI Data Archive. ActiveID must be initialized so that when the interfaces read it for the first time, it is not an error. ActiveID can also be used to force failover. For example, if the current primary is IF-Node 1 and ActiveID is 1, you can manually change ActiveID to 2. This causes the interface at IF-Node2 to transition to the primary role and the interface at IF-Node1 to transition to the backup role.	From 0 to the highest interface Failover ID number / None Updated by the redundant interfaces Can be changed manually to initiate a manual failover
Heartbeat 1	Updated periodically by the interface on IF-Node1. The interface on IF-Node2 monitors this value to determine if the interface on IF-Node1 has become unresponsive.	Values range between 0 and 31 / None Updated by the interface on IF-Node1

Point	Description	Value / Default
Heartbeat 2	Updated periodically by the interface on IF-Node2. The interface on IF-Node1 monitors this value to determine if the interface on IF-Node2 has become unresponsive.	Values range between 0 and 31 / None Updated by the interface on IF-Node2

PI Points

The following tables list the required UniInt failover control PI points, the values they will receive, and descriptions.

Active_ID Point Configuration

Attributes	ActiveID
Tag	<Intf>_ActiveID
CompMax	0
ExDesc	[UFO2_ActiveID]
Location1	Match # in /id=#
Location5	Optional, Time in min to wait for backup to collect data before failing over.
PointSource	Match x in /ps=x
PointType	Int32
Shutdown	0
Step	1

Heartbeat and Device Status Point Configuration

Attribute	Heartbeat 1	Heartbeat 2	DeviceStatus 1	DeviceStatus 2
Tag	<HB1>	<HB2>	<DS1>	<DS2>
ExDesc	[UFO2_Heartbeat: #] Match # in /UFO_ID=#	[UFO2_Heartbeat: #] Match # in /UFO_OtherID=#	[UFO2_DeviceStat: #] Match # in /UFO_ID=#	[UFO2_DeviceStat: #] Match # in /UFO_OtherID=#
Location1	Match # in /id=#	Match # in /id=#	Match # in /id=#	Match # in /id=#
Location5	Optional, Time in min to wait for backup to collect data before failing over.	Optional, Time in min to wait for backup to collect data before failing over.	Optional, Time in min to wait for backup to collect data before failing over.	Optional, Time in min to wait for backup to collect data before failing over.
Point Source	Match x in /ps=x	Match x in /ps=x	Match x in /ps=x	Match x in /ps=x
PointType	int32	int32	int32	int32
Shutdown	0	0	0	0
Step	1	1	1	1

Interface State Point Configuration

Attribute	Primary	Backup
Tag	<Tagname1>	<Tagname2>
CompMax	0	0

Attribute	Primary	Backup
DigitalSet	UFO_State	UFO_State
ExDesc	[UFO2_State: #] (Match /UFO_ID=# on primary node)	[UFO2_State: #] (Match /UFO_ID=# on backup node)
Location1	Match # in /id=#	Same as for primary node
PointSource	Match x in /ps=x	Same as for primary node
PointType	digital	digital
Shutdown	0	0
Step	1	1

The following table describes the extended descriptor for the above PI points in more detail.

PI Tag ExDesc	Required / Optional	Description	Value
[UFO2_ACTIVEID]	Required	Active ID point The ExDesc must start with the case sensitive string: [UFO2_ACTIVEID]. The PointSource must match the interfaces' Pointsource. Location1 must match the ID for the interfaces. Location5 is the COLD failover retry interval in minutes. This can be used to specify how long before an interface retries to connect to the device in a COLD failover configuration. (See the description of COLD failover retry interval for a detailed explanation.)	0 – highest Interface Failover ID Updated by the redundant interfaces
[UFO2_HEARTBEAT: #] (IF-Node1)	Required	Heartbeat 1 point The ExDesc must start with the case sensitive string: [UFO2_HEARTBEAT: #] The number following the colon (☺) must be the Failover ID for the interface running on IF-Node1. The PointSource must match the interfaces' /ps parameter. Location1 must match the ID for the interfaces.	0 – 31 / None Updated by the interface on IF-Node1
[UFO2_HEARTBEAT: #] (IF-Node2)	Required	Heartbeat 2 point The ExDesc must start with the case sensitive string: [UFO2_HEARTBEAT: #] The number following the colon (☺) must be the Failover ID for the interface running on IF-Node2. The PointSource must match the interfaces' /ps parameter. Location1 must match the ID for the interfaces.	0 – 31 / None Updated by the interface on IF-Node2

PI Tag ExDesc	Required / Optional	Description	Value
[UFO2_DEVICESTAT :#] (IF-Node1)	Required	<p>Device Status 1 Tag</p> <p>The ExDesc must start with the case sensitive string: [UFO2_DEVICESTAT :#]</p> <p>The value following the colon (:) must be the failover ID for the interface running on IF-Node1</p> <p>The PointSource must match the interfaces' /ps parameter.</p> <p>Location1 must match the ID for the interfaces.</p> <p>A lower value is a better status and the interface with the lower status will attempt to become the primary interface.</p> <p>The failover 1 device status point is very similar to the Unilnt Health Device Status point except the data written to this point are integer values. A value of 0 is good and a value of 99 is OFF. Any value between these two extremes may result in a failover. The interface client code updates these values when the health device status point is updated.</p>	0 – 99 / None Updated by the interface on IF-Node1
[UFO2_DEVICESTAT :#] (IF-Node2)	Required	<p>Device Status 2 point</p> <p>The ExDesc must start with the case sensitive string: [UFO2_DEVICESTAT :#]</p> <p>The number following the colon (:) must be the Failover ID for the interface running on IF-Node2</p> <p>The PointSource must match the interfaces' /ps parameter.</p> <p>Location1 must match the ID for the interfaces.</p> <p>A lower value is a better status and the interface with the lower status will attempt to become the primary interface.</p>	0 – 99 / None Updated by the interface on IF-Node2
[UFO2_STATE :#] (IF-Node1)	Optional	<p>State 1 point</p> <p>The ExDesc must start with the case sensitive string: [UFO2_STATE :#]</p> <p>The number following the colon (:) must be the Failover ID for the interface running on IF-Node1</p> <p>The failover state point is recommended.</p> <p>The failover state points are digital points assigned to a digital state set with the following values.</p> <p>0 = Off: The interface has been shut down.</p>	0 – 5 / None Normally updated by the interface currently in the primary role.

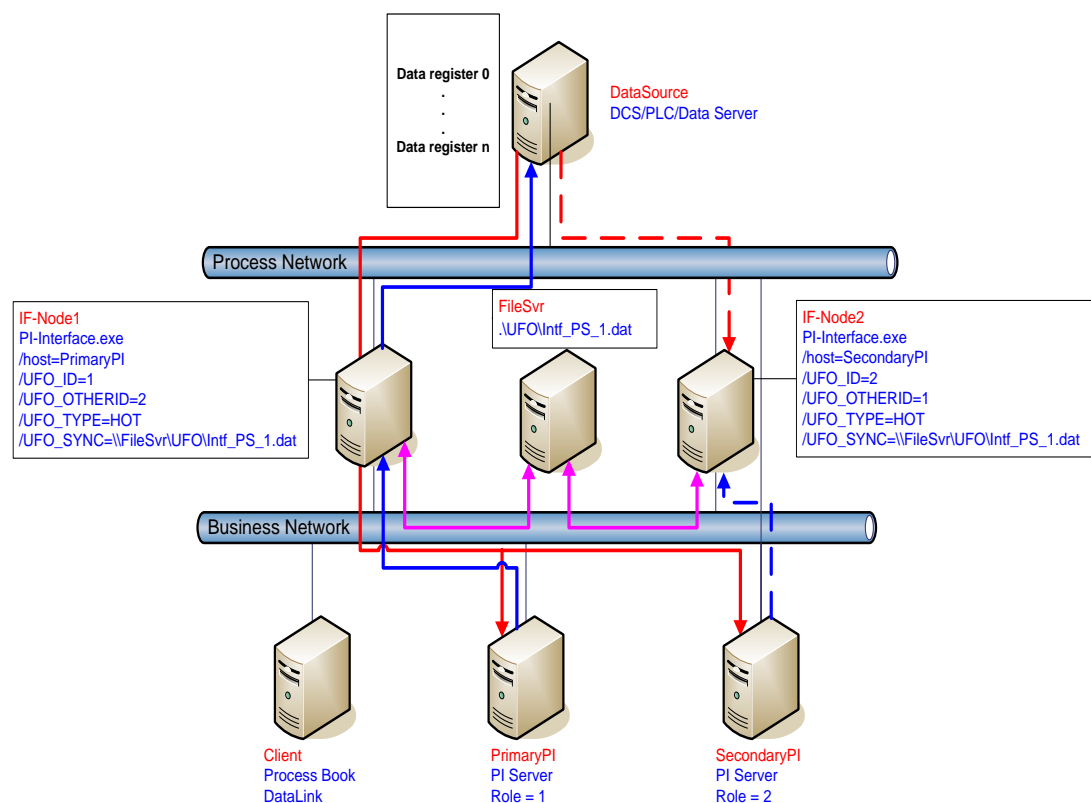
PI Tag ExDesc	Required / Optional	Description	Value
		<p>1 = Backup No Data Source: The interface is running but cannot communicate with the data source.</p> <p>2 = Backup No PI Connection: The interface is running and connected to the data source but has lost its communication to the PI Data Archive.</p> <p>3 = Backup: The interface is running and collecting data normally and is ready to take over as primary if the primary interface shuts down or experiences problems.</p> <p>4 = Transition: The interface stays in this state for only a short period of time. The transition period prevents thrashing when more than one interface attempts to assume the role of primary interface.</p> <p>5 = Primary: The interface is running, collecting data and sending the data to the PI Data Archive.</p>	
[UFO2_STATE: #] (IF-Node2)	Optional	<p>State 2 point</p> <p>The ExDesc must start with the case sensitive string: [UFO2_STATE: #]</p> <p>The number following the colon (☺) must be the Failover ID for the interface running on IF-Node2</p> <p>The failover state point is recommended.</p>	<p>Normally updated by the interface currently in the Primary state.</p> <p>Values range between 0 and 5. See description of State 1 point.</p>

Detailed Explanation of Synchronization through a Shared File (Phase 2)

In a shared file failover configuration, there is no direct failover control information passed between the data source and the interface. This failover scheme uses five PI tags to control failover operation, and all failover communication between primary and backup interfaces passes through a shared data file.

Once the interface is configured and running, the ability to read or write to the PI tags is not required for the proper operation of failover (unless connection to the shared file is lost). This solution does not require a connection to the PI Data Archive after initial startup because the control point data are set and monitored in the shared file. However, the PI point values are sent to the PI Data Archive so that you can monitor them with standard OSIsoft client tools.

You can force manual failover by changing the **ActiveID** on the PI Data Archive to the backup failover ID.



The preceding figure shows a typical network setup in the normal or steady state. The solid magenta lines show the data path from the interface nodes to the shared file used for failover synchronization. The shared file can be located anywhere in the network as long as both interface nodes can read, write, and create the necessary file on the shared file machine. OSIsoft strongly recommends that you put the file on a dedicated file server that has no other role in the collection of data.

The major difference between synchronizing the interfaces through the data source (Phase 1) and synchronizing the interfaces through the shared file (Phase 2) is where the control data is located. When synchronizing through the data source, the control data is acquired directly from the data source. We assume that if the primary interface cannot read the failover control points, then it cannot read any other data. There is no need for a backup communications path between the control data and the interface.

When synchronizing through a shared file, however, phase 2 cannot assume that loss of control information from the shared file implies that the primary interface is down. We must account for the possible loss of the path to the shared file itself and provide an alternate control path to determine the status of the primary interface. For this reason, if the shared file is unreachable for any reason, the interfaces use the PI Data Archive as an alternate path to pass control data.

When the backup interface does not receive updates from the shared file, it cannot tell definitively why the primary is not updating the file, whether the path to the shared file is down, whether the path to the data source is down, or whether the interface itself is having problems. To resolve this uncertainty, the backup interface uses the path to the PI Data Archive to determine the status of the primary interface. If the primary interface is still communicating with the PI Data Archive, then failover to the backup is not required. However, if the primary interface is not posting data to the PI Data Archive, then the backup must initiate failover operations.

The primary interface also monitors the connection with the shared file to maintain the integrity of the failover configuration. If the primary interface can read and write to the shared file with no errors but the backup control information is not changing, then the backup is experiencing some error condition. To determine exactly where the problem exists, the primary interface uses the path to the PI Data Archive to establish the status of the backup interface. For example, if the backup interface information indicates that it has been shutdown, it may have been restarted and is now experiencing errors reading and writing to the shared file. Both primary and backup interfaces must always check their status through the PI Data Archive to determine if one or the other is not updating the shared file and why.

Steady State Operation

Steady state operation is considered the normal operating condition. In this state, the primary interface is actively collecting data and sending its data to PI points. The primary interface is also updating its heartbeat value, monitoring the heartbeat value for the backup interface, checking the active ID value, and checking the device status for the backup interface every failover update interval on the shared file. Likewise, the backup interface is updating its heartbeat value, monitoring the heartbeat value for the primary interface, checking the active ID value, and checking the device status for the primary interface every failover update interval on the shared file. As long as the heartbeat value for the primary interface indicates that it is operating properly, the active ID has not changed, and the device status on the primary interface is good, the backup interface will continue in this mode of operation.

An interface configured for hot failover will have the backup interface actively collecting and queuing data but not sending that data to the PI Data Archive. An interface for warm failover in the backup role is not actively collecting data from the data source even though it may be configured with PI points and may even have a good connection to the data source. An interface configured for cold failover in the backup role is not connected to the data source and upon initial startup will not have configured PI points.

The interaction between the interface and the shared file is fundamental to failover. The discussion that follows only refers to the data written to the shared file. However, every value written to the shared file is echoed to the points in the PI Data Archive. Updating of the points in the PI Data Archive is assumed to take place unless communication with the PI Data Archive is interrupted. The updates to the PI Data Archive will be buffered by Bufserv or PIBufss in this case.

In a hot failover configuration, each interface participating in the failover solution will queue three failover intervals worth of data to prevent any data loss. When a failover occurs, there may be a period of overlapping data for up to three intervals. The exact amount of overlap is determined by the timing and the cause of the failover and may be different every time. Using the default update interval of 5 seconds will result in overlapping data between 0 and 15 seconds. The no data loss claim for hot failover is based on a single point of failure. If both interfaces have trouble collecting data for the same period of time, data will be lost during that time.

As mentioned above, each interface has its own heartbeat value. In normal operation, the heartbeat value in the shared file is incremented by UniInt from 1 to 15 and then wraps around to a value of 1 again. UniInt increments the heartbeat value in the shared file every failover update interval. The default failover update interval is 5 seconds. UniInt also reads the heartbeat value for the other interface copy participating in failover every failover update interval. If the connection to the PI Data Archive is lost, the value of the heartbeat will be incremented from 17 to 31 and then wrap around to a value of 17 again. Once the connection to the PI Data Archive is restored, the heartbeat values will revert back to the 1 to 15 range. During a normal shutdown process, the heartbeat value will be set to zero.

During steady state, the active ID will equal the value of the failover ID of the primary interface. This value is set by UniInt when the interface enters the primary state and is not updated again by the primary interface until it shuts down gracefully. During shutdown, the primary interface will set the active ID to zero before shutting down. The backup interface has the ability to assume control as primary even if the current primary is not experiencing problems. This can be accomplished by setting the active ID point in the PI Data Archive to the active ID of the desired interface copy.

As previously mentioned, in a hot failover configuration the backup interface actively collects data but does not send its data to PI points. To eliminate any data loss during a failover, the backup interface queues data in memory for three failover update intervals. The data in the queue is continuously updated to contain the most recent data. Data older than three update intervals is discarded if the primary interface is in a good status as determined by the backup. If the backup interface transitions to the primary, it will have data in its queue to send to the PI points. This queued data is sent to the PI points using the same function calls that would have been used had the interface been in a primary state when the data was collected. If UniInt receives data without a timestamp, the primary copy uses the current PI time to timestamp data sent to PI points. Likewise, the backup copy timestamps data it receives without a timestamp with the current PI time before queuing its data. This preserves the accuracy of the timestamps.

Failover Configuration Using PI ICU

The use of the PI ICU is the recommended and safest method for configuring the interface for UniInt failover. With the exception of the notes described in this section, the interface shall be configured with the PI ICU as described in the [Configuring the Interface with PI ICU](#) section of this manual.

Note: With the exception of the `/UFO_ID` and `/UFO_OtherID` startup command-line parameters, the UniInt failover scheme requires that both copies of the interface have identical startup command files. This requirement causes the PI ICU to produce a message when creating the second copy of the interface stating that the “PS/ID combo is already in use by the interface” as shown in

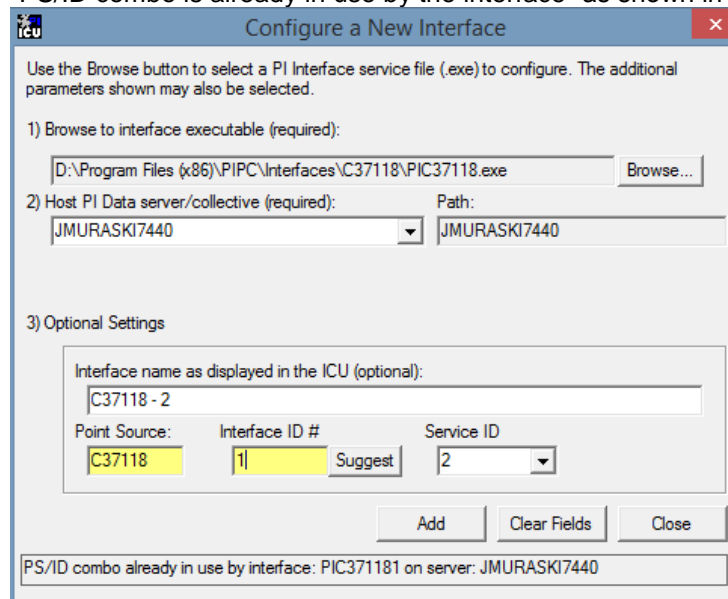


Figure below. Ignore this message and click the *Add* button.

Create the Interface Instance with PI ICU

If the interface does not already exist in the ICU it must first be created. The procedure for doing this is the same as for non-failover interfaces. When configuring the second instance for UniInt Failover the *Point Source* and *Interface ID #* boxes will be in yellow and a message will be displayed saying this is already in use. This should be ignored.

Configure a New Interface

Use the Browse button to select a PI Interface service file (.exe) to configure. The additional parameters shown may also be selected.

1) Browse to interface executable (required):

D:\Program Files (x86)\PIPC\Interfaces\C37118\PIC37118.exe Browse...

2) Host PI Data server/collective (required): Path:

JMURASKI7440 JMURASKI7440

3) Optional Settings

Interface name as displayed in the ICU (optional):

C37118 - 2

Point Source: Interface ID # Service ID

C37118 1 Suggest 2

Add Clear Fields Close

PS/ID combo already in use by interface: PIC371181 on server: JMURASKI7440

Figure 4: PI ICU configuration screen shows that the “PS/ID combo is already in use by the interface.” The user must ignore the yellow boxes, which indicate errors, and click the *Add* button to configure the interface for failover.

Configuring the UniInt Failover Startup Parameters with PI ICU

There are three interface startup parameters that control UniInt failover: `/UFO_ID`, `/UFO_OtherID`, and `/UFO_Interval`. The `UFO` stands for UniInt Failover. The `/UFO_ID` and `/UFO_OtherID` parameters are required for the interface to operate in a failover configuration, but the `/UFO_Interval` is optional. For more detail on each of these parameters, see the [Configuring UniInt Failover through a Shared File \(Phase 2\) Start-Up Parameters](#) section of this manual.

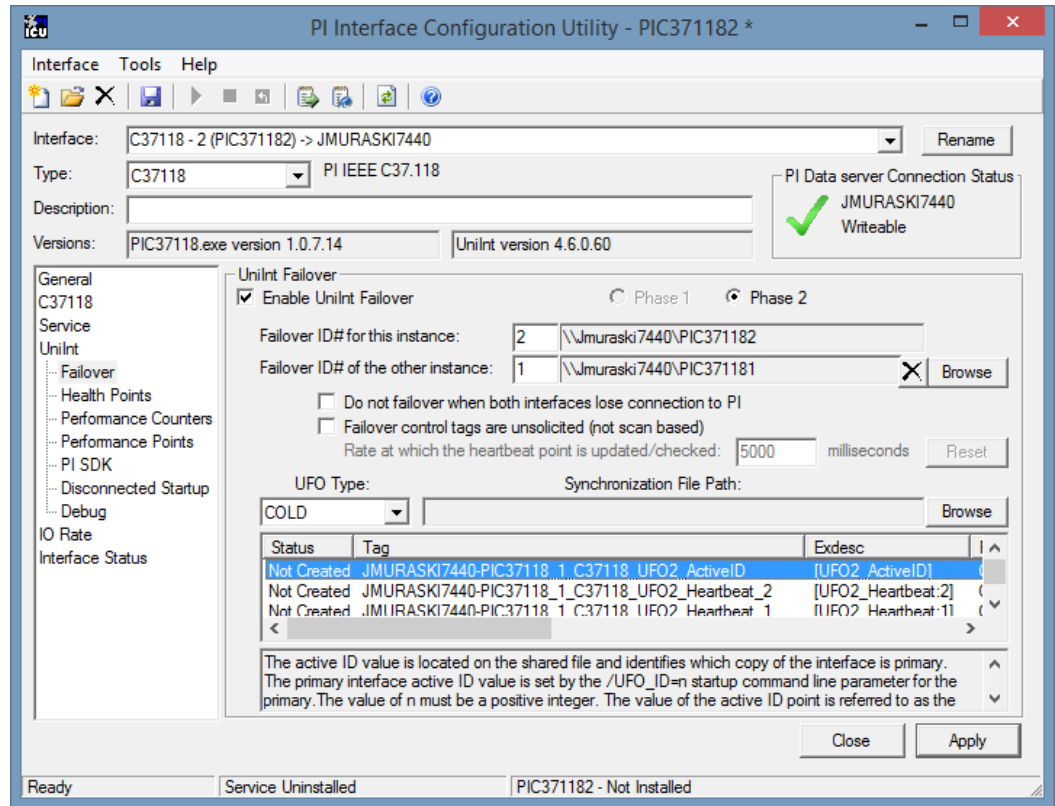


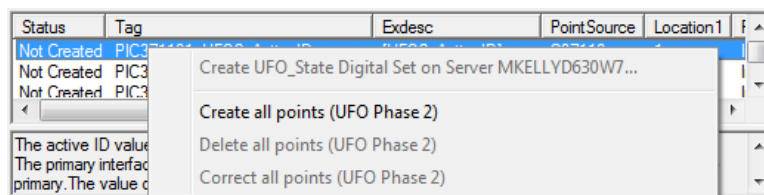
Figure 5: This figure illustrates the PI ICU failover configuration screen showing the UniInt failover startup parameters (Phase 2). This copy of the interface defines its Failover ID as 2 (`/UFO_ID=2`) and the other interfaces Failover ID as 1 (`/UFO_OtherID=1`). The other failover interface copy must define its Failover ID as 1 (`/UFO_ID=1`) and the other interface Failover ID as 2 (`/UFO_OtherID=2`) in its ICU failover configuration screen. It also defines the location and name of the synchronization file as well as the type of failover as COLD.

Creating the Failover State Digital State Set

The `UFO_State` digital state set is used in conjunction with the failover state digital point. If the `UFO_State` digital state set has not been created yet, it can be created using either the *Failover* page of the ICU (1.4.1.0 or later) or the Digital States plug-in in the SMT 3 Utility (3.0.0.7 or later).

Using the PI ICU Utility to create Digital State Set

To use the UniInt *Failover* page to create the `UFO_State` digital state set, right-click on any of the failover points listed in the *Tag* column and then click the *Create UFO_State Digital Set on Server XXXXXX...* command, where XXXXXX is the PI Data Archive where the points will be or are created.



This command is unavailable if the UFO_State digital state set already exists on the XXXXXX PI Data Archive.

Using the PI SMT 3 Utility to create Digital State Set

Optionally the *Export UFO_State Digital Set (.csv)* command on the shortcut menu can be selected to create a comma-separated file to be imported via the System Management Tools (SMT3) (version 3.0.0.7 or higher) or use the

UniInt_Failover_DigitalSet_UFO_State.csv file included in the installation kit.

The procedure below outlines the steps necessary to create a digital set on a PI Data Archive using the *Import from File* command found in the SMT3 application. The procedure assumes the user has a basic understanding of the SMT3 application.

1. Open the SMT3 application.
2. Select the appropriate PI Data Archive from the PI Data Archives window. If the desired server is not listed, add it using the PI Connection Manager. A view of the SMT application is shown in Figure 7 below.
3. From the *System Management Plug-Ins* window, expand *Points* then select *Digital States*. A list of available digital state sets will be displayed in the main window for the selected PI Data Archive. Refer to Figure 7 below.
4. In the main window, right-click on the desired server and select the *Import from File* command. Refer to Figure 7 below.

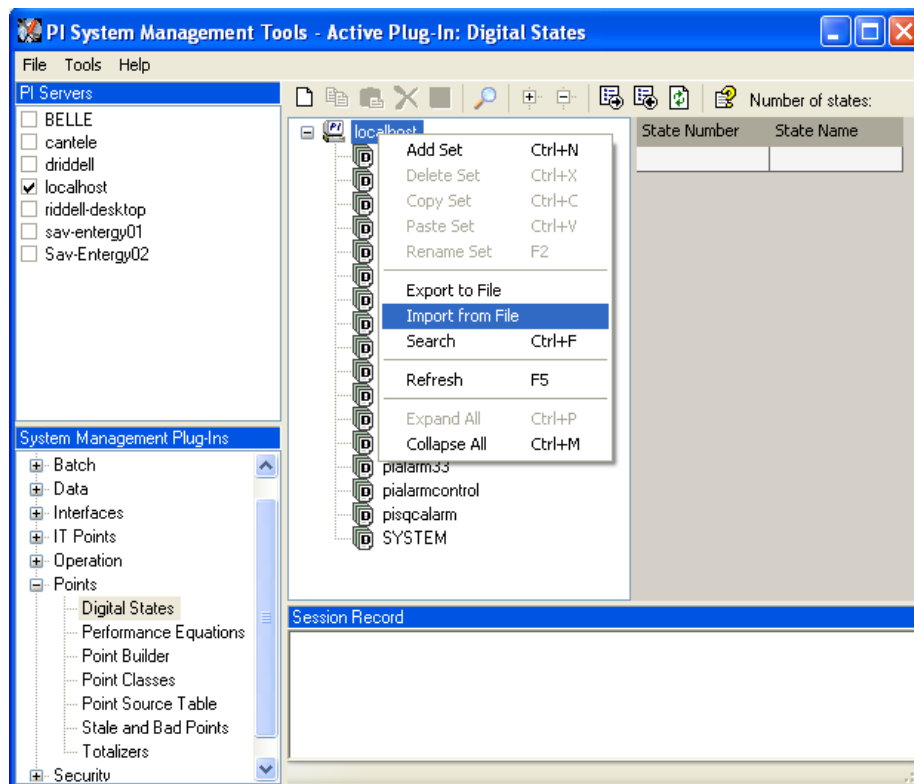


Figure 7: PI SMT application configured to import a digital state set file. The PI Data Archives window shows the “localhost” PI Data Archive selected along with the System Management Plug-Ins window showing the Digital States Plug-In as being selected. The digital state set file can now be imported by selecting the *Import from File* command.

5. Navigate to and select the `UniInt_Failover_DigitalSet_UFO_State.csv` file for import using the Browse icon on the display. Select the desired *Overwrite Options*. Refer to Figure below.

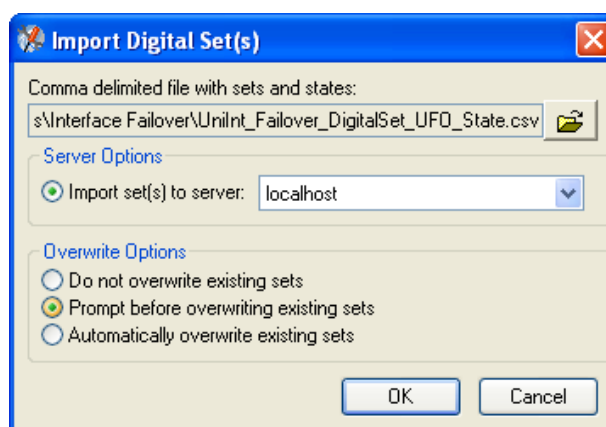


Figure 8: PI SMT application *Import Digital Set(s)* window. This view shows the `UniInt_Failover_DigitalSet_UFO_State.csv` file as being selected for import. Select the desired *Overwrite Options* by choosing the appropriate option button.

6. Click on the *OK* button. Refer to Figure above.

7. The UFO_State digital set is created as shown in Figure below.

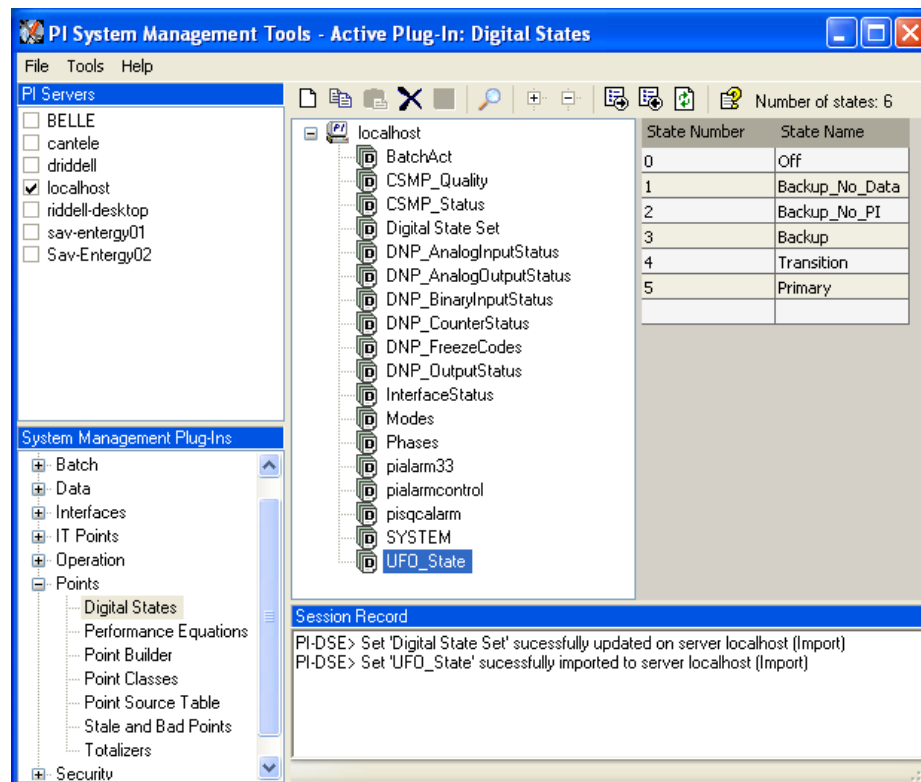


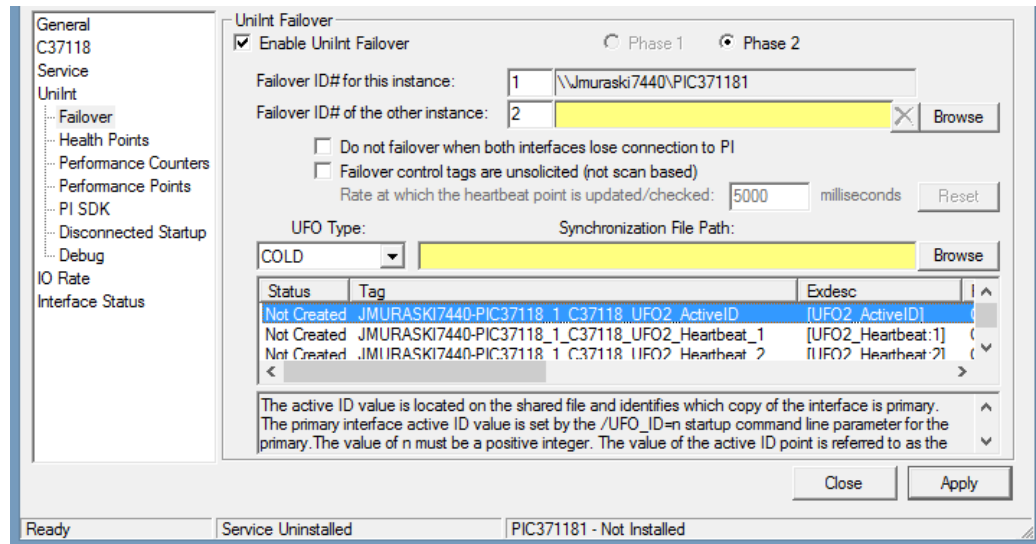
Figure 9: The PI SMT application showing the UFO_State digital set created on the “localhost” PI Data Archive.

Creating the Unint Failover Control and Failover State Points (Phase 2)

The ICU can be used to create the UniInt failover control and state points.

To use the ICU *Failover* page to create these points, simply right-click any of the failover points in the list and click the *Create all points (UFO Phase 2)* command.

If this menu command is unavailable, it is because the UFO_State digital state set has not been created in the PI Data Archive yet. *Create UFO_State Digital Set on Server xxxxxx...* on the shortcut menu can be used to create that digital state set. After this has been done then the *Create all points (UFO Phase2)* command should be available.



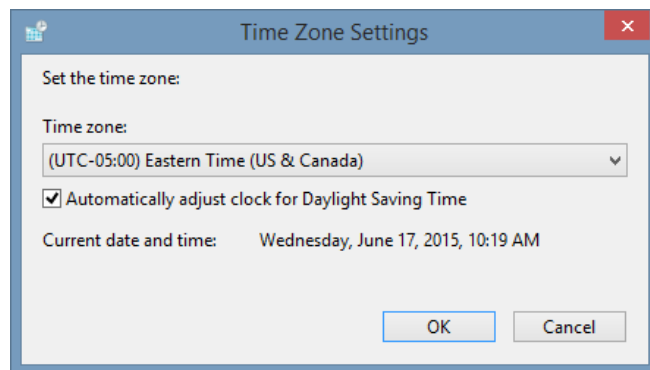
Once the failover control and failover state tags have been created the *Failover* page of the ICU should look similar to the illustration below.

Status	Tag	Exdesc
Created	JMUURASKI7440-PIC37118_1_C37118_UFO2_ActiveID	[UFO2_ActiveID]
Created	JMUURASKI7440-PIC37118_1_C37118_UFO2_Heartbeat_1	[UFO2_Heartbeat:1]
Created	JMUURASKI7440-PIC37118_1_C37118_UFO2_Heartbeat_2	[UFO2_Heartbeat:2]

The active ID value is located on the shared file and identifies which copy of the interface is primary. The primary interface active ID value is set by the /UFO_ID=n startup command line parameter for the primary. The value of n must be a positive integer. The value of the active ID point is referred to as the

Chapter 10. Interface Node Clock

Make sure that the time and time zone settings on the computer are correct. To confirm, run the Date/Time applet located in the Windows Control Panel. If the locale where the interface node resides observes Daylight Saving Time, check the *Automatically adjust clock for daylight saving changes* box. For example,



In addition, make sure that the TZ environment variable is not defined. All of the currently defined environment variables can be viewed by opening a Command Prompt window and typing `set`. That is,

```
C:> set
```

Confirm that TZ is not in the resulting list. If it is, run the System applet of the Control Panel, click the *Environment Variables* button under the *Advanced* tab, and remove TZ from the list of environment variables.

Chapter 11. Security

Windows

The PI Firewall Database and the PI Trust Database must be configured so that the interface is allowed to write data to the PI Data Archive.

The Trust Database, which is maintained by the PI Base Subsystem, replaces the Proxy Database used prior to PI Data Archive version 3.3. The PI Trust Database maintains all the functionality of the proxy mechanism while being more secure.

See “Manage Interface Authentication with PI Trusts” in the chapter “Manage Security” of the *PI Server Introduction to System Management Guide*.

If the interface cannot write data to the PI Data Archive because it has insufficient privileges, a -10401 error will be reported in the log file. If the interface cannot send data to a PI2 Data Archive, it writes a -999 error. See the section [Appendix A. Error and Informational Messages](#) for additional information on error messaging.

Authentication

Interface instances are usually configured to run as Windows services. Since a service runs in a non-interactive context, a PI trust is required to authenticate the interface service to the PI Data Archive. A PI trust is associated with one PI identity, PI user, or PI group. When an interface successfully authenticates through a trust, the interface is granted the access rights for the associated identity, user, or group.

OSIsoft discourages using highly-privileged identities, users, or groups in PI trusts for interfaces.



Security Note: Avoid using the piadmin super-user and piadmins group. The recommended best practice for PI Data Archive security is to create an identity, user, or group that has only the access rights which are necessary for the interface to operate.

PI Data Archive v3.3 and Later

Security Configuration using Trust Editor

The Trust Editor plug-in for PI System Management Tools edits the PI trust table.

See the “Manage Interface Authentication with PI Trusts” section in the PI Server Introduction to System Management manual for more details on security configuration.

Security configuration using piconfig

For PI Data Archive v3.3 and higher, the following example demonstrates how to edit the PI trust table with ipconfig.

```
C:\PI\adm> ipconfig
@table pitrust
@mode create
@istr Trust,IPAddr,NetMask,PIUser
a_trust_name,192.168.100.11,255.255.255.255,trust_identity
@quit
```

For the preceding example,

Trust: An arbitrary name for the trust table entry; in the above example,

a_trust_name

IPAddr: the IP address of the computer running the interface; in the above example,

192.168.100.11

NetMask: the network mask; *255.255.255.255* specifies an exact match with IPAddr

PIUser: the PI identity, user, or group the interface is entrusted as; in the example,

trust_identity

PI Data Archive v3.2

For PI Data Archive v3.2, the following example demonstrates how to edit the PI Proxy table:

```
C:\PI\adm> p100ointed100@table pi_gen,piproxy
@mode create
@istr host,proxyaccount
piapimachine,piadmin
@quit
```

In place of *piapimachine*, put the name of the interface node as it is seen by the PI Data Archive.

Authorization

For an interface instance to start and write data to PI points, the following permissions must be granted to the PI identity, user, or group in the PI trust that authenticates the interface instance.

Database Security	Permission	Notes
PIPOINT	r	

Point Database	Permission	Notes
PtSecurity	r	
DataSecurity	r,w	Unbuffered
	r	Buffered (the buffering application requires r,w for the interface points)

The permissions in the preceding table must be granted for every PI point that is configured for the interface instance. Observe that buffering on the interface node is significant to PI point permissions.

When the interface instance is running on an unbuffered interface node, the interface instance sends PI point updates directly to the PI Data Archive. Therefore, the DataSecurity attribute must grant write access to the PI identity, user, or group in the PI trust that authenticates the interface instance.

When the interface instance is running on a buffered interface node, the interface instance sends PI point updates to the local buffering application, which relays the PI point updates to the PI Data Archive. The buffering application is a separate client to the PI Data Archive and, therefore, authenticates independently of the interface instances. The DataSecurity attribute must grant write access to the PI identity, user, or group in the PI trust that authenticates the buffering application.

Chapter 12. Starting / Stopping the Interface


This section describes starting and stopping the interface once it has been installed as a service. See the *PI Universal Interface (UniInt) User Guide* to run the interface interactively.



Starting Interface as a Service

If the interface was installed as service, it can be started from PI ICU, the Services control panel or with the command:

```
PIC37118.exe / start [ /serviceid id ]
```

To start the interface service with PI ICU, use the  button on the PI ICU toolbar.

A message will inform the user of the status of the interface service. Even if the message indicates that the service has started successfully, double check through the Services control panel applet. Services may terminate immediately after startup for a variety of reasons, and one typical reason is that the service is not able to find the command-line parameters in the associated .bat file. Verify that the root name of the .bat file and the .exe file are the same, and that the .bat file and the .exe file are in the same directory. Further troubleshooting of services might require consulting the log file, Windows Event Viewer, or other sources of log messages. See the section [Appendix A. Error and Informational Messages](#) for additional information.


Stopping Interface Running as a Service

If the interface was installed as service, it can be stopped at any time from PI ICU, the Services control panel or with the command:

```
PIC37118.exe /stop [ /serviceid id ]
```

The service can be removed by:

```
PIC37118.exe /remove [ /serviceid id ]
```

To stop the interface service with PI ICU, use the  button on the PI ICU toolbar.

Chapter 13. Buffering for PI Interfaces

The interface node uses buffering to prevent data loss when PI Data Archive is not available. UniInt interfaces can buffer data to store point values when network communication to the PI Data Archive is unavailable. UniInt disconnected startup requires buffering, and it is highly recommended for failover. Buffering for interfaces is configured and enabled through PI ICU.

Buffering services

The PI System offers two services to implement buffering at interfaces:

- PI Buffer Subsystem (PIBufss)
- API Buffer Server (Bufserv)

PI Buffer Subsystem is the best option for most environments.

Use API Buffer Server only if one or more of the following conditions is true:

- The version of PI Data Archive receiving the buffered data is earlier than 3.4.375
- Your interfaces run on a non-Windows platform

If any of the above conditions apply to you, see the *PI Buffering Manager Help* (*PIPC/HELP/BufferManager.chm*) documentation for PI Server. Otherwise, use PI Buffer Subsystem.

Buffering and collectives

PI Buffer Subsystem 4.3 and later can buffer data to multiple independent servers, including those configured as PI Server collectives. For interfaces to use PI Buffer Subsystem with PI Server collectives, the PI Data Archive servers must be running PI Data Archive version 3.4.375 or later.



Caution:

API Buffer Server does not detect and validate the PI Server collective configuration. As a result, API Buffer Server requires manual configuration whenever a collective changes. Also, because API Buffer Server results in compression at the PI Data Archive machine, archives at different servers in the collective could contain different records.

Buffering configuration

Use PI Interface Configuration Utility (PI ICU) to configure interface buffering.

The **Tools > Buffering** option helps you configure buffering. Depending on your current configuration, this option does one of the following:

- If this computer is configured to buffer data using PI Buffer Subsystem 4.3 or later, the Buffering Manager window opens and shows a buffering dashboard. The dashboard shows information about the status of buffering on this computer.
- If this computer is not currently configured to buffer data, and PI Buffer Subsystem 4.3 or later is installed, you are prompted to configure PI Buffer Subsystem. If you click **Yes**, the Buffering Manager window opens and shows the installation wizard, which helps you configure PI Buffer Subsystem.
- If this computer is configured to buffer data using API Buffer Server (Bufserv), and PI Buffer Subsystem 4.3 or later is installed, you are prompted to convert to and configure PI Buffer Subsystem. If you click **Yes** at both prompts, the Buffering Manager window opens and shows the upgrade wizard, which helps you upgrade from API Buffer Server to PI Buffer Subsystem.
- If PI Buffer Subsystem 4.3 is not yet installed, the Buffering window opens for API Buffering or the earlier version of PI Buffer Subsystem.

For PI Buffer Subsystem 4.3, when configuring an interface to buffer data to a PI Data Archive server which has not been added to the buffered server list you must enable buffering. Click the **Enable** button in the **Buffering Status** box on the interface General page. To verify that the buffering status is **On**, exit PI ICU, then restart and select the interface.

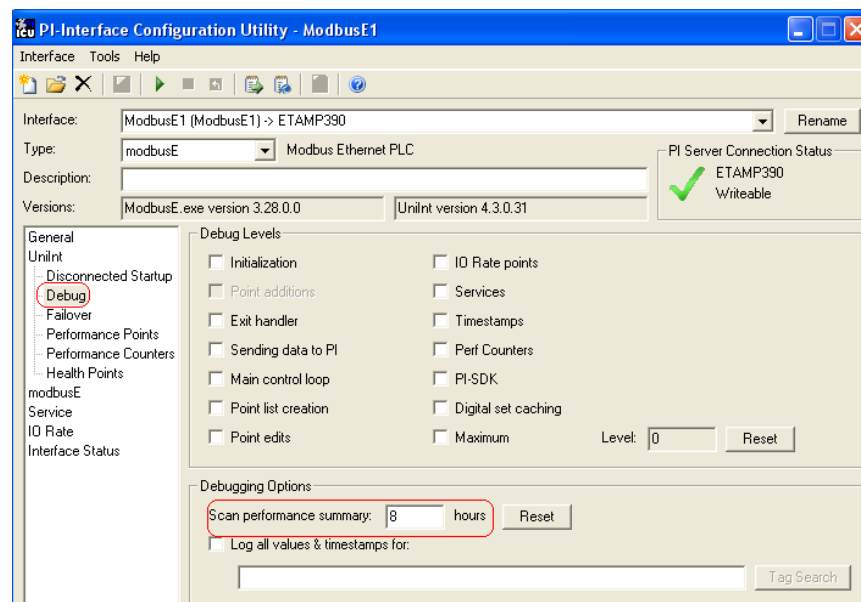
You can use Buffering Manager to configure, monitor, and troubleshoot buffering using PI Buffer Subsystem. PI Buffer Subsystem is recommended for applications connecting to PI Data Archive 3.4.375 or later. Older versions of PI Data Archive require API Buffer Server, as do some sites with custom solutions. See Buffering Manager help (*PIPC/HELP/BufferManager.chm*) for more information.

Chapter 14. Interface Diagnostics Configuration

The [PI Point Configuration](#) chapter provides information on building PI points for collecting data from the device. This chapter describes the configuration of points related to interface diagnostics.

Note: The procedure for configuring interface diagnostics is not specific to this interface. Thus, for simplicity, the instructions and screenshots that follow refer to an interface named **ModbusE**.

Some of the points that follow refer to a “performance summary interval”. This interval is 8 hours by default. You can change this parameter via the *Scan performance summary* box in the *Unint – Debug* parameter category page:

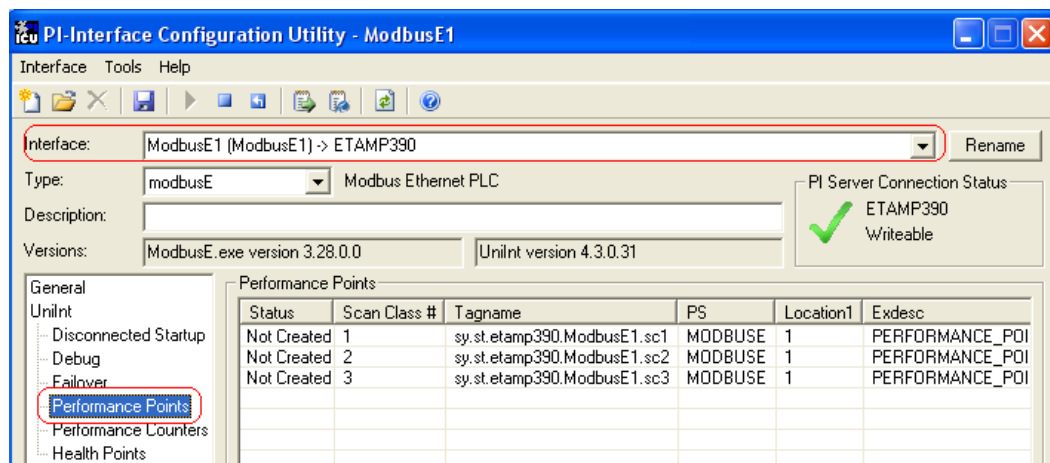


Scan Class Performance Points

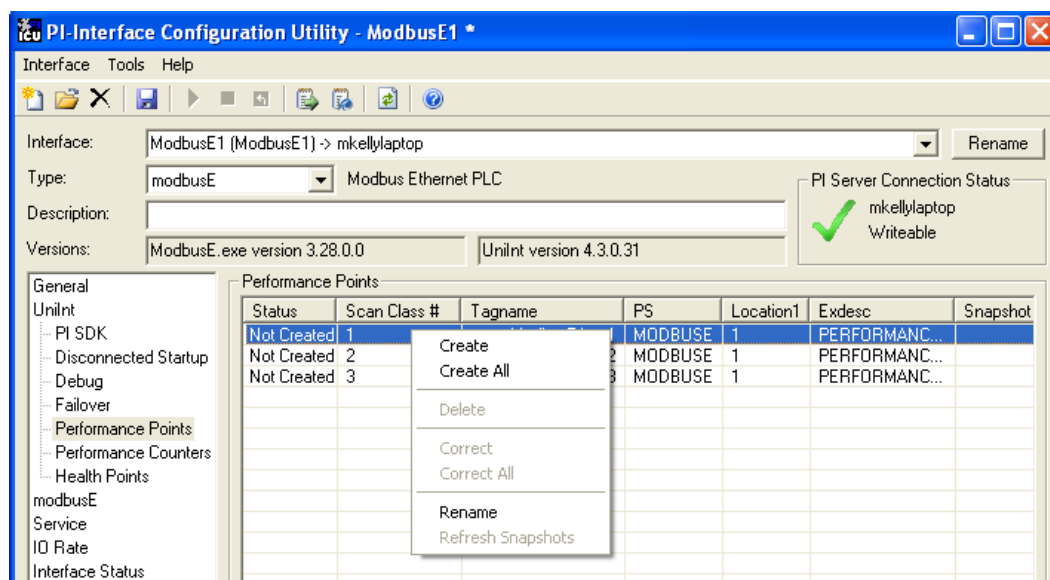
A scan class performance point measures the amount of time (in seconds) that this interface takes to complete a scan. The interface writes this scan completion time to millisecond resolution. Scan completion times close to 0 indicate that the interface is performing optimally. Conversely, long scan completion times indicate an increased risk of missed or skipped scans. To prevent missed or skipped scans, you should distribute the data collection points among several scan classes.

Interface Diagnostics Configuration

You configure one scan class performance point for each scan class in this interface. From the ICU, select this interface from the *Interface* drop-down list and click *UniInt-Performance Points* in the parameter category pane:



Right click the row for a particular *Scan Class #* to open the shortcut menu:



You need not restart the interface for it to write values to the scan class performance points.

To see the current values (snapshots) of the scan class performance points, right-click and select *Refresh Snapshots*.

Create / Create All

To create a performance point, right-click the line belonging to the point to be created, and select *Create*. Click *Create All* to create all the scan class performance points.

Delete

To delete a performance point, right-click the line belonging to the point to be deleted and select *Delete*.

Correct / Correct All

If the “Status” of a point is marked “Incorrect”, the point configuration can be automatically corrected by the ICU by right-clicking on the line belonging to the point to be corrected and selecting *Correct*. The performance points are created with the following PI attribute values.

Attribute	Details
Tag	Point name that appears in the list box
Point Source	Point Source for points for this interface, as specified on the <i>General</i> page
Compressing	Off
Excmax	0
Descriptor	<i>Interface name</i> + “ Scan Class # Performance Point”

If the ICU detects that a performance point is not defined with the correct attributes, the point will be marked *Incorrect*. To correct all points, click *Correct All*.

Rename

Right-click the line belonging to the point and select *Rename* to rename the performance point.

Column descriptions

Status

The *Status* column in the performance points table indicates whether the performance point exists for the scan class in the *Scan Class #* column.

Created – Indicates that the performance point does exist

Not Created – Indicates that the performance point does not exist

Deleted – Indicates that a performance point existed but was just deleted by the user

Scan Class #

The *Scan Class* column indicates to which scan class the performance point in the *Tagname* column belongs. There will be one scan class in the *Scan Class* column for each scan class listed in the *Scan Classes* box on the *General* page.

Tagname

The *Tagname* column holds the performance point name.

PS

This is the point source used for these performance points and the interface.

Location1

This is the value used by the interface for the */ID=#* point attribute.

ExDesc

This is used to tell the interface that these are performance points and the value is used to correspond to the `/ID=#` command line parameter if multiple copies of the same interface are running on the interface node.

Snapshot

The *Snapshot* column holds the snapshot value of each performance point that exists in the PI Data Archive. The *Snapshot* column is updated when the *Performance Points* page is selected, and when the interface is first loaded. You may have to scroll to the right to see the snapshots.

Performance Counters Points

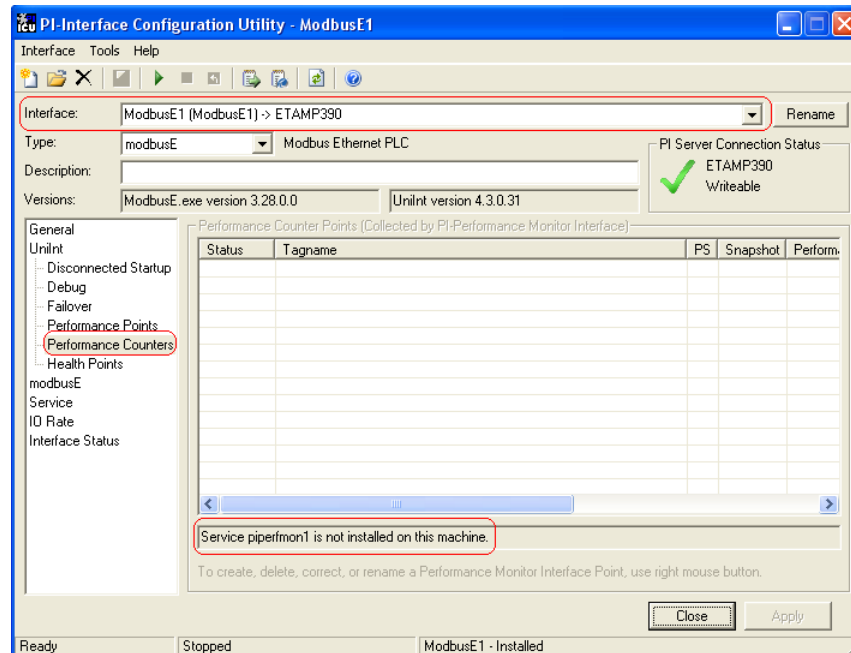
Installing the interface as a Windows service creates Windows performance counters to report performance data. Such data include items like:

- the amount of time that the interface has been running;
- the number of points the interface has added to its point list;
- the number of points that are currently updating among others

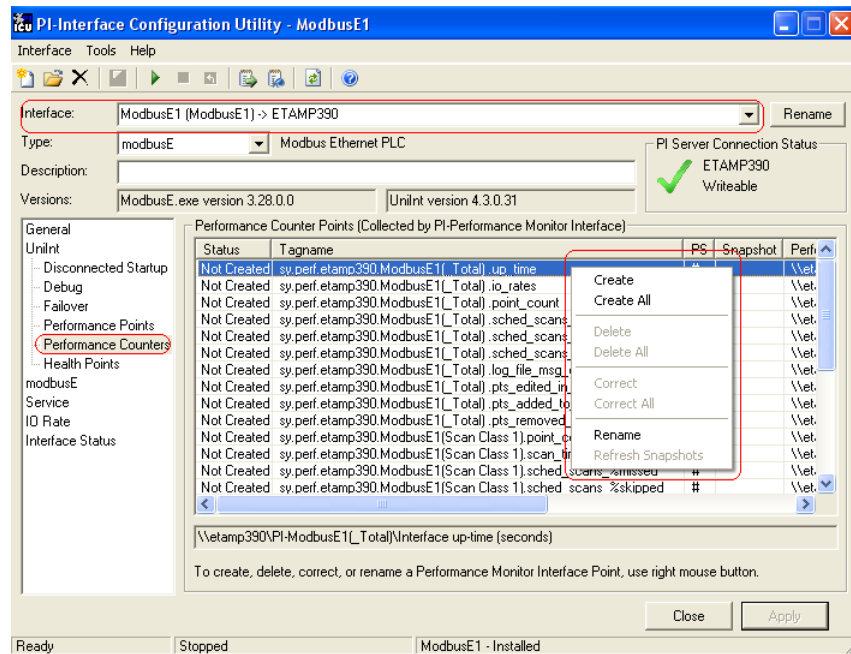
There are two types or instances of performance counters that can be collected and stored in PI points. The first is (*_Total*) which is a total for the performance counter since the interface instance was started. The other is for individual scan classes (Scan Class *x*) where *x* is a particular scan class defined for the interface instance that is being monitored.

OSIsoft's PI Interface for Performance Monitor is capable of reading these performance values and writing them to PI points. Please see the *PI Interface for Performance Monitor* for more information.

If there is no PI Interface for Performance Monitor registered with the ICU in the Module Database for the PI Data Archive the interface is sending its data to, you cannot use the ICU to create any interface instance's performance counters points:



After installing the PI Interface for Performance Monitor as a service, select this interface instance from the *Interface* drop-down list. Click *Performance Counters* in the parameter categories pane and then right-click on the row containing the performance counter point you wish to create. This will open the shortcut menu:



Click *Create* to create the performance counter point for that particular row. Click *Create All* to create all the performance counters points listed which have a status of Not Created.

To see the current values (snapshots) of the created performance counters points, right-click on any row and select *Refresh Snapshots*.

Note: The PI Interface for Performance Monitor – and not this interface – is responsible for updating the values for the performance counters points in the PI Data Archive. So, make sure that the PI Interface for Performance Monitor is running correctly.

Performance Counters

In the following lists of performance counters, the naming convention is:

“PerformanceCounterName” (.PerformanceCounterPointSuffix)

The name created by the ICU for each performance counter point is based on the setting found under the Tools → Options → Naming Conventions → Performance Counter Points. The default for this is “sy.perf.[machine].[if service]” followed by the performance counter point suffix.

Performance Counters for both (_Total) and (Scan Class x)

“Point Count” (.point_count)

A .sched_scans_%missed performance counter point is available for each scan class of this interface as well as a “_Total)” for the interface instance.

The .sched_scans_%missed performance counter point indicates the percentage of scans the interface missed per scan class or the total number missed for all scan classes since startup. A missed scan occurs if the interface performs the scan one second later than scheduled.

The ICU uses a naming convention such that the point containing “(Scan Class 1)” (for example, “sy.perf.etamp390.E1(Scan Class 1).point_count”) refers to scan class 1, “(Scan Class 2)” refers to scan class 2, and so on. The point containing “(_Total)” refers to the sum of all scan classes.

“Scheduled Scans: % Missed” (.sched_scans_%missed)

A .sched_scans_%skipped performance counter point is available for each scan class of this interface as well as a “_Total)” for the interface instance.

The .sched_scans_%skipped performance counter point indicates the percentage of scans the interface skipped per scan class or the total number skipped for all scan classes since startup. A skipped scan is a scan that occurs at least one scan period after its scheduled time. This point is similar to the [UI_SCSKIPPED] health point.

The ICU uses a naming convention such that the point containing “(Scan Class 1)” (for example, “sy.perf.etamp390.E1(Scan Class 1).sched_scans_%missed”) refers to scan class 1, “(Scan Class 2)” refers to scan class 2, and so on. The point containing “_Total)” refers to the sum of all scan classes.

“Scheduled Scans: % Skipped” (.sched_scans_%skipped)

A .sched_scans_%skipped performance counter point is available for each scan class of this interface as well as a “_Total)” for the interface instance.

The .sched_scans_%skipped performance counter point indicates the percentage of scans the interface skipped per scan class or the total number skipped for all scan classes since startup.

A skipped scan is a scan that occurs at least one scan period after its scheduled time. This point is similar to the [UI_SCSKIPPED] health point.

The ICU uses a naming convention such that the point containing "Scan Class 1)" (for example, "sy.perf.etamp390.E1(Scan Class 1).sched_scans_%skipped") refers to scan class 1, "Scan Class 2)" refers to scan class 2, and so on. The point containing "_Total)" refers to the sum of all scan classes.

“Scheduled Scans: Scan count this interval” (.sched_scans_this_interval)

A *.sched_scans_this_interval* performance counter point is available for each scan class of this interface as well as a "_Total)" for the interface instance.

The *.sched_scans_this_interval* performance counter point indicates the number of scans that the interface performed per performance summary interval for the scan class or the total number of scans performed for all scan classes during the summary interval. This point is similar to the [UI_SCSCANCOUNT] health point.

The ICU uses a naming convention such that the point containing "(Scan Class 1)" (for example, "sy.perf.etamp390.E1(Scan Class 1).sched_scans_this_interval") refers to scan class 1, "(Scan Class 2)" refers to scan class 2, and so on. The point containing "(_Total)" refers to the sum of all scan classes.

Performance Counters for (_Total) only

“Device Actual Connections” (.Device_Actual_Connections)

The *.Device_Actual_Connections* performance counter point stores the actual number of foreign devices currently connected and working properly out of the expected number of foreign device connections to the interface. This value will always be less than or equal to the Device Expected Connections counter.

“Device Expected Connections” (.Device_Expected_Connections)

The *.Device_Expected_Connections* performance counter point stores the total number of foreign device connections for the interface. This is the expected number of foreign device connections configured that should be working properly at runtime. If the interface can only communicate with 1 foreign device then the value of this counter will always be one. If the interface can support multiple foreign device connections then this is the total number of expected working connections configured for this interface.

“Device Status” (.Device_Status)

The *.Device_Status* performance counter point stores communication information about the interface and the connection to the foreign device(s). The value of this counter is based on the expected connections, actual connections and value of the **/PercentUp** command line option. If the device status is good then the value is '0'. If the device status is bad then the value is '1'. If the interface only supports connecting to 1 foreign device then the **/PercentUp** command line value does not change the results of the calculation. If for example the interface can connect to 10 devices and 5 are currently working then the value of the **/PercentUp** command line parameter is applied to determine the Device Status. If the value of the **/PercentUp** command line parameter is set to 50 and at least 5 devices are working then the DeviceStatus will remain good (that is, have a value of zero).

“Failover Status” (.Failover_Status)

The *.Failover_Status* performance counter point stores the failover state of the interface when configured for UniInt failover. The value of the counter will be ‘0’ when the interface is running as the primary interface in the failover configuration. If the interface is running in backup mode then the value of the counter will be ‘1’.

“Interface up-time (seconds)” (.up_time)

The *.up_time* performance counter point indicates the amount of time (in seconds) that this interface has been running. At startup the value of the counter is zero. The value will continue to increment until it reaches the maximum value for an unsigned integer. Once it reaches this value then it will start back over at zero.

“IO Rate (events/second)” (.io_rates)

The *.io_rates* performance counter point indicates the rate (in event per second) at which this interface writes data to its input points.

“Log file message count” (.log_file_msg_count)

The *.log_file_msg_count* performance counter point indicates the number of messages that the interface has written to the log file. This point is similar to the [UI_MSGCOUNT] health point.

“PI Status” (PI_Status)

The *.PI_Status* performance counter point stores communication information about the interface and the connection to the PI Data Archive. If the interface is properly communicating with the PI Data Archive then the value of the counter is ‘0’. If the communication to the PI Data Archive goes down for any reason then the value of the counter will be ‘1’. Once the interface is properly communicating with the PI Data Archive again then the value will change back to ‘0’.

“Points added to the interface” (.pts_added_to_interface)

The *.pts_added_to_interface* performance counter point indicates the number of points the interface has added to its point list. This does not include the number of points configured at startup. This is the number of points added to the interface after the interface has finished a successful startup.

“Points edited in the interface”(pts_edited_in_interface)

The *.pts_edited_in_interface* performance counter point indicates the number of point edits the interface has detected. The interface detects edits for those points whose PointSource attribute matches the /ps= parameter and whose Location1 attribute matches the /id= parameter of the interface.

“Points Good” (.Points_Good)

The *.Points_Good* performance counter point is the number of points that have sent a good current value to the PI Data Archive. A good value is defined as any value that is not a system digital state value. A point can either be Good, In Error, or Stale. The total of Points

Good, Points In Error, and Points State will equal the Point Count. There is one exception to this rule. At startup of an interface, the Stale timeout must elapse before the point will be added to the Stale Counter. Therefore the interface must be up and running for at least 10 minutes for all points to belong to a particular Counter.

“Points In Error” (.Points_In_Error)

The *.Points_In_Error* performance counter point indicates the number of points that have sent a current value to the PI Data Archive that is a system digital state value. Once a point is in the In Error count it will remain in the In Error count until the point receives a new, good value. Points categorized as in Error do not transition to the Stale Counter. Only good points become stale.

“Points removed from the interface” (.pts_removed_from_interface)

The *.pts_removed_from_interface* performance counter point indicates the number of points that have been removed from the interface configuration. A point can be removed from the interface when one of the point attributes is updated and the point is no longer a part of the interface configuration. For example, changing the PointSource, Location1, or Scan attribute can cause the point to no longer be a part of the interface configuration.

“Points Stale 10(min)” (.Points_Stale_10min)

The *.Points_Stale_10min* performance counter point indicates the number of good points that have not received a new value in the last 10 minutes. If a point is Good, then it will remain in the good list until the Stale timeout elapses. At this time if the point has not received a new value within the Stale Period then the point will move from the Good count to the Stale count. Only points that are Good can become Stale. If the point is in the In Error count then it will remain in the In Error count until the error clears. As stated above, the total count of Points Good, Points In Error, and Points Stale will match the Point Count for the interface.

“Points Stale 30(min)” (.Points_Stale_30min)

The *.Points_Stale_30min* performance counter point indicates the number of points that have not received a new value in the last 30 minutes. For a point to be in the Stale 30 minute count it must also be a part of the Stale 10 minute count.

“Points Stale 60(min)” (.Points_Stale_60min)

The *.Points_Stale_60min* performance counter point indicates the number of points that have not received a new value in the last 60 minutes. For a point to be in the Stale 60 minute count it must also be a part of the Stale 10 minute and 30 minute count.

“Points Stale 240(min)” (.Points_Stale_240min)

The *.Points_Stale_240min* performance counter point indicates the number of points that have not received a new value in the last 240 minutes. For a point to be in the Stale 240 minute count it must also be a part of the Stale 10 minute, 30 minute and 60 minute count.

Performance Counters for (Scan Class x) only

“Device Scan Time (milliseconds)” (.Device_Scan_Time)

A *.Device_Scan_Time* performance counter point is available for each scan class of this interface.

The *.Device_Scan_Time* performance counter point indicates the number of milliseconds the interface takes to read the data from the foreign device and package the data to send to the PI Data Archive. This counter does not include the amount of time to send the data to the Pi Data Archive. This point is similar to the [UI_SCINDEVSCANTIME] health point.

The ICU uses a naming convention such that the point containing “(Scan Class 1)” (for example, “*sy.perf.etamp390.E1 (Scan Class 1).device_scan _time*”) refers to scan class 1, “(Scan Class 2)” refers to scan class 2, and so on.

“Scan Time (milliseconds)” (.scan_time)

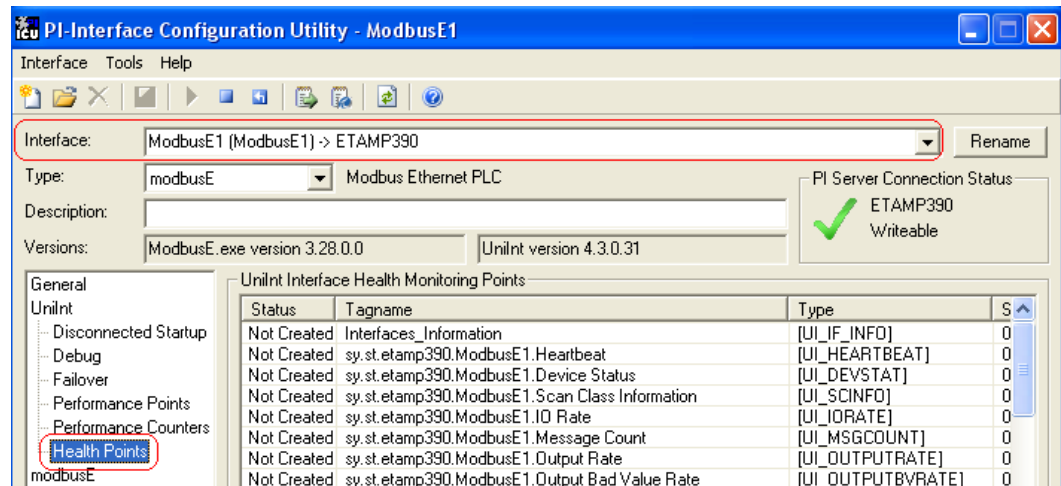
A *.scan_time* performance counter point is available for each scan class of this interface.

The *.scan_time* performance counter point indicates the number of milliseconds the interface takes to both read the data from the device and send the data to the PI Data Archive. This point is similar to the [UI_SCINSCANTIME] health point.

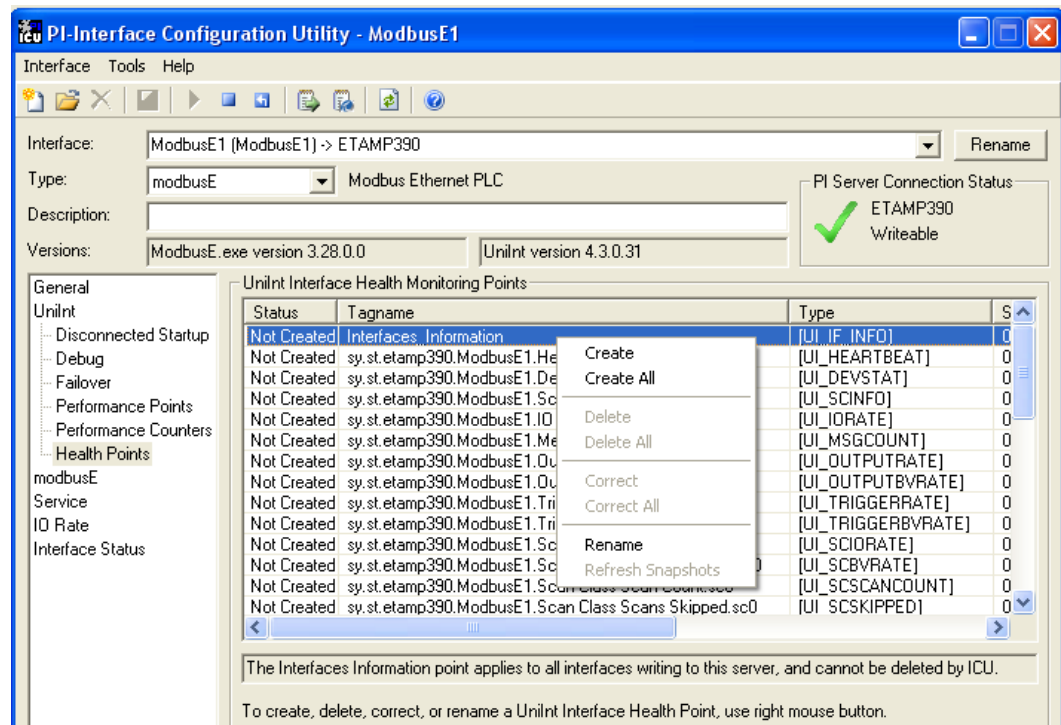
The ICU uses a naming convention such that the point containing “(Scan Class 1)” (for example, “*sy.perf.etamp390.E1 (Scan Class 1).scan_time*”) refers to scan class 1, “(Scan Class 2)” refers to scan class 2, and so on.

Interface Health Monitoring Points

Interface Health Monitoring Points provide information about the health of this interface. To use the ICU to configure these points, select this interface from the *Interface* drop-down list and click *Health Points* from the parameter category pane:



Right click the row for a particular health point to display the shortcut menu:



Click *Create* to create the health point for that particular row. Click *Create All* to create all the health points.

To see the current values (snapshots) of the health points, right-click and select *Refresh Snapshots*.

For some of the health points described subsequently, the interface updates their values at each performance summary interval (typically, 8 hours).

[UI_HEARTBEAT]

The [UI_HEARTBEAT] health point indicates whether the interface is currently running. The value of this point is an integer that increments continuously from 1 to 15. After reaching 15, the value resets to 1.

The fastest scan class frequency determines the frequency at which the interface updates this point:

Fastest Scan Frequency	Update frequency
Less than 1 second	1 second
Between 1 and 60 seconds, inclusive	Scan frequency
More than 60 seconds	60 seconds

If the value of the [UI_HEARTBEAT] health point is not changing, then this interface is in an unresponsive state.

[UI_DEVSTAT]

The [UI_DEVSTAT] health point provides an indication of the connection status between the interface and the source device. The possible values for this string point are:

- “1 | Starting | UI #.#.#” – The interface is currently starting up.
- “Good” – The interface is properly communicating and reading data from all configured PMUs. A value of “Good” does not mean that all points are receiving good values, but it is a good indication that there are no hardware or network problems.
- “3 | 1 device(s) in error | REP: 1 LEP 1” – The connection to one or more PMUs has failed.
- “4 | Intf Shutdown” – The interface has been shutdown.

Refer to the *PI Universal Interface (UniInt) User Guide.doc* file for more information on how to configure health points.

[UI_SCINFO]

The [UI_SCINFO] health point provides scan class information. The value of this point is a string that indicates

- the number of scan classes;
- the update frequency of the [UI_HEARTBEAT] health point ; and
- the scan class frequencies

An example value for the [UI_SCINFO] health point is:

3 | 5 | 5 | 60 | 120

The interface updates the value of this point at startup and at each performance summary interval.

[UI_IORATE]

The [UI_IORATE] health point indicates the sum of

1. the number of scan-based input values the interface collects before it performs exception reporting; and
2. the number of event-based input values the interface collects before it performs exception reporting

The interface updates this point at the same frequency as the [UI_HEARTBEAT] point. The value of this [UI_IORATE] health point may be zero. A stale timestamp for this point indicates that this interface has stopped collecting data.

[UI_MSGCOUNT]

The [UI_MSGCOUNT] health point tracks the number of messages that the interface has written to the log file since start-up. In general, a large number for this point indicates that the interface is encountering problems. You should investigate the cause of these problems by looking in log messages.

The interface updates the value of this point every 60 seconds. While the interface is running, the value of this point never decreases.

[UI_POINTCOUNT]

The [UI_POINTCOUNT] health point counts number of PI points loaded by the interface. This count includes all input and triggered input points. This count does NOT include any interface health points or performance points.

The interface updates the value of this point at startup, on change and at shutdown.

[UI_TRIGGERRATE]

The [UI_TRIGGERRATE] health point tracks the number of values that the interface writes to event-based input points. If there are no event-based input points for this interface, it writes the System Digital State `No Result` to this health point.

The interface updates this point at the same frequency as the [UI_HEARTBEAT] point. The interface resets the value of this point to zero at each performance summary interval.

[UI_TRIGGERBVRATE]

The [UI_TRIGGERBVRATE] health point tracks the number of system digital state values that the interface writes to event-based input points. If there are no event-based input points for this interface, it writes the system digital state `No Result` to this health point.

The interface updates this point at the same frequency as the [UI_HEARTBEAT] point. The interface resets the value of this point to zero at each performance summary interval.

[UI_SCIORATE]

You can create a [UI_SCIORATE] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “.sc1” (for example, `sy.st.etamp390.El.Scan Class IO Rate.sc1`) refers to scan class 1, “.sc2” refers to scan class 2, and so on.

A particular scan class's [UI_SCIORATE] health point indicates the number of values that the interface has collected. If the current value of this point is between zero and the corresponding [UI_SCPOINTCOUNT] point, inclusive, then the interface executed the scan successfully. If a [UI_SCIORATE] point stops updating, then this condition indicates that an error has occurred and the points in the scan class are no longer receiving new data.

The interface updates the value of a [UI_SCIORATE] point after the completion of the associated scan.

Although the ICU allows you to create the point with the suffix “.sc0”, this point is not applicable to this interface.

[UI_SCBVRATE]

You can create a [UI_SCBVRATE] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “.sc1” (for example,

`sy.st.etamp390.E1.Scan Class Bad Value Rate.sc1`) refers to scan class 1, “.sc2” refers to scan class 2, and so on.

A particular scan class's [UI_SCBVRATE] point indicates the number System Digital State values that the interface has collected.

The interface updates the value of a [UI_SCBVRATE] point after the completion of the associated scan.

Although the ICU allows you to create the point with the suffix “.sc0”, this point is not applicable to this interface.

[UI_SCSCANCOUNT]

You can create a [UI_SCSCANCOUNT] health point for each scan class in this interface.

The ICU uses a point naming convention such that the suffix “.sc1” (for example,

`sy.st.etamp390.E1.Scan Class Scan Count.sc1`) refers to scan class 1, “.sc2” refers to scan class 2, and so on.

A particular scan class's [UI_SCSCANCOUNT] point tracks the number of scans that the interface has performed.

The interface updates the value of this point at the completion of the associated scan. The interface resets the value to zero at each performance summary interval.

Although there is no “Scan Class 0”, the ICU allows you to create the point with the suffix “.sc0”. This point indicates the total number of scans the interface has performed for all of its Scan Classes.

[UI_SCSKIPPED]

You can create a [UI_SCSKIPPED] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “.sc1” (for example,

`sy.st.etamp390.E1.Scan Class Scans Skipped.sc1`) refers to scan class 1, “.sc2” refers to scan class 2, and so on.

A particular scan class's [UI_SCSKIPPED] point tracks the number of scans that the interface was not able to perform before the scan time elapsed and before the interface performed the next scheduled scan.

The interface updates the value of this point each time it skips a scan. The value represents the total number of skipped scans since the previous performance summary interval. The interface resets the value of this point to zero at each performance summary interval.

Although there is no “Scan Class 0”, the ICU allows you to create the point with the suffix “.sc0”. This point monitors the total skipped scans for all of the interface’s Scan Classes.

[UI_SCPOINTCOUNT]

You can create a [UI_SCPOINTCOUNT] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “sc1” (for example, `sy.st.etamp390.El.Scan Class Point Count.sc1`) refers to scan class 1, “sc2” refers to scan class 2, and so on.

This health point monitors the number of points in a scan class.

The interface updates a [UI_SCPOINTCOUNT] health point when it performs the associated scan.

Although the ICU allows you to create the point with the suffix “.sc0”, this point is not applicable to this interface.

[UI_SCINSCANTIME]

You can create a [UI_SCINSCANTIME] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “sc1” (for example, `sy.st.etamp390.El.Scan Class Scan Time.sc1`) refers to scan class 1, “sc2” refers to scan class 2, and so on.

A particular scan class’ [UI_SCINSCANTIME] point represents the amount of time (in milliseconds) the interface takes to read data from the device, fill in the values for the points, and send the values to the PI Data Archive.

The interface updates the value of this point at the completion of the associated scan.

[UI_SCINDEVSCANTIME]

You can create a [UI_SCINDEVSCANTIME] health point for each scan class in this interface. The ICU uses a point naming convention such that the suffix “sc1” (for example, `sy.st.etamp390.El.Scan Class Device Scan Time.sc1`) refers to scan class 1, “sc2” refers to scan class 2, and so on.

A particular scan class’ [UI_SCINDEVSCANTIME] point represents the amount of time (in milliseconds) the interface takes to read data from the device and fill in the values for the points.

The value of a [UI_SCINDEVSCANTIME] point is a fraction of the corresponding [UI_SCINSCANTIME] point value. You can use these numbers to determine the percentage of time the interface spends communicating with the device compared with the percentage of time communicating with the PI Data Archive.

If the [UI_SCSKIPPED] value is increasing, the [UI_SCINDEVSCANTIME] points along with the [UI_SCINSCANTIME] points can help identify where the delay is occurring: whether the reason is communication with the device, communication with the PI Data Archive, or elsewhere.

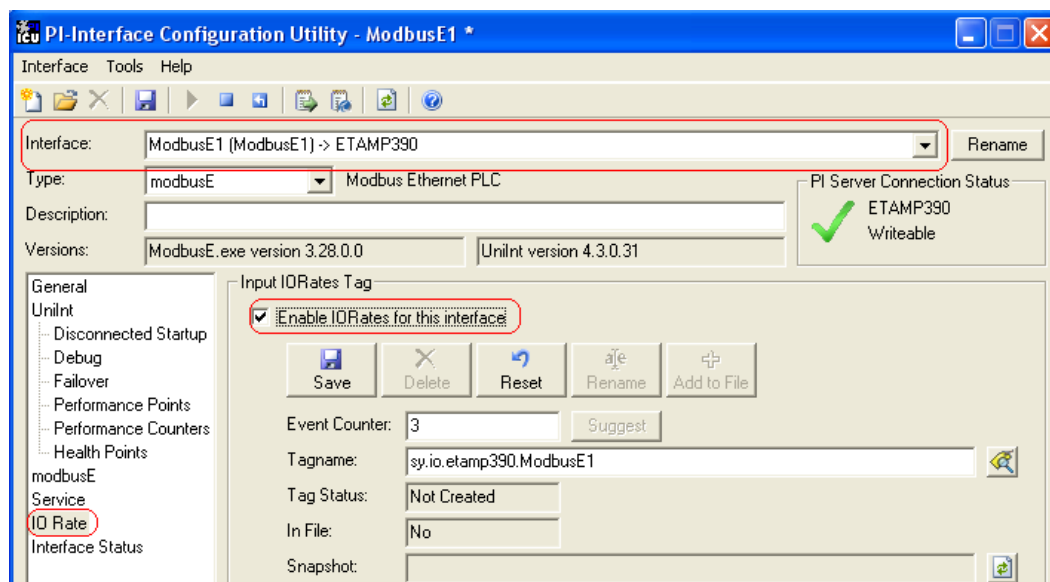
The interface updates the value of this point at the completion of the associated scan.

I/O Rate Point

An I/O Rate point measures the rate at which the interface writes data to its input points. The value of an I/O Rate point represents a 10-minute average of the total number of values per minute that the interface sends to the PI Data Archive.

When the interface starts, it writes 0 to the I/O Rate point. After running for ten minutes, the interface writes the I/O Rate value. The interface continues to write a value every 10 minutes. When the interface stops, it writes 0.

The ICU allows you to create one I/O Rate point for each copy of this interface. Select this interface from the *Interface* drop-down list, click *IO Rate* in the parameter category pane, and check *Enable IORates for this interface*.



As the preceding picture shows, the ICU suggests an *Event Counter* number and a *Tagname* for the I/O Rate point. Click the *Save* button to save the settings and create the I/O Rate point. Click the *Apply* button to apply the changes to this copy of the interface.

You need to restart the interface in order for it to write a value to the newly created I/O Rate point. Restart the interface by clicking the *Restart* button:



(The reason you need to restart the interface is that the *PointSource* attribute of an I/O Rate point is *Lab*.)

To confirm that the interface recognizes the I/O Rate point, look in the log file for a message such as:

```
PI-ModBus 1> IORATE: tag sy.io.etamp390.ModbusE1 configured.
```

To see the I/O Rate point's current value (snapshot), click the *Refresh* button of the *Snapshot* box:

Enable IORates for this Interface

The *Enable IORates for this interface* check box enables or disables I/O Rates for the current interface. To disable I/O Rates for the selected interface, uncheck this box. To enable I/O Rates for the selected interface, check this box.

Event Counter

The *Event Counter* correlates a point specified in the *iorates.dat* file with this copy of the interface. The command-line equivalent is */ec=x*, where *x* is the same number that is assigned to a point name in the *iorates.dat* file.

Tagname

The name listed in the *Tagname* box is the name of the I/O Rate point.

Tag Status

The *Tag Status* box indicates whether the I/O Rate point exists in the PI Data Archive. The possible states are:

- Created – This status indicates that the point exist in the PI Data Archive
- Not Created – This status indicates that the point does not yet exist in the PI Data Archive
- Deleted – This status indicates that the point has just been deleted
- Unknown – This status indicates that the PI ICU is not able to access the PI Data Archive

In File

The *In File* box indicates whether the I/O Rate point listed in the *Tagname* box and the number in the *Event Counter* is in the *IORates.dat* file. The possible states are:

- Yes – This status indicates that the point name and event counter are in the *IORates.dat* file
- No – This status indicates that the point name and event counter are not in the *IORates.dat* file

Snapshot

The *Snapshot* column holds the snapshot value of the I/O Rate point, if the I/O Rate point exists in the PI Data Archive. The *Snapshot* box is updated when the *IORate* page is selected, and when the interface is first loaded.

Create/Save

Create the suggested I/O Rate point with the name indicated in the Tagname box. Or, save any changes for the point name indicated in the Tagname box.

Delete

Delete the I/O Rate point listed in the *Tagname* box.

Rename

Change the name for the I/O Rate point listed in the *Tagname* box.

Add to File

Add the point indicated by the name in the Tagname box to the IORates.dat file with the event counter listed in the *Event Counter* box.

Search

Search the PI Data Archive for a previously defined I/O Rate point.

Appendix A. Error and Informational Messages

A string *NameID* is pre-pended to error messages written to the message log. *Name* is a non-configurable identifier that is no longer than 9 characters. *ID* is a configurable identifier that is no longer than 9 characters and is specified using the */id* parameter on the startup command-line.

Message Logs

The location of the message log depends upon the platform on which the interface is running. See the *PI Universal Interface (UniInt) User Guide* for more information.

For more information about logs for interfaces running on Windows, see *UniInt Interface Message Logging for UniInt 4.5.0.x and later Interfaces* or [OISsoft KB article - KB00401 - How to read new UniInt 4.5.0.x and later Interface message logs](#) on the OSIsoft technical support web site.

Messages are written to `[PIHOME]\dat\pipc.log` at the following times.

- When the interface starts many informational messages are written to the log. These include the version of the interface, the version of UniInt, the command-line parameters used, and the number of points.
- As the interface loads points, messages are sent to the log if there are any problems with the configuration of the points.
- If the UniInt `/dbUniInt` parameter is found in the command-line, then various informational messages are written to the log file.

Messages

Interface Configuration Error Messages

Message	ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Couldn't create XML DOM Schema instance, make sure MSXML 4.0 is installed. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Couldn't create XML DOM Document instance, make sure MSXML 4.0 is installed. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Couldn't create XML DOM Schema Collection object, make sure MSXML 4.0 is installed.
Cause	There was a problem created one of the DOM XML objects. Typically indicates that the Microsoft MSXML 4.0 API is not installed.
Resolution	Download the API from Microsoft and install.

Message	ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - XML config document load failed: System error: -2146697210.
Cause	The specified XML file could not be loaded.
Resolution	Check the file name and path specified via the /confile argument.

Message	ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Couldn't find root <PI_C37_118.Session> element in the XML configuration document.
Cause	The root node root <PI_C37_118.Session> couldn't be located.
Resolution	Check the format of the XML file. If using the PIC37118 ICU (recommended) this should be correct.

Message	ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for DataSortingAction must be 0, 1, 2 or 3. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for SyncErrorAction must be 0, 1, 2 or 3. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for PMUErrorAction must be 0, 1, 2 or 3. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for DataSortingAction must be 0, 1, 2 or 3. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for LeapSecondAction must be 0, 1, 2 or 3. Or ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - Value for WriteIOTimeout must be 0 or 1.
Cause	The value specified for the one of the above attributes in the root section of the XML configuration file is out of range.
Resolution	Use PIC37118 ICU to ensure this value is correct.

Message	ERROR> Hardware initialization - C37118XMLConfig::C37118XMLConfig - No <LocalEndPoint> elements found in the PIC37118_Arbiter_Example.xml XML configuration document.
Cause	No <LocalEndPoints> section was found in the configuration document.
Resolution	The must be at least one LocalEndPoints section in the file.
Message	LocalEndPoint::LocalEndPoint(-1) - ID attribute missing from LocalEndPoint definition.
Cause	The required ID attribute is missing from a LocalEndPoint section.
Resolution	Add the ID attribute to the LocalEndPoint definition.

Message	LocalEndPoint::LocalEndPoint(2) - Type element in LocalEndpoint configuration must be IP or Serial.
Cause	The Type attribute in LocalEndPoint with ID of 2 has invalid communications type specified.
Resolution	Use the PIC37118 ICU to ensure this attribute is correct.

Message	LocalEndPoint::LocalEndPoint(2) - Invalid data protocol specified in <DataProtocol>, must be TCP or UDP(case insensitive). Or LocalEndPoint::LocalEndPoint(2) - Invalid data protocol specified in <ConfigProtocol>, must be TCP or UDP(case insensitive).
Cause	The DataProtocol or ConfigProtocol attribute must be TCP or UDP.
Resolution	Use the PIC37118 ICU to ensure this attribute is correct. This parameter is not case sensitive.

Point Loading Error Messages

The following messages are logged when there is a problem loading a point. In each case the point name and point id is reported along with the error information.

Message	LocalEndPoint Id: 1 not found in interface LocalEndPointCollection
Cause	There is not a LocalEndPoint defined in the XML configuration file that matches the Location2 attribute of the point.
Resolution	Add the appropriate LocalEndPoint.

Message	RemoteEndPoint Id: 1 not found in interface RemoteEndPointCollection
Cause	There is not a RemoteEndPoint defined in the XML configuration file that matches the Location3 attribute of the point.
Resolution	Add the appropriate RemoteEndPoint

Message	Delimiter character '\\' not found in InstrumentTag specification (PMU.FREQ)
Cause	The InstrumentTag attribute requires the Measurement Type and Measurement be delimited by a "/" character and was not found.
Resolution	Added appropriate delimiter.
Message	RemoteEndPoint tag of type LEAPSEC is already defined
Cause	There can only be a single point for a given RemoteEndPoint type(REP) for each RemoteEndPoint. For example there cannot be two points with an InstrumentTag field of REP/LEAPSEC configured for the same RemoteEndPoint. Any subsequent points are rejected.
Resolution	Remove duplicate point.

Message	PMU Id: 66 not found in interface PMU Collection
Cause	The PMU ID specified in the Location5 attribute was not found in the XML configuration file.
Resolution	Fix XML configuration file to include ID.

Message	Phasor measurement string XX is shorter than shortest valid keyword Or Invalid Phasor sub type (XXX) specified
Cause	The Phasor measurement specified in the InstrumentTag attribute is invalid. This value must be PhasorReal, PhasorImag, PhasorMag or PhasorAngle.
Resolution	Correct XML configuration file.

Message	Phasor channel (XXX) is not configured Or Analog channel (XXX) is not configured Or Digital channel (XXX) is not configured
Cause	The Phasor, analog or digital channel name is not valid for the PMU. This value is not case sensitive but imbedded spaces must remain if they are present in the channel names reported by the PMU. Trailing spaces may be removed.
Resolution	Correct XML configuration file.

Message	Phasor tag with channel name of XXX and sub-type of IMAG is already defined for PMU Or Analog tag with channel name of XXX is already defined for PMU Or Digital tag with channel name of XXX is already defined for PMU
Cause	There can only be one PI point configured for each Analog or Digital channel. Each Phasor channel can have one PI point for each sub type on a channel.
Resolution	Correct by removing the duplicate point entry.

Message	Invalid PMU measurement type (XXX) specified
Cause	The PMU measurement specified in the InstrumentTag is not valid. For example: PMU/XXX is not valid but PMU/FREQ is.
Resolution	Correct the invalid measurement type.

Communications Related Error Messages

The following messages are logged when there is a problem communications to the C37.118 device. In each case the communications parameters are reported along with the error information.

Message	PDC/PMU communication timeout.
Cause	The communications with a PDC or PMU have timed out. The interface will continue to attempt a reconnection.

Message	Error code: -1, Generic Error
Cause	The interface was unable to determine the exact nature of the error. Any available information is displayed.

Error and Informational Messages

Message	Debug code: -2, Debug Message.
Cause	Miscellaneous debug message.

Message	Error code: -3, Truncated String.
Cause	The C37.118 message was truncated.

Message	Error code: -4, Failed to create communication instance.
Cause	The interface was unable to create a communications thread. Check system resources.
Resolution	Check system resources.

Message	Error code: -5, Bad memory allocation.
Cause	The UFO_ID pa
Resolution	Check system resources.

Message	Error code: -6, Required parameter passed a NULL pointer.
Cause	An internal function passed a NULL pointer for a required variable. This should never occur.
Resolution	Contact OSIsoft technical support.

Message	Error code: -7, Failed to stop communication thread.
Cause	The communications thread could not be stopped. This would only occur during interface shutdown, but should never occur.
Resolution	Contact OSIsoft technical support.

Message	Error code: -8, PDC/PMU communication timeout.
Cause	The communications with a PDC or PMU have timed out. The interface will continue to attempt a reconnection.

Message	Error code: -9, Sync byte not found.
Cause	The 0xAA sync byte was not found in the data stream. The interface will attempt to resynchronize the data stream by closing the socket and reconnecting.
Message	Error code: -10, Message IDCODE not for PDC / PMU.
Cause	The message received was valid but the IDCODE in word 3 of the frame was not for the configured PDC / PMU.
Resolution	Check the XML configuration file and PDC / PMU configuration.

Message	Error code: -11, Message frame size error.
Cause	The frame size in the C37.118 message did not match the actual size.

Message	Error code: -12, Message CRC error.
Cause	The CRC validation failed on a C37.118 message.
Resolution	This error may occur occasionally. If it occurs with any frequency, check the communications hardware.

Message	Error code: -13, No Message Buffer Available.
Cause	No message buffer was available. The interface will continue to operate but will be missing data.
Resolution	Contact OSIsoft technical support.

System Errors and PI Errors

System errors are associated with positive error numbers. Errors related to PI are associated with negative error numbers.

Error Descriptions

Descriptions of system and PI errors can be obtained with the pidiag utility:

```
\PI\adm\pidiag /e error_number
```

UniInt Failover Specific Error Messages

Informational

Message	16-May-06 10:38:00 C37118 1> UniInt failover: Interface in the "Backup" state.
Meaning	Upon system startup, the initial transition is made to this state. While 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 Data Archive 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 may cause the queue to hold up to five failover update intervals worth of data.
Message	16-May-06 10:38:05 C37118 1> UniInt failover: Interface in the "Primary" state and actively sending data to PI. Backup interface not available.
Meaning	While in this state, the interface is in its primary role and sends data to the PI Data Archive as it is received. This message also states that there is not a backup interface participating in failover.
Message	16-May-06 16:37:21 C37118 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 Data Archive 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.

Errors (Phase 1 & 2)

Message	16-May-06 17:29:06 C37118 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 point 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 C37118 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 point 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	17-May-06 09:06:03 C37118 1> 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 C37118 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 2)

Unable to open synchronization file

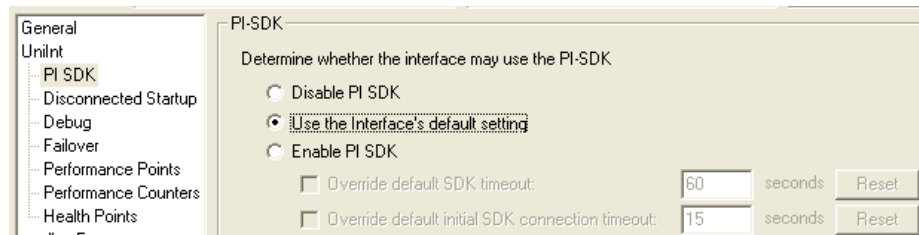
Message	27-Jun-08 17:27:17 PI Eight Track 1 1> Error 5: Unable to create file '\\georgiaking\GeorgiaKingStorage\UnIntFailover\PIEightTrack_eight_1.dat' Verify that interface has read/write/create access on file server machine. Initializing UniInt library failed Stopping Interface
Cause	This message will be seen when the interface is unable to create a new failover synchronization file at startup. The creation of the file only 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 likely the result of a permissions problem.
Resolution	Ensure the account the interface is running under has read and write permissions for the folder. The "log on as" property of the Windows service may need to be set to an account that has permissions for the folder.

Error Opening Synchronization File

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
Cause	This message will be seen when the interface is unable to open the failover synchronization file. The interface failover will continue to operate correctly as long as communication to the PI Data Archive is not interrupted. If communication to the PI Data Archive is interrupted while one or both interfaces cannot access the synchronization file, the interfaces will remain in the state they were in at the time of the second failure, so the primary interface will remain primary and the backup interface will remain backup.
Resolution	Ensure the account the interface is running under has read and write permissions for the folder and file. The "log on as" property of the Windows service may need to be set to an account that has permissions for the folder and file.

Appendix B. PI SDK Options

To access the PI SDK settings for this interface, select this interface from the *Interface* drop-down list and click *Unlnt – PI SDK* in the parameter category pane.



Disable PI SDK

Select *Disable PI SDK* to tell the interface not to use the PI SDK. If you want to run the interface in disconnected startup mode, you must choose this option.

The command line equivalent for this option is `/pisdsk=0`.

Use the Interface's default setting

This selection has no effect on whether the interface uses the PI SDK. However, you must not choose this option if you want to run the interface in disconnected startup mode.

Enable PI SDK

Select *Enable PI SDK* to tell the interface to use the PI SDK. Choose this option if the PI Data Archive version is earlier than 3.4.370.x or the PI API is earlier than 1.6.0.2, and you want to use extended lengths for the Tag, Descriptor, ExDesc, InstrumentTag, or PointSource point attributes. The maximum lengths for these attributes are:

Attribute	Enable the Interface to use the PI SDK	PI Data Archive earlier than 3.4.370.x or PI API earlier than 1.6.0.2, without the use of the PI SDK
Tag	1023	255
Descriptor	1023	26
ExDesc	1023	80
InstrumentTag	1023	32
PointSource	1023	1

However, if you want to run the interface in disconnected startup mode, you must not choose this option.

The command line equivalent for this option is `/pisdsk=1`.

Appendix C. Communication Plug-In Selection

This appendix describes the selection criteria for the Communications plug-in for the PDCs and PMUs which have been tested. Current Plug-In selections are PIC37118_Comm001, PIC37118_Comm002 and PIC37118_Comm003.

PDC/PMU	Data Protocol	Command Protocol	Command and Data on Same Port	Data Transmission Mode	Close Cmd Comm Setting	Plug-In
Arbiter 1133	UDP	UDP	Yes	Unicast	Yes	001
Arbiter 1133	TCP	TCP	Yes	Unicast	Yes	001
Arbiter 1133	TCP	UDP	No	Unicast	Yes	002
Arbiter 1133	TCP	UDP	No	Broadcast	Yes	002
Arbiter 1133	TCP	UDP	No	Multicast	Yes	002
Areva 897	TCP	TCP	Yes	Unicast	Yes	001
GE D60	TCP	TCP	Yes	Unicast	Yes	001
Electric Power Group PDC	UDP	TCP	No	Unicast	Yes	002
NI Rio	TCP	TCP	Yes	Unicast	Yes	001
Siemens R-PMU	TCP	TCP	Yes	Unicast	Yes	001
SEL 3306	TCP	UDP	No	Unicast	No	002
SEL 3378 (SVP operating in PDC mode)	TCP	TCP	Yes	Unicast	Yes	001
SEL 421	UDP	UDP	Yes	Unicast	Yes	001
SEL 421	TCP	TCP	Yes	Unicast	Yes	001
SEL 421	TCP	UDP	No	Unicast	Yes	002
SEL 421	TCP	UDP	No	Broadcast	Yes	002
SEL 421	TCP	UDP	No	Multicast	Yes	002
SEL 451	UDP	UDP	Yes	Unicast	Yes	001
SEL 451	TCP	TCP	Yes	Unicast	Yes	001
SEL 451	TCP	UDP	No	Unicast	Yes	002
SEL 451	TCP	UDP	No	Broadcast	Yes	002
SEL 451	TCP	UDP	No	Multicast	Yes	002
SEL 487E	TCP	TCP	Yes	Unicast	Yes	001
GE D60	TCP	TCP	Yes	Unicast	Yes	001
GE D60	TCP	TCP	Yes	Unicast	Yes	001
Siemens R-PMU	TCP	TCP	Yes	Unicast	Yes	001
TVA Proxy	Udp	n/a	n/a	Unicast	No	003
Serial (All Devices)	n/a	n/a	n/a	n/a	No	001

Appendix D. Terminology

To understand this interface manual, you should be familiar with the terminology used in this document.

Interface Specific Terms

IEEE

IEEE stands for the Institute of Electrical & Electronic Engineers.

PMU

A Phasor Measurement Unit (PMU) is a generic device that produces synchronized phasors from voltage and/or current inputs and synchronizing signal.

Phasor

A Phasor is the complex equivalent of a simple sine wave quantity such that the complex modulus is the sine wave amplitude and complex angle (in polar form) is the sine wave phase angle.

PDC

A Phasor Data Concentrator (PDC) provides Phasor data from multiple PMUs.

General Terms

Buffering

Buffering refers to an interface node's ability to store temporarily the data that interfaces collect and to forward these data to the appropriate PI Data Archives.

N-Way Buffering

If you have PI Data Archives that are part of a collective, PIBufss supports n-way buffering. N-way buffering refers to the ability of a buffering application to send the same data to each of the PI Data Archives in a collective. (Bufserv also supports n-way buffering to multiple PI Data Archives however it does not guarantee identical archive records since point compressions attributes could be different between PI Data Archives. With this in mind, OSIsoft recommends that you run PIBufss instead.)

ICU

ICU refers to the PI Interface Configuration Utility. The ICU is the primary application that you use to configure PI interface programs. You must install the ICU on the same computer on which an interface runs. A single copy of the ICU manages all of the interfaces on a particular computer.

You can configure an interface by editing a startup command file. However, OSIsoft discourages this approach. Instead, OSIsoft strongly recommends that you use the ICU for interface management tasks.

ICU Control

An ICU Control is a plug-in to the ICU. Whereas the ICU handles functionality common to all interfaces, an ICU Control implements interface-specific behavior. Most PI interfaces have an associated ICU Control.

Interface Node

An interface node is a computer on which

- the PI API and/or PI SDK are installed, and
- PI Data Archive programs are not installed.

PI API

The PI API is a library of functions that allow applications to communicate and exchange data with the PI Data Archive. All PI interfaces use the PI API.

PI Data Archive Collective

A PI Data Archive Collective is two or more replicated PI Data Archives that collect data concurrently. Collectives are part of the High Availability environment. When the primary PI Data Archive in a collective becomes unavailable, a secondary collective member node seamlessly continues to collect and provide data access to your PI clients.

PI Data Archive Node

A PI Data Archive node is a computer on which PI Data Archive programs are installed. The PI Data Archive runs on the PI Data Archive node. In earlier documentation, PI Data Archive was referred to as the PI Server. (See [PI Server Node](#).)

PIHOME

PIHOME refers to the directory that is the common location for PI 32-bit client applications.

A typical *PIHOME* on a 32-bit operating system is C:\Program Files\PIPC.

A typical *PIHOME* on a 64-bit operating system is C:\Program Files (x86)\PIPC.

PI 32-bit interfaces reside in a subdirectory of the *Interfaces* directory under *PIHOME*.

For example, files for the 32-bit Modbus Ethernet Interface are in

[*PIHOME*]\PIPC\Interfaces\ModbusE.

This document uses [*PIHOME*] as an abbreviation for the complete *PIHOME* or *PIHOME64* directory path. For example, ICU files in [*PIHOME*]\ICU.

PIHOME64

PIHOME64 is found only on a 64-bit operating system and refers to the directory that is the common location for PI 64-bit client applications.

A typical *PIHOME64* is C:\Program Files\PIPC.

PI 64-bit interfaces reside in a subdirectory of the *Interfaces* directory under *PIHOME64*.

For example, files for a 64-bit Modbus Ethernet Interface would be found in

```
C:\Program Files\PIPC\Interfaces\ModbusE.
```

This document uses [*PIHOME*] as an abbreviation for the complete *PIHOME* or *PIHOME64* directory path. For example, ICU files in [*PIHOME*]\ICU.

PI Message Log

The PI message log is the file to which OSIsoft interfaces based on UniInt 4.5.0.x and later write informational, debug and error messages. When a PI interface runs, it writes to the local PI message log. This message file can only be viewed using the *PIGetMsg* utility. See the *UniInt Interface Message Logging.docx* file for more information on how to access these messages.

PI SDK

The PI SDK is a library of functions that allow applications to communicate and exchange data with the PI Data Archive. Some PI interfaces, in addition to using the PI API, require the use of the PI SDK.

PI Server Node

In earlier documentation, the term “PI Server” was used as a nickname for the PI Data Archive and a PI Server node was a computer on which PI Data Archive programs were installed. While the PI Data Archive remains a core server of the PI Server product, the product name “PI Server” now refers to much more than the PI Data Archive. OSIsoft documentation, including this user manual, is changing to use “PI Server” in this broader sense and “PI Data Archive” to refer to the historian core. (See [PI Data Archive Node](#).)

PI SMT

PI SMT refers to PI System Management Tools. PI SMT is the program that you use for configuring PI Data Archives. A single copy of PI SMT manages multiple PI Data Archives. PI SMT runs on either a PI Data Archive node or an interface node.

Pipc.log

The `pipc.log` file is the file to which OSIsoft interfaces based on UniInt versions earlier than 4.5.0.x write informational and error messages. When a PI interface runs, it writes to the `pipc.log` file. The ICU allows easy access to the `pipc.log`.

Point

The PI point is the basic building block for controlling data flow to and from the PI Data Archive. For a given timestamp, a PI point holds a single value.

A PI point does not necessarily correspond to a “point” on the data source device. For example, a single “point” on the data source device can consist of a set point, a process value,

an alarm limit, and a discrete value. These four pieces of information require four separate PI points.

Service

A Service is a Windows program that runs without user interaction. A service continues to run after you have logged off from Windows. It has the ability to start up when the computer itself starts up.

The ICU allows you to configure a PI interface to run as a service.

Tag

The Tag attribute of a PI point is the name of the PI point. There is a one-to-one correspondence between the name of a point and the point itself. Because of this relationship, PI System documentation uses the terms “tag” and “point” interchangeably.

Interfaces read values from a device and write these values to an input tag.

Appendix E. Technical Support and Resources

For technical assistance, contact OSIsoft Technical Support at +1 510-297-5828 or 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
- 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

To ask questions of others who use OSIsoft software, join the OSIsoft user community, [PI Square](#). Members of the community can request advice and share ideas about the PI System. The [PI Developers Club](#) space within PI Square offers resources to help you with the programming and integration of OSIsoft products.

Appendix F. Revision History

Date	Author	Comments
13-Jun-2007	Jhartman	Initial Release
18-Jul-2007	Jloe	Version 1.0.1.3 Revision H: updated headers, table formatting, IORates information
26-Jul-2007	Mkelly	Version 1.0.1.3 Revision I: Updated the ICU Control and XML section. Fixed headers and footers. Corrected filenames references for csv files included with distribution.
28-Jun-2008	Jhartman	Version 1.0.2.0, Added /TagConfig argument information and updated version.
10-Oct-2008	Mkelly	Version 1.0.2.0 Revision A; Updated screenshots, replace buffering section, fixed headers and links, updated TOC.
14-Jan-2009	Jhartman	Updated version to 1.0.3.0
02-Feb-2009	Mkelly	Version 1.0.3.0, Revision A; Updated copyright date, added Terminology section, updated TOC.
07-Apr-2009	Jhartman	Changed ranges for EndPoint and PMU Id's to 0-65535 to reflect unsigned short value.
11-Oct-2010	Sbondar	Updated with skeleton Version 3.0.27. Updated the list of tested PMU/PDC
17-Dec-2010	Sbondar	Updated with PDC / PMU Data Source Level Failover and some other additional parameters.
29-Dec-2010	Sbranscomb	Version 1.0.4.x Revision A; Updated with skeleton Version 3.0.31.
28-Jan-2011	Mkelly	Version 1.0.4.x Revision B; Update ICU Control screenshots for new parameters.
10-Feb-2011	Sbondar	Version 1.0.4.x Revision C; Update tables of Attributes and Elements with the new parameters.
15-Feb-2011	Sbondar	Version 1.0.4.x Revision D; Change phrase from 'server-level failover' to 'Data Source Level Failover' to be consistent with other documents. The reason for that is PMU / PDC are considered to be more like a data source rather than a server.
21-Jul-2011	Twilliams	Version 1.0.4.x – 1.0.5.x; Updated the version number for a rebuild with new Unilnt 4.5.2.0.
21-Feb-2013	Mkelly	Version 1.0.6.x Updated ICU Control screenshots for new MulticastSourceIPAddress XML Element.
25-Feb-2013	Sbranscomb	Version 1.0.6.x; Revision A; Updated with skeleton Version 3.0.36.
28-Feb-2013	Sbondar	Version 1.0.6.x Revision B; Updated description for new MulticastSourceIPAddress XML Element

Revision History

Date	Author	Comments
01-Apr-2013	Sbondar	Version 1.0.6.x Revision C; Updated /TagConfig and /configfile parameters in command-line. Parameters table
01-Apr-2013	Sbondar	Version 1.0.6.x Revision D; Updated InstrumentTag attribute description.
01-July-2013	Sbondar	Version 1.0.6.x Revision E; Updated Trademarks page.
13-Apr-2015	WZhang	Version 1.0.6.x Revision F; Update terminology
01-Jun-2015	JMuraski	Version 1.1.0.x Updated with skeleton version 3.0.42
11-Aug-2015	TWilliams	Version 1.1.0.x. Added text to more clearly define how to configure a service using Local System