PI Data Archive 3.4.390

System Management Guide

Included with PI Server 2014

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

PI Data Archive 3.4.390 System Management Guide

Version: 3.4.390

Published: 8 April 2014

Contents

Introduction to PI Server system management........ 1
About this book........ 1
PI System overview........ 1
PI Server directory structure........ 2
File system best practices........ 2
PI Server subsystems........ 3

Tools for System Management........ 5
PI System Management Tools (SMT)........ 5
PI System Tray........ 5
Add PI Tag Configurator utility to Excel........ 6
Run PI Interface Configuration Utility (ICU)........ 6
Run Collective Manager........ 6
PI SDK Utility and About PI-SDK........ 7
Tools for working with PI Asset Framework........ 7

Start and stop PI Server........ 9
Start PI Server........ 9
Start PI Server as a Windows service........ 9
Configure automatic startup of services........ 9
Startup verification........ 10
Start PI Server in interactive (troubleshooting) mode........ 10
Stop PI Server........ 10
Stop Windows services........ 11
Change the shutdown wait time........ 11
Stop interactive mode........ 11
Shutdown events........ 12
Set shutdown events for specific points........ 12

Time specifications and considerations........ 15
PI time........ 15
PI time abbreviations........ 15
PI time expressions........ 16
Time-stamp specification........ 16
Time-interval specification........ 17
PI time terminology........ 18
Daylight Saving Time considerations........ 18
Determine whether a time string is ambiguous........ 18
Determine your time zones........ 19
Specify time zones........ 19
How to display time zone information........ 19
Time zone information and Daylight Saving Time transition rules........ 20
Display options........ 20
Display local and UTC time........ 21
Display a different time zone........ 21
Customize standard and Daylight Saving Time changes........ 21
How to translate time formats........ 21
String to integer format sample output........ 22
Integer format to string sample output........ 22

Back up PI Server.. 23
PI Server scheduled backup configuration.. 23
Scheduled backups overview.. 23
Recommended disk configuration for PI Server backups.. 24
Configure a PI Server scheduled backup.. 25
PI Server collectives and backups.. 31
View backup history of a PI Server.. 31
PI Server backup types.. 33
Change the number of backups shown in the Backup History table.. 35
View backup information summary.. 36
View backup details.. 36
Troubleshooting PI Server backups.. 39
View PI Server backup logs and messages.. 39
Common problems with VSS backups.. 39
PI Server backup failure due to offline subsystem.. 40
Perform an on-demand PI Server backup.. 41
Export PI Server backup history.. 42
PI Server backup scripts.. 42
pibackuptask.bat.. 43
pibackup.bat.. 43
pisitebackup.bat.. 43
pintbackup_3.4.370.bat.. 44
afbackup.bat.. 44
How PI Server backup verification works.. 44
Restore PI Server from a backup.. 45
Restore a backup to an existing PI Server.. 45
Restore a PI Server backup to a new computer.. 46

Manage PI points.. 49
PI point classes and attributes.. 49
Predefined point classes.. 50
Predefined attribute sets.. 51
Base class point attributes.. 54
Classic point class attributes.. 63
COM Connector point attributes.. 64
Default values for point attributes.. 65
System-assigned attributes.. 67
Exception reporting and compression testing.. 68
Compression guidelines.. 68
Compression testing.. 69
Exception reporting.. 71
Change PI point type.. 74
Allowable point type coercions.. 74
Point type changes and archives.. 75
Valid point type edits.. 76
Configure error handling.. 76
Modification of PI point classes and attribute sets.. 76
Required access permissions and other restrictions.. 77
Attribute set database edit.. 79
Point classes database edit.. 86
Point class rename.. 90
Editing a point's point class.. 91

Point database audit....92
Digital state sets....92
Digital state name conventions....92
Default set System state set....92
Define a digital state set....93

PI archives....95
Archive management tools....95
Archive files....96
Primary archive files....96
Archive records....97
Index records....97
Fixed and dynamic archives....97
Manage PI archives....100
Open the Archives tool in PI SMT....100
Create new archives with PI SMT....100
Manage archive size....102
Manage archive registration....105
Manage archive shifts....108
Maintain annotations....110
Manage automatic archive creation....112
Manage archive gaps....117
Configure number of archives monitored for readwrite errors....117
Prevent archive changes....117
Export archives to a file....120
Move archive files....120
Back up archives....121
Delete an archive event....121
Delete an archive....122
Manage backfilling of data....122
About backfilling data....122
Prepare PI Server for backfilling....124
Backfill data with a piconfig script....125
Backfill data into a new PI System....126
Backfill existing archives with data from new PI points....128
List archive record details (archive walk)....130
Example of an archive walk....131
Determine archive sequence numbers....131
Display archive records....131
Examine broken pointers....133
Verify the integrity of the archive files....133
Find and report errors in archive files....134
Manage offline archive files....135
Tips for using the offline archive utility....135
Options for the offline archive utility....136
Run the offline archive utility....137
Time ranges....138
Correcting event timestamps (-tfix -tfixend)....140
Combine or divide archives....141

PI interfaces overview....143
UniInt interface framework....144
PI API....144

PI SDK... 144
Interfaces included with PI Server.. 145
Ramp Soak interface point configuration... 145
Random interface point configuration..147
Interface installation checklist.. 150
Interface installation.. 150
Advanced interface features (optional)...151
Interface diagnostics (optional)..152
Additional documentation on interfaces..152
Interfaces Quickstart Series.. 153

Monitor the PI Server.. 155
PI System messages... 155
View PI System messages with PI SMT...155
View messages in the Windows event log.. 155
Message structure..155
Severity levels..156
PI message definition file.. 157
Interpret error codes... 157
Subsystem health checks (RPC resolver error messages)...157
Log file naming convention... 157
PI Server daily health check... 158
Windows performance counters.. 159
PI Server backup performance counters.. 160

PI Server tuning parameters...161
Configurable tuning parameters.. 161
Edit tuning parameters.. 161
Add a tuning parameter to the list... 162
Pending update limit... 162
When to change the pending update limit...163

PI SQL Subsystem.. 165
PI SQL Subsystem architecture...165
Statement handles.. 165
Concurrency.. 166
PI SQL Subsystem configuration.. 166
Hierarchy of PI SQL Subsystem settings.. 166
Initialization file settings..166
pisqlss... 167
PI SQL Subsystem troubleshooting... 169
PI SQL Subsystem trace files.. 169
Clear expensive query problems.. 171
Performance counters... 171
PI System unexpected shutdown or power outage recovery plan... 172
Technical support and problem reports.. 172

PI data retrieval with foreign data sources.. 173
Point configuration.. 174
Retrieval of snapshot data... 174
Retrieval of archive data...175
Archive files.. 176

Snapshot updates........176
Compression........176
Point security........177

Troubleshooting........179
Troubleshooting checklist........179
General troubleshooting tasks........181
Verification that PI Server processes are running........182
Connection checking........184
Update Manager checking........185
Configure Dr. Watson for Windows........186
Specific problems........188
Slow connections and connection failures........188
PI ProcessBook trend flatlines........189
Abusive usage........190
Problems with COM connectors........192

Repairs........199
Repair the archive manager data file........199
Archive file corruption........200
Recovery of a non-primary archive........200
Recover a primary archive........200
Correction of archive event time stamps........201
Time conversion file format........202
Snapshot recovery........203
Recover from future times in the snapshot........203
Rebuild the snapshot file........204
Removal of future-time snapshots........205
Database repair........205
Repair the PI Module Database........206
Diagnose and repair PI Server database files........206
List the header and index........206
Compress a file base file........207
Recover file base file........207
Recover from accidental system time change........208
Recover from accidental time change at interface node that uses PI Buffer Subsystem........210
Excessive CPU usage by utilities........210

Technical support and other resources........213

Introduction to PI Server system management

Topics in this section

- About this book
- PI System overview
- PI Server directory structure
- File system best practices
- PI Server subsystems

About this book

This book provides detailed instructions for configuring, maintaining, and troubleshooting a PI Server. It also discusses other PI components that are relevant to PI Server system management. These include PI interfaces as well as client tools that can be used to manage your PI System.

This guide assumes that you have a basic knowledge of the PI Server and how to perform typical system administration tasks. See the *Introduction to PI System Management* guide for this information.

PI Server security (authentication, access permissions) is documented in the *Configuring PI Server Security* guide.

PI System overview

The PI System collects, stores, and manages data from your plant or process. The PI System includes different types of products. PI interfaces retrieve data from your data sources and send it to one or more PI Servers. Users on other computers can get data from PI Servers and display it with client tools. The PI System includes:

- Data sources: Data sources are the instruments that generate your data. They can be almost anything, and they can connect to the interface nodes in a variety of different ways. PI Performance Equations, PI ACE, and Totalizer are also considered data sources, even though they may be hosted on the PI Server computer.
- Interfaces: PI interfaces get the data from the data sources and send it to the PI Server. Each different data source needs a PI interface that can interpret it. OSIsoft has over 300 different interfaces.
- PI Servers: The PI Server is the heart of the PI System. It gets the data and routes it in real time throughout the PI System and your entire information infrastructure, making it possible for everyone to work from a common set of real-time data. Operators, engineers, managers, and other plant personnel can use client applications to connect to the PI Server and view manufacturing data from the PI data archives or from external data storage systems.

  PI Server typically runs on a separate computer from those that run PI interfaces and client applications. This distributed data collection architecture is scalable, robust, and flexible.

When the high availability (HA) architecture is used, the PI Server runs on two or more computers that are automatically synchronized and act as one logical PI Server, called a PI Server collective. These computers can be geographically dispersed.

- PI Asset Framework (PI AF): PI AF allows the definition of consistent representations of organizational assets and/or equipment and uses these representations in simple or complex analyses that yield critical and actionable information.
- Client applications: Operators, engineers, managers and other plant personnel use a variety of client applications to connect to PI Servers and PI application servers to view plant data. PI Coresight, PI ProcessBook, PI DataLink, and PI WebParts are all client applications.

PI Server directory structure

When you install the PI Server, the setup program prompts you for a location to store the PI Server files. By default, PI Server installs its files in a folder called `PI` on the disk with the most available space. Although you can rename the `PI` directory to whatever you like, we will refer to the top PI Server directory as the `PI` directory. The **`PISERVER`** environment variable points to this directory.

The `PI` directory includes the following subdirectories:

Subdirectory	Contents
`Program Files\PI\adm`	Administrative tools.
`Program Files\PI\bin`	Subsystem or PI service executables.
`Program Files\PI\dat`	Databases and tables such as Point Database and Digital State Table. This is also the default directory for archive files and the event queue.
`Program Files\PI\interfaces`	Interfaces that were installed with previous versions of PI. This directory is not present on new PI Server installations, but might be present on servers that are running upgrades.
`Program Files\PI\log`	Log files.
`Program Files\PI\setup`	Files for install and uninstall.

In addition to the PI directory, the PI Server installation creates the `pipc` directory, if it does not exist. The `pipc` directory is actually created when you install the PI SDK, which is included in the PI Server installation. The `pipc` directory contains files for the PI SDK, for bundled PI interfaces, and for a variety of other tools and utilities, including PI SMT, Collective Manager, and PI ICU. The **`PIHOME`** environment variable points to this directory.

File system best practices

- Disable virus scanning on the `PI\dat` and archive folders. Virus scanning might affect the integrity of archive or other database files. The problem with virus scanning is that, because the data is random, it might have a bit pattern that matches a known virus signature. The virus scanning software then locks and quarantines the data file.
- Use Windows file system compression with caution. File compression might slow down the PI Server's access to archive files. Compression can significantly reduce archive size, but

more CPU resources are required to access a compressed file. Do not use file compression on files that are frequently accessed, such as recent archives.

PI Server subsystems

The PI Server subsystems are a set of several interdependent processes, referred to as subsystems. Some subsystems depend on other subsystems for proper behavior. All subsystems wait at startup for any dependent subsystems. The executable for each of the PI subsystems is installed in the `PI\bin` directory. PI Server subsystems run as Windows services. You can check the status of these services in the Windows Services tool, or in PI SMT (**Operation** > **PI Services**).

Generally, PI Server requires seven core subsystems to function at a minimum level.

PI Server core subsystems

Subsystem	Executable	Purpose	Dependencies
Archive	`piarchss.exe`	Stores and serves the data after it comes out of the snapshot subsystem. Data consists of multiple time-stamped measurements for each data point. Values represent on/off, pressures, flows, temperatures, set points, and so on.	Snapshot, Update Manager, and License Manager.
Base	`pibasess.exe`	Maintains the Point Database, Digital State Table, and configuration databases for authentication. Hosts the PI Module Database.	Update Manager and License Manager.
License Manager	`pilicmgr.exe`	Maintains license information for the PI Server and all connected applications.	
Message	`pimsgss.exe`	Records status and error messages for the PI Server in a log file.	Messages are routed to the Windows event log if this subsystem is not available.
Network Manager	`pinetmgr.exe`	Manages communication between PI Server subsystems, interfaces and client applications. Also validates clients at time of connection. Clients may be standard products such as PI ProcessBook, or they may be custom PI API or PI SDK programs.	

Subsystem	Executable	Purpose	Dependencies
Snapshot	`pisnapss.exe`	Stores the most recent event for each point, applies compression, sends data to the event queue, serves snapshot events, and sends updates for client applications to PI Update Manager.	Update Manager and License Manager.
Update Manager	`piupdmgr.exe`	Queues notifications of changes in data values, point attributes, modules, and so on to any interface or client application that is signed up for notification.	Essential for proper operation of a PI Server; it is required by most of the PI subsystems and most client applications.

In addition to the core PI Server subsystems, the PI Server includes additional subsystems that are not essential to run the PI Server. Some of these subsystems are licensed separately and might not be installed on your PI Server.

PI Server non-core subsystems

Subsystem	Executable	Purpose
AF Link	`piaflink.exe`	Synchronizes PI MDB and PI AF.
Alarm	`pialarm.exe`	Provides alarm capabilities for PI points.
Backup	`pibackup.exe`	Manages backups of the PI Server.
Batch*	`pibatch.exe`	Detects and records batch activity.
Performance Equations*	`pipeschd.exe`	Performs PI Performance Equation (PE) calculations for PI PE points.
Recalculator	`pirecalc.exe`	Recalculates values of PE points after historical changes.
Redirector	`piudsrdr.exe`	Obtains data from external systems and sends it to the Base, Archive, and Snapshot subsystems. Used in connection with COM Connectors*.
Shutdown	`pishutev.exe`	Determines when the PI System was stopped and writes shutdown events to points configured to receive these events; It runs only at startup and then stops.
SQL	`pisqlss.exe`	Prepares and executes SQL statements directed at the PI Server; The primary users of this subsystem are the PI ODBC Driver and the PI SDK.
Totalizer*	`pitotal.exe`	Performs post-processing calculations on a point in the snapshot and stores the results in a PI Totalizer point.

* indicates a separately licensed subsystem

Tools for System Management

OSIsoft provides several tools and utilities for managing a PI System:

- PI System Management Tools (SMT) for performing routine PI Server administration tasks.
- PI System Tray monitors your PI servers and AF servers. You can see normal, error, or critical status at a glance.
- PI Interface Configuration Utility (ICU) for configuring PI interfaces.
- Collective Manager for creating and managing PI collectives for implementing high availability (HA) in your PI Server.
- PI SDK Utility for troubleshooting tasks.
- PI System Explorer for creating and managing assets including analytics and event frames in PI AF.
- PI Builder for creating and editing tags and PI AF objects in an Excel spreadsheet. In PI Server 2014 and later PI Builder replaces PI Tag Configurator and PI AF Builder.

OSIsoft also provides powerful command-line utilities, described in the *PI Data Archive Reference Guide*. The PI Server also includes powerful command line utilities. See *PI Data Archive Reference Guide* for PI Data Archive utilities and *PI System Explorer User Guide* for PI AF utilities.

PI System Management Tools (SMT)

PI System Management Tools (SMT) is a set of easy-to-use tools that allow you to perform all the basic PI Server administration tasks. PI SMT is included in the PI Server installation, but you can get the latest version of PI SMT on the OSIsoft Technical Support website (http://techsupport.osisoft.com).

PI System Tray

PI System Tray displays as a small icon on your Windows task bar. The icon shows the status of the PI Servers and PI AF servers that it is monitoring. The PI System Tray icon shows a green health indicator when all servers are running normally and there are no errors on any monitored server. If problems occur, a notification appears, and the health indicator's color changes to yellow or red depending on the severity of the problem.

PI System Tray also provides shortcuts for viewing system messages, starting and stopping PI Servers and PI AF servers, and starting PI System Management Tools (SMT) and PI System Explorer.

By default, PI System Tray monitors the default PI Server (or PI Server collective) and the PI AF application service associated with the default PI AF server. You can monitor additional servers or change the monitored servers if needed.

PI System Tray is installed with PI System Management Tools 2012 and later. It launches automatically when you install PI SMT. To launch it manually from the Windows Start menu, choose **Start** > **All Programs** > **PI System** > **PI System Tray**.

Add PI Tag Configurator utility to Excel

You must add the PI Tag Configurator utility to the Excel **Add-Ins** menu before you can use it.

Procedure

1. Open Excel.
2. Open the **Add-Ins** menu:
   - In Excel 2003 and earlier: Select **Tools** > **Add-Ins**.
   - In Excel 2007: From the **Office** button click **Excel Options**, choose **Add-Ins**, then click the **Go** button
3. Click **Browse** to open the **Browse** explorer window.
4. Browse to the `PIPC\SMT` folder.
5. Select `PITagCnf.xla`.
6. Click **OK** to return to the Add-Ins window.
7. Select **PI-TagConfigurator 32bit**.
8. Click **OK** to exit the **Add-Ins** window. This adds the **PI-SMT** menu to Excel.

After you finish

Use the PI SMT help file for instructions about using the **PI SMT** menu.

Run PI Interface Configuration Utility (ICU)

To configure a PI interface with the ICU, you must run it directly on that interface node. This means you need to install the ICU on each interface node. You can get the latest version of the ICU on the OSIsoft Technical Support website (http://techsupport.osisoft.com).

Procedure

- To run the ICU, click **Start** > **Programs** > **PI System** > **PI Interface Configuration Utility**.

Run Collective Manager

You can get the latest version of Collective Manager on the OSIsoft Technical Support website (http://techsupport.osisoft.com).

Procedure

1. To run Collective Manager, from the Windows Start menu, choose **Programs** > **PI System** > **Collective Manager**.
2. To view and edit a collective's properties, click the collective name under **Collectives.** The collective properties and a diagram of servers in the collective appear on the right side of Collective Manager. An icon in the diagram represents each server in the collective. A green check mark on the icon indicates that the server is communicating properly. A red X indicates that the server is unavailable. A yellow warning icon indicates that the server is available but has errors. **Status** and **Connection Status** show the associated errors.

After you finish

For further details about Collective Manager, see the Collective Manager Help files.

PI SDK Utility and About PI-SDK

Most clients (and some interfaces) communicate with the PI Server through the PI SDK. PI SDK Utility provides troubleshooting features, such as checking connectivity, performing tag searches, viewing message logs, and enabling tracing. PI SDK Utility is a replacement for About PI-SDK.

Procedure

- PI SDK versions 1.4 and later include PI SDK Utility. To run PI SDK Utility, click **Start** > **All Programs** > **PI System** > **PI SDK Utility**.
- PI SDK versions earlier than 1.4 include About PI-SDK. To run About PI-SDK, click **Start** > **All Programs** > **PI System** > **About PI-SDK**.

Tools for working with PI Asset Framework

OSIsoft provides two tools for creating and editing PI AF objects:

- PI System Explorer provides a graphical user interface for creating, editing and managing PI AF objects. Use PI System Explorer to create and manage your asset framework including PI AF databases elements, templates, and all other PI AF objects. If you are new to PI AF, start with PI System Explorer.

  To open PI System Explorer from the Windows Start menu, choose **All Programs** > **PI System** > **PI System Explorer**.
- PI Builder is a Microsoft Excel add-in that allows you to work with PI AF objects in bulk. For more on PI Builder, see the *PI Builder User Guide*.

In addition, any PI client applications allow users to view PI AF elements and attributes.

Start and stop PI Server

PI Server runs as a collection of Windows services. These services are typically configured to start automatically at computer startup. If you need to shut down or restart the Windows operating system, always stop the PI Server first. Otherwise, you could lose data due to the service timeouts. You could also lose data that is still in memory and not flushed to disk.

Start PI Server

Topics in this section

- Start PI Server as a Windows service
- Configure automatic startup of services
- Startup verification
- Start PI Server in interactive (troubleshooting) mode

Start PI Server as a Windows service

Procedure

1. Log on to a Windows account that has full access to the PI Server files and permission to start PI services.
2. At a command prompt, go to the `PI\adm` directory.
3. Use the `pisrvstart.bat` script to start PI Server as Windows services:

   ```
   pisrvstart.bat [-nosite] [-base]
   ```

   The optional **`nosite`** parameter runs PI Server without starting interfaces and other site-specific programs.

   The optional **`base`** parameter starts only the core subsystems. When you use this parameter, these subsystems will start in the following order: Network Manager, Base, Message, License Manager, Snapshot, Archive, Backup, and Update Manager. This can be useful for troubleshooting.

Configure automatic startup of services

PI Server on Windows normally runs as a collection of services. You can set the PI Server reboot startup behavior with the Windows Services tool.

In order to control the services, you must be logged on with an account that has sufficient privileges. If you are not, you get a message like this:

```
Error 5: Access Denied
```

For diagnostic purposes, you can also start the PI Server in interactive mode.

Startup verification

The PI Server subsystems may take several minutes to start. The PI Server subsystems that must start up before interfaces and other applications can connect to the PI Server are Archive, Base, License Manager, Network Manager, Snapshot, and Update Manager. When these subsystems are ready to service requests, the TCP/IP listener is opened.

To verify that the PI server has started, you can:

- Determine that the PI Server port is open:

  ```
  netstat -an
  ```

  ```
  netstat -an
  ```

- Or, review the PI Server log for the following message:

  ```
  >> TCP/IP connection listener opened on port: 5450
  ```

  Connection attempts will fail if the port is not open. Most interfaces and client applications will retry automatically until a connection is established.

Start PI Server in interactive (troubleshooting) mode

For troubleshooting purposes, you can start the PI Server interactively. Do this only if you need to monitor, test, or troubleshoot the PI Server.

Procedure

1. Log on to a Windows account that has full access to the PI Server files and permission to start PI Server services.
2. At a command prompt, go to the `PI\adm` directory and type:

   ```
   pistart.bat [-nosite] [-stdout] [-base]
   ```

   The optional **`nosite`** parameter runs the PI Server without starting the interfaces and other site-specific programs.

   The optional **`stdout`** parameter prevents PI Message Subsystem from being started; all messages will be sent to the standard output instead of the PI Server message log.

   The optional **`base`** parameter starts only the core PI Server subsystems. When you use this parameter, these subsystems will start in the following order: Network Manager, Base, Message, License Manager, Snapshot, Archive, Backup, and Update Manager.

   Note:

   Some Windows interfaces cannot be run as services. Refer to the interface documentation for details.

Stop PI Server

Topics in this section

- Stop Windows services
- Change the shutdown wait time
- Stop interactive mode

Stop Windows services

Use the `pisrvstop.bat` file to stop the PI Server if its processes are running as Windows services.

Procedure

1. Log on to a Windows account that has full access to the PI Server files and permission to start PI services.
2. At a command prompt, go to the `PI\adm` directory:
3. Type the following command:

   ```
   pisrvstop.bat
   ```

   This stops all of the interfaces and programs listed in `pisrvsitestop.bat`, and then the PI processes. To stop PI Server without shutting down interfaces and other site-specific programs, enter the optional **`nosite`** parameter. You may also enter a reason for shutting down the PI Server by entering:

   ```
   [ -reason "Reason for shutting down PI Server" ]
   ```

   Note:

   If you plan to completely start or stop a PI Server, it is important that you use the startup and shutdown scripts; these files will start or stop services in the order required by system dependencies.

Change the shutdown wait time

Windows has a registry entry that defines the maximum wait time for a service to exit. On PI Servers with large point counts, the maximum wait may need to be increased to allow the services enough time to shut down properly. The PI Server installer sets the default value to 300,000 milliseconds, or 5 minutes. This is generally enough time for proper shutdown on systems with fewer than 50,000 points. Larger servers may require more time.

Failing to allow proper shutdown of PI Server can result in lost data or corrupted data files.

Procedure

1. Use the `pisrvstop.bat` file to stop PI Server.
2. Record the time it takes PI Server to shutdown.
3. If it takes more than 5 minutes for the server to shut down, check the registry entry:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WaitToKillServiceTimeout
   ```

4. To increase the wait time:
   a. Edit the WaitToKillServiceTimeout parameter to equal the time required for shutdown.
   b. Reboot to have the change to the registry entry take effect.

Stop interactive mode

To shut down a PI Server that was started in interactive mode, type Ctrl+C in each of the command prompt windows corresponding to the PI Server processes.

Procedure

- Shut down these processes in the following order:

  a. Utilities (such as **`piconfig`**).

  b. Interfaces (such as RampSoak and Random).

  c. The Network Manager service.

     When you stop the Network Manager service, the remaining processes are told to exit in the proper order before the Network Manager service itself stops.

Shutdown events

PI points have a configurable attribute to determine whether shutdown events are written. The time stamp of the shutdown event normally represents the actual shutdown time of the PI Server as recorded by PI Snapshot Subsystem. If the PI Server is shut down ungracefully, this time stamp will be accurate to within 15 minutes by default.

The shutdown attribute has two possible values: 1 (On) and 0 (Off). If the shutdown attribute of a point is set to 1, then PI Shutdown Subsystem writes a shutdown event whenever the PI Server shuts down. Beginning with PI Server PR1 SP1, unless you configure points to receive shutdown events, only test points such as sinusoid and sinusoidu will receive shutdown events.

OSIsoft recommends using the default setting of 0 (Off) for points collected by remote interfaces that are configured for buffering or high availability (HA). The reason: if you run remote interfaces with buffering or PI Server collectives, shutdown events are not an accurate indicator of data loss when a PI Server is shut down. With properly configured buffering, data will simply be queued up for a PI Server while it is shut down, provided the remote interfaces continue running. Also, if a PI Server is part of a collective, shutting down one member has no effect on the other members' ability to continue receiving and serving data.

Note:

OSIsoft recommends that you do not run interfaces on the same machine as the PI Server; however, if you do use such a configuration, these local interfaces should be configured for shutdown events. Unlike most PI Server subsystems, PI Shutdown Subsystem exits after completion.

Set shutdown events for specific points

Points that receive shutdown events are specified in the `PI\dat\shutdown.dat` file.

On PI Server version 3.4.375.38 (PR1) or later, the default configuration of the `shutdown.dat` file targets only points with a point source of R *and* a shutdown attribute set to 1. If you upgrade to PI Server 3.4.375.38 (PR1) or later, the installer will not change the configuration of the `shutdown.dat` file.

You can edit the `shutdown.dat` file to restrict shutdown events to certain groups of points. To specify more than one tag name use a tag mask. Use the wildcards * and ?. An asterisk (*) matches all possibilities with any number of characters. The question mark (?) matches a single character and may be used any number of times.

Note:

Do not specify additional tags by appending comma-separated tag masks or by using additional lines. You can specify only one tag mask. You must specify at least one tag mask to enable the shutdown system to operate without errors. To prevent all shutdown events, specify a tag mask that does not match any tag.

You can use other point attributes and values in addition to, or instead of, the shutdown flag. All conditions are logically combined with AND. If no point attributes are specified, all tags specified by the tag mask are selected to receive shutdown events.

For example, this configuration file entry selects only tags that start with s, have the location1 attribute set to 0, and the point source set to H. No other tags receive shutdown events:

```
! tag mask
s*
! point attributes
location1,0
pointsource,H
```

Time specifications and considerations

PI Server tracks time according to the Windows clock, including the time zone and Daylight Saving Time (DST) settings. If the system clock is wrong, the PI Server data is not correct.

OSIsoft recommends that you check the system clock regularly. If you need to make an adjustment, adjust the clock only in small increments (for example, one second per minute). Keep a record of all adjustments you make.

Archive time stamps in PI Server are stored as the number of seconds past January 1, 1970. Two-digit years from 00 through 69 are interpreted as 21st century. Two-digit years from 70 through 99 are interpreted as the 20th century (1900s). For example, 70 translates to 1970; 00 translates to 2000; and 37 translates to 2037.

PI time

You can use a special syntax, called PI time, to specify inputs for time stamps and time intervals. PI time uses specific abbreviations, which you combine to create time expressions.

Topics in this section

- PI time abbreviations
- PI time expressions
- Time-stamp specification
- Time-interval specification

PI time abbreviations

When specifying PI time, you can use specific abbreviations that represent time units and reference times.

Time-unit abbreviations

Abbreviation	Time unit
s	second
m	minute
h	hour
d	day
mo	month
y	year
w	week

To specify time units, you can specify the abbreviation, the full time unit, or the plural version of the time unit, such as `s`, `second`, or `seconds`. You must include a valid value with any time unit. If specifying seconds, minutes, or hours, you can specify a fractional value, such as `1.25h`. You cannot specify fractional values for other time units.

Reference-time abbreviations

Abbreviation	Full version	Reference time
*		Current time
t	today	00:00:00 (midnight) of the current day
y	yesterday	00:00:00 (midnight) of the previous day
sun[1]	sunday	00:00:00 (midnight) on the most recent Sunday
jun[2]	june	00:00:00 (midnight) on the current day in June of the current year
dec *DD*	december *DD*	00:00:00 (midnight) on the *DD*th day of December in the current year
YYYY		00:00:00 (midnight) on the current day and month in year *YYYY*
M-D or *M/D*		00:00:00 (midnight) on the *D*th day of month *M* in the current year
DD		00:00:00 (midnight) on the *DD*th day of the current month

[1]: Use the first three letters as an abbreviation for any day of the week: sun, mon, tue, wed, thu, fri, or sat.
[2]: Use the first three letters as an abbreviation for any month of the year: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec.

PI time expressions

PI time expressions can include fixed times, reference-time abbreviations, and time offsets. A time offset indicates the offset direction (either + or -) and the offset amount (a time-unit abbreviation with a value). Valid PI time expressions might include:

- Only a fixed time, such as `24-aug-2012 09:50:00`
- Only a reference-time abbreviation, such as `t`
- Only a time offset, such as `+3h`
- A reference-time abbreviation with a time offset, such as `t+3h`

Include at most one time offset in an expression. Including multiple offsets can lead to unpredictable results. For example, the following time expressions are *not* valid:

- `*+1d+4h`
- `t-1d+12h`

Time-stamp specification

To specify inputs for time stamps, you can enter time expressions that contain:

- Fixed times

  A fixed time always represents the same time, regardless of the current time.

Input	Meaning
23-aug-12 15:00:00	3:00 p.m. on August 23, 2012
25-sep-12	00:00:00 (midnight) on September 25, 2012

- Reference-time abbreviations

  A reference-time abbreviation represents a time relative to the current time.

Input	Meaning
*	Current time (now)
3-1 or 3/1	00:00:00 (midnight) on March 1 of the current year
2011	00:00:00 (midnight) on the current month and day in the year 2011
25	00:00:00 (midnight) on the 25th of the current month
t	00:00:00 (midnight) on the current date (today)
y	00:00:00 (midnight) on the previous date (yesterday)
tue	00:00:00 (midnight) on the most recent Tuesday

- Reference-time abbreviations with a time offset

  When included with a reference-time abbreviation, a time offset adds or subtracts from the specified time.

Input	Meaning
*-1h	One hour ago
t+8h	08:00:00 (8:00 a.m.) today
y-8h	16:00:00 (4:00 p.m.) the day before yesterday
mon+14.5h	14:30:00 (2:30 p.m.) last Monday
sat-1m	23:59:00 (11:59 p.m.) last Friday

- Time offsets

  Entered alone, time offsets specify a time relative to an implied reference time. The implied reference time might be the current clock time or another time, depending on where you enter the expression.

Input	Meaning
-1d	One day before the current time
+6h	Six hours after the current time

Time-interval specification

Time-interval inputs define intervals for collecting or calculating values during a time period. For example, you might specify a 60-minute interval to compute an hourly average over a 12-hour period. To specify time-interval inputs, enter a valid value and time unit:

- Positive values define intervals that begin at the earlier time in the period and that finish at or before the later time in the period.

Start time	2:00:00
End time	3:15:00

Time interval	30m
Returned intervals	2:00:00 to 2:30:00 2:30:00 to 3:00:00

- Negative values define intervals that finish at the later time in the period and that begin at or after the earlier time in the period.

Start time	2:00:00
End time	3:15:00
Time interval	-30m
Returned intervals	2:15:00 to 2:45:00 2:45:00 to 3:15:00

PI time terminology

The PI Server 2014 documentation uses improved terminology to describe PI time, the syntax you can enter in PI System utilities when prompted for dates and times. The syntax has not changed; only the words used to describe the syntax have changed. Specifically, the documentation no longer uses the following terms to describe PI time:

- Relative time
- Absolute time
- Combined time

For the current explanation of PI time, see PI time.

Daylight Saving Time considerations

Some time zones observe Daylight Saving Time. Daylight Saving Time introduces some considerations for PI time strings.

Procedure

1. Determine whether a time string is ambiguous.
2. Determine your time zones.
3. Specify time zones.

Determine whether a time string is ambiguous

Use **`pidiag -tz`** to determine if a specific time string is considered ambiguous:

Procedure

1. Open a command window and navigate to the `PI\adm` directory.
2. Type:

```
pidiag -tz "timeString"
```

where *timeString* is your time string. For example:

```
pidiag -tz "25-oct-98 1:30:00"
```

The **pidiag** utility returns several lines of output.

```
Passed Time: 25-Oct-98 01:30:00* PST Local: 909279000 UTC:909307800
```

The asterisk in the above line indicates that this is an ambiguous time string unless you include the time zone name.

Determine your time zones

Use **pidiag -tz** to determine names of your time zones:

Procedure

1. Open a command window and navigate to the `PI\adm` directory.
2. Type:

   ```
   pidiag -tz
   ```

   The **pidiag** utility returns several lines of output, including your time zone names.

Specify time zones

In almost all cases, PI can accurately determine whether daylight saving time is in effect. If you wish to be specific, then use the DD-MMM-YY hh:mm:ss.ssss absolute time format and append the time zone.

PI supports both long time zone names (such as Pacific Standard Time) and short time zone names (such as PST). You may specify either name. Comparisons are not case sensitive. The following time strings are equivalent:

```
25-Oct-98 01:30 Pacific Daylight Time
25-Oct-98 01:30 pdt
25-Oct-98 01:30 D
```

Other commonly-used time zone codes include S for standard time, D for daylight time, and MET for Middle European Time.

How to display time zone information

Use **pidiag -tz** to determine names of your time zones:

Procedure

1. Open a command window and navigate to the `PI\adm` directory.
2. Type:

   ```
   pidiag -tz
   ```

   The **pidiag** utility returns several lines of output, including your time zone names. For more detailed results, use the **pidiag -tz** display options.

Procedure

1. Time zone information and Daylight Saving Time transition rules.
2. Display options.

3. Display local and UTC time.
4. Display a different time zone.
5. Customize standard and Daylight Saving Time changes.

Time zone information and Daylight Saving Time transition rules

Without the optional **TZ** parameter, `pidiag -tz` displays the time zone information and Daylight Saving Time (DST) transition rules that are being used by the PI server. If the file `PI\dat\localhost.tz` is present and valid, then the time zone information is from the file. Otherwise, the information is from the operating system.

StartYear, **EndYear**, **Month**, **Week**, **Day**, **Time**, and **Offset** define the daylight and standard time transition rules.

The transition rules are:

- **StartYear** is the first year that the rule is in effect
- **EndYear** is the last year that the rule is in effect
- **Month** is the month (1-12) that the rule is applied.
- Week is the week (1-5) that the rule is applied. If **Week** is 5 and there are only four weeks in the month, then 5 designates the last week in the month. If **Week** is -1, then **Week** is ignored and day becomes absolute.
- If **Week** is greater than 0, then **Day** is the relative day (1-7) that the rule is applied. A **Day** of 1 represents Sunday, a **Day** of 2 represents Monday, and so on. For example, a **Week** of 1 and a **Day** of 1 means the first Sunday in April. If **Week** is -1, then **Day** is an absolute day (1-31).
- **Time** is the time in seconds after midnight that the rule is applied.
- **Offset** is the time in seconds to subtract from standard time to get the local time. For example, when daylight saving time is in effect, -3600 is subtracted from standard time.

If your time zone does not observe daylight saving time, the output indicates this.

```
C:\PI\adm> pidiag -tz
TZ environment variable: <not set>
Standard Time Name: US Mountain Standard Time (UMST)
Daylight Saving Time: <not observed>
```

See Customize standard and Daylight Saving Time changes to change this setting.

Display options

The **-check** option generates no output at all, unless the time zone settings on your system are invalid.

The **-dump** option dumps the whole time zone table. This includes fall/spring changes in every year. The dump is in comma-separated variable (CSV) format and can be loaded directly into a spreadsheet, if all time-change information for the local time zone.

The **-dump** option can be combined with **-brief**. The output with this option includes the year and spring and fall time changes, each marked with D or S to denote daylight or standard time.

The **-full** option can be used to display additional information about the `localhost.tz` file, such as the file's UID, creator, creation time, and so on. The information is valid only if `localhost.tz` was successfully loaded by the PI server.

Display local and UTC time

Use **pidiag -tz** to display the Local and UT times (in seconds) for a specified time:

Procedure

1. Open a command window and navigate to the `PI\adm` directory.
2. Type:

   ```
   pidiag -tz "timeString"
   ```

   where *timeString* is your time string. For example:

   ```
   pidiag -tz "25-oct-98 1:30:00"
   ```

   The **pidiag** utility returns several lines of output, including:

   ```
   Passed Time: 25-Oct-98 01:30:00* PST Local: 909279000 UTC:909304200
   ```

Display a different time zone

If, in addition to time parameter, the **TZ** (time zone) parameter is specified, **pidiag -tz** displays the time zone information of the provided time zone and converts the time as if the provided time zone were in effect. Note that the **TZ** argument follows the time/year argument, so you must provide a time string or year to use this feature. The specified time zone can be different from the local time zone.

```
C:\PI\adm>pidiag -tz "*" GMT0BST
TZ environment variable: GMT0BST
Standard Time Name: GMT (GMT)
Daylight Time Name: BST (BST)
StartYear, EndYear, Month, Week, Day, Time, Offset
1970, 2037, 4, 1, 1, 7200, -3600
1970, 2037, 10, 5, 1, 7200, 0
Passed Time: 8-Oct-03 20:27:04 BST Local: 1065644824 UTC:
1065641224
```

Customize standard and Daylight Saving Time changes

PI uses an internally constructed table to determine when changes between Standard Time and Daylight Saving Time (DST) occur. By default, PI builds the table using the single time change rule available from Windows. However, when the `PI\dat\localhost.tz` file is present and valid, PI builds the table using the change rules specified in the file.

How to translate time formats

Use **pidiag -tz** to determine if a specific time string is considered ambiguous:

Procedure

1. Open a command window and navigate to the `PI\adm` directory.
2. Type:

```
pidiag -t time [U]
```

This provides translation between time string formats and internal formats: If time starts with 0 (zero) an integer format (seconds since 1-jan-70) is translated to string representation. **pidiag** assumes local time, unless the third argument is U or UTC, in which case the argument is taken to be universal time (GMT).If the first character is not 0, the time argument is treated as time string, absolute or relative, and translated into an integer value. Both local time and UTC integer values are displayed.

Procedure

1. String to integer format sample output.
2. Integer format to string sample output.

String to integer format sample output

If you type the following command:

```
C:\PI\adm>pidiag -t 1-sep
```

Then you get output that looks like this:

```
1-Sep-98 00:00:00 PDT - Local: 904608000 UTC: 904633200
```

If you type the following command:

```
C:\PI\adm>pidiag -t t+1h
```

Then you get output that looks like this:

```
21-Oct-98 01:00:00 PDT - Local: 908931600 UTC: 908956800
```

If you type the following command:

```
C:\PI\adm>pidiag -t "*"
```

Then you get output that looks like this:

```
21-Oct-98 20:00:10 PDT - Local: 909000010 UTC: 909025210
```

Integer format to string sample output

If you type the command:

```
C:\PI\adm>pidiag -t 0909000010
```

Then you get output that looks like this:

```
21-Oct-98 20:00:10 PDT - Local: 909000010 UTC: 909025210
```

Back up PI Server

Topics in this section

- PI Server scheduled backup configuration
- View backup history of a PI Server
- Troubleshooting PI Server backups
- PI Server backup scripts
- How PI Server backup verification works
- Restore PI Server from a backup

PI Server scheduled backup configuration

Topics in this section

- Scheduled backups overview
- Recommended disk configuration for PI Server backups
- Configure a PI Server scheduled backup
- PI Server collectives and backups

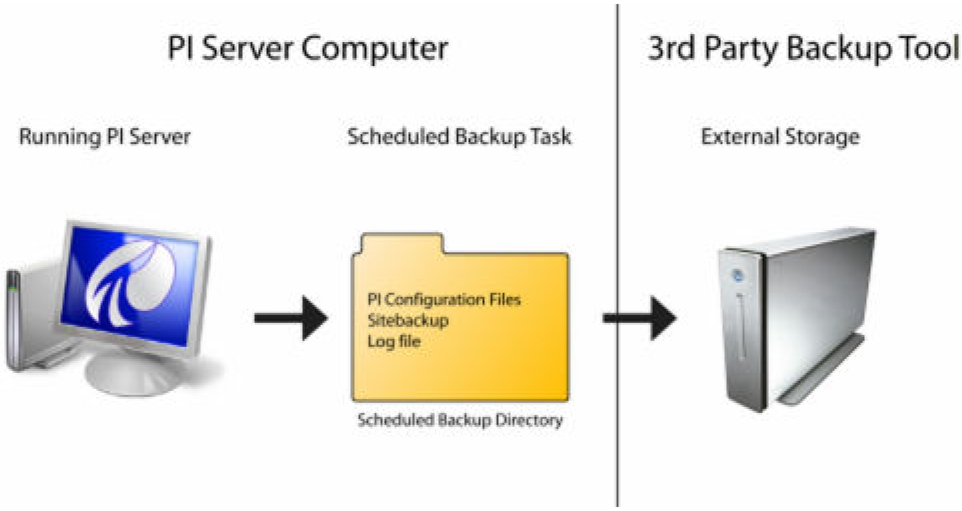
Scheduled backups overview

OSIsoft recommends that you configure an automatic daily backup of the PI Server. The PI Server installation includes a script that you can use to set up a daily backup as a Windows service. In this document, we refer to this backup task as the scheduled backup task. The scheduled backup task performs an incremental, verified backup each day.

The scheduled backup task places the backup files in a directory specified by the Windows task, which we refer to as the scheduled backup directory. The scheduled backup directory holds only the most recent verified backup.

In addition to the scheduled backup task, OSIsoft strongly recommends that you use a third-party backup application to back up the scheduled backup directory to a safe location. The backup of the scheduled backup directory will allow you to restore the backup directory at a point in time in the past.

The following diagram illustrates the two steps that should occur each day: the backup of the running PI Server, and the backup of the scheduled backup directory.



PI Server access during daily backups

You can access the PI Server as usual while the scheduled backup task is running. You can create points, push data to the archives, and so on. To minimize performance impact during backups, use the recommended disk configuration. See Recommended disk configuration for PI Server backups.

Note:

By default, the backup task uses Microsoft's Volume Shadow Copy Services (VSS) to enable access to the PI Server during backups. If VSS is not supported on the PI Server computer, then there are some limitations to access during backups. See Comparison of VSS and non-VSS backups for more information.

Recommended disk configuration for PI Server backups

VSS backups require at least one NTFS partition on the machine where the PI Server is installed. For optimum performance during backups the following files should be on separate drives:

- The PI data archives and event queue.
- The paging file of the operating system.
- The scheduled backup directory (this drive can be a remote network drive or NAS).

For example, the PI Server may be installed on the `C:` drive, which is usually where the paging file is located, while the archives and event queue are configured to be on the `D:` drive and the intermediate backup directory is on the `E:` drive or a remote network drive.

All archives to be backed up must be on the PI Server. The backup will fail if the archive to be backed up is on a remote drive, such as a mapped network drive. (This is true for all VSS backups.)

Procedure

1. Performance during PI Server backups.

Performance during PI Server backups

Any backup on the PI Server computer has the potential to impact PI server performance. You can largely avoid this impact by using the recommended disk configuration. However, some impact is unavoidable because CPU resources and file system cache resources are consumed.

Monitor your PI Server during a backup to determine how the backup affects archiving rate, archive reads, and the CPU usage of the PI Server. Also, monitor the Windows Performance counters **`Avg Disk Write Queue Length`** and **`Avg Disk Read Queue Length`** for the Physical Disk performance object. If the disk queue length is greater than one, then the disk is falling behind.

Configure a PI Server scheduled backup

The recommended strategy for automating backups of PI Server is as follows:

Procedure

1. Establish a baseline backup in the scheduled backup directory. The scheduled backup directory is the directory that contains the backup files generated by the scheduled backups. (You do not need to perform this step for new installations.)
2. Set up a PI Server backup to run as a Windows automatic task. The PI Server includes scripts for creating the task and performing the backups. The task backs up files and copies them to the scheduled backup directory. The name and location of the backup directory is configurable, as is the time of the backup.
3. Use a third-party backup tool to automate a regular backup of the files in the backup directory. This backup should store the files on a different computer from the PI Server. The PI Server includes a script that you can use for this, if a third-party tool is not an option.
4. Run a test backup to make sure that the scheduled backup task is working correctly.

Topics in this section

- Establish a baseline backup
- Create the scheduled backup task
- Add the PIPC directory to the PI Server daily backup task
- Configure automatic backup of the PI Server scheduled backup directory
- Run a test backup

Establish a baseline backup

Procedure

1. In a command prompt window, navigate to the `PI\adm` directory.
2. Create the backup.
   - If you have the disk space to include all the archives, then type:

     ```
     piartool.exe -backup backupdir - type full -arcdir -wait
     ```

where *backupdir* is the full path to the backup directory, as described in File paths in baseline backup commands.

> **Note:**
>
> Empty archives are not identified for backup.

- If you do not have the disk space to include all the archives and want to include only a subset of the archives in the baseline backup, then type:

  ```
  piartool -backup backupdir -numarch num -arcdir -wait
  ```

  where *backupdir* is the full path to the backup directory as described in File paths in baseline backup commands, and *num* is the number of archives you want to keep in the backup directory. For example, specifying **`-numarch 2`** backs up the primary archive and archive 1.

Topics in this section

- File paths in baseline backup commands
- Baseline backup command examples

File paths in baseline backup commands

When you issue the baseline backup comand, you specify the path to the scheduled backup directory. The path can be a UNC path. Here are a few examples of valid paths:

```
C:\pibackup\
\\myserver\c$\pibackup\
\\myserver\share\pibackup\
```

> **Note:**
>
> The UNC path can be a path to a shared directory on a remote computer, but mapped network drives cannot be used in the full path.

Baseline backup command examples

- To establish a baseline backup with `D:\PI\backups` as the backup directory, you would go to the `PI\adm` directory and type the following command:

  ```
  piartool -backup D:\PI\backups -type full -arcdir -wait
  ```

- To establish a baseline backup with four archives and `D:\PI\backups` as the backup directory, you would go to the `PI\adm` directory and type the following command:

  ```
  piartool -backup D:\PI\backups -numarch 4 -arcdir -wait
  ```

Create the scheduled backup task

Procedure

1. On the PI server computer, log in with a Windows account that has administrator privileges.
2. Open a command prompt window.
3. Change to the `PI\adm` directory.

   For example, if the PI server is installed on the `D` drive, you would type:

   ```
   cd /d "%piserver%adm"
   ```

4. Select a target directory for your backups, for example `e:\pibackup`. Ideally, the `e:` drive does not correspond to the system drive or the drive where your archives are stored.

   ```
   pibackup e:\pibackup -install
   ```

   This command creates a Windows Scheduled Task named "PI Server Backup" .

   On Windows 2000 Server, the task is named `At`*n*, where *n* is the next available task number when the task was created.

5. If you have installed the scheduled task on Windows 2000, rename the scheduled task to *PI Server Backup* by right-clicking the task name and choosing **Rename**.

6. Open the scheduled task to verify that it was created.

After you finish

Use the backup tool of your choice to back up the `pipc` directory, which is not included in the PI Server scheduled backup task. If no backup tools are available, you can configure the PI Server scheduled backup task to back up this directory See Add the PIPC directory to the PI Server daily backup task.

Configure an automatic backup of the backup files generated by the scheduled backup task. See Configure automatic backup of the PI Server scheduled backup directory

Topics in this section

- Customize the default backup
- Change scheduled time of PI Server automated backup

Customize the default backup

After installing the PI Server backup as a scheduled task, you can customize the task.

Before you start

To make any of these changes, open the Windows Task Scheduler, and double-click the PI Server Backup entry.

Procedure

- Change backup time.

  The default time is 3:15 AM.

- Change the Windows user under which the task runs.

  By default, the task runs under the System account. If you are using the `pisitebackup.bat` script to backup files to a remote computer, then you will need to run the scheduled task as a user that has sufficient privileges to write to the target directory on the remote computer.

- Change the path to the scheduled backup directory.

Change scheduled time of PI Server automated backup

To change the scheduled time of the automatic backup, edit the Windows task.

Procedure

1. On the PI Server computer, log on to a Windows account that has administrator privileges.
2. In Control Panel, open **Scheduled Tasks**.
3. Double-click the **PI Server Backup** task.

   A configuration dialog box opens.
4. Click the **Schedule** tab.
5. Select a **Start time** for the backup.
6. Click **OK**.

Add the PIPC directory to the PI Server daily backup task

If you choose not to use third-party softwareto back up the files in the `pipc` directory, you can configure the scheduled backup task to include include them.

Procedure

1. On the PI Server computer, log on to a Windows account that has administrator privileges.
2. Open a Windows command window.
3. Change to the `PI\adm` directory: `cd %piserver%adm`
4. Find the file called `pisitebackup.bat.example`.
5. Rename this file to `pisitebackup.bat` by removing the `.example` extension.

   This sets up a backup of a default set of files in the `pipc` directory. This backup runs immediately after the automated PI Server Backup task executes. The backup files are placed in a subdirectory called `sitebackup` within the PI Server backup directory.
6. By default, `pisitebackup.bat` backs up the `.bat`, `.log`, `.ini`, `.txt`, and `.sql` files from the `pipc` directory. To change which files are backed up, edit the `pisitebackup.bat` file.

After you finish

Note:
You can edit `pisitebackup.bat` to include any scripts you want to run immediately after the scheduled PI Server Backup task executes. For more information, read the instructions in the `pisitebackup.bat.example` file.

Configure automatic backup of the PI Server scheduled backup directory

Procedure

1. Use a third-party backup tool to configure regularly scheduled backups of the scheduled back up directory.

   You can use any backup strategy that is available with the third-party backup program. For example, you might choose to do a combination of full and incremental backups, full and differential backups, a combination of full, incremental, and differential backups, and so on. The exact terminology and strategies vary among backup programs.

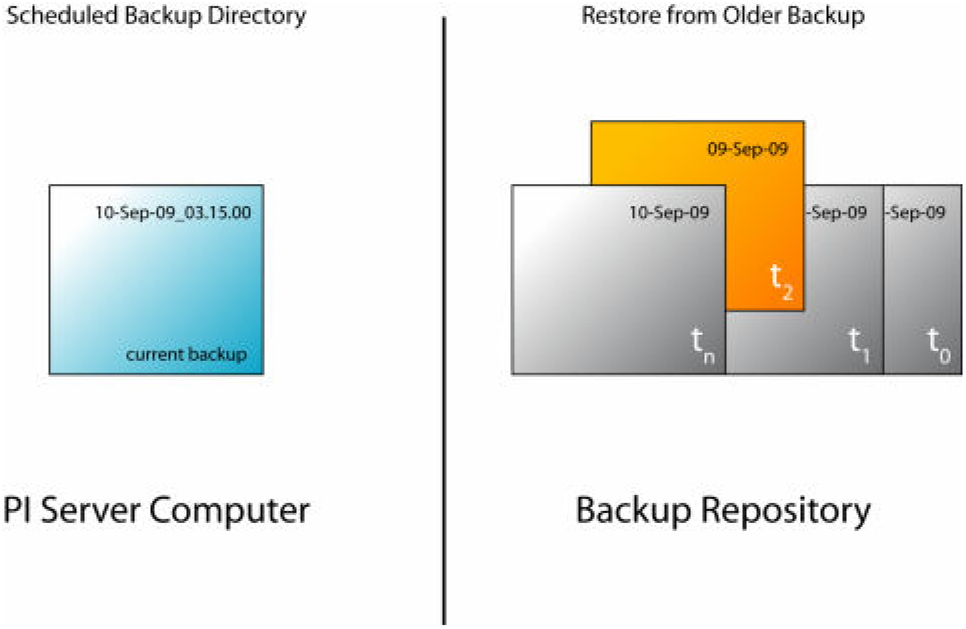
Third-party software will significantly reduce your backup administrative tasks. For example, typical third-party backup software will automatically discard old backups after a configurable expiration period. Also, third-party backup software can typically be managed by IT so that the PI system manager can attend to other tasks.

Note:
If third-party backup software is not available, you can use the `pisitebackup.bat` script to automatically copy the backup directory to a remote computer.

Why back up the scheduled backup directory?

Backing up the files in your backup directory is a crucial step to safeguarding your PI server. The backup directory contains only the most recent backup. As new backup files are copied into the backup directory, the old backup files are overwritten. This means that the backup directory contains only the most current verified backup. This does not help you if you need to restore to an older backup.

For example, suppose you discover that two days ago you accidentally deleted a point. You cannot recover the deleted point from the files in the backup directory because the last backup occurred after the point was deleted. You need to recover the point from an earlier backup. Your backups of the PI server backup directory will provide the backup history that allows you to restore the point.



Run a test backup

Run a couple of test backups. In this example, the backup directory is assumed to be `E:\pibackup\`. The PI Server directory is assumed to be `C:\pi\`.

Procedure

1. Run the **`pigetmsg -f`** command to monitor messages that are written to the PI Message log during the backup.
2. In the Windows Task Scheduler, start a test backup by right-clicking on the PI Server Backup scheduled task and selecting **Run**. Files will be backed up to the following directories.
   - Archives will be backed up to `E:\pibackup\arc`.
   - Files from `C:\pi\dat`, `C:\pi\adm`, `C:\pi\log`, and `C:\pi\bin` are backed up to `E:\pibackup\dat`, `E:\pibackup\adm`, `E:\pibackup\log`, and `E:\pibackup\bin`, respectively.
3. Monitor the PI Message Log messages from the **`pigetmsg -f`** command. At the beginning of the backup, you should see the message `Backup operation started`. You may see -15033 errors because PI Message Subsystem is briefly unavailable for logging during the backup. Because of this, you may not see the `Backup operation completed successfully` message in the **`pigetmsg -f`** output, even though the message does eventually get written to the message log. You can tell when the backup is complete by examining the task status in the Windows Task Scheduler.
4. After the backup is complete, examine the backup log in `E:\pibackup`. The log will have a name of the form `pibackup_`*`dd-MMM-yy_hh.mm.ss`*`.txt`.
   - Near the beginning of the log you will see the list of the currently registered archives. This archive list can be helpful during restore operations. For example, when restoring the PI server, it is helpful to know at the time of the backup which archive was the primary and which directories the archives were in.
   - At the very end of the log you should see the message `pibackup.bat script completed successfully` if the backup was successful.
   - The log contains a `Verbose File Backup Report` indicating which files were copied to the backup directory.
   - The log displays the list of subsystems that were registered for backup and the subsystem version numbers.
   - The log also contains a summary of the backup that looks similar to the following.

```
         Backup in Progress:     FALSE
               Files Copied:     24
              Files Skipped:     36
         File Copy Failures:     0
           Total File Count:     60
  Last Backup Start Pending:     1-Nov-09 03:15:05
          Last Backup Start:     1-Nov-09 03:15:26
                        End:     1-Nov-09 03:15:42
                     Status:     [0] Success
           Last Backup Type:     INCREMENTAL, VSS, Component Mode
          Last Backup Event:     BACKUPSHUTDOWN
     Last Backup Event Time:     1-Nov-09 03:15:43
    Verification Start Time:     1-Nov-09 03:15:42
              VSS Supported:     TRUE
```

The type of the backup should be INCREMENTAL, which is true for all newly installed backup tasks in 3.4.380. The first incremental backup of a newly installed PI Server should be equivalent to a full backup. A backup type of NUMARCH/CUTOFF is possible only if the backup task is left over from an upgrade.

PI Server collectives and backups

At a minimum, configure backups for the primary server in a PI server collective. PI server collectives are not a substitute for backing up your PI server. Backups provide a means of recovery after unintended configuration changes and database corruption. An example of an unintentional configuration change is an accidental point deletion. Replication will propagate this mistake to all nodes in the collective. Depending on when the mistake is detected, the only means to recover the deleted point may be to restore a backup that was taken prior to the deletion of the point.

Consider whether to configure backups for the secondary servers as well as the primary. There are several good reasons to backup secondary servers.

- Not all configuration information is replicated. Non-replicated data includes tuning parameters and PI Server message logs. In part, these files can be enumerated by the **`piartool -backup -identify -verbose`** command; the nonreplicated components where the data may differ between the primary and secondary nodes include the timeout parameters, pimsgss, and pibatch components. However, non-replicated data also includes customized batch scripts, `.ini` files, and logs that can be backed up with the `pisitebackup.bat` script.
- Database corruption can occur independently on primary and secondary nodes. The validation step at the end of the backup may, for example, detect corruption on a secondary node that did not occur on the primary node.
- If the secondary and primary are geographically separated across a slow network, then it might be more expedient to restore the secondary from a backup, rather than reinitializing from the primary.

The start and end time of archives are not the same on primary and secondary nodes. Reinitializing a secondary typically requires manual steps to eliminate data gaps. If a secondary is restored from backup, there will be no data gaps.

If you restore a primary PI Server from a backup, you must reinitialize all secondary servers from the primary PI Server . If you restore the primary PI Server from a backup of a secondary PI Server, you must reinitialize the other secondary servers.

View backup history of a PI Server

Procedure

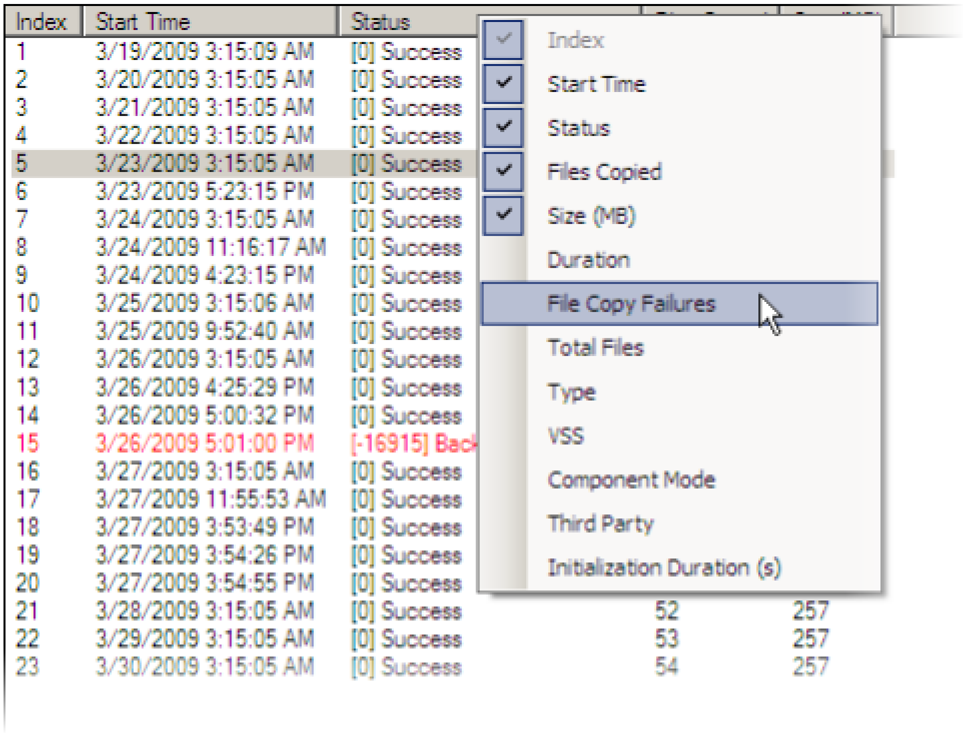
1. Open PI SMT.
2. Under **Collectives and Servers**, select the server you want to check.
3. Under **System Management Tools**, select **Operation** > **Backups.**
4. In the **PI Server** drop-down list, select the server you want to examine. The list includes all of the servers selected under **Collectives and Servers**.

   The **Backup History** table shows the backup history for the server selected in the **PI Server** list. By default, the table lists the last 100 backups and shows a subset of these available data:

Column	Data
Index	Number to represent the order in which the backups occurred.

Column	Data
Start Time	Time the backup started.
Status	Status code and the status code description for each backup.
Files Copied	Number of files copied during the backup.
Size	Total size of the backup.
Duration	Time it took to complete the backup.
File Copy Failures	Number of files for which the backup failed.
Total Files	Number of files selected for backup.
Type	Copy, Incremental, Differential, Full, and Numarch/Cutoff. See PI Server backup types.
VSS	**True** for VSS backups; **False** for non-VSS backups.
Component Mode	**True** for component mode backups; **False** if not.
Third Party	**True** if a third-party backup application was used to backup the PI Server; **False** if the PI Server Backup Subsystem was used.
Initialization Duration	Time elapsed before the backup began after the backup request.

5. Right-click a column heading to see a complete list of columns you can display.



Index	Start Time	Status		
1	3/19/2009 3:15:09 AM	[0] Success		
2	3/20/2009 3:15:05 AM	[0] Success		
3	3/21/2009 3:15:05 AM	[0] Success		
4	3/22/2009 3:15:05 AM	[0] Success		
5	3/23/2009 3:15:05 AM	[0] Success		
6	3/23/2009 5:23:15 PM	[0] Success		
7	3/24/2009 3:15:05 AM	[0] Success		
8	3/24/2009 11:16:17 AM	[0] Success		
9	3/24/2009 4:23:15 PM	[0] Success		
10	3/25/2009 3:15:06 AM	[0] Success		
11	3/25/2009 9:52:40 AM	[0] Success		
12	3/26/2009 3:15:05 AM	[0] Success		
13	3/26/2009 4:25:29 PM	[0] Success		
14	3/26/2009 5:00:32 PM	[0] Success		
15	3/26/2009 5:01:00 PM	[-16915] Back		
16	3/27/2009 3:15:05 AM	[0] Success		
17	3/27/2009 11:55:53 AM	[0] Success		
18	3/27/2009 3:53:49 PM	[0] Success		
19	3/27/2009 3:54:26 PM	[0] Success		
20	3/27/2009 3:54:55 PM	[0] Success		
21	3/28/2009 3:15:05 AM	[0] Success	52	257
22	3/29/2009 3:15:05 AM	[0] Success	53	257
23	3/30/2009 3:15:05 AM	[0] Success	54	257

Topics in this section

- PI Server backup types
- Change the number of backups shown in the Backup History table
- View backup information summary
- View backup details

PI Server backup types

The **Backup History** table might show the following backup types:

- **Copy**

  The backup type for unscheduled backups, that is, those run with the **Backups** tool.

- **Incremental**

  The backup type for the regularly scheduled PI Server backups.

- **Differential**

  A backup type if you are using third-party backup software to back up the PI Server.

- **Full**

  A backup type if you are using third-party backup software to back up the PI Server.

- **Numarch/Cutoff**

  The backup type for regularly scheduled backups that were configured on PI Server versions 3.4.370 or 3.4.375.

Topics in this section

- VSS and non-VSS backups
- Comparison of VSS and non-VSS backups

VSS and non-VSS backups

The PI Server can use one of two different backup mechanisms: VSS or non-VSS. VSS backups are not supported on all operating systems. Non-VSS backups have some restrictions on PI Server access while the backup is taking place.

- VSS Backups: The PI Server's preferred online backup mechanism uses Volume Shadow Copy Services (VSS). With these backups, archiving is interrupted only for a brief time, typically less than a second. During the backup you can create points, push data to the archives, and so on. The PI Server supports VSS backups on Microsoft Windows 2003 SP1 and greater.
- Non-VSS Backups: On operating systems where VSS backups are not supported, the PI Server still provides online backup functionality by doing non-VSS backups. Non-VSS backups have the following limitations:
  - Files are read-only while they are being backed up
  - You cannot create new points during the backup
  - You cannot push new data into the archives during the backup

Comparison of VSS and non-VSS backups

The following table compares the PI Server impact for VSS and non-VSS backups.

Database	Operation	Applies to	Impact
Snapshot Database	Read and Write	VSS and Non-VSS	The snapshot remains available for read and write operations. For example, a PI ProcessBook trend that remains open during the course of a backup will continue to receive data without interruption. However, if the revert button is pressed, data that arrived during the course of the backup may not be available from the archive depending on whether the data can be written during the backup. See the Write operation under Archive Databases below.
Message Logs	Read and Write	VSS	The message logs are unavailable for a brief time, typically less than a second.
		Non-VSS	The message logs are unavailable during the entire time period that it takes for the message log files to be backed up.
Archive Databases	Read	VSS and Non-VSS	All data that is in the archive remains available for read without interruption.
	Write	VSS	Archiving is turned off only for a brief time, typically less than 1 second.
		Non-VSS	While each archive is backed up, archiving is turned off. The total amount of time that archiving is turned off depends on the total size of all archive files that are included in the backup and the speed at which PI Backup Subsystem can be copy the archives to the destination directory for the backup.

Database	Operation	Applies to	Impact
Base Databases (for example, point and Module configuration data)	Read	VSS and Non-VSS	Configuration data remains available for read without interruption during the entire backup. For example, a tag search can be done at any point during the backup.
	Write	VSS	The base databases are unavailable for configuration changes only for a brief period of time, typically less than a second. For example, PI points and PI Modules can be created or edited during the course of a backup. Although database changes are possible, it is not recommended to make large-scale configuration changes during a backup.
		Non-VSS	The Point Database cannot be altered when the snapshot, archive, or base databases are backed up. Other base databases cannot be altered when the database itself is backed up.

Change the number of backups shown in the Backup History table

Set the Backup_MaxHistory tuning parameter to change the number of backups shown in the **Backup History** table.

Procedure

1. Open PI SMT.
2. Under **Collectives and Servers**, select the server you want to check.
3. Under **System Management Tools**, select **Operation** > **Tuning Parameters.**
4. In the Tuning Parameters tool, click the **Backup** tab.
5. Double-click **Backup_MaxHistory** to open the window where you set the parameter value.

View backup information summary

The **Summary** fields below the **Backup History** table contain information about the selected backup.

Procedure

- Select a backup in the **Backup History** table to view information about the backup in the **Summary** fields below the table:
  - **Status**

    The status code of the backup: either **Success** or an error code for the error that occurred.
  - **Method**

    Information about the backup method:
    - **Type**

      Backup types include incremental, copy, full, differential, or Numarch/Cutoff (see PI Server backup types).
    - **VSS/non-VSS**

      Method use, as describe in VSS and non-VSS backups.
    - **Component Mode**

      Meaningful only for third-party backups. Some third-party backup applications cannot do component mode backups. These backup applications do not provide feedback to the PI Server as to the success or failure of the backup.
    - **Third party**

      Appears if PI Backup Subsystem is not the application used to perform the backup.
  - **Start Time**

    The time the backup started.
  - **Duration**

    The time taken for the backup to complete.

View backup details

The Backups tool provides access to details about PI Server backups.

Procedure

- Double-click a backup in the **Backup History** table to open the Backup Details window. The Backup Details window shows a summary of backup details and a list of backed up files.

Topics in this section

- Backup details summary
- Backed up file list

Backup details summary

Use the **Summary** tab in the Backup Details window to view details about the selected backup. At the top left of the tab, the **Index** field shows the index of the backup you are currently viewing.

The index is a number representing the order in which the backup occurred, relative to the other backups in the list. Use these options to see details for a different backup:

- Click the arrows on the **Index** field
- Enter an index number directly in the **Index** field
- Use the **Previous** and **Next** arrows at the top of the window

The **Summary** tab displays all the information that is included in the **Summary** area under the main **Backup History** table. See View backup information summary for more on these fields. The **Summary** tab also provides:

- Backup statistics on each subsystem and archive included in the backup. The statistics shown are different for VSS and non-VSS backups.
- A list of the subsystems that were available for backup at the time the backup was executed.

Topics in this section

- Backup statistics for VSS backups
- Backup statistics for non-VSS backups

Backup statistics for VSS backups

For VSS backups, the following statistics are displayed:

- **Freeze Start Time**

  Time at which all subsystems entered a frozen state; if the backup is successful, the last backup time of archive files will be updated with this time stamp.

- **Freeze Duration Time**

  Maximum amount of time that the databases could have been frozen (to writes) for a subsystem that was backed up.

- **Freeze Transition**

  Time spent for all subsystems to go from an unfrozen to frozen state.

- **Initialization Duration**

  Time elapsed between the issue of the backup request and the start of the backup.

Backup statistics for non-VSS backups

For non-VSS backups, the following statistics are displayed:

- **Name**

  Name of subsystem.

- **Backup Start**

  Time the backup started.

- **Freeze Transition**

  Time spent for all subsystems to go from an unfrozen to frozen state.

- **Backup duration**

  Time spent to complete the backup.

Backed up file list

The **Backed Up File List** tab displays some brief summary information and a list of all files that were backed up. In the case of an aborted or otherwise unsuccessful backup, the files that were *not* backed up are also listed, in red.

The summary information consists of:

- **Total Files**

  Number of files selected for backup.

- **File Copy Failures**

  Number of files that should have been backed up and were not.

- **Files Backed Up**

  Number of files that were backed up.

- **Unchanged Files for Incremental Backup**

  Number of files that were not backed up because they were unchanged since the last backup; this field appears only if the selected backup was an incremental backups. See PI Server backup types.

The table at the bottom of the tab shows lists the backed-up files:

Column	Description
File Name	Name of the file
Source Directory	Name of the directory in which that file is located on the PI Server
Report	Action taken or the reason the backup failed
Destination	Name of the directory to which this file was backed up
Status	A status code: `success` for a successful backup or an error for an unsuccessful backup
Size	The size of the file in KB
Component Description	The name of the component to which the file belongs.

You can:

- Click a column heading to sort the list by that column.
- Right-click a column heading to select the columns you want to display.

Troubleshooting PI Server backups

Topics in this section

- View PI Server backup logs and messages
- Common problems with VSS backups
- PI Server backup failure due to offline subsystem
- Perform an on-demand PI Server backup
- Export PI Server backup history

View PI Server backup logs and messages

Check the following log files for error messages:

- Backup script log

  The backup script log is written to the target directory of the backup with a name of the form `pibackup_`*`dd-mmm-yy_hh.mm.ss`*`.txt`. For example:

  ```
  pibackup_5-Aug-05_14.16.22.txt.
  ```

- PI message log

  To display all error messages between the start and end time of a backup, use a command of the form:

  ```
  pigetmsg -sl E –st starttime –et endtime
  ```

  If any errors occur during a backup, the output of this command is automatically dumped to the backup script log.

  To display all messages related to backup, use a command of the following form:

  ```
  pigetmsg -src pibackup –st starttime –et endtime
  ```

  To display only those messages from the PI Backup subsystem itself, use a command of the following form:

  ```
  pigetmsg -pr pibackup –st starttime –et endtime
  ```

- Windows application event log

  Look for messages from VSS and COM+ event sources.

- Windows system event log

  Look for messages from VOLSNAP and NTFS event sources.

Common problems with VSS backups

These are common reasons why VSS backups fail:

- Service Pack 1 or greater of Windows Server 2003 is not installed. There were many bug fixes with regard to VSS in Service Pack 1 of Windows Server 2003. VSS backups are not reliable in Windows Server 2003 without service pack 1.
- `Ole32.dll` is not registered. Sometimes the system can get into a state where the `ole32.dll` becomes unregistered. If `ole32.dll` becomes unregistered then VSS backups will not work. This problem can be solved by re-registering `ole32.dll` with the following command:

  ```
  regsvr32 ole32.dll
  ```
- VSS backups fail if either of the following services is disabled:
  - Microsoft Software Shadow Copy Provider
  - Volume Shadow Copy
- Typically, the Volume Shadow Copy service should not be running. It is started on demand whenever it is needed. If the service is running, this could mean that the service is stuck in a bad state. You can stop the service with the following command:

  ```
  net stop "Volume Shadow Copy"
  ```

  If this command does not work, you might need to end the `VSSVC.exe` process in Windows Task Manager.
- Third-party VSS providers can cause VSS backups to fail. VSS backups of the PI Server have been known to fail if the OfmProvider from St. Bernard software is installed on the PI Server machine. You can determine whether this software provider is installed by running the **`vssadmin list providers`** command and looking for the `OfmProvider` in the output. There are no other known problems with third-party VSS providers.
- For VSS backups, all archives to be backed up must be on the PI Server node. The VSS backup will fail if the archive to be backed up is on a remote drive, such as a mapped network drive.
- VSS backups require at least one NTFS partition on the machine where the PI Server is installed.

PI Server backup failure due to offline subsystem

When a subsystem registers for backup, the subsystem must remain online during the next backup or else the backup will fail with the following error:

```
Backup start failed, Status: [16896] RPC Resolver offline for a subsystem to
which the backup subsystem is communicating. Find -10733 error in message log to
identify problematic RPC
```

Two messages will appear in the PI server message log with the `-10733` error similar to the following:

```
E 19-Oct-09 13:54:57 pibackup                           (5059)
>> Callback failed for <pibatch> for the IDENTIFY event. Error [10733]
PINET: RPC Resolver is Off-Line.
E 19-Oct-09 13:54:57 pibackup                           (5061)
>> Error [10733] PINET: RPC Resolver is Off-Line., failed to send the IDENTIFY
backup event to pibatch
```

To fix the problem, you can either start the subsystem that is not running, or you can do the following, if the subsystem was purposefully stopped:

1. Run the command **`piartool -backup -query`** and make note of the subsystems that are currently registered for backup.
2. Restart PI Backup Subsystem.
3. Wait for the previously registered subsystems to register for backup again, with the exception of the problematic subsystem. Subsystems may take up to 5 minutes to re-register for backup after the backup subsystem has been restarted.

Perform an on-demand PI Server backup

Use the Backups tool in SMT to run on-demand backups of the PI Server for testing and troubleshooting. On-demand backups run in component mode.

> **Note:**
> Do not use the Backups tool as a substitute for scheduled backups. Your daily backups should be set up as a Microsoft Windows scheduled task.

Before you start

In the default PI Server configuration, you must have `piadmin` privileges to perform a backup. To allow other users to back up the PI Server, use the **Database Security** tool to change permissions in the PIBACKUP table.

Procedure

1. In the Backups window, select the PI Server from the **PI Server** list.
2. Click **Backup Now** .
3. In the server Backup window, select the PI Server components to back up .

   As you make a selection, the right side of the window shows the list of files that will be backed up.
4. In the **Backup Location** field, type the path for the target directory in which to save the backup files.

   You can specify a UNC path or a local path on the PI Server that you are backing up. Alternatively, click the browse button ... and browse to the target directory.

   > **Note:**
   > If you are not running PI SMT on the same PI Server that you are backing up, then you cannot use the browse button unless you have Windows administrator access to that PI Server.
5. If your operating system supports VSS, select or clear the **Use VSS Backup** check box to indicate the type of backup to perform. you have the option to choose whether to do a VSS or non-VSS backup. .
6. Click the **Backup** button.

   The bottom of the **Backup History** window shows the progress of the backup.
7. To stop the backup, click **Abort**.

The backup stops immediately and logs a status to indicate the backup was aborted.

8. To verify whether the backup is complete or aborted, view the backup details in the Backup History window. If the backup was not successful, an error message is displayed.

Export PI Server backup history

You can use the Backups tool to export a backup history report to an XML file.

Procedure

1. Click the **Export Backup Reports** button to open the Save Backup History As window.
2. Browse to the save location.
3. Enter the file name.
4. Click **Save**.

PI Server backup scripts

The PI Server backup scripts are located in the `PI\adm` directory:

- **pibackup.bat**

  Use this script to launch a backup or to install a backup as a scheduled task.

- **pibackuptask.bat**

  This script calls `pibackup.bat` and redirects the standard output to a log file.

- **pisitebackup.bat**

  This custom backup scriptdoes not exist by default.

- **pintbackup.bat**

  This custom backup script does not exist by default and typically should not be implemented unless you have upgraded your PI Server.

- **pibackup_3.4.370.bat**

  The PI Server installation creates this script only when upgrading from 3.4.370 to 3.4.375 or greater.

With the exception of installing a backup task with `pibackup.bat`, you should not need to run any of these backup scripts directly.

Note:
If the PI AF server is installed on the same computer as the PI Server, then an additional script, called `afbackup.bat`, is installed in the `pipc` directory (`pipc\AF\sql`).

Topics in this section

- pibackuptask.bat
- pibackup.bat
- pisitebackup.bat
- pintbackup_3.4.370.bat

- afbackup.bat

pibackuptask.bat

The scheduled backup task calls the `pibackuptask.bat` script to launch the backup. The script calls `pibackup.bat` and redirects the standard output to a backup log. The backup log is written to the target directory of the backup and the log file has a name of the form:

```
pibackup_dd-mmm-yy_hh.mm.ss.txt
```

For example:

```
pibackup_5-Aug-05_14.16.22.txt
```

pibackup.bat

This script is used to install a backup as described in Create the scheduled backup task. The `pibackup.bat` script performs the actual backup of the PI Server and calls the `pisitebackup.bat` script after backing up the PI Server. When the backup task runs, the `pibackuptask.bat` script is called directly, which itself then calls `pibackup.bat`.

The `pibackup.bat` script starts the backup of the PI Server by running a single **`piartool -backup`** command.

After you run `pibackup.bat` to set up the backup service, you should not need to run it directly again. If you want to launch your regularly scheduled backup prior to its scheduled time, you should open the Scheduled Tasks Windows control panel and run the PI Server Backup scheduled task from there. If you want to run a manual backup (one that does not change the last backup time for your scheduled backups) use the Backups tool in PI SMT. See *Introduction to PI Server System Management Guide* for more information.

Note:

Do not directly customize the `pibackup.bat` script. This script is overwritten on PI Server upgrades.

pisitebackup.bat

After the backup of the PI Server has completed, `pibackup.bat` calls `pisitebackup.bat`, provided that `pisitebackup.bat` exists.

The `pisitebackup.bat` script can be used to:

- Back up site-specific files.
- Copy files from the backup directory to a safe location. This should be done only if a third-party backup program is not available.
- Trigger a backup of the backup directory with a third-party backup program.

The `pisitebackup.bat` does not exist by default. However, the PI Server installs the `pisitebackup.bat.example` file to the `PI\adm\` folder. By simply removing the `.example` extension, the script backs up all files ending in `.bat`, `.log`, `.ini`, `.txt`, and `.sql` under the 32-bit and 64-bit `PIPC` home directories. If you want to backup any other files or do any other task, you must customize the script. Customization instructions are in the `pisitebackup.bat.example` file itself.

Note:

The `pisitebackup.bat` script is not overwritten when the PI Server is upgraded.

pintbackup_3.4.370.bat

The `pibackup_3.4.370.bat` script is created by the PI Server installation only when upgrading from 3.4.370 to 3.4.375 or greater. The purpose of the `pibackup_3.4.370.bat` script is to maintain the behavior of the backups from 3.4.370 so that a user's site-specific backup is not broken by the upgrade.

afbackup.bat

PI AF Server 2010 and later includes an additional backup script, called `afbackup.bat`. The `pibackup.bat` script calls `afbackup.bat`, if it exists. Since the `afbackup.bat` is part of the AF Server installation, it will exist on the PI Server only if the AF Server is installed on the same computer as the PI Server. The `afbackup.bat` script is installed in the `pipc` directory under `pipc\AF\sql`.

By default, the `afbackup.bat` script backs up an instance of SQL Server called `./sqlexpress`. If the PI AF database is called `sqlexpress` and it is located on the PI Server computer, then the scheduled backup task will back up that database. The PI AF database backup is written to:

```
pibackupdir\AF\
```

where *pibackupdir* is the backup directory passed to the `pibackup.bat` script.

You can edit the `afbackup.bat` file to back up a different SQL Server instance, such as `sqlserver`. The `afbackup.bat` script is a site-specific file. If you upgrade the PI Server, this file is not overwritten.

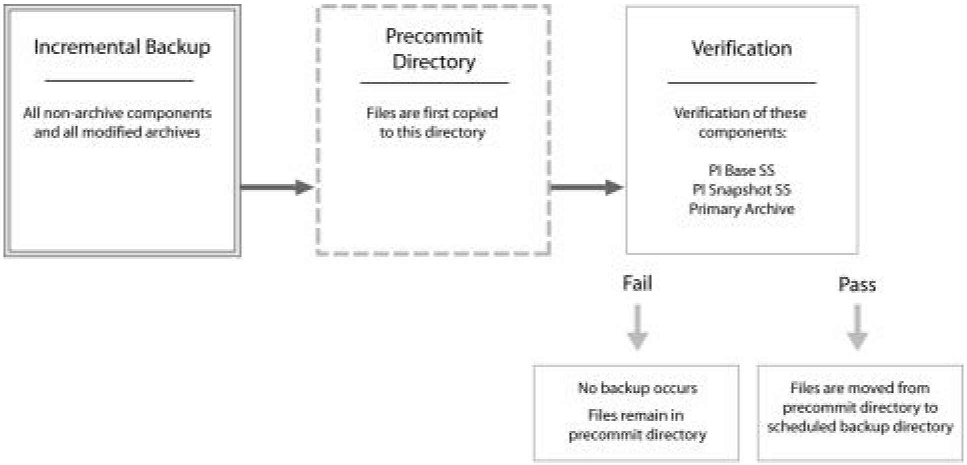
How PI Server backup verification works

By default, the PI Server backup task performs an incremental backup of PI server files.

The PI Backup subsystem attempts to maintain a consistent backup without corrupt or partially copied files in the PI Server backup directory. It does this by temporarily copying files to a `precommit` directory before moving the files to their final destination. This `precommit` directory is a subdirectory under the PI Server backup directory. If the backup is aborted, if the PI Backup subsystem crashes, or if the files in the `precommit` directory do not pass verification, then the files in the `precommit` directory are not moved to their final destination. Therefore, the last good backup should always be available.

The backup process works like this:

1. The files corresponding to the components selected for backup are copied to a `precommit` directory.
2. The primary archive and the files that correspond to the base and snapshot subsystems are verified in the `precommit` directory.
3. If verification passes, the files are moved to the backup directory. Any files with the same name that already exist in the backup directory are renamed before the move operation.
4. If all move operations are successful, the renamed files are deleted.


Incremental Backup
All non-archive components and all modified archives
Precommit Directory
Files are first copied to this directory
Verification
Verification of these components:
PI Base SS
PI Snapshot SS
Primary Archive
Fail
Pass
No backup occurs
Files remain in precommit directory
Files are moved from precommit directory to scheduled backup directory

If the backup fails verification, the files are left in the `precommit` directory, and the reason for the failed backup is written to a `pibackupverify_*.log` file in the PI Server backup directory. If successive backups fail verification, files will start accumulating in the precommit directory. After the next successful backup, all files are copied to their final destination.

The following table shows commands that the PI Backup subsystem spawns to perform the verification.

Component	Verification Command
Archive components	**`pidiag -archk`**
pibasess	**`pibasess -verifydb`**
pibasess	**`pidiag -fb`**

Although only the primary archive is verified by default, the number of archives to be verified can be set with the **`BackupVerification_NumArch`** tuning parameter. Alternatively, all verification can be suppressed by setting the **`BackupVerification`** tuning parameter to 0.

Although the last good backup should not be corrupt, it is still imperative to backup the PI server backup directory, preferably with third-party backup software. For example, if you accidentally delete a PI point and subsequently perform a backup, the PI point is deleted in the last good backup as well. To retrieve the deleted PI point, you might need to restore a previous backup. If you are not keeping multiple backups of your PI Server backup directory, there will be no means to do this restore.

Restore PI Server from a backup

Topics in this section

- Restore a backup to an existing PI Server
- Restore a PI Server backup to a new computer

Restore a backup to an existing PI Server

This section explains how to restore your PI Server from a backup. Follow these instructions to restore the PI Server to the same computer that it was running on.

Procedure

1. Isolate your PI Server from the network.
2. Stop PI Server.

   ```
   \pi\adm\pisrvstop.bat
   ```

3. Delete the following file:

   ```
   \pi\dat\PIModuleUnitDb.dat
   ```

   This file is automatically regenerated when you restore from backup.

4. Restore the backup to a temporary directory, such as `C:\TempRestoreDir`.

   For example, if you back up your backup directory with a third-party backup application, restore the desired backup to `C:\TempRestoreDir`. Alternatively, if you are restoring the latest backup, you can restore the PI Server directly from the latest backup directory. This procedure assumes that you have restored the desired backup to a folder of the name `C:\TempRestoreDir`.

5. Copy the files from `C:\TempRestoreDir\dat` to `PI\dat`.
6. Copy the files from `C:\TempRestoreDir\adm` to `PI\adm`.
7. Copy the files from `C:\TempRestoreDir\bin` to `PI\bin`.
8. Copy the files from `C:\TempRestoreDir\log` to `PI\log`.
9. Copy the archive files from the `C:\TempRestoreDir\arc` directory in your backup folder to their original location on PI Server.

   If you are not sure where your archive files were located on PI Server, look in the backup log file in `C:\TempRestoreDir\`. The log contains the archive list at the time of the backup.

   Because you are restoring to an existing server, you do not have to restore all archives. At a minimum you must restore the primary archive. Restore other archives as needed.

10. If a site backup was performed (if, for example, `C:\TempRestoreDir\sitebackup` exists), then copy the files from the site backup directories to the corresponding 32-bit and 64-bit `pipc` directories.
11. Restart PI Server.
12. Restore PI Server's connection to the network.
13. Use the MDB to AF Synchronization tool in PI SMT to check the status of the synchronization between MDB and AF.
14. If MDB and AF are out of sync, then use the MDB to AF Synchronization tool to reset MDB.

Restore a PI Server backup to a new computer

This procedure guides you through restoration of a complete PI Server from backup and the original installation disk. This is suitable for cases of disk crashes or disasters of similar magnitude.

Before you start

This procedure assumes that you are restoring the PI Server on a computer where the PI Server was never previously installed. You need to know the following information about the PI Server installation that you are restoring:

- The version of the PI Server you are restoring (for example, PI Server 2012). If you are restoring a version of the PI Server earlier than 3.4.370, then please consult earlier versions of the PI Server documentation or contact technical support.
- The full directory path, including drive letter, for the PI Server installation that you are restoring (for example, `C:\PI`).

Procedure

1. Change the computer name of the new computer to the name of the old PI Server computer. Restart the computer.
2. Restore the backup to a temporary directory, such as `C:\TempRestoreDir`.

   For example, if you have been backing up your backup directory with a third-party backup application, restore the desired backup to `C:\TempRestoreDir`. Alternatively, if you are restoring the latest backup, you can restore the PI Server directly from the latest backup directory. This procedure assumes that you have restored a previous backup to a folder of the name `C:\TempRestoreDir`.
3. Copy the installation kit to the new computer.

   The installation kit should be for the same version of the PI Server as the version you are restoring. Do not try to upgrade the PI Server as part of a restore.

   > **Note:**
   >
   > If you are restoring an old backup, use the PI Server version that was installed at the time that the backup was taken. The PI Server version can typically be found in the backup log, which should have been restored to `C:\TempRestoreDir`.
4. Disconnect the computer from the network. Disconnecting from the network is important so that data is not lost from buffered interfaces in subsequent steps.
5. Run the PI Server installation kit. Install the to the same drive letter and directory path as on the old PI Server node.
6. Verify that the PI Server is disconnected from the network before proceeding to the next step.
7. Start the PI Server.
8. Confirm that the PI Server started up correctly and then stop the PI Server.

   This accomplishes the "run once" functions performed after an installation. Since the PI Server is disconnected from the network at this point, data will not be lost from buffered server nodes.
9. Restore (using Windows Explorer or the copy command) all files from the `C:\TempRestoreDir\dat\` directory to the new `PI\dat\` directory.
10. Restore all the message log files (`pimsg_xxxxxxx.dat`) from the `C:\TempRestoreDir\log\` to the `PI\log` directory.

11. Restore all files from the `C:\TempRestoreDir\adm\` directory to the new `PI\adm\` directory.

12. Restore all files from the `C:\TempRestoreDir\bin\` directory to the new `PI\bin\` directory.

13. Restore the archives from the `C:\TempRestoreDir\arc\` directory to the same directory that they were installed on the old PI Server computer.

    You can determine the directories from the archive list in the restored backup log. If you restore the archives to a different directory, then you will need to do the following additional step.

    a. Use PI SMT to register the primary archive in the new location (**Operation** > **Archives**). If you are uncertain which of the backed up archives is the primary archive then examine the archive dates. The primary should have the latest start date and an end date of Current time.

14. Start the PI Server.

15. Register additional archives in PI SMT as needed.

16. Use **`piartool -al`** and the client tools (PI ProcessBook and PI DataLink) to verify that all the data has been recovered. If the data is intact, then the restoration is complete. Run the clients locally, since the PI Server should be isolated from the network. It is very important to confirm correct PI Server recovery before exposing the restored server to buffered data. Failing to do so may cause data loss.

17. Connect the PI Server to the network. Verify the PI Server is reachable from clients on the network. Monitor all buffered interface nodes.

18. Use the MDB to AF Synchronization tool in PI SMT to check the status of the synchronization between MDB and AF.

19. If MDB and AF are out of sync, then use the MDB to AF Synchronization tool to reset MDB.

Manage PI points

Note:
The *Introduction to PI Server System Management* guide provides basic information about working with PI points. This document does not repeat the information provided there, but discusses additional topics in detail.

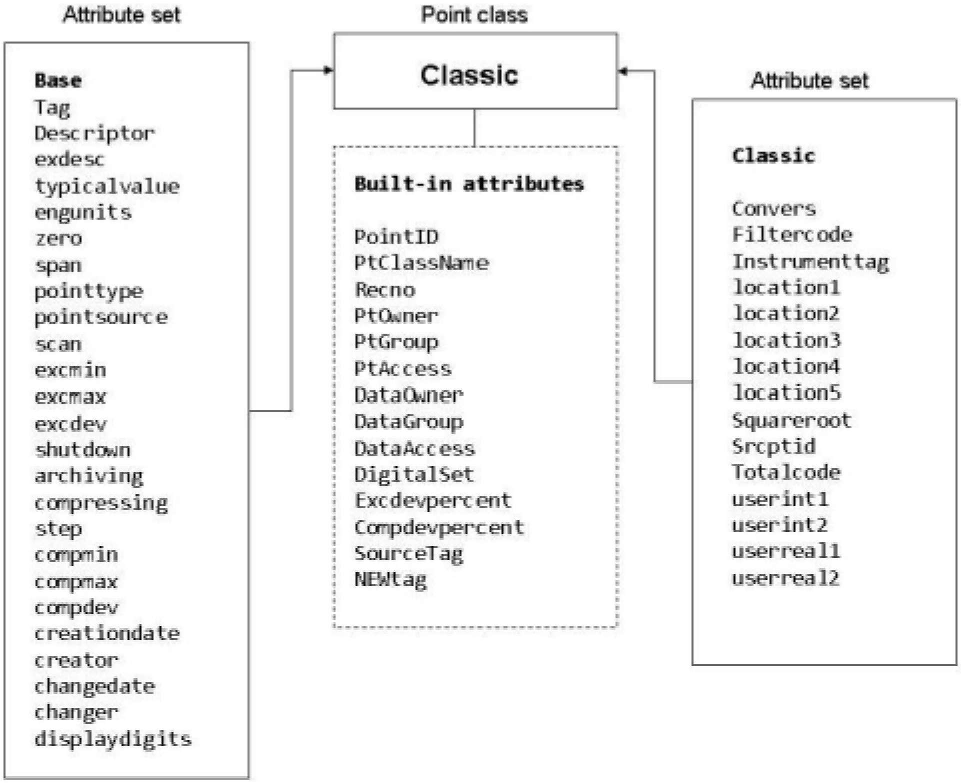
Topics in this section

- PI point classes and attributes
- Exception reporting and compression testing
- Change PI point type
- Modification of PI point classes and attribute sets
- Digital state sets

PI point classes and attributes

A point class represents the schema or template of a point. It determines what attributes you can define for a point of that type. Essentially a point class is just a group of point attribute sets. Each attribute set consists of a group of individual attributes. Point class is assigned when the point is created. The default point class is Base point class.

No two attribute sets within a point class can contain the same attribute. The Point Database has several different point classes, such as Base and Classic. The structure of the Classic point class is depicted in the following figure.



You can create point classes and attribute sets. You can also edit and delete both attribute sets and point classes (PI Server version 3.4.370 and later) although this poses risks and it is rare that you should need to do so.

Topics in this section

- Predefined point classes
- Predefined attribute sets
- Base class point attributes
- Classic point class attributes
- COM Connector point attributes
- Default values for point attributes
- System-assigned attributes

Predefined point classes

The following table lists predefined point classes.

Point class	Attribute sets that make up the point class
Alarm	base alarmparam
Base	base

Point class	Attribute sets that make up the point class
Classic	base classic
SQC_Alarm	base sqcalm_parameters
Totalizer	base totals

Predefined attribute sets

alarmparam

Attribute	Type	Default
action1	String	
action2	String	
action3	String	
action4	String	
action5	String	
AutoAck	String	yes
ControlAlg	String	
ControlTag	String	
Deadband	Float32	0
Options	String	
ReferenceTag	String	
Srcptid	Int32	0
test1	String	
test2	String	
test3	String	
test4	String	
test5	String	
txt1	String	
txt2	String	
txt3	String	
txt4	String	
txt5	String	

base

Attribute	Type	Default
Archiving	BYTE	1
Changedate	TimeStamp	31-Dec-69 16:00:00
Changer	Uint32 for 3.4.380 and later (Uint16 for earlier versions)	0
Compdev	Float32	2.
Compmax	Uint32	28800
Compmin	Uint16	0
Compressing	BYTE	1
Creationdate	TimeStamp	31-Dec-69 16:00:00
Creator	Uint32 for 3.4.380 and later (Uint16 for earlier versions)	0
Descriptor	String	
DisplayDigits	BYTE	-5
EngUnits	String	
Excdev	Float32	1.
ExcMax	Uint32	600
ExcMin	Uint16	0
ExDesc	String	
PointSource	String	Lab
PointType	UBYTE	12
Scan	BYTE	1
Shutdown	BYTE	1
Span	Float32	100.
Step	BYTE	0
TypicalValue	Float32	50.
Zero	Float32	0.

classic

Attribute	Type	Default
Convers	Float32	1.
Filtercode	Int16	0
InstrumentTag	String	
location1	Int32	0
location2	Int32	0
location3	Int32	0
location4	Int32	0
location5	Int32	0

Attribute	Type	Default
Squareroot	Int16	0
Srcptid	Int32	0
Totalcode	Int16	0
userint1	Int32	0
userint2	Int32	0
userreal1	Float32	0.
userreal2	Float32	0.

sqcalm_parameters

Attribute	Type	Default
AutoAck	String	yes
ChartType	Int32	0
ClearOnLimitChange	String	true
ClearOnStart	String	false
CLTag	String	
CommentTag	String	
LCLTag	String	
LSLTag	String	
Mixture	String	
OneSideofCL	String	
Options	String	
OutsideControl	String	
OutsideOneSigma	String	
OutsideTwoSigma	String	
PIProductLimits	String	no
ProductTag	String	
ReferenceTag	String	
ResetTag	String	
SQCAlarmPriority	Int32	0
Srcptid	Int32	0
Stratification	String	
TestStatusTag	String	
Trend	String	
UCLTag	String	
USLTag	String	
WaitOnLimitChange	String	false

totals

Attribute	Type	Default
CalcMode	String	timeweighted
CompValue	String	ON
Conversion	Float32	1
EventExpr	String	
FilterExpr	String	
Function	String	Total
MovingCount	Int16	2
Offset	String	+0m
Offset2	String	+0m
Options	String	
PctGood	Float32	85
Period	String	+1h
Period2	String	+2m
RateSampleMode	String	natural
ReportMode	String	Running
Srcptid	Int32	0
TotalCloseMode	String	clock
Zerobias	Float32	0

Base class point attributes

The Base class is a common set of attributes that all other point classes include. Some of these attributes can be changed only by the system. These attributes are described in System-assigned attributes.

Archiving

The Archiving flag must be set to ON (1) for a point to be archived. This flag can be set to OFF (0) to stop archiving of a point.

Compression flag

Set compression to ON (1) for most points. Set compression OFF for laboratory and manually entered tags so every value is recorded in the archive. The number of events for these tags is usually small. With compression off, every value sent to the snapshot is saved in the archive.

Compression affects digital points, since a new value is recorded only when the current value changes. Points of types Blob and string have a similar behavior; new events pass compression only when the value changes. String values are compared ignoring case. (VaLuE and valUe are equal.) For Blob events, any change is significant.

CompDev, CompMin, CompMax, and CompDevPercent

When a new snapshot arrives, the previous one is evaluated according to the compression specifications to see if it is a significant event. If so, it is sent to the event queue. If not, it is discarded. The result is that only significant data is written to the archive. This process is called compression.

The compression specifications consist of a deviation (CompDev), a minimum time (CompMin), and a maximum time (CompMax):

- CompMin: An event that comes before CompMin time elapsed is discarded. For points associated with interfaces that send exception reports, the CompMin should be 0.
- CompMax: Events are archived if discarding them would cause a gap greater than CompMax. The recommended maximum time specification is one work shift (that is, 8 hours). Duplicate values are archived if the elapsed time exceeds CompMax. Under no circumstances does this cause PI to generate events; it only filters events that are externally generated.
- CompDev: The most important compression specification is the deviation, CompDev. Setting this value too low causes too little data compression and wastes space in the archive. Setting this value too high causes loss of useful data.
- CompDevPercent: This is similar to CompDev, but it specifies the compression deviation in percent of Span rather than in engineering units. If one is changed by user, the other is automatically changed to be compatible. If both are changed, CompDevPercent overrides CompDev.

For non-numeric tags, CompDev and CompDevPercent are set to zero and ignored.

See Exception reporting and compression testing.

Descriptor

The Descriptor is a text field that appears on various client application displays and can be used in reports. It can be of any length up to 65,535 characters. When this value is read through the PI API it is truncated to 26 characters.

Some interfaces use the descriptor for tag configuration on external system. Having quotes or wild card characters might lead to confusion when these attributes are passed to other applications.

DigitalSet

The DigitalSet attribute specifies the name of the digital state set associated with the tag (see Digital state sets). This attribute applies to digital type tags only. It is ignored for all other types of tags.

There is a special digital state set called the System digital state set (see Default set System state set). All tags, regardless of type, are associated with the System digital state set.

DisplayDigits

The DisplayDigits attribute controls the format of numeric values on screens and in reports. Zero or a positive number indicates the number of digits to display to the right of the decimal point. A negative number indicates the number of significant digits to display. In this case, the absolute value of DisplayDigits is the number of significant digits.

The following table shows how a value of 23.45 would appear on the screen for different values of DisplayDigits:

DisplayDigits	Format
3	23.450
2	23.45
1	23.5
0	23
-1	2E+001
-2	23
-4	23.45

EngUnits

The Engineering Unit string describes the units of the measurement. You may use any string, and the string may be of any length. However, the PI API retrieves only the first 12 characters. The PI SDK does not truncate the string.

Examples include:

```
DEGF
Degrees Centigrade
Gal/Min
Gallons Per Minute
' " Hg
```

Engineering Unit strings are case preserving, but not case sensitive on search. The system trims leading and trailing blanks during input.

ExDesc

The *extended descriptor* is a text field of any length (although it is truncated to 80 characters when reported through PI API). It is typically used to provide additional information for documentation. Several interfaces use the ExDesc attribute to encode additional configuration information.

ExcDev, ExcMin, ExcMax, and ExcDevPercent

Most interface programs use exception-reporting specifications to determine when to send data to the snapshot. The exception reporting specifications consist of a deviation (ExcDev), a minimum time (ExcMin), and a maximum time (ExcMax). ExcDevPercent is similar to CompDev, but it specifies the exception deviation in percent of Span rather than in engineering units. If you change one, the other is automatically changed to be compatible. If both ExcDev and ExcDevPercent are changed, ExcDevPercent overrides ExcDev.

For digital, string and Blob tags, ExcDev and ExcDevPercent are set to zero and ignored. See Exception reporting and compression testing.

NewTag

The NewTag attribute is used for renaming tags.

Point Security

You can independently configure security for point attributes and point data. Use the Point Security and Data Security attributes respectively (older versions of some client tools refer to these as the Point Access and Data Access attributes).

Setting the values of security attributes is different for PI Server 3.4.380 and later than it is for earlier versions of the PI Server. On PI Server versions 3.4.380 and later, you can set point security access permissions for any PI Identities, PI Users, and PI Groups. Earlier versions of the PI Server use the owner/group model.

In the owner/group model, you define an owner and a group for each security attribute. The owner must be a PI User and the group must be a PI Group. You then set access permissions for owner and for group, as well as world access.

See the *Configuring PI Server Security* guide for complete details on security.

PointSource

The PointSource attribute is a string that associates a tag with an interface or PI application. An interface uses the point source to retrieve all its points.

The default point source is Lab. Use this for points that are not associated with any interface to specify lab-input points.

Avoid using the % (percent) character as a point source, as it has also special meaning for SQL and other applications. Similarly, avoid using ? (question mark), * (asterisk), and _ (underscore) as point source characters.

PointType

There are many point types in the PI Server. The PointType value is assigned when the point is created. Prior to PI Server version 3.4.370, this attribute could not be changed, but in versions 3.4.370 or later it can be edited. For details, see Modification of PI point classes and attribute sets.

Point Type	Used for
Digital	Points whose value can only be one of several discrete states, such as ON/OFF or Red/Green/Yellow. Nearest equivalent to the PI Server 2.x Digital type.

Point Type	Used for
Int16	Points whose values are 15-bit unsigned integers (0 to 32767). Nearest equivalent to the PI 2.x Integer type.
Int32	Points whose values are 32-bit signed integers (-D2147450880 to 2147483647). PI reserves the lowest 32K values of the 32bit range for digital states.
Float16	Floating point values, scaled. The accuracy is one part in 32767. Nearest equivalent to the PI 2.x Real type.
Float32	Single-precision floating-point values, not scaled.
Float64	Double-precision floating-point values, not scaled.
String	Storing string data of up to 972 characters if annotated, 976 otherwise.
Blob	Storing any type of binary data up to 972 bytes if annotated, 976 otherwise.
Timestamp	Storing values of type Timestamp. Any Time/Date in the Range 1-jan-1970 to 1-Jan-2038.

Topics in this section

- Point types
- Attributes that depend on point type

Point types

For points collected by interfaces, use the point type that most closely matches the point type in the source system. For example, if the point originates from a transmitter that supplies an analog measurement with 14 bits of accuracy, use the float16 point type. This point type provides 15 bits of precision.

For accumulators and other high precision values, use the higher precision point types: either Float32 or Float64.

The higher precision point types require more disk space for each stored value. Float16 points use 16 bits or 2 bytes per value. Float32 and float64 use 4 and 8 bytes per value, respectively. Int16 and int32 values use 2 and 4 bytes, respectively. Int16 is similar to a PI 2 integer type; it supports only 15-bit unsigned integers.

If you want to store negative integers, select the int32 point type. Note that PI reserves some values in the negative range of a 32-bit integer. The smallest value that can be stored is shown in PointType.

Interface manuals sometimes refer to point types `R` (real), `I` (integer), and `D` (digital).

- Use float16 or float32 for type `R`. If the data is coming from an analog-to-digital converter (ADC), float16 is sufficient
- Use int16 or int32 for type `I` or integer values
- Use digital for type `D` or discrete values

OSIsoft recommends that you use float32 as the default type for floating point data. The space-saving of float16 can reduce the amount of I/O, but this is significant only on very large data retrievals, such as yearly average calculations or long term trends. Also, float16 data is scaled; it can cause incorrect results in some applications such as SQC, if your zero and span settings are not set correctly.

Attributes that depend on point type

Some point attributes are not relevant for some point types:

- Only Digital type tags use the DigitalSet attribute. It is irrelevant for other type tags and can be ignored.
- For Digital, string and Blob type tags, the values of CompDev, CompDevPercent, ExcDev and ExcDevPercent are not applicable. The value of these attributes is automatically returned to applications as 0.

For Digital, string and Blob type tags, the Span and Zero attributes are not applicable. For digital tags, Zero is automatically set to the digital set number. For PI Server version 3.4.380 and later, span is no longer relevant for digital points. On earlier versions of the PI Server, Span is automatically set to the number of states minus 1 in the set.

Finally, for all non-numeric types the step flag is set to TRUE.

PtClassName

PtClassName specifies the point class. The point class must be defined before the point is created. Prior to PI Server 3.4.370, the point class could not be changed once the point was created. In versions 3.4.370 or later, it can be edited (see Modification of PI point classes and attribute sets).

Scan flag

Some interface programs use a **Scan** flag. Interfaces that honor this attribute do not update points whose scan flag is set to **OFF**. See the interface documentation to find out if your interface program uses it.

Shutdown flag

In some cases it is useful to record to PI points, when the archive was shut down. That way there is a clear indication of a gap in the data collection. Points may be configured so that PI Server automatically records a time-stamped event to indicate when a shutdown occurs. These are called *shutdown events*.

The shutdown flag for a point is set to ON (1) to indicate that shutdown events should be recorded for this tag. The default is ON.

For points collected from interfaces on distributed collection nodes, set this flag to OFF (0) because data buffering in PINet or PI API retains the data until the home node is running again. Therefore, there are no data gaps to identify with shutdown events.

Many sites configure points that store laboratory test values so that the lab test points do not receive shutdown events. Lab values are assumed to be constant between tests. Inserting shutdown events for these points can be misleading.

The shutdown flag is used in conjunction with the configuration file `dat\shutdown.dat`.

SourceTag

For data output to other systems, the SourceTag is the PI tag of the data source. For example, you can define a tag ABC to receive data using point source A, and then define another tag DEF to send this information to another instrument system using point source B. The source tag for tag DEF would be ABC. This attribute is used by some interfaces, while other interfaces use the extended descriptor (ExDesc) for this information.

The interface performs the reading and writing of data when this attribute is defined.

The SourceTag attribute is not stored in the Point Database. It is only a more readable representation of the srcptid attribute that holds the source point ID. srcptid does not exist in all point classes. For example, it is part of the classic point class but not of base.

Span

The Span is the difference between the top of the range and the bottom of the range. It is required for all numeric data type points.

For float16 point types, the Span is used with the Zero for scaling values in the archive. The Span must be a positive value. If the value for a point type float16 point is greater than the top of range, it is recorded in the archive as Over Range. For other point types, Zero and Span do not affect the values recorded in the archive.

The Span is also used when defining a PI ProcessBook trend with a vertical scale of database.

This attribute is not used for non-numeric points.

The Span for a tag can be changed without affecting data already in the archive. For points of type float16, the old Span is used for retrieving the archive data collected before the edit. The new Span is used for data collected after the edit. When Span is changed, the exception and compression deviation percents are preserved. This means that the ExcDev and CompDev fields, which are expressed in engineering units, are modified internally. If any of the deviation fields is specified in the editing operation they take precedence.

Note:

Some interfaces might use Span information to filter incoming data. These interfaces often convert out- of-range data to digital states over range and under range. However, interfaces might use Span configuration in other ways. The PI Server itself does not change out of range data except for tags of type float16.

Step

The Step flag affects only numeric points. It defines how numeric archived values are interpolated. The default behavior, step OFF (0), treats archived values as a continuous signal. Adjacent archived values are linearly interpolated. For example, at 12:00:00, the value 101.0 is archived and at 12:01:00, the value 102.0 is archived. A request for the archive value at 12:00:30 would return 101.5.

A step flag of ON (1) treats the archived values discretely. Adjacent archived values are not interpolated; an archived value is assumed constant until the next archived value.

For example:

- At 12:00:00, the value 101.0 is archived
- At 12:01:00, the value 102.0 is archived
- A request for the value at 12:00:30 would return 101.0

> **Note:**
>
> For points with non-numeric type (digital, string, and timestamp) the Step attribute is always ON (1). You cannot turn it off.

In general, data coming from continuous signals should be archived in points with the step flag OFF. Examples might include signals from thermocouples and flow meters. Data coming from discrete measurements should be archived in points with the step flag on. Examples are sampled lab data, batch charge weight.

The step attribute setting affects both display and compression.

Data for points with this attribute set to 1 is assumed to remain fixed between events, whereas for points with step=0 data is assumed to change linearly between valid numeric events.

The swinging-door compression, explained above, is not used when the step flag is set. Instead, an exception calculation is applied using the CompDev value. If the absolute difference between the current snapshot and the last archive value is greater than CompDev then the snapshot is sent to the archive.

Compression maximum and minimum limits work the same as for tags with the step flag not set.

Tag

The Tag attribute is the name of the point. Each Tag must be unique to a PI System. Since the tag is the name that identifies the point to users, use a consistent tag-naming convention that is meaningful to people in your organization. For example, you could reserve the first two characters of a tag to indicate a unit name or an area of the plant. You could reserve another six characters to match the standard instrument tag, and so on.

Tags may be any length and can include letters, numbers, and spaces. Tags are subject to the following constraints:

- The first character must be alphanumeric, an underscore (_), or a percent sign (%).
- Control characters, such as linefeeds or tabs, are not allowed.
- The following characters are not allowed:
  ```
  * ' ? ; { } [ ] | \ ` ' "
  ```

These characters are allowed, however, in other tag attributes, such as the descriptor.

Any tags that follow the above rules are, technically, allowed. However, be aware that other applications use some legal characters in special ways. For example, SQL uses the underscore

(_) and percent sign (%) as wild cards. Therefore, tags that contain these characters may cause problems with these applications. Similarly, some functions and components restrict the length of tags:

- PI API functions pipt_tag and pipt_updates truncate the tag to 12 characters. Functions pipt_findpoint, pipt_wildcardsearch, pipt_taglong, and pipt_tagpreferred report only the first 80 characters.
- PI SQL Subsystem can only process tags with at most 1016 characters. Joins that involve longer tags will return no row found. Queries without joins return rows but truncate tags to 1016 characters.

Topics in this section

- Case sensitivity
- Changing tag names

Case sensitivity

The system preserves the case of all strings, including the tag, but searches are not case-sensitive. For example, a string entered as BatchStart is stored exactly as entered. Subsequent retrievals of this string retain the same capitalization. A search for this string does not require that the capitalization match.

Changing tag names

To change the tag attribute, use the **piconfig** utility and the NewTag attribute in the PIPoint table. Keep in mind that when you change a tag, certain programs that retrieve data using that tag, such as PI DataLink spreadsheets, might also have to be updated. PI ProcessBook displays automatically use the new tag name.

TypicalValue

The Typical Value is used only to document an example of a reasonable value for this point. For a numeric tag, it must be greater than or equal to the Zero point attribute, and less than or equal to the Zero plus the Span point attributes. Some interfaces use this as an initial or default value.

Zero

A Zero attribute is required for all numeric data type points to indicate the lowest value possible. It does not have to be the same as the instrument Zero, but that is usually a logical choice. Certain interfaces require that the Zero and Span match the instrument system range; see the interface documentation for details.

The Zero is the bottom of the range used for scaling float16 values in the PI archive. If the value for a float16 type point is less than the bottom of range, the value is recorded in the archive as the Under range state when the archive cache is flushed to disk. The Zero is also used when defining a PI ProcessBook trend with a vertical scale of database.

This attribute is not used for non-numeric points.

A tag's Zero attribute can be changed without affecting data already in the archive. For points of type float16, the old Zero is used for retrieving the archive data collected before the edit. The new Zero is used for data collected after the edit.

Note:

Some interfaces might use Zero information to filter incoming data. These interfaces often convert out- of-range data to digital states over range and under range, however, interfaces might use Zero configuration in other ways. The PI Server itself does not change out of range data except for tags of type float16.

Classic point class attributes

Many OSIsoft interfaces rely on classic attributes. Use the Classic point class for all PI Interface points if the interface uses the InstrumentTag or location code attributes.

Filtercode

The Filtercode indicates the time constant of a first-order filter used to smooth incoming data. While it does impact the compressed data, it does not affect exception reporting.

We recommend not altering incoming data by leaving this code at its default value of 0. The other options are:

Code	Time Constant (Seconds)
1	10
2	60
3	120
4	600

Instrument tag

When a value is retrieved from or sent to an external system such as a DCS, the instrument tag is used by some interfaces as the tag in the external system. The InstrumentTag field can be any length. However, most interfaces only use the first 32 characters of this attribute. Some interfaces use the extended descriptor (ExDesc) instead.

Location1, Location2, Location3, Location4, and Location5

There are five integer location codes. Their meanings depend on the interface. For many PI Interfaces, you use the Location1 attribute to specify the interface ID number and the Location4 attribute to assign scan class. For instrument interfaces, the location codes often describe a hardware or software address for reading or writing the value. See the interface documentation for details on how to set these point attributes.

Ranges of ExcMax and CompMax

In early releases of PI Server 3, the values of these two point attributes were stored as unsigned 16-bit integers, which meant that the maximum value of each was 65,535 seconds. This continues to be true for existing systems upgraded from PI Server 3.3 or earlier, but since PI Server 3.4.370.x or later, these attributes can be edited to uint32 type. See Attribute set database edit.

In new installations of PI Server 3.3 or greater releases, the values of these two point attributes are stored as unsigned 32-bit integers, and the maximum value of each is 4,294,967,295 seconds.

The PI API protocol defines the ExcMax and CompMax attributes as a signed 16-bit integer. If the PI Server stores a value that is larger than 32,767, the value returned by the PI API is 32,767.

PI SDK applications obtain from the PI Server a signed 32bit integer values for ExcMax and CompMax.

SquareRoot

Some interface programs use the square root code. Check the manual for your interface.

Srcptid

Srcptid is the PI point number corresponding to the tag specified in the SourceTag attribute. If this attribute is edited, PI Server changes SourceTag to the corresponding tag. Do not directly alter the Srcptid attribute; change SourceTag instead.

UserInt1, UserInt2, UserReal1, and UserReal2

PI reserves these four attributes for user applications. Most PI applications do not use these attributes. UserInt1 and UserInt2 are 32-bit integers. UserReal1 and UserReal2 are 32-bit floating-point numbers.

COM Connector point attributes

COM Connectors allow the PI Server to obtain data from foreign data systems. To do this, you must create a special PI Server point whose attributes identify the location of the data in the foreign system.

The term *map* is used throughout this manual to mean making a relationship between a point on the foreign system and a point on the PI Server. During PI Server operation, clients make requests for data by using PI Server point information. The PI Server then obtains data from the foreign system point to which the PI Server point is mapped.

For mapped points you must define a point class that includes the following attributes:

Attribute	Description
ctr_progid	COM Program ID, as stored in the Windows registry. This name is used to instantiate the COM Connector object.
ctr_lmap	Longword mapping parameter.
ctr_strmap	String mapping parameter.

A point is identified as a PI Server mapped point if it includes these three attributes.

The ctr_progid is used by the PI Server to load the COM Connector. The mapping parameters ctr_lmap and ctr_strmap are passed to the COM Connector through a COM method call so that it can locate the appropriate foreign system point. The usage of these two attributes is always specified in the manual for any COM Connector.

The PI Server has a script file called `classicctr.dif` that can be processed by the **`piconfig`** utility to define a point class called classicctr. This point class has all the attributes of the classic point class plus the three attributes that define mapped points. The `classicctr.dif` file can be used directly, or as a template for custom point class definitions.

Default values for point attributes

When you create a point you must, at a minimum, name the point with the tag attribute. If you do not assign values to all other attributes, the PI Server uses the default value.

Note:

Programmatic access to some of the attributes may be limited or unavailable from the PI API.

Point Class	Field Name	Default Value	Limits
Base	Archiving	ON	ON, OFF, 1, or 0
Base	ChangeDate		System-assigned
Base	Changer		System-assigned
Base	CompDev	2 eng units	
Built-in	CompDevPercent	Comes from CompDev	0 ≤ x ≤ 100
Base	CompMax	28800 sec	
Base	CompMin	0 sec	0 ≤ x ≤ 65535
Base	Compressing	On	ON, OFF, 1, or 0
Classic	Convers	1	
Base	Creationdate		System-assigned
Base	Creator		System-assigned
Built-in	DataAccess	O:rw g:r w:r	Owner, group, and world may be assigned read, write, or no access (blank)
Built-in	DataGroup	piadmins	In PI Identity Database
Built-in	DataOwner	piadmins	In PI Identity Database

Point Class	Field Name	Default Value	Limits
Built-in	DataSecurity	piadmin: A(r,w) | piadmins: A(r) | PIWorld: A(r)	Each identity may be assigned read, write, or no access (blank)
Base	Descriptor	blank	
Built-in	DigitalSet	no default	Used only for digital tags This must be specified when the point is created.
Base	DisplayDigits	-5	-20 ≤ x ≤ 10
Base	EngUnits	blank	
Base	ExcDev	1 eng units	
Built-in	ExcDevPercent	Comes from ExcDev	0 ≤ x ≤ 100
Base	ExcMax	600 sec	
Base	ExcMin	0 sec	0 ≤ x ≤ 65535
Base	ExDesc	blank	
Classic	Filtercode	0	
Classic	InstrumentTag	blank	
Classic	Location1	0	
Classic	Location2	0	
Classic	Location3	0	
Classic	Location4	0	
Classic	Location5	0	
Built-in	NEWTag		
Built-in	PointID		System-Assigned
Base	PointSource	Lab	
Base	PointType	Float32	
Built-in	PtAccess	o:rw g:r w:r	Owner, group, and world may be assigned read, write, or no access (blank)
Built-in	PtClassName	Base	Base, classic, Totalizer, alarm
Built-in	PtGroup	piadmins	In PI Identity Database
Built-in	PtOwner	piadmin	In PI Identity Database
Built-in	PtSecurity	piadmin: A(r,w) | piadmins: A(r) | PIWorld: A(r)	Each identity may be assigned read, write, or no access (blank)
Built-in	RecNo		System-assigned
Base	Scan	On	ON, OFF, 1, or 0
Base	Shutdown	True	
Built-in	SourceTag	blank	
Base	Span	100	x ^{3} 0
Classic	SquareRoot	0	On, Off, or 0 ≤ x ≤ 10

Point Class	Field Name	Default Value	Limits
Classic	Srcptid	0	
Base	Step	OFF for all numeric points ON for all non-numeric points	
Base	Tag	no default	This must be specified when the point is created.
Classic	Totalcode	0	
Base	TypicalValue	50	Zero ≤ x ≤ (Zero + Span)
Classic	UserInt1	0	
Classic	UserInt2	0	
Classic	UserReal1	0	
Classic	UserReal2	0	
Base	Zero	0	
Other			The Totalizer Point class includes other attributes, which are discussed in the *PI Server Applications User Guide*.

System-assigned attributes

When you create a point, several attributes are assigned by the system. You cannot change the values of these attributes.

ChangeDate

The date and time when the point was last edited.

Changer

The last user to edit the point.

CreationDate

The date and time when the point was created.

Creator

The user who created the point.

PointID

The unique number that identifies the point internally. PointID is never reused, even when a point is deleted. PointID is the PI point identifier that is passed as a parameter to most of the PI API functions. In the PI API Manual, this identifier is referred to as the point number, or PtNum.

RecNo

The record number contains the point's primary record number in the archive. This is useful when using tools such as **`piartool -aw`** to examine the archives.RecNo is not to be confused with the PointID attribute.

Exception reporting and compression testing

You can tune your PI points for maximum efficiency with the configurable attributes that specify compression and exception reporting. The configuration of these specifications impacts the flow of data from the interface node to the server for that point (exception reporting) and the efficiency of data storage in the archive for that point (compression testing).

- Compression testing

  PI Snapshot Subsystem performs compression testing on the PI Server to enhance data storage efficiency and thereby conserve disk space. The compression test uses a sophisticated algorithm, called the swinging door compression algorithm, to determine which events should be stored in the PI archive. PI Server needs to store only those events deemed significant by the compression test; it can essentially recreate other events through extrapolation of surrounding events.

- Exception reporting

  This process filters out noise, and thereby reduces the communication (I/O) burden between the PI Server and the interface node. As networks have improved and I/O capacity has become less of an issue, some PI System Managers have essentially turned off exception reporting, by setting the exception deviation to 0. OSIsoft recommends that you set the exception deviation to slightly smaller than the precision of the instrument. Exception reporting is a simple linear test that occurs on the interface node.

Topics in this section

- Compression guidelines
- Compression testing
- Exception reporting

Compression guidelines

Using compression gives you the flexibility to configure on a per-point basis, with the option of archiving relevant information. Compression greatly impacts performance, bandwidth, and data access. It is not intended only for saving storage space. You want to store only meaningful data: no noise, no rounding, and no averages. OSIsoft's compression method is designed to remove noise out of the signal, because noisy signals are prevalent in process data. PI Server stores the actual values received from the source, not interpolations or averages or approximations as do some alternative compression methods.

You have complete control over the amount of compression used (including turning it off), from the most compression (lossy) to the least compression (lossless). For example:

- Turn compression on for noisy signals.
- Turn compression on and set the compression deviation (compdev) attribute to zero. With this setting, successive identical values (or values aligning perfectly) are not archived. This is more efficient than turning compression off.
- Turn compression off for non-noisy signals like lab measurements. When compression is turned off, all exceptions are archived.

  It is appropriate to turn off data compression for manually-entered and totalized data, and for other tags where each event is significant in itself and not merely representative of an underlying flow.

In general, a PI Server installation with default compression values is appropriate for most cases.

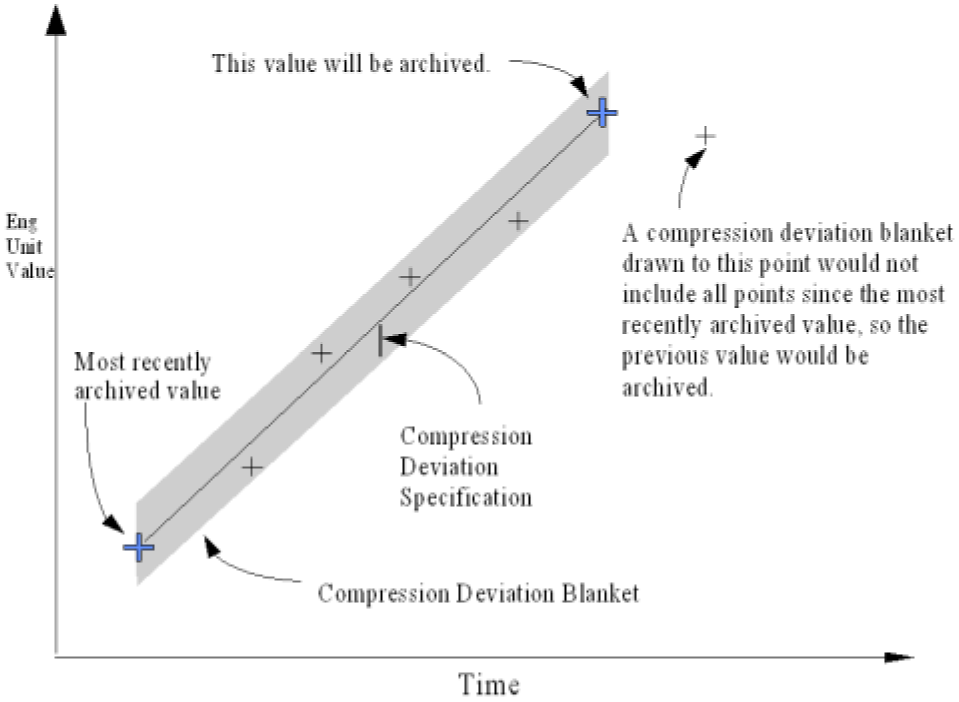
Setting compression requires you to apply your process knowledge about the nature of the signal being compressed. One compression deviation specification will not work for all measurements. It is a function of sensor type, instrument accuracy, and so on. Fortunately, there are broad categories of measurements in a process plant. All similar flow meters, pressure gauges, and thermocouples have exactly the same degree of repeatability and reproducibility in their measurements. Setting the compression specifications for these measurements at the accuracy of the sensing device is easy and will compress out measurements that represent noise in the signal. This dramatically improves performance for end-users and does not cause loss of any significant data.

Compression testing

When a new snapshot arrives, the previous one is evaluated according to the compression specifications to see if it is a significant event. If so, it is sent to the event queue. If not, it is discarded. This process is called *compression*. The point of compression testing is to store just enough data to accurately reproduce the original signal.

PI uses a sophisticated compression algorithm to determine which events it needs to keep in order to provide an accurate data history. The compression method used by PI allows PI to keep orders of magnitude more data online than conventional scanned systems. The data is also much more detailed than in an archiving system based on averages or periodic samples.

The compression method is called *swinging door compression*. Swinging door compression discards values that fall on a line connecting values that are recorded in the archive. When a new value is received by PI Snapshot Subsystem, a new archive value will be recorded and the timestamp of that new value will be that of the last snapshot value received before the latest snapshot value. This value is recorded only if any of the values since the last recorded value do not fall within the compression deviation blanket. The deviation blanket is a parallelogram extending between the last recorded value and the new value with a width equal to twice the compression deviation specification.


This value will be archived.
Eng Unit Value
A compression deviation blanket drawn to this point would not include all points since the most recently archived value, so the previous value would be archived.
Most recently archived value
Compression Deviation Specification
Compression Deviation Blanket
Time

Each point has three attributes that comprise the compression specifications: CompDev (compression deviation), CompMin (compression minimum time), and CompMax (compression maximum time). CompDev is the half-width of the deviation blanket (as shown in the illustration). CompDevPercent is similar to CompDev, but it specifies the compression deviation in percent of Span rather than in engineering units.

CompMin and CompMax are limits that refer to the time between events in the archive. A new event is not recorded if the time since the last recorded event is less than the compression minimum time for the point. The snapshot event is always recorded if the time since the most recent snapshot event is greater than or equal to the compression maximum time.

Note:

The maximum time specification does not guarantee that a value will be written to the archive within a certain time. The archive waits for events to be sent to it. It does not check to see if a point has timed out. It does not *create* new values.

You can adjust the compression parameters to produce efficient archive storage without losing significant data. The compression maximum time is usually set to one value for all points in the system. It should be large enough that a point that does not change at all uses very little archive space. A compression maximum time of one work shift (for example, 8 hours) is often a good choice.

Use the compression minimum time (CompMin) to prevent an extremely noisy point from using a large amount of archive space. This parameter should be set to 0 for any point coming from an interface that does exception reporting. In this case, the exception minimum time should be used to control particularly noisy points. For a data acquisition system with a slow scan time, this parameter is not important. There are few cases where you want to use a non-zero compression minimum time.

The most significant compression parameter is the deviation specification, CompDev. This parameter is often adjusted after the point is defined. A reasonable starting point is one or two percent of Span for transmitters and 0.5 to 1.0 degrees for thermocouples. Look at trend displays to find points for which the reproduction of the data is not acceptable. The goal is to filter out instrument and process noise and still record significant process changes. The effect of changing the compression deviation is not predictable.

For digital points, any change is a significant change. Only the compression maximum and minimum time are important. The compression deviation specification is ignored for digital points.

There are three instances where an event will bypass the compression process and be put in the event queue:

- If the Compressing attribute for the point is set to OFF.
- If the timestamp is older than the timestamp of the current snapshot. Such an event is considered out of order.
- If the Status attribute of the point has changed.

Procedure

1. Step flag.

Step flag

The step attribute setting affects both display and compression.

Data for points with this attribute set to 1 is assumed to remain fixed between events, whereas for points with step=0 data is assumed to change linearly between valid numeric events.

The swinging-door compression, explained above, is not used when the step flag is set. Instead, an exception calculation is applied using the CompDev value. If the absolute difference between the current snapshot and the last archive value is greater than CompDev then the snapshot is sent to the archive.

Compression maximum and minimum limits work the same as for tags with the step flag not set.

Exception reporting

PI interfaces use the exception reporting process to evaluate the significance of new events. The interface sends significant events to the PI Server and discards events that are not significant. The purpose of exception reporting is to avoid sending changes that are smaller than the instrument can measure, from the interface to the PI Server.

The interface compares each new value to the previously sent value. The interface sends the new value to the PI Server only if it is different from the previous value by an amount larger than the value in the ExcDev attribute.

Exception reporting uses a simple deadband algorithm to determine whether to send events to PI. For each point, you set exception reporting specifications (the ExcDev, ExcMin and ExcMax attributes) to create the deadband. The interface ignores values that fall inside the deadband.

Interface programs that do exception reporting apply the following algorithm rules whenever a new value is received: A new value is compared to the last value reported; if the new value does not fall within the deadband, an exception occurs; when an exception occurs, the interface sends the event (both timestamp and value) that caused the exception and the previous event to the snapshot.

The new value is not reported:

- Unless the difference between the new value and the last value is greater than the exception deviation specification, *and* the difference between the times of the new and last values is greater than or equal to the exception minimum time specification.
- *Or*, the difference between the timestamp of the new value and the timestamp of the last reported value is greater than or equal to the exception maximum time specification.

Note:

The time between exception reports might be greater than the exception maximum time if no new values are received by the interface for a point. Neither the PI Server nor the interface will *create* data.

Some interfaces do not support exception reporting. See the documentation for your interface to determine whether it supports this capability. Manually entered data is not normally reported by exception so that every value can be retained.

Most OSIsoft interfaces report new events on exception. The exception algorithm relies on the following parameters:

- Exception Maximum: Maximum time span between exceptions, expressed in seconds. This value is configured for each point in the attribute ExcMax.
- Exception Minimum: Minimum time span between exceptions, expressed in seconds. This value is configured for each point in the attribute ExcMin.
- ExcDev: Deadband when exceeded causes an exception. This is configured for each PI point in either the ExcDev or ExcDevPercent attribute.
- OldEvent: Value/status/timestamp of last event sent to the snapshot; this is the last event that passed exception report.
- PrevEvent: Value/status/timestamp of last event compared to determine whether or not to send to the snapshot.
- NewEvent: Value/status/timestamp of event to test for exception.

Exception reporting works by comparing the new event to the old event as follows.

- If the time new event timestamp and old event timestamp is greater than or equal theExcMax, the new event is sent to the snapshot.
- For digital points, if the new value differs from the old value, the new event is sent to the snapshot regardless of ExcMin time.
- For numeric points, if the status changes from good to bad, or bad to good, the new event is sent to the snapshot.
- For numeric points, if the time between the old event and the new event is greater than or equal to ExcMin and the absolute value of the difference between the new value and the old value is greater than ExcDev, the value is sent to the snapshot.
- If the new event was sent to the snapshot, the old event is replaced by the new event.

The last step is a test to see if the PrevEvent should also be sent to the snapshot. If the PrevEvent was not equivalent to the original OldEvent, the PrevEvent is sent to the snapshot. The only time the PrevEvent is not sent to the snapshot is when two consecutive exception reports send the new event to the snapshot. The PrevEvent is used to accurately indicate what really happened to the value; without it, a step change would look like a ramp change. Basically, if a measurement holds steady for hours, then makes a step change, just sending the new value to the snapshot results in interpolating between the old value and the new value. By also sending the PrevEvent, the step change is stored.

Topics in this section

- ExcDev and ExcDevPercent
- ExcMin
- ExcMax
- Turning off exception reporting

ExcDev and ExcDevPercent

The ExcDev attribute (Exception Deviation) specifies in engineering units how much a value may differ from the previous value before it is considered to be a significant value. The ExcDevPercent attribute specifies the same thing as a percentage of the Span attribute. A typical value is 1 percent of Span. The exception deviation should be less than the compression deviation by at least a factor of 2.

You can set either the ExcDev or the ExcDevPercent attribute. If you change one, the other is automatically changed to be compatible. If you try to change both at once, ExcDevPercent takes precedence.

ExcMin

The Exception Minimum attribute, ExcMin, is a deadband after the previous value. This is used to suppress noise. It is specified in seconds. A new data value that is received before the end of the ExcMin interval is discarded.

ExcMax

The Exception Maximum attribute, ExcMax, puts a limit on the length of time that values can be discarded due to exception testing. For example, it is possible for the incoming data to be a single value for many days. If ExcMax is set to 28800 seconds (8 hours) then a value will not be discarded due to exception if the previous event timestamp was more than 28800 seconds before that. Note that the interface does not manufacture data. If there are no incoming values within 28800 seconds, then nothing is passed to the PI Server.

Turning off exception reporting

To turn off exception reporting (that is, to generate an exception for every event):

Procedure

1. Set ExcMin=0
2. Set ExcMax = 0

Change PI point type

In PI Server 3.4.370 or later, you can edit the type attribute of a point, just as you change other attributes.

You can use PI SMT, PI Tag Configurator, or **`piconfig`** to change point types.

Procedure

1. Stop the PI interface that collects data for the point you plan to change.
2. Open PI SMT.
3. Navigate to **Points** > **Point Builder**.
4. Search for and select the point for which you would like to change the type attribute.
5. In **Point type**, select the desired point type.
6. Save your changes.

Topics in this section

- Allowable point type coercions
- Point type changes and archives
- Valid point type edits
- Configure error handling

Allowable point type coercions

In order for a point type attribute to be changed successfully, you must change between point types that can be coerced.

Data from the previous type(s) is coerced to the current type at retrieval time if possible. In order for a point type attribute to be successfully coerced and subsequently changed, you must make changes only between point types that are allowed. .

If an event in the archive cannot be coerced to the edited point type, the digital state Coercion Failed is returned by default. The Coercion Failed digital state acts as a placeholder for an event that PI Snapshot Subsystem failed to coerce. Out-of-order events may also result in a Coercion Failed digital state.

You can use the parameter Archive_DataCoercionPolicy to translate a digital state, as appropriate. For details, see Configure error handling.

Allowable point type coercions

	int16	int32	float16	float32	float64	digital	string	blob	timestamp
int16		ok	ok[5]	ok	ok	ok	ok	N/A	N/A

	int16	int32	float16	float32	float64	digital	string	blob	timestamp
int32	ok[1]		ok[5]	ok	ok	ok[3]	ok	N/A	ok
float16	ok[1]	ok[2]		ok	ok	ok[3]	ok	N/A	N/A
float32	ok[1]	ok[2]	ok[5]		ok	ok[3]	ok	N/A	ok
float64	ok[1]	ok[2]	ok[5]	ok		ok[3]	ok	N/A	ok
digital	ok	ok	ok	ok	ok		ok	N/A	N/A
string[5]	ok	ok	ok	ok	ok	ok[4]		N/A	ok
blob	N/A	N/A	N/A	N/A	N/A	N/A	N/A		N/A
timestamp	N/A	ok	ok	ok	ok	N/A	ok	N/A	

[1] Assuming values in the range of 0 to 32767

[2] Assuming values in the range of -2,147,450,880 to 2,147,483,647

[3] Assuming positive, integer values that are lower than number of digital states

[4] Assuming exact, case-insensitive match with a state string

[5] Assuming the range of the source is compatible with the range of the target

[6] When going from string type, the coercion is possible only if the string values are in numerical form. For example, the string "3.5" can be coerced to a float, but the string "hello" cannot.

Note:

When you change point types to int16 or digital, you must enter a value for the Zero and Span attributes.

Point type changes and archives

When you change a point type attribute, the current archive record is closed and a new record is open. The new record uses the changed point type attribute.

Point type edits affect how users will see both newly generated data and data that was previously archived. For this reason, you should consider whether you want to:

- Preserve the original point type in the archive history, or
- Convert all archives to reflect the point type change.

By default, the original point type is preserved in the archives. That is, the events that were created and archived prior to the point type edit will reflect the point type that was used before the point type edit.

If you want the previously archived data to reflect the new point type, you can reprocess your archives off-line to convert the stored events to the new point type. See Manage offline archive files.

Valid point type edits

PI Server logs a message in the PI Message Log when you successfully edit a point type. To successfully edit a point type, the point type must receive snapshot values that are valid for the new point type.

If the snapshot value cannot be coerced, the edit fails.

For example, if you change a point type to int16 and the current snapshot has a negative value, the edit fails and the following error is returned because int16 does not translate negative values:

```
[-10005] Subscript Under Range
```

Configure error handling

If PI Archive Subsystem cannot coerce a stored point type to an edited point type, values are replaced as specified by the Archive_DataCoercionPolicy tuning parameter. You can use the PI SMT Tuning Parameters tool to configure this parameter, or use **`piconfig`** to update this parameter in the PI Timeout table.

An Archive_DataCoercionPolicy parameter can have one of these values:

0	DTC_MarkBad	Failed events are returned as DS -315 (“Coercion Failed”)
1	DTC_Leave	Original events are returned (mixed types)
2	DTC_Zero	Returned as 0 or blank depending on the type
3	DTC_Hide	Hidden (skip that event)

Modification of PI point classes and attribute sets

In PI Server version 3.4.370 or later, users can edit and delete both attribute sets and point classes.

> **!** **Caution:**
>
> The modification of point classes and attribute sets can trigger a cascade of edits through a multitude of points and across a multitude of group and user boundaries. Do this only if absolutely necessary. Do a backup before you begin.

Point classes can be modified as follows:

- Attributes ExcMax and CompMax in base attribute set can be edited from uint16 to uint32.
- It is possible to move a point from one class to another. Data collection can continue for this point even though the method of collection may change. For example, one may convert a Totalizer point, which belongs to the Totalizer point class and is populated by PI Totalizer Subsystem, to a Performance Equation point, which belongs to the Classic point class and is populated by the performance equation scheduler.

- Users can change the attributes of a given point by modifying the attribute sets of the parent point class. This type of modification applies to all points that belong to the same point class.
- Adding, removing, and editing attributes of a point class are generally motivated by changes in the data collection methods. These operations have little or no effect on the retention of history for individual points.

Point attributes can be modified as follows:

- Change the point class of a point to another point class that contains the desired attributes. To do this, change the PtClassName attribute of the selected point.
- Change the point class explicitly by any combination of deleting and adding attribute sets.
- Change the point class implicitly by modifying existing attribute sets. A modification of an attribute set triggers a cascade of edits through all point classes that use the attribute set.
- Both implicit and explicit point class modifications trigger edits of all points belonging to the modified point class. If modification of all points in a point class is not the intended effect, change the point classes of individual points instead.

Topics in this section

- Required access permissions and other restrictions
- Attribute set database edit
- Point classes database edit
- Point class rename
- Editing a point's point class
- Point database audit

Required access permissions and other restrictions

Only the piadmin user or the piadmins group can create, delete or edit attribute sets and point classes. You must also have read and write permissions on the Point Database (PIPOINT). The piadmin user always has read/write access to PIPOINT, but if you are using piadmins, you might need to set the permissions explicitly.

The following restrictions apply when editing and deleting attribute sets and point classes (but not when creating them):

- Attribute sets and point classes may be edited or deleted only in stand-alone mode, in which PI Net Manager closes the TCP/IP listener and disallows any client connections. Existing PI SDK, PI API and PI Server Application connections close, and reconnection attempts are refused for the duration of the stand-alone mode. Subsystems and locally run utilities such as **`piconfig`** and **`piartool`** can connect. Default-only attribute edits are supported in normal mode.

  The command `piartool -sys -standalone on` puts the system in stand-alone mode, that is, no clients can connect, and **`piartool -sys -standalone off`** sets it in normal operating mode. Use `piartool -sys -standalone query` to query the system's current mode.
- You may add attributes to any set, except the Base attribute set.

Note:

Any attribute added to a predefined set cannot be removed. The predefined attribute sets are required by predefined point classes, which cannot be deleted, so they are always in use. When expanding a point class, we recommend that you create a new attribute set with new attributes rather than adding new attributes to a predefined set. For a list of predefined attribute sets and point classes, see Predefined point classes and Predefined attribute sets.

.

- You may delete attributes from any set (only if not in use in any point class) except:
  - Required attributes contained in sets predefined in PI Server
  - Attributes contained within in-use attribute sets
- You may not rename attributes
- You may rename attribute sets, except predefined attribute sets
- You may delete any attribute sets except:
  - Predefined attribute sets
  - In-use attribute sets
- You may add attribute sets to any point class
- You may delete attribute sets from a point class (only if not used by any point) except:
  - Required attribute sets contained in predefined point classes (see Predefined point classes)
  - In-use point classes
- You may rename any point classes, except predefined point classes
- You may delete any point classes, except:
  - Predefined attribute sets
  - In-use point classes

These restrictions protect the attribute sets and point classes that PI System uses as building blocks. These restrictions can limit the user's ability to easily undo some actions. Always make a backup of the PI Point Database before attempting to edit attribute sets or point classes. You can delete attribute sets from predefined point class as long as the class is not in use and the set to be deleted is not a required set for that point class. Any attribute added to a predefined attribute set can never be removed.

The **`piconfig`** utility can be used to perform these edits and deletes after placing the PI Server in stand-alone mode using the **`piartool`** utility.

Attribute set database edit

Note:

Only the piadmin user or the piadmins group can create, delete or edit attribute sets and point classes. You must also have read and write permissions on the Point Database (PIPOINT). The piadmin user always has read/write access to PIPOINT, but if you are using piadmins, you might need to set the permissions explicitly.

Topics in this section

- Attribute set creation
- Attribute set deletion
- Attribute set edits
- Attribute set rename

Attribute set creation

To create an attribute set, specify the set name, attribute names, types, and default values. If the type is not specified, float32 is assigned. If a default value is not specified, PI sets the value. Supported attribute types and defaults lists allowed types and defaults. Types not listed are not supported, and are either rejected at attribute set creation time or have unexpected behavior.

Topics in this section

- Supported attribute types and defaults
- Disallowed attribute names
- Example: Create an attribute set

Supported attribute types and defaults

- String (“”)
- Int16 (0)
- Int32 (0)
- BYTE (0)
- UBYTE (0)
- Uint16 (0)
- Uint32 (0)
- Timestamp (“31-Dec-69 16:00:00”)
- Float32 (0)
- Bool (0)

Note:

Boolean values show as either 1 or 0 instead of true or false. All non-zeros are interpreted as true and 0 is interpreted as false.

Disallowed attribute names

The following attribute names are not allowed in any user-defined attribute set:

Built-in attribute names

DataAccess	CompDevPercent	PointID	PtOwner
DataGroup	DigitalSet	PtAccess	PtSecurity
DataOwner	Excdevpercent	PtClassName	RecNo
DataSecurity	NEWtag	PtGroup	SourceTag

Reserved names

Class	NEWSET
NEWCLASS	Set

Base attribute names

Archiving	Creationdate	ExcMin	Tag
ChangeDate	Creator	PointSource	TypicalValue
Changer	Descriptor	PointType	Zero
CompDev	DisplayDigits	Scan	ExcMax
CompMax	EngUnits	Shutdown	
CompMin	ExDesc	Span	
Compressing	ExcDev	Step	

The built-in attributes are added to all points. Users cannot modify their types and defaults. However, users can modify the default values of non-system-assigned attributes, such as PtSecurity, DataSecurity, PtOwner, PtGroup, PtAccess, DataOwner, DataGroup, DataAccess, DigitalSet, ExcDevPercent,CompDevPercent, and SourceTag.

OSIsoft creates the base attribute set. See Attribute set edits for details. Attribute name checks are case-insensitive.

Example: Create an attribute set

This example uses the **piconfig** utility to create an attribute set. Stand-alone mode is not required for creating an attribute set.

Procedure

1. In a command prompt window, change to the `PI\adm` directory
2. Type:

   ```
   piconfig
   ```
3. At the **piconfig** prompt, type:

```
@table piatrset
@mode create
@istru set
@istru attrib,type,default
@istru ...
MyAttributeSet
MyAttribute1,BYTE,
MyAttribute2,int32,2
MyAttribute3,string,"Default string"
MyAttribute4,,
@ends@
```

MyAttribute4 is of type float32 and default of 0.0.

4. To list the attribute set just created:

```
@table piatrset
@mode list
@ostru set
@ostru attrib,type,default
@ostru ...
@select set=MyAttributeSet
@ends
```

The output will look like:

```
MyAttributeSet
MyAttribute1,BYTE,0
MyAttribute2,Int32,2
MyAttribute3,String,Default string
MyAttribute4,Float32,0.
* End Repeat...
*----------
```

Attribute set deletion

An attribute set can be removed by simply specifying the set name.

Predefined attribute sets are used as building blocks for PI point classes and may not be removed from the database. When an attribute set deletion is requested, whether it is a removable attribute set is checked. If not a removable set, an error is returned. The following sets are predefined sets and may not be removed.

- Alarmparam
- Base
- Classic
- Sqcalm_parameters
- Totals

If the set to be removed is in use by any point class, an error is returned.

Procedure

1. Example: Delete an attribute set.

Example: Delete an attribute set

Procedure

1. In a command prompt window, change to the `PI\adm` directory.
2. Place the system in stand-alone mode. Type:

   ```
   piartool -sys -standalone on
   ```

3. Type:

   ```
   piconfig
   ```

4. At the **`piconfig`** prompt, type:

   ```
   @table piatrset
   @mode delete
   @istru set
   MyAttributeSet
   @ends
   ```

5. When finished, place the system back in normal mode:

   ```
   piartool -sys -standalone off
   ```

Attribute set edits

An attribute set can be edited by adding, removing attributes and/or changing attribute types and default values. Be sure to make a backup before you edit an attribute set.

Edits other than default-only edits require that the system be put in stand-alone mode. Use the **`piartool`** utility to enter stand-alone mode and to return to normal mode:

- To enter stand-alone mode, enter:

  ```
  piartool -sys -standalone on
  ```

- To return to normal mode, enter:

  ```
  piartool -sys -standalone off
  ```

Default-only edits do not require stand-alone mode.

Default-only edits modify the existing sets directly. Default edit triggers implicit point class edits but *do not* trigger implicit point edits. This is because the new defaults are applied only to new points. In the rest of this document, an edit implies non-default-only edits unless stated otherwise.

Topics in this section

- Implicit point class and point edits
- Built-in attributes
- Base attributes and allowed types
- Example: Editing an attribute set
- Informational messages

Implicit point class and point edits

When an attribute set is edited, all dependent point classes and points are edited without user intervention. These edits are known as *implicit edits*.

With implicit point edits, the existing attribute values are not changed if they are compatible with the new types. If the new attribute type is not compatible with the old one, the new default takes precedence over the existing attribute's value. Additional attributes are assigned default values.

Built-in attributes

Built-in attributes are a part of every PI point, but do not belong to any particular attribute set. The types and defaults of built-in attributes do not belong to any attribute set explicitly and cannot be edited.

Base attributes and allowed types

The only Base attribute set edits allowed are the conversion of types of CompMax, ExcMax, Creator, and Changer attributes from uint16 to uint32 and changes to the default values of any attributes in this set. All other edits to the Base attribute set are not allowed.

Note:

In PI Server versions prior to 3.3, ExcMax and CompMax were type uint16.

Name	Allowed type
Descriptor	String
ExDesc	String
TypicalValue	float32
EngUnits	string
Zero	float32
Span	float32
PointType	ubyte
PointSource	string
Scan	byte
ExcMin	uint16
ExcMax	uint16 or uint32
Excdev	float32
shutdown	byte
Archiving	byte
Compressing	byte
Step	byte
Compmin	uint16
Compmax	uint16 or uint32
Compdev	float32
Creationdate	timestamp

Name	Allowed type
Creator	uint16
Changedate	timestamp
Changer	uint16
DisplayDigits	byte

Example: Editing an attribute set

If you edit an attribute set, PI Base Subsystem edits its dependent point classes, and subsequently dependent points, internally. You do not need to explicitly edit the PI point classes database. Such indirect edits are referred to as implicit edits.

To illustrate, suppose you want to change a set called MyAttributeSet. First, put the system in stand-alone mode using **piartool**:

```
piartool -sys -standalone on
```

Then list the existing attributes in the **piconfig** utility:

```
@table piatrset
@mode list
@ostru set
@ostru attrib,type,default
@ostru ...
@select set=MyAttribSet
@ends
```

Suppose the attributes and their types and defaults of this attribute set appear as:

```
MyAttribSet
MyAttrib1,Int32,22
MyAttrib2,BYTE,0
MyAttrib3,Float32,5.
```

To change the attribute MyAttrib2 to type String and add another attribute, MyAttrib4 of type uint16 in **piconfig**:

```
@table piatrset
@mode edit
@istru set
@istru attrib,type,default
@istru ...
MyAttribSet
MyAttrib1,int32,22
MyAttrib2,String,default string
MyAttrib3,float32,
MyAttrib4,uint16,1
@ends
```

Now list the resulting set:

```
@mode list
@ostru set
@ostru attrib,type,default
@ostru ...
@select set=MyAttribSet
@ends
MyAttribSet
MyAttrib1,Int32,22
MyAttrib2,String,default string
MyAttrib3,Float32,0.
MyAttrib4,Uint16,1
```

When editing an attribute set, you must explicitly redefine the attribute name, type, and default. If a pre-existing attribute is not specified in the new definition, it is permanently removed from the set. If you had not wanted to edit the existing attributes, but only wanted to add a new attribute MyAttrib4, you would still need to specify all attributes in his definition. For example:

```
@table piatrset
@mode edit
@istru set
@istru attrib,type,default
@istru …
MyAttribSet
MyAttrib4,uint16,1
@ends
```

produces MyAttribSet containing only one attribute, MyAttrib4.

When you edit an attribute set, you must completely specify all the attributes, exactly as on creation. If an attribute set is edited and its pre-existing attribute name (but not the type and default) is specified, float32 and value 0.0 is assigned, overwriting the original type and default. If the user specifies only the type, a new default is assigned even if the type is identical to the previous one. The default of MyAttrib3 attribute was changed to 0.0 from the original 5.0 because it was not explicitly specified in the edit.

When you are finished with the edit, place the system back in normal mode so that applications can connect.

```
piartool -sys -standalone off
```

Renaming an attribute set does not trigger any implicit edits of point classes or points and does not require stand-alone mode.

See Required access permissions and other restrictions for attribute set edit restrictions.

Informational messages

Some messages are not directly returned to the application that initiates the edit, such as **`piconfig`**, but are instead sent to the PI Message Subsystem. Examples of these messages are information regarding the status (success or failure) of the edit steps (rename the old set, add a new set, implicitly edit dependent point classes and points, and remove the old set) and the number of dependent point classes found. These messages are useful in verifying that the steps correctly followed during the edit.

Attribute set rename

An attribute set may be renamed via edit unless it is one of the predefined attribute sets. This rename does not require stand-alone mode.

Procedure

1. Example: Renaming an attribute set.

Example: Renaming an attribute set

Procedure

1. In a command prompt window, change to the `PI\adm` directory
2. Type:

   ```
   piconfig
   ```

3. At the **`piconfig`** prompt, type:

   ```
   @table piatrset
   @mode edit
   @istru set,newset
   MyAttribSet,MyNewAttribSet
   @ends
   ```

Point classes database edit

For details on indirect (that is, implicit) edit of PI Point Class Database, see Attribute set edits. This section explains how to explicitly create, edit, or delete a point class.

> **Note:**
>
> Only the piadmin user or the piadmins group can create, delete or edit attribute sets and point classes. You must also have read and write permissions on the Point Database (PIPOINT). The piadmin user always has read/write access to PIPOINT, but if you are using piadmins, you might need to set the permissions explicitly.

Topics in this section

- Point class creation
- Point class deletion
- Point class edit

Point class creation

Once a new point class is created, you can start assigning points to this class. Create a point class using **`piconfig`**, specifying which attribute sets to include. This does not require stand-alone mode. All point classes must include the base attribute set.

Procedure

1. Example: Create a point class.

Example: Create a point class

Procedure

1. In a command prompt window, change to the `PI\adm` directory.
2. Type:

   ```
   piartool -sys -standalone on
   ```

3. At the **`piconfig`** prompt, type:

```
@table piptcls
@mode create
@istru class
MyPtClass
@ends
```

4. To go back to normal mode, enter:

```
piartool -sys -standalone off
```

Point class deletion

Predefined point classes and in-use point classes cannot be deleted.

Procedure

1. Example: Delete a point class.

Example: Delete a point class

Procedure

1. In a command prompt window, change to the `PI\adm` directory.

2. Type:

```
piconfig
```

3. At the **`piconfig`** prompt, type:

```
@table piptcls
@mode delete
@istru class
@istru set,...
MyPtClass
Base,MyAttribSet
@ends
```

Point class edit

You can explicitly edit a point class by adding or removing attribute sets that form the point class.

`piconfig` version 3.4.370.x or later can display which attribute sets form a point class:

```
@table piptcls
@ostru class
@ostru set,...
@select class=MyPtClass
@ends
```

This feature makes it easier to determine what attribute sets are being used to form the point class.

Topics in this section

- Example: Edit a point class
- Restrictions on attribute set edit
- Informational messages

Example: Edit a point class

A point class list in **piconfig** shows the following:

```
* (Ls - ) piconfig> @table piptcls
* (LS - PIPTCLS) piconfig> @mode list
* (Ls - PIPTCLS) piconfig> @ostru class
* (Ls - PIPTCLS) piconfig> @ostru set,...
* (Ls - PIPTCLS) piconfig> @select class=MyPtClass
* (Ls - PIPTCLS) piconfig> @ends
MyPtClass
base,classic
*----------
```

Procedure

1. Add the attributes MyAttribute1 (string) and MyAttribute2 (int32) to this point class. To do this, create an attribute set, MyAttributeSet, as follows.

   ```
   @table piatrset
   @mode create
   @istru set
   @istru attrib,type,default
   @istru ...
   MyAttributeSet
   MyAttribute1,string,my default string
   MyAttribute2,int32,22
   ```

2. Check that the attribute was correctly created:

   ```
   @table piatrset
   @mode list
   @ostru set
   @ostru attrib,type,default
   @ostru ...
   @select set=MyAttributeSet
   @ends
   ```

   You should see:

   ```
   MyAttributeSet
   MyAttribute1,String,my default string
   MyAttribute2,Int32,22
   * End Repeat...
   *----------
   ```

3. Edit MyPtClass to include this attribute set. The system must be in stand-alone mode. Enter at a command prompt:

   ```
   piartool -sys -standalone on
   ```

4. In **piconfig**, define the attribute sets that should belong to the point class:

   ```
   @table piptcls
   @mode edit
   @istru class
   @istru set,...
   MyPtClass
   base,classic,MyAttributeSet
   ```

5. Check that these attributes now form MyPtClass.

   ```
   * (Ed - PIPTCLS) piconfig> @mode list
   * (Ls - PIPTCLS) piconfig> @ostru class
   * (Ls - PIPTCLS) piconfig> @ostru set,...
   * (Ls - PIPTCLS) piconfig> @select class=MyPtClass
   * (Ls - PIPTCLS) piconfig> @ends
   ```

You should see:

```
MyPtClass
base,classic,MyAttributeSet
*----------
```

6. To see all attributes that are in this point class, enter:

```
@table pipoint
@ptclass MyPtClass
@?atr
```

The following list appears:

```
1 - Tag String D: !#!#!# C:
2 - NEWTag String D: C:
3 - archiving BYTE D: 1 C:
4 - changedate TimeSta D: 31-Dec-69 16:00:00 C:
5 - changer Uint16 D: 0 C:
6 - compdev Float32 D: 2. C:
7 - Compdevpercent Float32 D: 2 C:
8 - CompMax Uint32 D: 28800 C:
9 - CompMin Uint16 D: 0 C:
10 - compressing BYTE D: 1 C:
11 - convers Float32 D: 1. C:
12 - creationdate TimeSta D: 31-Dec-69 16:00:00 C:
13 - creator Uint16 D: 0 C:
14 - DataAccess String D: o:rw g:r w:r C:
15 - DataGroup String D: piadmin C:
16 - DataOwner String D: piadmin C:
17 - datasecurity String D: piadmin: A(r,w) | piadmins (r) | PIWorld (r) C:
18 - descriptor String D: C:
19 - DigitalSet String D: system C:
20 - displaydigits BYTE D: -5 C:
21 - engunits String D: C:
22 - excdev Float32 D: 1. C:
23 - Excdevpercent Float32 D: 1 C:
24 - ExcMin Uint32 D: 600 C:
25 - excmin Uint16 D: 0 C:
26 - exdesc String D: C:
27 - filtercode Int16 D: 0 C:
28 - instrumenttag String D: C:
29 - location1 Int32 D: 0 C:
30 - location2 Int32 D: 0 C:
31 - location3 Int32 D: 0 C:
32 - location4 Int32 D: 0 C:
33 - location5 Int32 D: 0 C:
34 - myattribute1 String D: my default string C:
35 - myattribute2 Int32 D: 22 C:
36 - PointID Int32 D: 0 C:
37 - pointsource String D: Lab C:
38 - pointtype String D: Float32 C:
39 - PtAccess String D: o:rw g:r w:r C:
40 - PtClassName String D: MyPtClass C:
41 - PtGroup String D: piadmin C:
42 - PtOwner String D: piadmin C:
43 - ptsecurity String D: piadmin: A(r,w) | piadmins (r) | PIWorld (r) C:
44 - Recno Int32 D: 1 C:
45 - scan BYTE D: 1 C:
46 - shutdown BYTE D: 1 C:
47 - SourceTag String D: C:
48 - span Float32 D: 100. C:
49 - squareroot Int16 D: 0 C:
50 - srcptid Int32 D: 0 C:
51 - step BYTE D: 0 C:
52 - totalcode Int16 D: 0 C:
53 - typicalvalue Float32 D: 50. C:
54 - userint1 Int32 D: 0 C:
```

```
55 - userint2 Int32 D: 0 C:
56 - userreal1 Float32 D: 0. C:
57 - userreal2 Float32 D: 0. C:
58 - zero Float32 D: 0. C:
```

7. Place the system back in normal mode:

   ```
   piartool -sys -standalone off
   ```

Restrictions on attribute set edit

Some notes on point class edits are:

- All point classes contain the base attribute set.
- Any attribute set can be added to a point class.
- Attribute sets cannot be deleted from point classes that are in use.
- Required attribute sets cannot be deleted from predefined point classes even if they are not in use.
- Predefined classes cannot be renamed.
- Renaming a point class does not trigger any implicit edits of points.

Informational messages

Some messages are not directly returned to the user but are instead are sent to the PI Message Subsystem. Examples of such messages are information regarding the status of the steps involved in point class edit (rename the original class, add a new class, implicitly edit dependent points, remove the original class) and the number of dependent points found.

Point class rename

A point class may be renamed by an edit unless it is one of the pre-defined point classes. This rename does not require stand-alone mode.

Procedure

1. Example: Rename a point class.

Example: Rename a point class

Procedure

1. In a command prompt window, change to the `PI\adm` directory.
2. Type:

   ```
   piconfig
   ```

3. At the **`piconfig`** prompt, type:

   ```
   @table piptcls
   @mode edit
   @istru class,newclass
   MyPointClass,MyNewPointClass
   @ends
   ```

Editing a point's point class

You can change the point class of a PI point. As in the case of implicitly edited point, the attributes of the point are rebuilt. The important difference is that unlike in an implicit point edit, some existing attributes may be removed because a point class edit does not allow removing any attributes if there are any dependent points. This prevents points from inadvertently losing existing attributes. However, if a user deliberately moves a point from one class to another and the new point class does not contain some of this point's current attributes, they are deleted without prompting.

When you change a point's point class, any new attributes are added and set to default values. Attributes that do not belong to the new point class are removed. Existing attributes are copied if they are in the new point class. Compatible types retain their values and incompatible types are set to new default values.

When editing a point with **`piconfig`**, new attributes can be modified simultaneously. That is, it is acceptable to edit the PtClassName attribute and include new attributes that only belong to the new point class and did not previously belong to the point's old class. However, the target class must be set before such an edit is attempted.

To illustrate, try editing a point that belongs to Totalizer point class to Classic point class in **`piconfig`** as follows:

```
@table pipoint
@ptclass Totalizer
@mode edit
@istru tag,ptclassname,location4,pointsource
```

The following error is returned:

```
*piconfig Err> Unknown parameter <location4> in structure
*@istru tag,ptclassname,location4,pointsource
*piconfig Err> Complete Structure line removed
*@istru tag,ptclassname,location4,pointsource
```

This is because `@ptclass Totalizer` sets the environment for this edit to Totalizer point class, which does not have the location4 attribute. If you want to edit the PtClassName attribute and new attributes unique to the target point class at the same time, set the environment to the target point class, Classic, by using `@ptclass classic` first:

```
@ptclass classic
@istru tag,ptclassname,location4,pointsource
tagname,classic,1,C
```

If it is not necessary to edit the PtClassName attribute and new attributes at the same time, issuing:

```
@ptclass classic
```

because PtClassName is a built-in attribute and every point has this attribute.

Point class of a point can be edited using **`piconfig`** in PI Server 3.4.370 or later.

Procedure

1. Converting COM Connector classes to and from native PI classes.

Converting COM Connector classes to and from native PI classes

Special handling is required in case of a native PI point's PtClassName edit to a COM Connector point class or vice versa. The difficulty arises from the fact that in order to allow transparent retrieval of data for a point that has some data in a foreign database and some in a PI archive, the PI System must be aware of the cutoff times and go to the correct source. The possibility that the conversion may occur multiple times adds to the complexity.

History of the conversions is ignored and a data request is directed to the current data source.

Point database audit

The Audit Database includes both attribute sets and point classes. The EnableAudit parameter in the PI Timeout Table bit (which starts from 0) is used for Attribute Sets Audit Database and bit 4 for Point Classes Audit Database.

Database	Bit	Value to Enable (decimal)
Point DB	0	1
Attribute Sets DB	2	4
Point Classes DB	4	16

Use the AuditViewer tool to work with the Audit Database. See the AuditViewer Help and the *Auditing the PI Server* guide for more details.

Digital state sets

A digital state set has a name and consists of a list of discrete states, sometimes called *digital state strings*. For example, we can define the digital state set *Valve-state-set* as containing the two states OPEN and CLOSE. You can use the default set or you can define your own digital state.

Topics in this section

- Digital state name conventions
- Default set System state set
- Define a digital state set

Digital state name conventions

For digital state set names and digital state names, note that:

- Digital set names cannot have the prefix "DIGSET_". The DIGSET_ prefix is reserved for deleted digital state sets.
- Digital state names are not case-sensitive. If you define a state with all capital letters, such as OFF, any later references to that state are not case-sensitive. That is, either off or Off are valid references to the state named OFF.
- Leading and trailing blanks are removed from state names.

Default set System state set

The default set is called the *System digital state set* and contains over 300 digital states that may apply to any point. States may be added to this set, but states in the offset range 193-320 are reserved for use by the PI System and their meaning should not be modified. The last possible state in the System digital state set is number 16383. It is reserved for internal PI Server use.

Note:

OSIsoft recommends that you keep changes to the System digital state set to a minimum. At most, you can translate the states into another language without changing their meaning. Digital points should use a user-defined digital set, not the System digital state set.

A sample of pre-defined digital states that represent typical system states are as follows:

State	Description
I/O Timeout	Interfaces use this state to indicate that communication with a remote device has failed.
No Data	Data-retrieval functions use this state for time periods where no archive values for a tag can exist 10 minutes into the future or before the oldest mounted archive.
Under Range	For float16 point types, this state indicates a value that is less than the zero for the tag.
Over Range	For float16 point types, this state indicates a value that is greater than the top of range (Zero+Span) for that tag.
Pt Created	This state is assigned to a tag when it is created. This is a tag's value before any value is entered into the system.
Shutdown	All tags that are configured to receive shutdown events are set to this state on system shutdown.
Arc Off-line	Used by data-retrieval functions to indicate a period of time not covered by any mounted archive.
Bad Input	Interfaces use this state to indicate that a device is reporting bad status.

Define a digital state set

You can define digital state sets with the Digital States tool in PI SMT (**Points** > **Digital States**) or **`piconfig`**. For digital points, there are typically only a small number of valid states. Before you can configure a digital point, you need to define the digital state set that it will use.

For example, you can configure a point to use an instrument system that reports a valve position as having a digital code value of 0 or 1 and assigns the value 0 to a state of CLOSED and the value 1 to a state of OPEN.

Before you configure this point, you must establish a digital state set with two strings, CLOSED and OPEN. You might name this digital state set "Valve Position" . You could also define other points that also have CLOSED and OPEN states to use the same Valve Position digital state set. Points that have states of ON and OFF would use a different digital state set, which you could name "Switch Position" .

PI Server determines the digital code value, such as 0 or 1, based on the position of the digital state string in the Digital State Table. The first value is 0, the second is 1; the third is 2, and so on. The position in the set is termed `digcode` in PI API.

When retrieving state strings, PI API returns a maximum of 79 characters. Do not use the prefix "DIGSET_" for digital set names. The DIGSET_ prefix is reserved for deleted digital state sets.

Note:

There is no need to include system states in custom digital sets because the system states contained in the System State Set is used where needed by the PI Server. A total of 16384 sets may exist on the system, and the first set is the system digital state set. You may define up to 16383 state sets with up to 16383 states in each set. For details, see Default set System state set.

PI archives

Archive management tools

PI System provides client tools in PI System Management Tool (SMT) and command-line utilities for working with archives:

- SMT archive management tools
    - Archive Editor

      This tool is for working with the data in PI archives. Use Archive Editor to view, edit, insert, and delete values for PI point events in a PI archive. To open the tool, select **Data** > **Archive Editor** in SMT.
    - Archives tool

      This tool displays a list of registered archives for each connected PI Server. The archive list contains columns that describe the status and properties of each archive. Toolbar functions and a context menu allow you to monitor and manage archive use. To open the tool, select **Operation** > **Archives** in SMT.
    - Snapshot and Archive Statistics tool

      Use this tool to monitor snapshot and PI archive activity and usage statistics on connected PI Servers. This tool is one of the most important gauges available to the PI System manager. If you periodically review these statistics, you can solve a system or data issue before it becomes a large problem. Many of the statistics, such as Overflow Data Record Count are informational; others are valuable for predictive maintenance. To open the tool, select **Operation** > **Snapshot and Archive Statistics** in SMT.
- PI System command-line utilities

  You can perform most PI archive management tasks with the PI SMT Archives tool. Some tasks, however, such as performing an archive walk or backfilling data require you to use command-line utilities.

  For information on using command-line tools for managing archives, see the *PI Server Reference Guide*.
    - **`piartool`**

      Use this utility for most archive management tasks such as creating, registering, and unregistering archives, forcing archive shifts, and listing archive file details.

      For a complete list of the parameters for **`piartool`**, enter `piartool ?` from the `PI\adm` directory or see "piartool utility" in the *PI Server Reference Guide*.
    - **`piarcreate`**

      Use this utility to create new archives.
    - **`piarchss`**

      Use this utility to process existing offline archives, including:

Archive files

PI archives are the files that the PI System uses to store PI data. Each archive file contains events for a time period specified by the archive start time and end time. When you run the setup program, automatic archive creation is enabled by default.

The archive receiving current data is called the *primary archive*. When the primary archive becomes full, an *archive shift* occurs and the next available archive becomes the new primary archive.

Archive files can be either *fixed* or *dynamic*. Fixed archive files are always the same size, regardless of how much data they contain. Dynamic archive files grow in size as they receive data. Archive files range in size from 1 MB to 2 TB.

Note:

As of version 3.4.375, fixed archives will become dynamic whenever they are full, if there is sufficient disk space. A fixed archive that has become dynamic will remain shiftable. When the changed archive becomes primary, it will convert back to fixed-sized. The allocated disk space for archives will not be modified. For details, see If fixed archives are full.

See Manage PI archives for tasks related to archive management.

Topics in this section

- Primary archive files
- Archive records
- Index records
- Fixed and dynamic archives

Primary archive files

The primary archive is the archive file that covers the current time range. For the duration of time that its state is the primary archive, it has a defined start time but no defined end time. The end time is always assumed to be now.

The end time for the primary archive is redefined when an archive shift occurs. An archive shift is the process of replacing the primary archive with a new or cleared archive. The time of the shift becomes the end time and that archive is no longer the primary archive.

If registered, an empty archive is selected to be the new primary archive. If no empty archive is registered, then the oldest archive becomes the primary archive and its existing data is overwritten.

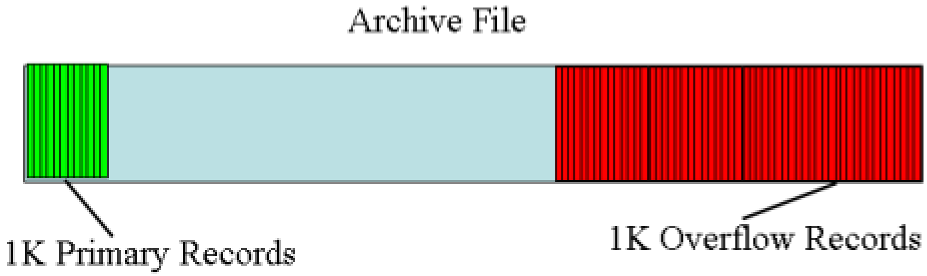
You may also set up automatic archive creation using the Archive_AutoArchiveFileRoot and the Archive_AutoArchiveFileSize tuning parameters. If these are enabled, PI Archive Subsystem creates a new archive with the same characteristics as the current primary archive with the specified file prefix and in the specified path and with the specified size.

See Manage automatic archive creation.

The PI Server ensures that some space is still available at the time of the shift so that out-of-order events can still be stored in the archive after it is no longer the primary archive.

Archive records

PI System stores archive events as *data records*. Data records are either *primary records* or *overflow records*. Each point in the PI point database has one primary record allocated in every archive file. Primary records are stored at the very beginning of the archive file. When the primary record for a point fills up, the data for that event goes to an *overflow* record in the archive file. Overflow records start at the end of the archive file and are filled backwards. Each record is one kilobyte.



A point can have as many overflow records as needed. Points that receive events at a slow rate might never need to allocate an overflow record, whereas points that receive events at a fast rate might need to allocate many overflow records. The PI System uses *index records* to keep track of multiple overflow data records for a point.

When the archive allocates a new overflow record for a point, it immediately writes to disk both the new record and any existing records that reference the new record.

Index records

Index records are records that index the overflow data of a point by time. After one overflow record for a point is full, PI Server creates an index record for that point and a new overflow record. An index record can hold between 150 and 160 record points. When the index record is full, PI Server creates a second index record and these index records are linked. Archive data retrieval for points with chains of index records are marginally slower than for points with zero or one index records.

Fixed and dynamic archives

PI archives can be either fixed or dynamic. Fixed archive files are always the same size, regardless of how much data they contain. Dynamic archive files grow in size as they get data. By default, PI archives are fixed.

Topics in this section

- Fixed archives
- Dynamic archives
- Automatic conversion of archives from fixed to dynamic
- Advantages of using fixed archives
- If fixed archives are full

Fixed archives

Use fixed archives for all normal operations. The default archives that are installed with a PI Server are fixed archives. All the disk space for a fixed archive is allocated at creation time. An empty archive and a full archive take the same amount of disk space. A fixed archive may or may not participate in archive shifts, depending on the point count-to-archive size ratio and the state of the shift and write flags. You can add new points to a fixed archive if it is the primary archive.

Note:

As of version 3.4.375, fixed archives will become dynamic whenever they are full, if there is sufficient disk space. A fixed archive that has become dynamic will remain shiftable. When the changed archive becomes primary, it will convert back to fixed-sized. The allocated disk space for archives will not be modified. For details, see If fixed archives are full.

Dynamic archives

In general, in PI Server 3.4.375 and later, it is no longer necessary to create dynamic archives since auto-dynamic archive conversion was introduced. The feature converts non-primary archives that become full to dynamic archives in order to incorporate backfilled data when necessary.

As of PI Server 2012, you can backfill existing archives with data from new PI points. See Manage backfilling of data.

Dynamic archives are useful as non-primary archives for receiving backfilled data since they do not need pre-allocated space and can grow to incorporate the historical data.

After non-primary fixed archives are converted to dynamic archives and data is backfilled, determine if you want to maintain these archives as dynamic archives or convert them back to fixed archives using**piarchss**, the Offline Archive Utility.

The size of dynamic archives is flexible, enabling disk space to be allocated only as needed. Dynamic archives cover a specific time range. They do not contain unallocated space waiting to be used for overflow records; the file grows as overflow records are added, up to the specified maximum size, or maxsize, but no larger than 2 terabytes. You must specify a maximum archive size when you create a dynamic archive.

Automatic conversion of archives from fixed to dynamic

As of PI Server 2012, auto-dynamic archive conversion is the default behavior. Fixed archives become dynamic whenever they are full, if there is sufficient disk space. A fixed archive that has become dynamic remains shiftable. The allocated disk space for archives will not be modified. Auto-dynamic archives *preserve their shift flag* and turn into fixed size archives again if they become primary archives.

PI Server version	Auto-dynamic archive conversion behavior
Prior to 3.4.375	If you upgrade from a version of the PI Server prior to 3.4.375 to a version prior to 2012, the Archive_Enable_AutoDynamic parameter may not be enabled.

PI Server version	Auto-dynamic archive conversion behavior
PI Server PR1 SP1a (3.4.375.80)	Beginning with PI Server PR1 SP1a (3.4.375.59), auto-dynamic archives *preserve their shift flag* and turn into fixed size archives again if they become primary archives. The Archive_Enable_AutoDynamic tuning parameter is enabled by default.
PI Server 2012	The Archive_Enable_AutoDynamic tuning parameter is deprecated. Automatic conversion of full fixed archives to dynamic archives is the default behavior.

Enable AutoDynamic for PI Server versions prior to 3.4

Enable the automatic conversion of full fixed archives to dynamic archives for PI Server versions prior to 3.4.375.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Tuning Parameters** and click the **Archive** tab.
2. Select the **Archive_Enable_AutoDynamic** parameter and set the **Value** to 1.

Advantages of using fixed archives

Use fixed archives over dynamic archives for the following benefits:

- Less maintenance

  Once you determine how much data you want online and then create and register a fixed number of archives, there is no risk of running into disk space errors because old archives are overwritten. Once all archives become full, the oldest archive with data becomes the new primary archive, and old data will be overwritten as new events enter this archive.

- Automatic archive creation

  If you do not want a fixed number of archives, you can configure the Archive Subsystem to create new archives as needed. If the Archive_AutoArchiveFileRoot parameter is enabled, once the primary archive becomes full, the Archive Subsystem will create an empty fixed archive in the location of your choice to shift into. Note that with this feature enabled, you run the risk of encountering disk space errors because new fixed archives are automatically created.

  See Manage automatic archive creation for information on the Archive_AutoArchiveFileRoot parameter.

- Better I/O performance

  Fixed archives can be created from non-fragmented disk space. Since dynamic archives grow as needed they are likely to become significantly fragmented over time. creating inefficiencies that can be avoided with fixed, pre-allocated files.

- Easier to backup and reprocess

  Smaller archives are faster to reprocess than larger archives. Also, with incremental backups, it is potentially much faster to backup several smaller, fixed archives rather than one large, dynamic archive. With smaller, fixed archives, only those that have been modified since the last backup are copied. If a large, dynamic archive has only been modified for a small segment of time, the entire archive still has to be copied in an incremental backup. For these reasons, OSIsoft recommends that at least the primary archive be kept as a fixed archive.

If fixed archives are full

As of version 3.4.375, fixed archives will become dynamic whenever they are full, if there is sufficient disk space. A fixed archive that has become dynamic will remain shiftable. When the changed archive becomes primary, it will convert back to fixed-sized. The allocated disk space for archives will not be modified.

Prior to version 3.4.375, it is possible for a fixed archive to fill completely. Once an archive is full, incoming data events for that time range have nowhere to go. This can occur if a much larger than usual quantity of data is sent to the primary archive and confuses the attempt to predict the shift based on event rates. It can also happen when old data is sent to an archive that is already full.

If such a condition occurs, PI Archive Subsystem attempts to convert this fixed archive into a dynamic archive to prevent data loss. This means that the file size is extended and additional overflow records are appended at the end. If the archive that was filled is the primary, an archive shift is also scheduled immediately. This allows all the incoming events to be stored so there is no data loss. The fixed archive that was converted is marked as non-shiftable, like all non-empty dynamic archives. As a result, some additional management may be required. For example, you may need to register a new empty archive for the next shift or reprocess the converted archive to fit a certain size. Converted archives are clearly marked as a different type on archive displays such as **`piartool -al`**.

Manage PI archives

PI Server stores data in archive files. You can perform most PI archive management tasks with the PI SMT Archives tool. Some tasks, however, such as performing an archive walk or backfilling data require you to use command-line utilities. For information on using command-line tools for managing archives, see the *PI Server Reference Guide*.

The topics in this section explain how to perform tasks in PI SMT wherever possible and provide command-line instructions where necessary.

Open the Archives tool in PI SMT

The Archives tool displays a list of registered archives for each connected PI Server. The archive list contains columns that describe the status and properties of each archive. Toolbar functions and a context menu allow you to monitor and manage archive use.

To open the Archives tool, click **Start** > **PI System Management Tools** > **Operation** > **Archives.**

Create new archives with PI SMT

Use the Archives tool in PI SMT to create archives.

Note:

You can also use the **`piarcreate`** command-line utility to create archives, as described in the *PI Server Reference Guide*.

Topics in this section

- Create a new archive
- Create multiple archives for backfilling
- Archive names

Create a new archive

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.**
2. Right-click an archive file from the target PI Server and choose **Create New.**
3. In the Create a New Archive dialog box, select **Single archive.**
4. Click the browse button to change the archive path, if desired.

   You can store an archive in any local or network directory accessible by PI Server. Local storage with other archives provides the most reliable option for managing archives.
5. Enter a name for the file in **Archive name**, or accept the chronologically-numbered default name.

   If the text field is yellow, then the archive name is already in use by another file, possibly an unregistered archive. You may want to cancel the procedure and use the existing archive, if empty.
6. Select a source option to create the archive:
   - Select **Clone primary archive fixed size** to create a new, empty archive of fixed type, based on the size of the current primary archive. A dynamic archive may also be used, but is not recommended. This option is not available if the current primary archive is not of fixed type.
   - Select **Create archive larger than current primary** to create a new, empty archive larger than the current primary archive, and use the accompanying field to specify the desired size in megabytes (MB). The size must be equal to or greater than the size of the current primary archive, up to a maximum of 2 TB for a PI Server 3.4 and later, and 1 GB for a PI Server prior to 3.4.
   - Select **Create archive with fixed start and end time** to create a new, empty archive to be used only for a specified time period.
7. Choose the **Type** of archive to create: a fixed archive with size equal to the current primary archive or a dynamic archive.
8. Enter **Start time** and **End time** in the provided fields using PI time format.

   A yellow background indicates that a timestamp was entered in a format that is not recognized by PI Server.

Note:

Start and end times must not overlap an existing archive.

9. Click **OK**.

   The Archives tool attempts to register the newly created archive automatically. If the registration succeeds, the new archive appears in the archive list.

Create multiple archives for backfilling

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.**
2. Right-click an archive file from the target PI Server and choose **Create New.**
3. In the Create a New Archive dialog box, select **Multiple archives for backfilling.**
4. Click the browse button to change the archive path, if desired.
5. Enter a prefix for the file in **Archive name**, or accept the default prefix.

   The start time and end time will be automatically appended to the archive name depending on the archives being created.
6. Define the **Maximum archive duration** for each new archive file.
7. Enter **Start time** and **End time** for the new archive files using PI time format.

   Note:

   Start and end times must not overlap an existing archive.
8. Click **OK**.

   The Archives tool registers the newly created archives automatically.

Archive names

Use a naming convention for your PI archives that is valid for the underlying operating system and ensure that the file location has sufficient disk space. There are no other limitations to name PI archives.

The default archive file names are `piarch.`*`xxx`*, where *`xxx`* is `001, 002, 003`, and so on.

The associated annotation file has the same full path name as its archive file with `.ann` appended. For example, the annotation file for the `piarch.001` archive file would be `piarch.001.ann`.

Manage archive size

Topics in this section

- Archive size guidelines
- Archive size and point count limits
- Archive size and shift frequency

- View the snapshot status to see current number of points
- Estimate archive utilization
- Set maximum file size or maximum number of points for a dynamic archive

Archive size guidelines

Archive files can have a fixed size or can be configured to grow dynamically as they receive more data. The archive size affects backups, backfilling, frequency of shifting, and total number of points allowed.

- If you are backfilling data, see "Manage Backfilling of Data" in the *PI Server System Management Guide*.
- Your archives must be sized with at least 2KB for each point in the system, and an archive size cannot exceed 2 TB.However, if your PI Server will have 5,000 points or less then you can safely use the default value (currently 256MB).
- If you have more than 50,000 points, run the 64-bit PI Server on 64-bit Windows OS and set the archive size to 4-8 KB x the total number of points, with the following consideration on memory resources.

Select a size so that at least two archive files can fit in the Windows File System Cache (FSC). At most times, the PI server writes to and/or reads from the 2-3 most recent archive files. The FSC is capped at approximately 1GB on 32-bit systems, but can use all of the RAM on 64-bit systems. Therefore, 256MB for 32-bit systems, and (RAM / 3) for 64-bit is a safe upper limit for archive files.

Note:

Many people size their archives based on a size that is convenient to use with their desired backup media. As a rule of thumb, your Snapshot Event Queue should be set to half of the archive size.

Archive size and point count limits

It is important to note that the archive size limits the number of points that may be created. No more than half of the archive records of a fixed archive can be primary records. If the allotment of primary records is used and you try to create an additional point, although the primary archive is not full, you will get the error:

```
PI_AR_ARCHIVEFULL = -11053, /* no more available records in this archive */
```

To resolve this, force the archives to shift into a larger archive before creating additional points:

- See Force an archive shift (with PI SMT Archive tool)
- Use the command **`piartool -fs`** as described in the *PI Server Reference Guide*.

Archive size and shift frequency

The larger the PI Server archive files, the less frequently archive shifts will occur. To decide what archive size is optimal for your system, consider that the following factors will determine shift frequency:

- Backup device
- Available disk space
- Average incoming data rate

View the snapshot status to see current number of points

To view the current point count, enter `piartool -ss` to list the snapshot status. The maximum number of archive points should be at least the current number and greater if you expect to create new points.

Estimate archive utilization

Examine output from the **`piartool -al`** command to determine the utilization of your archives.

Procedure

1. At a command prompt, go to the `PI\adm` directory.
2. Type the following command:

   ```
   piartool -al
   ```

   Output from the command lists archives in reverse chronological order (archives with newer data before archives with older data). Next to the Used label for each archive file, the output shows the percentage of that archive that is used. For fixed sized archives, this is the percentage of the total available records in use and is an indication of how much space is available in the archive for data. For dynamic archives this is the percentage of primary records in use and is an indication of how close one is to the maximum number of points that can be created for that archive.

Set maximum file size or maximum number of points for a dynamic archive

You can set a limit on the file size (maxsize) or the number of points (maxpoints) of a dynamic archive. To do this, you must use either the **`piarcreate`** or **`piartool`** commands (you cannot do this using the PI SMT Archive Editor tool).

- Use **`piartool`** to create and register a dynamic archive file and set the `maxpoints` and `maxsize` values using the following syntax:

  ```
  piartool -acd <path> <maxpoints> <maxsize>
  ```

- Use **`piarcreate`** to create a new archive file and set the `maxpoints` and maxsize values using the following syntax:

```
piarcreate –d <path> <maxpoints> <maxsize>
```

For example:

```
C:\PI\adm>piarcreate -d D:\PI\dat\piarch.115 2000 20000
Attempting to create a 2000 record, dynamic archive: D:\PI\dat\piarch.115 with
a maximum size of 20000 Mbytes.
Initializing archive file: D:\PI\dat\piarch.115
Archive D:\PI\dat\piarch.115 is prepared to be registered
```

- Use **`piarcreate`** to change the maxpoints parameter:

```
G:\pi\adm>piarcreate -?
Usage: piarcreate -v
piarcreate -d path maxpoints maxsize(Mb)
piarcreate path size(Mb)
```

The following listing is for a 2048 MB archive; the maximum number of configurable points for the archive is 1,048,576 (half the total number of records).

```
D:\PI\adm>piartool -al
Archive shift prediction:
    Shift Time: 5-Oct-05 19:42:01
  Target Archive: e:\pi\arc\piarch-2GB.1
Archive[0]: e:\pi\arc\piarch-2GB.3 (Used: 53.4%)
    PIarcfilehead[$Workfile: piarfile.cxx $ $Revision: 101 $]::
     Version: 7 Path: e:\pi\arc\piarch-2GB.3
     State: 4 Type: 0 Write Flag: 1 Shift Flag: 1
     Record Size: 1024 Count: 2097152 Add Rate/Hour: 154207.3
     Offsets: Primary: 253063/1048576 Overflow: 1231270/2097152
     Annotations: 1/65535 Annotation File Size: 2064
        Start Time: 5-Oct-05 06:11:09
         End Time: Current Time
       Backup Time: Never
      Last Modified: 5-Oct-05 13:26:21
```

Manage archive registration

The PI Server archive registry contains the name, location, size, count of records, and record size for each archive file. This information is stored in the binary file, `PI\dat\piarstat.dat`. In order to for an archive to be shiftable, it must be registered. Once an archive is registered, it is available to the system and participates in shifts and storage and retrieval of event data.

Topics in this section

- Archive registration
- Register archives
- Unregister archives
- Display an unregistered archive
- Bulk archive registration

Archive registration

The PI Server archive registry contains the name, location, size, count of records, and record size for each archive file. This data is stored in the binary file, `PI\dat\piarstat.dat`. In order to for an archive to be shiftable, it must be registered. Once an archive is registered, it is available to the system and participates in shifts and storage and retrieval of event data. See Manage archive registration.

Register archives

If you want PI Server to recognize a file as an archive file, you must register the file. By registering an archive file, you tell PI Server that the file exists and is available for PI Archive Subsystem to use for storing data. PI Server cannot access data in unregistered archives, nor can the PI client applications. To register an existing archive for use with a particular server, use the PI SMT Archives tool.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.** The Archives tool lists all the archives registered on the selected server. Unregistered archive files do not appear in the list.
2. Right-click an archive file from the target PI Server and choose **Register Archive.**
3. Browse to the archive file you want to register and click **Open.** The list of archives is refreshed.

   To register multiple archives at once, press Ctrl-Click or Shift-Click when you select the files and click **OK.**

   > **Tip:**
   >
   > If you sort files by size, press Shift-Click to select all archives and click **OK**. All selected archives are registered with the PI Server.

Unregister archives

If the PI Server is not on the local machine, this task requires the PI Server's piadmin password.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.**
2. Right-click an archive file from the target PI Server and choose **Unregister Archive.**
3. Click **Yes** to unregister the archive, or **No** to cancel the operation.

Display an unregistered archive

Use the Archives tool to display an unregistered archive. After displaying an unregistered archive, you can register and use that archive.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.**
2. Right-click an archive file from the target PI Server and choose **Display Unregistered Archive.**
3. Browse to the correct directory and select the unregistered archive file on the server, and click **Open.**

   PI SMT adds the unregistered archive to the list of archives with an unregistered archive icon and **State** set to **Dismounted.**

Bulk archive registration

Use the **`piartool`** utility to register or unregister archives in bulk. The **`piartool`** utility allows you to use the wildcard *** and question mark *?* symbols to perform bulk operations. The symbol *** matches all possibilities with any number of characters. The symbol *?* matches a single character and may be used any number of times.

Topics in this section

- Register archives in bulk
- Unregister archives in bulk

Register archives in bulk

Procedure

- To register archives, you use the `piartool -ar` command. The syntax is:

  ```
  piartool -ar path
  ```

  The specified *path* must be a complete, not relative, path of an existing archive file.

  The following command registers a single archive called `piarch.006` in the `PI\dat` directory on the `D` drive:

  ```
  piartool -ar D:\PI\dat\piarch.006
  ```

  You can use the wildcard characters *** and *?* to register archives in bulk. The symbol *** matches all possibilities with any number of characters. The symbol *?* matches a single character and may be used any number of times.

  For example, the following command registers all archive files in the `PI\dat` directory that begin with the `piarch.0` prefix:

  ```
  piartool -ar D:\PI\dat\piarch.0*
  ```

Unregister archives in bulk

Procedure

- To unregister an archive, use the `piartool -au` command. The syntax is:

  ```
  piartool -au path
  ```

  where *path* specifies a complete, not relative, pathname.

  For example, the following command unregisters the archive called `piarch.006` in the `PI\dat` directory on the `D` drive:

  ```
  piartool -au D:\PI\dat\piarch.006
  ```

  You can use the wildcard characters *** and *?* to register archives in bulk. The symbol *** matches all possibilities with any number of characters. The symbol *?* matches a single character and may be used any number of times.

  For example, the following command unregisters all archive files that begin with the `piarch.0` prefix and are located in the `PI\dat` directory:

  ```
  piartool -au D:\PI\dat\piarch.0*
  ```

Manage archive shifts

Topics in this section

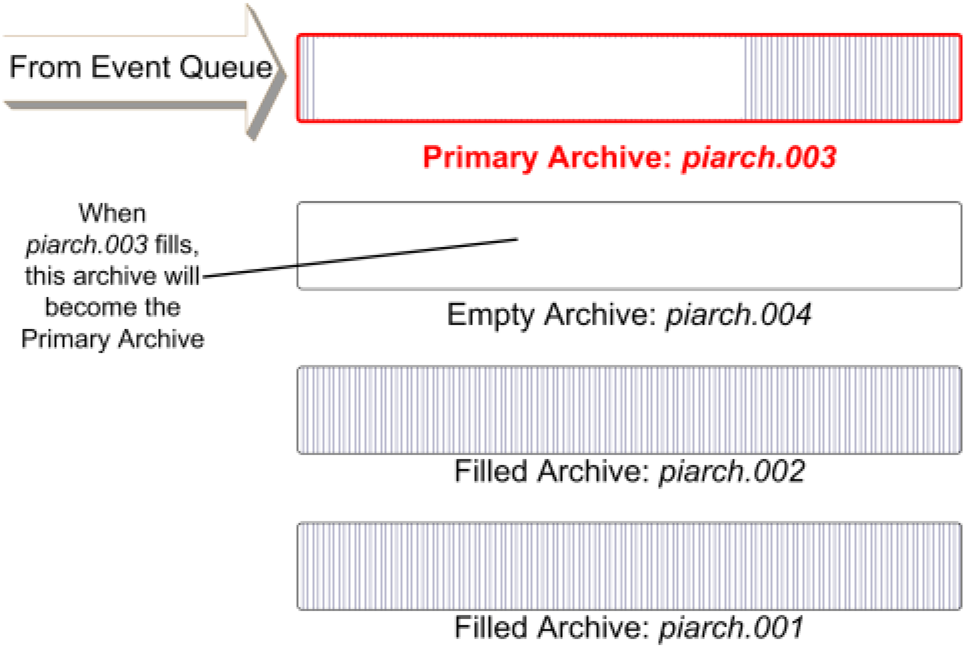
- About archive shifts
- View the next predicted archive shift time
- Change the shift flag for an archive
- Force an archive shift (with PI SMT Archive tool)
- Failed archive shifts

About archive shifts

The primary archive is the archive that is receiving current data. When the primary archive becomes full, an archive shift occurs and the next available archive becomes the new primary archive.

PI Server performs an archive shift before the primary archive is completely full. As a result, the archive contains some extra space so that, if necessary, you may add data later.

For an archive file to be eligible to be the new primary archive, it must be registered, shiftable, writable, and large enough to handle the current size of the PI point database. If an archive does not meet these criteria, it is skipped over during an archive shift. By making an archive non-shiftable or read only, an archive may be excluded from the shift cycle.


From Event Queue
Primary Archive: *piarch.003*
When *piarch.003* fills, this archive will become the Primary Archive
Empty Archive: *piarch.004*
Filled Archive: *piarch.002*
Filled Archive: *piarch.001*

If no empty archives are available for an archive shift, PI Server uses the oldest available filled archive as the new primary archive and, in the process, overwrites the data in the old archive. For example, in the preceding illustration there are no empty registered archives left on the server after the shift from `piarch.003` to `piarch.004`. If the PI System administrator does not create new archives on this PI Server, filled archive: `piarch.001` becomes the next primary archive and the PI Server overwrites the existing data in that archive.

During an archive shift you are not allowed to add, edit, or delete points.

View the next predicted archive shift time

The PI SMT Archives tool (**Operation** > **Archives**) has a shift prediction column that predicts the time for the next archive shift, based on the average number of archive records consumed per hour, plus the rate of consumption. If the primary archive is less than 20 percent full, the prediction is based on the previous archive rates.

Change the shift flag for an archive

Procedure

- You can change the shift flag for an archive.
    - To make the archive non-shiftable, right-click the archive in list view and choose **Make Non-shiftable**.
    - To make the archive shiftable, right-click the archive in list view and choose **Make Shiftable**.

Force an archive shift (with PI SMT Archive tool)

During normal operations, you should not force an immediate archive shift. However, it may be useful to force an archive to shift while testing your system or when performing advanced archive management.

Use this procedure to force a selected server to shift from one archive to another.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives**.
2. Right-click the server's primary archive in the list and select **Force Shift**.

   A dialog box prompts for confirmation before forcing the shift.

Failed archive shifts

If an archive shift fails for any reason, all further shifts are disabled and a message is sent to the log. When the cause of the failure is resolved, restart PI Archive Subsystem to enable archive shifts. If the cause of failure was a lack of an available archive to shift into, then register a new empty archive to automatically resolve the situation and enable a shift into the new archive.

Failed shifts do not cause any data loss since the archive goes into read-only mode and prevents data from being written to the archive file. All incoming data is queued in the event queue by PI Snapshot Subsystem.

Maintain annotations

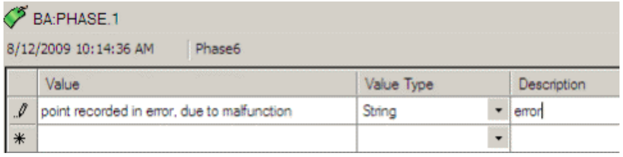
Use the PI SMT Archive Editor tool to view, add, and edit annotations. Annotations allow you to associate arbitrary information, such as text comments and other binary data, with a PI archive value.

Use the PI Annotations Editor in the PI SMT Archive Editor to view, edit, insert, and delete annotations to PI point values. Annotations can include comments, notes, supplementary values with specified data types, and even files.

There are two modes you can use to maintain annotations.

- **Standard/Default mode**

  Provides a table format that can include alternate values with assigned data types. This mode is best if the annotation data is likely to be structured, read programmatically, or exported for use by another application.

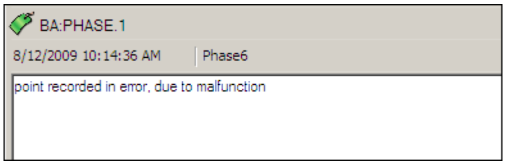
  BA:PHASE.1

  8/12/2009 10:14:36 AM | Phase6

	Value	Value Type	Description
	point recorded in error, due to malfunction	String	error
*			

- **String/VARIANT mode**

  Stores annotation data as an unspecified **VARIANT** data type. This mode is best for simple string annotations, annotations that do not require structured data, and to conform with legacy annotations from earlier versions of PI Server.

  BA:PHASE.1

  8/12/2009 10:14:36 AM | Phase6

  point recorded in error, due to malfunction

Topics in this section

- Annotations
- Maintain PI annotations with Archive Editor
- Add or edit annotations
- Import a file to an annotation

Annotations

Each archive file has a single associated annotation file, with an .ann extension. The annotation file is created if it does not exist. It is important the archive and annotation files are stored together, especially when a backed up archive file is restored.

> **Note:**
>
> Any operation on an annotation translates into an actual I/O, bypassing archive caching. Annotated events are much less efficient than non-annotated events.

Use the PI SMT Archive Editor tool to view, add, and edit annotations. Annotations allow you to associate arbitrary information, such as text comments and other binary data, with a PI archive value.

Every value in the snapshot or the archive may be annotated. Annotated events always bypass compression. An annotation can be of any binary data type. The size of an annotation is controlled by the Snapshot_AnnotationSizeLimit tuning parameter.

(Prior to PI Server 2012 annotation size was controlled by the Snapshot_EventQueuePageSize tuning parameter. This parameter was deprecated for 2012.).

Maintain PI annotations with Archive Editor

Procedure

1. Select **Start** > **PI System Management Tools** > **Data** > **Archive Editor**.
2. Select an event in the archived events list and click the **Annotations** button 📎, or right-click the value and select **Annotations**.

   The **PI Annotations Editor** opens in **Standard** mode by default. To use **String/Variant** mode, select the **Use String Annotations?** check box in the search panel.
3. Right-click an annotation row to select from these options (the **PI Annotations Editor** toolbar also provides these options):
   - **Delete**

     Delete the selected event.
   - **Import**

     Import a file object into an annotation row.
   - **Export**

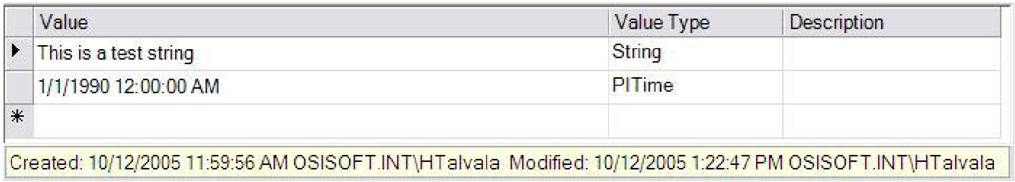
     Export a file object from an annotation row.
   - **Show Details**

     Show detailed information about the annotation record in the status bar.

Add or edit annotations

Procedure

1. Select an event in the archived events list and click the **Annotations** button 📎, or right-click the value and select **Annotations**.
2. In the PI Annotations Maintenance window, modify the following information, or enter new rows containing:
   a. Point values or any other data that requires a specified data type in the Value column. If your annotation consists only of a string, enter it in the Value column.
   b. Data types to match corresponding values in the Value Type column. Value Type is set automatically, and should be changed only if it is incorrect.



	Value	Value Type	Description
▸	This is a test string	String	
	1/1/1990 12:00:00 AM	PITime	
*			

Created: 10/12/2005 11:59:56 AM OSISOFT.INT\HTalvala Modified: 10/12/2005 1:22:47 PM OSISOFT.INT\HTalvala

If you need to change a Value Type, select one of the following.

- String (default type)
- Byte, Short, Long
- Single, Double
- Boolean
- PITime, DateTime

Other data types displayed are for internal use, and cannot be used for annotations.

c. Related information and secondary annotations in string format in the Description column.

3. Enter as many rows as necessary and click **Save**.

Import a file to an annotation

Procedure

1. Select an event in the archived events list and click the **Annotations** button, or right-click the value and select **Annotations**.
2. In the PI Annotations Maintenance window, set the Value Type to **File**, and click in the **Value** cell.
3. Click the **Import** button.
4. Select a file and click **Open**.
5. Click **Save**.

Manage automatic archive creation

In PI Server 3.4.375.67 and later, you can configure the PI Server so that it generates new archives automatically when shifting.

Note:

OSIsoft recommends using MCN Health Monitor to monitor disk space on the PI Server and using PI Notifications to warn when there is low free space on the PI Server.

When relying solely on automatic archive creation, you expose yourself to more points of failure. It might be easier to keep track of available disk space than to keep track of what archives are mounted, what they contain, and whether or not they can be allowed to be overwritten. If you are not monitoring the disk space on your PI Server, your hard disk could fill up and PI could either overwrite old archives or data could back up in the event queue without being archived.

Topics in this section

- Automatic archive creation
- Automatic archive creation tuning parameters
- Configure automatic archive creation
- Disable automatic archive creation

Automatic archive creation

In PI Server 3.4 and later, you can set the Archive_AutoArchiveFileRoot parameter so that the PI Server generates new archives automatically when shifting.

The Archive_AutoArchiveFileSize tuning parameter allows you to define the size for automatically created archives. If the parameter is not present, by default the new archive will be the same size of the current primary archive.

See Manage automatic archive creation.

Automatic archive creation tuning parameters

The following tuning parameters govern the behavior of automatic archive creation.

Tuning Parameter	Applicable Versions	Description
Archive_AutoArchiveFileRoot	PI Server 3.4.375.67 and later	This parameter enables automatic archive creation. If present, this parameter defines the path and file name prefix to be used for new archives. For example, a setting of "C:\PI\arc \auto_" defines that newly created archives should be placed in the `C:\PI\arc` folder with a file name prefix of "auto_" . The tuning parameters Archive_AutoArchiveFileFormat and Archive_AutoArchiveFileExt govern how the remainder of the archive's name is formed.
Archive_AutoArchiveFileExt	PI Server 3.4.375.67 and later	This parameter specifies the file extension to be added to the name of the archive file to be created. The default value of this parameter is ".arc" .

Tuning Parameter	Applicable Versions	Description

Archive_AutoArchiveFileFormat	PI Server 3.4.375.67 and later	This parameter governs the generated portion of the file name to be given to the archive file to be created. Possible values for this parameter and the corresponding file names are:
		`0 - _D_Mon_YYYY_H_M_S.ext`
		`1 - prefix_YYYY-MM-DD_HH-MM-SS.ext`
		`2 - prefix_UTCSECONDS.ext`
		Where *prefix* is the file name prefix specified in the Archive_AutoArchiveFileRoot parameter and `.ext` is the file extension specified in the Archive_AutoArchiveFileExt parameter.

Archive_AutoArchiveFileSize	PI Server 2012 and later	Default 0. If set to 0, the size of the new primary archive will always be the same as the current primary archive. If set to 1, specifies the size of the new primary archive when an automatic archive create shift occurs. You may change the size of new archives created via automatic archive creation by tuning this parameter.If you tune it smaller than the current primary archive, this size will only take place if the requested size is large enough to satisfy the primary records for points currently in use.

Tuning Parameter	Applicable Versions	Description
Archive_OverwriteDataOnAutoShiftFailure	Enabled by default in PI Server 3.4.380. Disabled by default in PI Server 3.4.380SP1 and later, it is disabled by default.	This parameter determines how the PI Archive subsystem behaves if it encounters an error (such as insufficient disk space or disk error) while attempting to create a new archive. When this parameter is enabled (set to "1") and an error is detected while attempting to create a new archive file (such as insufficient disk space or disk error), it will shift into the oldest filled archive on the hard drive and begin overwriting older data. When this parameter is disabled (set to "0"), and an error is detected while attempting to create a new archive file , archiving is disabled and data will queue in the event queue file. You can later detach the event queue file and reprocess it into an archive, but this is time consuming and can be disruptive.

Configure automatic archive creation

Procedure

1. Add the Archive_AutoArchiveFileRoot parameter to the PI Tuning Parameters and set the **Value** field.
   a. Select **Start** > **PI System Management Tools** > **Operation** > **Tuning Parameters**.
   b. Click the **Archive** tab.
   c. Right-click an existing parameter and choose **New**.
   d. Add the **Archive_AutoArchiveFileRoot** parameter and set the **Value** to a valid path and file name prefix for the new archives, such as `C:\pi\arc\piauto_`.
2. If running PI Server 3.4.380 or later, set the Archive_OverwriteDataOnAutoShiftFailure parameter.
   a. Right-click an existing parameter and choose **New**.
   b. Add the Archive_OverwriteDataOnAutoShiftFailure parameter and set the **Value** to 1 to enable or 0 to disable, depending on your preference.
3. Add the **Archive_AutoArchiveFileFormat** parameter and set the **Value** field.
   a. Right-click an existing parameter and choose **New**.
   b. Add the **Archive_AutoArchiveFileFormat** parameter and set the **Value** to 0, 1 or 2 to determine the format of the filename for new archives:

    - 0 - *_D_*Mon*_YYYY_H_M_S.ext*
    - 1 - *prefix_YYYY-MM-DD_HH-MM-SS.ext*
    - 2 - *prefix*_UTCSECONDS.*ext*

4. Add the Archive_AutoArchiveFileExt parameter and set the **Value** field.
    a. Right-click an existing parameter and choose **New.**
    b. Add the parameter Archive_AutoArchiveFileExt set the **Value** to the extension to use for the filename for new archives.
5. Optionally, add the Archive_AutoArchiveFileSize parameter and set the **Value** field. If the parameter is not defined, by default the new archive will be the same size of the current primary archive.
6. Make sure your primary archive is a fixed size primary archive.

    Note:

    Prior to PI Server 2012, auto-archive creation will not work with a dynamic primary archive. The size of automatically created archives will always be taken from the size of that initial primary archive.

    a. Select **Start** > **PI System Management Tools.**
    b. Look at the **Type** for the primary archive.
7. If running PI Server prior to 3.4.380, make sure you have one valid, shiftable, empty target archive available, even though it will not be used if automatic archives can be created. This archive can be dynamic or fixed, unlike the primary archive. The purpose for this archive is to ensure you always have an archive to shift to if for some reason the automatic archive creation fails.
8. To test if the parameters are properly set, you can force an archive shift .
    a. Select **Start** > **PI System Management Tools** > **Operation** > **Archives.**
    b. Right-click the server's primary archive in the list and select **Force Shift.**

        A dialog box prompts for confirmation before forcing the shift.

Disable automatic archive creation

Procedure

- Disable automatic archive creation by turning off the Archive_AutoArchiveFileRoot parameter:
    a. Select **Start** > **PI System Management Tools** > **Operation** > **Tuning Parameters** and click the **Archive** tab.
    b. Select the **Archive_AutoArchiveFileRoot** timeout parameter and clear the **Value** field (leave it blank).

Manage archive gaps

An archive gap is a range of time when no archive file is registered. If an event is sent to the archive and no archive file is registered within the appropriate time range, the event is discarded and an error is logged. If data retrieval is attempted for a time range that overlaps with a gap, the returned data includes a digital state Arc Offline that indicates the beginning of the gap. This prevents values from being interpolated when data is missing.

PI archive files meet chronologically end-to-end, accounting for all of history with no gaps and no overlaps. If an archive gap occurs, it is important to identify and fix it as soon as possible. You can use PI SMT or the **`pidiag`** tool to fix archive gaps.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Archives**.

   All the archives registered on the selected PI Server are listed. Any archive gaps are labeled and highlighted in red.

2. Right-click on the line displaying the archive gap and select **Create New**.

   The Create New Archive dialog box appears. The dialog box is already populated with the correct start and end times to fill the archive gap.

3. Click **OK**.

   The new archive is created and registered and an archive gap no longer appears in the archive list.

Configure number of archives monitored for readwrite errors

By default, PI Archive Subsystem stops writing time-series data to archive files after detecting an error when reading or writing data. This feature protects archive data.

Procedure

- The Archive_DisableArchivingOnIOError tuning parameter enables this feature. With this parameter enabled, the Archive_DisableArchivingOnIOErrorRange parameter sets the number of archives monitored.

  PI Archive Subsystem will monitor this number of files for read-write errors, starting with the primary archive. For example, if set to 3 (default value), PI Archive Subsystem will monitor the primary archive plus the two previous archives, but not prior archives. A value of 0 indicates that PI Archive Subsystem will monitor all files.

  Note:

  If PI Archive Subsystem loses its connection to the primary archive, the subsystem does not try to reconnect to the archive. Data will flow into the event queue, but will not be stored in the archive file. You must stop and restart the subsystem in order to reconnect and store data in the archive.

Prevent archive changes

In PI Server 3.4.375 and later you can configure a time limit in number of days prior to the current time for insertion and editing of events. The snapshot rejects events with timestamps earlier than the limit. By default there is no limit, which is consistent with earlier versions of PI. PI Snapshot Subsystem must be restarted for the changes to take effect.

This section discusses the ramifications of read-only archives and explains how to configure a time limit on insertions and event edits with the EditDays tuning parameter. Using the EditDays tuning parameter is preferable to making an archive read-only which can result in data loss.

Topics in this section

- Read-only archive files
- Modify the EditDays tuning parameter
- Set archive to writable or read-only

Read-only archive files

! **Caution:**

Setting an archive to read-only can result in data loss. The preferred method for preventing archive changes is to use the EditDays tuning parameter to set a time limit for archive changes.

The **Write Flag** column of the archives list in SMT indicates the state of each registered archive. Any archive listed with the Write flag of Read-Only is subject to data loss. Archive files that have a read-only file-system attribute, or files on a read-only device (CD ROM) are mounted as read-only. Their status will show up on the `piartool -al` display as not writable.

Read-only files cannot participate in archive shifts and they cannot be modified, therefore any new events received that should have gone into that archive are discarded. This includes attempts to edit, delete or annotate events in a read-only archive.

When this occurs, the PI Archive Subsystem reports this message in the PI Server log:

```
[-11078], Target archive is not writable
```

However, no error is returned to the application writing data.

Here is an example scenario:

1. A PI administrator marks a historical archive as read-only (to prevent changes).
2. PI client user writes historical data (for example, with PI Manual Logger) that should go in the read-only archive.
3. PI Server reports an error in the message log, but discards the data.
4. PI client user is unaware the data has not been archived since no error is returned.
    - Use the EditDays tuning parameter, so that you do not need to manually mark historical archives as read-only, and data older than the value set with the EditDays parameter will not change. See Modify the EditDays tuning parameter.
    - Ensure that any PI archives that may receive event data are not marked as read-only. See Set archive to writable or read-only.

To prevent historical data from being edited and to mitigate this data loss scenario:

- Use the EditDays tuning parameter, so that you do not need to manually mark historical archives as read-only, and data older than the value set with the EditDays parameter will not change. See Modify the EditDays tuning parameter.
- Ensure that any PI archives that may receive event data are not marked as read-only. See Set archive to writable or read-only.

Note:

The EditDays parameter can be used in conjunction with read-only archives, as long as only historical archives older than EditDays are marked read-only.

Modify the EditDays tuning parameter

You can modify the value of the EditDays parameter with the number of past days where events can be modified in the Snapshot or Archive databases. A value of zero means no time check is done.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Tuning Parameters.**
2. In the **Collectives and Servers** box, select the PI Server on which you want to edit the parameter.
3. Clear the check boxes for the other PI Servers.
4. Click the **Archives** tab for the subsystem where you want to edit the tuning parameter value.
5. If necessary, add the EditDays tuning parameter to the parameter list. See Add a tuning parameter to the list.
6. Right-click the **EditDays** parameter in the list, and select **Edit.**
7. Enter a **Value** and click **OK**.
8. Stop and restart the PI Server for the changes to take effect.

Set archive to writable or read-only

Caution:

Primary archives may not be set to read only. Setting an archive to read-only can result in data loss. The preferred method for preventing archive changes is to use the EditDays tuning parameter to set a time limit for archive changes.

Procedure

- To change the protective **Write** flag for an archive, right-click the archive in list view, and:
  - Choose **Make Read-Only** to make a writable archive read only.
  - Choose **Make Writable** to make a read-only archive writable.

Export archives to a file

Procedure

1. Select an archive from the target PI Server from the list and click the **Export Archive List** button

   .
2. Select a file type to export the archives to:
   - comma separated values (`.csv`)
   - registration `.bat` file
3. In the Save Archive List As dialog box, select a location to store the archive file.
4. Click **Save**.

Move archive files

To change the location of a primary archive, you must create a new primary archive. (You cannot move a primary archive, because you cannot unregister it while the PI archive process is running.) If you are using the automatic archiving feature, you must configure the full path to the new archive directory. If automatic archiving is disabled, you must create a new registration list before moving the archives.

Procedure

1. Change the location of the primary archive:
   a. If empty archives exist in the original directory, unregister them.
   b. Move them to the new directory.
   c. Re-register them in the new directory.

   One of these archives will become the new primary archive.
2. If you are using automatic archiving, change the AutoArchive path.

   Use the Archive_AutoArchiveFileRoot tuning parameter to configure the full path to the new archive directory. The new primary archive is automatically created in the new directory.
3. Verify that there is at least one empty archive registered in the new directory. Create one if it does not exist.
4. Stop the PI Server.
5. Run **`pidiag -ar`** to point to the primary archive in the new directory.
6. Start the PI Server.
7. Force an archive shift. This creates a new, empty, primary archive in the new directory.
8. Move the secondary archives:
   a. Unregister the secondary archives.
   b. Move the secondary archives and associated annotation files to a new directory.
   c. Re-register the secondary archives.

See also

Manage automatic archive creation

Manage archive registration

Create a new archive

Force an archive shift (with PI SMT Archive tool)

Back up archives

Topics in this section

- Back up archives for PI Server versions 3.4.375 or later
- Back up archives for PI Server versions 3.4.370 or earlier

Back up archives for PI Server versions 3.4.375 or later

See Back up PI Server.

Back up archives for PI Server versions 3.4.370 or earlier

Use the Archives tool to back up an archive and its corresponding annotation file. Backups can be created only for local PI Servers, version 3.4.370 or earlier.

Procedure

1. Select the archive to back up in the list, and click the **Backup** button . Alternatively, right-click the archive that you want to back up, and select **Backup**.
2. Browse to the desired backup directory.
3. Enter a new file name or select an existing backup file.
4. Click **Save**.

Delete an archive event

OSIsoft strongly recommends that you practice deleting small amounts of data on a test system before deleting real data.

> **Note:**
>
> Always back up your data before deleting.

There is no way to undo a delete with any utility. Deleting large amounts of data may affect availability of an online PI Server. If this is necessary on a periodic basis, it is best to understand the root cause and consider alternatives, such as using appropriate exception and compression settings, increasing the available disk space, moving older archives to a second tier storage and so on, instead of deleting data in bulk.

For details about deleting larger amounts of data, see KB article 3065OSI8 at https://techsupport.osisoft.com/Troubleshooting/KB/3065OSI8 (https://techsupport.osisoft.com/Troubleshooting/KB/3065OSI8).

Procedure

1. Under **Collectives and Servers**, select the server from where you want to remove data.
2. Under **System Management Tools**, select **Data** > **Archive Editor**.
3. Right-click the value you want to delete and select **Delete**.
4. Click **Save**.

> **Note:**
>
> There is no prompt to confirm deleting values.

Delete an archive

You must first unregister an archive before you can delete it. Use the Archives tool (**Operation** > **Archives**) in PI SMT to delete the archive. Then delete the related annotation file.

Manage backfilling of data

Topics in this section

- About backfilling data
- Prepare PI Server for backfilling
- Backfill data with a piconfig script
- Backfill data into a new PI System
- Backfill existing archives with data from new PI points

About backfilling data

You can use **`piconfig`** scripts or your own custom applications to import historical data into PI Server. OSIsoft also offers various interfaces for backfilling, including the PI Relational Database (RDBMS via ODBC), the RDBMSPI Interface, and the Universal File and Stream Loader Interface (UFL). If you are planning on backfilling many points and more than few days worth of data, please contact Technical Support for recommendations before you begin.

OSIsoft recommends that you use dynamic or auto-dynamic archives for backfilling data. Backfilling into a fixed archive can result in the target archive filling before all data is backfilled. Therefore, it is generally considered a good idea to use dynamic archives for backfilling. You can reprocess the dynamic archives into fixed archives after the backfilling is complete.

> **Note:**
>
> OSIsoft recommends using MCN Health Monitor to monitor disk space on the PI Server and using PI Notifications to warn when there is low free space on the PI Server.

For PI Servers prior to 2012, in order to backfill data into PI tags that did not already exist when an older archive was created, the archive needs to be reprocessed in order to create a primary record header for the backfilling PI tags.

With PI Server 2012, reprocessing of the archives is no longer necessary because all the archives on the PI Server will be aware of any PI tags that are created.

To create multiple archives for backfilling, see Create new archives with PI SMT.

Topics in this section

- Backfilling optimization
- Backfilling large amounts of data
- Backfill on an offline PI Server

Backfilling optimization

PI Server 2012 (3.4.390) or later is strongly recommended. Significant enhancements make backfilling data a much easier and faster process, including:

- Archive reprocessing is no longer necessary.
- "Pt Created" events for newly created points no longer need to be explicitly deleted from snapshot in order to backfill with compression.
- Writing data to PI Server is significantly faster.

  OSIsoft recommends backfilling with compression to avoid archiving unnecessary data and to ensure that the backfilling is done as efficiently as possible.

OSIsoft recommends backfilling with compression to avoid archiving unnecessary data and to ensure that the backfilling is done as efficiently as possible.

For PI Server versions prior to 2012, in order to compress data during backfilling, you must first clear the snapshot for the tag, and then send the data in chronological order. To clear the snapshot, you must have PI 3.3 or later.

Evaluate the data that you are backfilling. Determine the number and configuration of new tags, the time period covered by all tags, and the approximate amount of data you need to import.

Always run a backfill test with a small amount of data first, and then do the rest of the data. This way you can verify your piconfig script and make sure that the data is importing properly.

Check the archive and snapshot statistics during the test to see how the backfilling affects the PI Server performance.

Some factors that are critical for optimizing backfilling of historical data are:

- Your point database must be accurate and complete prior to start of processing.
- Freeze changes to the point database during processing.
- Process data from all tags in discrete batches for similar time periods.
- If you backfill out-of-order data, it will not be compressed. It is most efficient if the data for each tag is in order, with or without compression.

- You cannot backfill if there are errors in the data.
- You cannot backfill with compression into tags that already contain data.

Backfilling large amounts of data

If you have large amounts of data, consider writing a custom application to do the backfilling, rather than creating comma-separated text files and backfilling with **`piconfig`**.

Backfilling large amounts of data can stress a PI Server severely. Data in files is sent to the PI Server at a very high rate, much higher than the rate of the control system. To minimize the load on the PI system:

- Always contact OSIsoft Technical Support for recommendations before preparing a large backfill.
- Always backfill data in chronological order, from oldest to newest, within each point.
- When backfilling from data files, use several smaller files rather than a single larger file (try one day at a time). You can throttle the rate at which the PI Server processes the data using the "@wait 1" line within the data file at intervals. The "1" is one second. You can increase this if you like.

Backfill on an offline PI Server

OSIsoft highly recommends, whenever possible, to do backfilling jobs on an offline PI Server to avoid excessive burden on your main production server. This also offers an opportunity to verify the backfill is successful without posing risk to your real data on the PI Server.

Prepare PI Server for backfilling

Procedure

1. Evaluate the data that you are backfilling.

   Determine the number and configuration of new tags, the time period covered by all tags, and the approximate amount of data you need to import.

2. Create the points to backfill.

   If the points correspond to active interfaces, make sure current data is not being sent to the point from the interface. One way to do this is to create the points with the Scan attribute set to 0 (zero, which is off), or set the Point Source attribute for the points to L for Lab point. You can change these later. (You can import data into existing points that already contain values, but you will not be able to compress the data.)

3. Check existing archive files, using the Archives plug-in in PI SMT, or the **`piartool -al`** command. Note the start time, end time, and filename (including the path) of all archives within the time range of the backfill data.

4. Make a backup of your PI Server, including all archives you plan to reprocess (if necessary) and backfill. See Back up PI Server.

5. Reprocess old archives to create primary records for the new points.

   You need to reprocess any existing non-primary archives with dates within the range of the backfill data. This creates primary records for the new points in those archives. In addition, you should reprocess them as dynamic archives (using the **`-d`** option) to allow the archives to accommodate new data.

> **Note:**
>
> Skip this step for PI Server 2012 or later. Reprocessing to add the primary record and reprocessing fixed-size archives to dynamic is no longer necessary for PI Server 2012.

To reprocess:

a. Select **Operation** > **Archives** in PI SMT.

b. Right-click the archive file and select **Unregister Archive**.

c. Open a command prompt window and change directories to `PI\bin`.

d. Issue the command:

```
piarchss -if full path of archive -of full path of reprocessed archive -dup
```

This command creates a new dynamic archive file as it reprocesses. It preserves the same start time and end time of the original archive.

e. In the Archives tool, right-click the reprocessed archive and select **Register Archive**.

f. Repeat steps 2-5 for all archives that need to accommodate new data.

6. Create additional archives, as needed.

   If the data to be backfilled include values prior to the oldest archive, create a new dynamic archive with a Start Time at or earlier than the oldest time stamp, and an End Time equal to the start time of the current oldest archive. See Create new archives with PI SMT.

7. Clear the snapshot value for the new points by deleting the snapshot value for each point at the time the point was created.

   You can use **`piconfig`** or the Archive Editor tool in PI SMT to delete the snapshot. See Clearing the Pt Created snapshot.

8. Verify that the oldest value is now in the snapshot for new points.

   At this point your PI System is ready to accept the backfill data. If using piconfig, see Backfill data with a piconfig script

Backfill data with a piconfig script

This topic describes how to set up your data in a comma-separated value (`*.csv`) text file and add a `piconfig` script to the file so you can easily backfill the data into a PI archive.

Procedure

1. Create a comma-separated text file containing the data.

   Format your text file as follows:

   - One tag value per line.

     Each line must include the tag name, timestamp, and value. For example:

```
A1HV074B,08-Aug-01 11:00:00,3659
```

- Values from multiple tags can be included in a single file.
- Values must be in time order (oldest to newest) for each tag to backfill with compression.

2. If you have a large amount of data, separate the data into smaller files so you can easily manage and track the backfilling (for example, one day at a time)

3. Add the following **`piconfig`** script to the beginning of the file:

   ```
   @mode edit,t
   @table pisnap
   @istr tag, time, value
   A1HV074B,08-Aug-01 11:00:00,3659
   ... [followed by the rest of your data]
   ```

4. Save the file as a `*.csv` file, such as `data.csv`.

5. Test the **`piconfig`** script with a small sample of data.

   Always run a backfill test with a small amount of data first, and then do the rest of the data. This way you can verify your **`piconfig`** script and make sure that the data is importing properly. Check the archive and snapshot statistics during the test to see how the backfilling affects the PI Server performance.

   Check the archive and snapshot statistics during the test to see how the backfilling affects the PI Server performance.

6. Force an archive shift to avoid backfilling into a primary archive if you are backfilling on a production system. The easiest way to do this is with the PI SMT Archive Editor plug-in.

7. Open a command prompt window.

8. Change to the `PI\adm` directory.

9. Run **`piconfig`** and redirect your previously prepared data file. Substitute the actual path and file name of your prepared text file for `c:\tags\data.csv` in the example below:

   ```
   piconfig < c:\tags\data.csv
   ```

10. Verify your data using DataLink or ProcessBook.

Backfill data into a new PI System

Topics in this section

- Backfill data into a new PI System with compression
- Backfill data into a new PI System without compression

Backfill data into a new PI System with compression

To backfill data with compression, you backfill the archives by sending the source data in time sequential order so that the data is compressed.

Procedure

1. Install PI Server, start PI Server, create all points, stop PI Server.

2. Isolate the PI Server from all incoming process data.

   This means shutting down PI interfaces on all PI API and PINet nodes.

Another way to do this is to disallow all PI API connections at the server. To do this, start **piconfig** without starting PI. Disregard messages about failure to connect and enter:

```
@table pifirewall
@mode edit,t
@istr hostmask,value
"*.*.*.*",DISALLOW
```

> **Note:**
>
> Entries that allow access to specific names or addresses override the `DISALLOW`. Edit all other entries to `DISALLOW`. Local connections are not affected by `PIFirewall` table entries; verify that applications that may write data are not running.

3. Start PI Server with the `-base` parameter. This ensures that PI Server starts up with only the minimum required subsystems. Enter the command:

   ```
   pisrvstart.bat -base
   ```

4. Use **`piartool -acd`** to create and register an archive file for the backfilling period. You can also use the Archives tool in PI SMT to create multiple files for backfilling. See Create multiple archives for backfilling.

5. Delete all the `Pt Created` events from the snapshot. This can be done with the PI SMT Archive Editor tool (**Data** > **Archive Editor**), a PI API or PI SDK program or with the **piconfig** utility.

   > **Note:**
   >
   > Skip this step for PI Server 2012 unless the Snapshot_DoNotReplacePTCreatedOnOOO parameter is enabled.

6. Backfill the archives by sending the source data in time sequential order so that the data is compressed.

   > **Caution:**
   >
   > For the points that are being backfilled, make sure that no data sources are writing to those points. Otherwise, compression will be bypassed for data that is written prior to the snapshot time.

7. If you used the technique of modifying the PIFirewall table in step 2 above, run **piconfig** to either change the `hostmask` value to `ALLOW` or delete the above `hostmask` altogether.

8. Start the remaining PI Server applications by running `pisrvstart.bat` without the `-base` flag.

Backfill data into a new PI System without compression

To backfill data without compression, specify the start time as the timestamp of the oldest data to be backfilled; and specify the end time as the start time of the oldest archive as listed by **`piartool -al`**. The data that you backfill is not compressed, since the start time of the oldest archive occurs prior to the snapshot time.

If the backfill archive requires more than 2 TB of disk space and if you realize this after backfilling has started, you must delete the backfill archive, and create multiple, dynamic backfill archives to store the data. Divide the target time range between the dynamic archives, and then retry the data backfilling.

Note:

If using PI Server 2012, the data is compressed by default. If you do not want compression, you can disable the Snapshot_DoNotReplacePTCreatedOnOOO tuning parameter.

Procedure

1. Install PI Server and create all points that you will backfill data into.
2. Ensure there are archives created and registered to cover the time span being backfilled into. Use **`piartool -ac`** or **`piartool -acd`** to create and register the archives.
3. Backfill the data.

Backfill existing archives with data from new PI points

Topics in this section

- Backfill existing archives from new PI points with compression
- Backfill existing archives from new PI points without compression
- Clearing the Pt Created snapshot

Backfill existing archives from new PI points with compression

Procedure

1. Add the new points to the existing PI Server archive. Use Tag Configurator to batch load points, or use the PI SMT Point Builder tool to create individual points.
2. If the tags correspond to active interfaces, make sure current data is not being sent to the tag from the interface.
3. Use the PI SMT PI Archive Manager tool or the **`piartool -al`** command to check your existing archive files. Note the start time, end time, and filename (including the path) of all archives within the time range of the backfill data.
4. Make a backup of your PI Server including all archives that you plan to backfill.
5. For PI Server 2010 (3.4.385) or earlier, use the Offline Archive Utility to reprocess all archives that you wish to add data to for these new tags. See Manage offline archive files. Skip this step if you are using PI Server 2012 (3.4.390) or later.
6. For versions of PI Server prior to 3.4, for each of the new tags that you wish to backfill, add one value with a timestamp before or at the start time of the backfill period.

   Skip this step for 3.4 and later.
7. For PI Server 3.3 and later, delete all the `Pt Created` events from the snapshot. This can be done with the PI SMT Archive Editor tool (**Data** > **Archive Editor**), a custom application, or the **`piconfig`** utility. See Clearing the Pt Created snapshot.

Note:

Skip this step for PI Server 2012 unless the Snapshot_DoNotReplacePTCreatedOnOOO parameter is enabled.

8. Backfill the data.

Backfill existing archives from new PI points without compression

Procedure

1. Add the new points to the existing PI Server archive.

   Use Tag Configurator to batch load points, or use the PI SMT Point Builder tool to create individual points.

2. Use the Offline Archive Utility to reprocess all archives that you wish to add data to for these new tags. See Manage offline archive files.

   Note:

   Skip this step if you are using PI Server 2012 or later.

3. Backfill the data.

   Note:

   If using PI Server 2012, the data is compressed by default. If you do not want compression, you can disable the `Snapshot_DoNotReplacePTCreatedOnOOO` tuning parameter.

Clearing the Pt Created snapshot

If you are using a PI Server version earlier than PI Server 2012 (3.4.390) or if the Snapshot_DoNotReplacePTCreatedOnOOO tuning parameter is disabled, follow these steps to clear the Pt Created snapshot.

Note:

You can skip this procedure if you have PI Server 2012 (3.4.390) or later, and the Snapshot_DoNotReplacePTCreatedOnOOO tuning parameter has not been set to 1 (the default is 0).

Use the Archive Editor in PI SMT to delete a `Pt Created` snapshot value one point at a time.

Use **`piconfig`** to delete data in large batches. Follow these steps to export the points with snapshot values of `Pt Created` and the delete those snapshot values from the snapshot table.

Procedure

1. Create a **`piconfig`** script file called `gettags.txt` to export all points whose most current value is `Pt Created`. The file should contain the following commands:

```
@table pisnap
@select status = Pt Created, tag = *
@ostr tag, time
@output snap_to_delete.txt
@ends
```

2. Save and close the `gettags.txt` file to your C: drive.

3. Open a command prompt window.
4. Go to the `PI\adm` directory.
5. Run **`piconfig`** by redirecting the previously created script file. (Substitute the actual path and file name of your script file for `c:\gettags.txt` in the example below):

   ```
   Piconfig <c:\gettags.txt
   ```

   This will generate a list of points whose snapshot you want to delete.
6. Open the newly created `snap_to_delete.txt` file from the `PI\adm` directory to verify which points have Pt Created value in the snapshot, and remove any points from the list that you are not planning to backfill.
7. Save this file.
8. Run **`piconfig`** with the following commands, and use the `snap_to_delete.txt` file as an input for which snapshot values to remove:

   ```
   @table piarc
   @mode edit, t
   @istr tag, time
   @modi mode = remove
   @input snap_to_delete.txt
   @ends
   ```

For PI Server version 3

Before you clear the snapshot, you must first add the oldest value to be backfilled to the archive for each point. Use the Archive Editor in PI SMT or a **`piconfig`** script such as in this example:

```
@mode edit,t
@table pisnap
@istr tag,time,value
mytag1,01-Jun-00 11:00:00,99.2
...
@ends
```

List archive record details (archive walk)

Use **`piartool-aw`** to read the contents of an archive directly from the file. Use this command primarily for troubleshooting.

When a new archive is created, data for each point flows into its own separate primary record. When this primary record fills up, then an overflow record is reserved for the new data. The primary record points to the first overflow record, which can point to a second, and so forth. Following this chain of records is referred to as an *archive walk*. When the number of free unused overflow records in an archive gets below a configurable level, an archive shift initiates.

The key to reading archive data this way is to understand that every PI point has a unique record number (RecNo) which corresponds to a primary record in the archive. This can be found using **`piconfig`** or PI SMT tools.

Topics in this section

- Example of an archive walk
- Determine archive sequence numbers

- Display archive records
- Examine broken pointers
- Verify the integrity of the archive files
- Find and report errors in archive files

Example of an archive walk

To view a detailed listing of archive records, use **`piartool -aw`**. After issuing this command, you are prompted for the target archive number and the target record.

Use **`piartool -al`** to determine the archive sequence number. See Determine archive sequence numbers for more details. After you determine the archive sequence number, you can display archive records, view record details and examine broken pointers.

Determine archive sequence numbers

Some **`piartool`** commands require an archive sequence number; for example, **`archive backup`**(**`piartool -backup`**) and **`archive walk`** (**`piartool -aw`**). The archive sequence number is chronologically assigned with zero being the primary archive. Archive sequence numbers are arbitrarily assigned to unused archives.

Procedure

1. In a command prompt window, change to the `PI\adm` directory.
2. Type: `piartool -al`

Results

In the output, the archive sequence number for each archive is displayed in the brackets immediately following the word `Archive`.

For example, if the archive sequence number is one (1) then the line looks something like this:

```
Archive[1]: g:\pi\arc\piarc144.arc (Used: 16.2%)
```

Display archive records

Topics in this section

- View record headers only
- View archive record event data
- Event record details

View record headers only

Procedure

1. In a command prompt window, change to the `PI\adm` directory.
2. Type: `piartool -aw`
3. Enter an archive number and a record number at the prompts.

The tool returns output that looks something like this:

```
Enter Archive Number: 0
Enter Record Number: 40
    Point ID: 18 Record Number: 40
    Chain Record Number - Next: 80531 Prev: 0 Index: 0
    Record Version: 3 Data type: 11 Zero: 600 Span: 500
    Flags - Index:0 Step:0 Del:0 Dirty:0
    Sizes - Record 1024 Data: 998
    Parent Archive 00000000 Data Head: 26
    Event Count: 214
    Storage (bytes) - Available: 990 Used: 987
Enter Record #, <CR> next rec (p)rev (e)vents (a)rchive # (q)uit:
```

The last line in the record is a toggle to display event data for the record you viewed. To view event data about this record, see View archive record event data.

To toggle off the display, enter h at the prompt:

```
Enter Record # <CR> next rec (p)rev (e)vents (a)rchive # (q)uit:
```

View archive record event data

By default, the **piartool -aw** command displays only the record header. To view the data in the record, enter **e** when prompted for the next record ID.

Event data is displayed as shown in this example. Every archive record must contain at least one event.

```
Enter Record #, <CR> next rec (p)rev (e)vents (a)rchive # (q)uit: e
    PIarcrecord[$Workfile: piarrec.cxx $ $Revision: 75 $]::
     Point ID: 4 Record Number: 59421
     Chain Record Number - Next: 0 Prev: 71411 Index: 4
     Record Version: 3 Data type: 101 Digital State Set: 3
     Flags - Index:0 Step:1 Del:0 Dirty:0
     Sizes - Record 1024 Data: 998
     Parent Archive 00000000 Data Head: 26
     Event Count: 121
     Storage (bytes) - Available: 994 Used: 288
     121 Event(s):
     0: 9-Sep-05 18:57:04, S,O,A,S,Q [3,1,0,0,0]:
     1: 9-Sep-05 18:58:14, S,O,A,S,Q [3,2,0,0,0]:
     2: 9-Sep-05 18:59:24, S,O,A,S,Q [3,3,0,0,0]:
     3: 9-Sep-05 19:00:34, S,O,A,S,Q [3,2,0,0,0]:
     4: 9-Sep-05 19:01:44, S,O,A,S,Q [3,1,0,0,0]:
     5: 9-Sep-05 19:05:14, S,O,A,S,Q [3,2,0,0,0]:
     6: 9-Sep-05 19:06:24, S,O,A,S,Q [3,3,0,0,0]:
       etc.
```

The `S, O, A, S, Q` array in the event record output indicates these values:

Event Type Coding	Definition
S	StateSet
O	Offset in StateSet; 248 corresponds to "No Data"
A	Annotated (0=no, 1=yes)
S	Substitute (0=no, 1=yes)
Q	Questionable (0=no, 1=yes)

Event record details

`Index` shows whether the values in the records are data values or pointers to data records, where 0 indicates that it is *not* an index record. If they are pointers, the pointers are actually record numbers corresponding to the start time. When events for a point exceed two records in a single archive, an index record is created. An index record holds about 150 pointers to data records.

`RecNo` (record number) is a read-only point attribute which indicates the position of the point's primary record in the archive. This is useful when using tools such as **`piartool -aw`** to examine the archives. Do not confuse `RecNo` with the PointID attribute. If the point is deleted, the `RecNo` can be reused but the PointID cannot.

This table shows commonly used data types:

Data Type	Definition
8	Int32(index records are also of data type 8=, as they contain record numbers)
12	Float32
101	digital
102	Blob

Examine broken pointers

In cases of hardware failure, record chains can become inconsistent. You can use the following archive check utility to examine archive integrity:

```
pidiag -archk path
```

See Verify the integrity of the archive files for more details.

The archive offline utility repairs any record chaining problem. See Manage offline archive files for details.

Verify the integrity of the archive files

Run this command on unregistered archive files or when the server is inactive.

Procedure

1. At a command prompt, go to the `PI\adm` directory.
2. Type the command: `pidiag -archk` *`path`*

   Where *path* is the full path to the archive.

   Note:

   You can specify the optional `complete` parameter, as in: `pidiag -archk` *`path`* `complete`, to include details about each record in the report, including the start time, number of events, and fill ratio of the data record.

   The command generates a report that displays:

- List of points sorted by record number (RecNo)
- Number of index records (indices)
- Number of data records
- Count of events in all records and the average fill ratio (fr)

Find and report errors in archive files

Review archive statistics to find problems in archive files. On average, points should not have more than one or two index records. If this is not the case for many points, you should review compression parameters for these points or make the archive files smaller.

Use the archive check command to detect and report any errors in the archive file. Errors are summarized at the end of the report but when running the command, they are sent to the standard error (`stderr`) stream. As a result, when redirecting the output to a file, the following syntax should be used so that errors are inserted into the output file `report.txt`:

```
pidiag -archk "archive_file" > report.txt 2>&1
```

Alternatively, the following construct can be used to redirect the output to two different files:

```
pidiag -archk "archive_file" 1> report.txt 2> errors.txt
```

The archive errors or corruptions could be the result of failures (crash, unexpected termination, power failure, and so on) or software bugs. If this occurs, use the offline archive utility to reprocess the corrupted archive file, recover the data and fix all issues..

The `-archk` command can also be run with the optional argument `complete`, in order to extract detailed information about the archive file structure. This extra information is similar to what is provided by the archive walk command (see `piartool -aw`). It notably includes point types and statistics (start time, event count, fill ratio) for every index or data record and for each point in the archive file. The archive header information is also included at the beginning of the report.

Here is an example of the detailed mode:

```
D:\PI\adm>pidiag -archk D:\PI\dat\piarch.001 complete
Analyzing archive: D:\PI\dat\piarch.001
--------------------------------------------------------------------------
PIarcfilehead[$Workfile: piarfile.cxx $ $Revision: 101 $]::
 Version: 7 Path: D:\PI\dat\piarch.001
 State: 3 Type: 0 Write Flag: 1 Shift Flag: 1
 Record Size: 1024 Count: 131072 Add Rate/Hour: 1.7
 Offsets: Primary: 20/65536 Overflow: 128665/131072
 Annotations: 10826/65535 Annotation File Size: 434144
    Start Time: 19-Oct-05 12:39:10
    End Time: Current Time
    Backup Time: Never
    Last Modified: 19-Dec-05 18:09:15
recno: 1, id: 1, events: 636, annotations: 0, fr: 89.4% - (Float32)
  index array size: 1
0: idxrec id:    1, record pointers: 5, total events: 636
  record array size: 5
0: record id: 130516, start: 19-Oct-05 12:39:10, events: 142, fr: 99.4%
1: record id: 130811, start: 30-Oct-05 15:33:27, events: 142, fr: 99.7%
2: record id: 130515, start: 12-Nov-05 09:29:36, events: 142, fr: 99.9%
3: record id: 130210, start: 22-Nov-05 04:44:08, events: 142, fr: 99.9%
4: record id: 128814, start: 15-Dec-05 13:31:42, events: 68, fr: 47.9%
[...]
```

Manage offline archive files

The offline archive utility uses the **piarchss** executable that runs PI Archive Subsystem.

The offline archive utility works with archive files that are not registered with a running PI Server (that is, offline files). The running PI Server can continue to archive current data, while you manipulate offline archive files. If you attempt to use the offline archive utility on a registered archive file, the utility unregisters the file. You can use the offline archive utility to:

- Combine a number of archives together
- Divide a large archive file into smaller archives
- Extract a specific time period from an archive
- Recover a corrupted archive
- Recover events from an event-queue file

The offline archive utility can create fixed or dynamic archive files. Created files have the same format as archive files created by **piartool -ac**. Every archive file has a parallel annotation file created with the archive. The annotation file, which has the `.ann` file extension, must remain in the same directory as its associated archive file.

To run the offline archive utility, run **piarchss** at the command prompt using special command-line arguments.

Topics in this section

- Tips for using the offline archive utility
- Options for the offline archive utility
- Run the offline archive utility
- Time ranges
- Correcting event timestamps (-tfix -tfixend)
- Combine or divide archives

Tips for using the offline archive utility

When working with the offline archive utility, note the following.

- The full path name of the input archive must be specified.

  Note:

  The **piartool -al** command lists only registered archives.

- If the input file is registered, the offline archive utility unregisters it when processing begins.
- If the input archive is the primary archive, it cannot be unregistered. To work around this, force an archive shift using the **piartool -fs** command or temporarily shut down PI Archive Subsystem.
- If the output file does not exist, the utility creates it.

- If a full path name is not specified for the output archive, the utility places the output archive in the current directory.
- At the end of processing, neither the input nor the output archives are registered.
- Starting in PI Server 2012, the input archive's integrity is checked prior to being used. This behavior can be disabled by specifying the `–noinputcheck` option.
- By default, the offline archive utility creates dynamic archives. Use the **`-f`** option to specify a fixed archive.

> **Note:**
>
> Dynamic archives become non-shiftable once a single overflow record is generated, but remain shiftable if no overflow records are generated.

- You can run the offline archive utility while the PI Server including PI Archive Subsystem is running. At a minimum, the PI Network Manager, PI Base Subsystem, and PI Snapshot Subsystem must be running, because the utility needs to access the PI point database during offline operations.

Options for the offline archive utility

The offline archive utility supports several options. You may enter options in any order. The offline archive utility requires the **`-if`** and **`-of`** options. Subsequent sections discuss how you might use some of the options. Type `piarchss -?` to see a list of available options.

Option	Name	Description
`-acceptprompts`	Accept all prompts	When specified, all prompts the utility present during the reprocessing are accepted by default.
`-dup`	Duplicate records	Allow input archive records with duplicate times. By default duplicates are ignored.
`-evq`	Event queue file	Indicate that the input file is a PI 3 event-queue file (for example, `pimapeq.dat`).
`-f` *size*	Make output archive a fixed archive	Specify size in MB. If size = 0, the input file size is used. Default is dynamic archive.
`-filter` *start end*	Filter	Process events only within the time range (inclusive) specified. Requires both timestamps. See Time filters.
`-filter_ex` *start end*	Filter excluding end time	Process events only within the time range (inclusive of start time; exclusive of end time) specified. Requires both timestamps. See Time filters.
`-id` *pathname*	Specify ID conversion file	Specify the ID conversion binary path and file name. See Specify an ID conversion file.
`-idci` *input_file* **`-idco`** *outfile*	ID conversion file creation	Create ID conversion file from specified input file.
`-if` *pathname*	Input archive file	Required. The full path, including drive letter is required. This is true for all file arguments passed.
`-noinputcheck`	Disable input check of input archive	Disables the integrity check of the input archive.

Option	Name	Description
-oet *option*	Output file end time	Specify the end time for the output file. See Specify an end time for the output file for details.
-of *pathname*	Output archive file	Required.
-ost *option*	Output file start time	Specify the start time for the output file. See Specify a start time for the output file for details.
-outputcheck	Enables checking output archive	After reprocessing is complete, the output archive will be checked to ensure it has integrity.
-silent	Silent mode	Suppress warning messages.
-tfix	Time fix	Apply a specified time transformation to input data. See Correcting event timestamps (-tfix -tfixend).
-**tfixend***timestamp*	Time fix end	Specify a timestamp after which no time transformation is to be performed. Optional; only used in conjunction with **-tfix**.
-tzf *pathname*	Time zone specification file	Use when input is different from standard DST.
-vah	Validate annotation handles	Apply a validation algorithm. Multiple events referencing a single annotation are detected and fixed. Batch Database annotations are checked for consistency.

Run the offline archive utility

To run the offline archive utility (**piarchss**), enter an input archive file and an output archive file, along with relevant command parameters.

Procedure

1. At a command prompt, go to the `PI\bin` directory
2. Type the following command:

   ```
   piarchss -if inputPath -of outputPath
   ```

   where *inputPath* is the full path (including file name) of the input archive file and *outputPath* is the full path (including file name) of output archive file.

Results

The offline archive utility takes the input file, processes it according to the command parameters, and then outputs the processed file to the specified location. The offline archive utility does not modify the input file under any circumstances.

Topics in this section

- Offline archive utility exit codes
- Digital state reprocessing
- Specify an ID conversion file

Offline archive utility exit codes

To facilitate batch file processing, the offline archive utility returns an exit code to the operating system.

Code	Definition
`0`	No errors—at least one input record processed.
`1`	Errors during input phase..
`2`	No processing errors—0 records processed, possibly an empty input file.
`<0`	An error returned from the output processing check log messages.

Digital state reprocessing

Digital states are stored in the archives as offsets in a digital state set. The digital state set number is registered in every archive record of a digital point. System digital states can appear in any record.

When an archive is reprocessed these offsets are preserved, but the digital set becomes the current digital set of the point. This can cause confusion when the digital state of a point was changed. For example, if data is stored for a point with a set of *On,Off* this data amounts to `0` and `1`. Later the point is changed to use a set with *Open, Close*. When reprocessed, the old data displays as *Open* and *Close*. This might be a desired behavior in some cases and confusing in others. One possible solution is to create a new set with both old and new states, but that works only if new data sent corresponds to the new offsets, in this case `2` and `3`.

Specify an ID conversion file

Use the `-id` option to specify an ID conversion file when reprocessing archives, such as when moving a PI archive file to a different PI Server. The ID conversion file is a binary file that maps the source archive point ID into the target system point ID. When you specify an ID conversion file, the offline archive utility processes and coverts points included in the file.

Always use this option when bringing an archive file from another PI Server.

Create the binary file from an input text with the `-idci` option:

```
piarchss -idci ID_conversion_input_file -idco ID_conversion_binary_file
```

The *ID_conversion_input_file* is the full path and file name for the input text file.

The *ID_conversion_binary_file* is the full path and name for the binary file to be created.

The offline archive utility reports any point in the input file that does not exist in the target system.

Time ranges

You can specify the time range of the records that the offline archive utility processes, and you can specify the start time and end time of the output file that the offline archive utility produces.

Topics in this section

- Time filters
- Specify a start time for the output file
- Specify an end time for the output file

Time filters

To process events that occurred during a specific time period, use a filter flag.

- `-filter`

  Specifies the period between the start time and end time, and includes both the start time and end time. The offline archive utility discards events outside this range. The usage is:

  `-filter` *`starttime endtime`*

  Start time must be before end time.

- `-filter_ex`

  Specifies the period between the start time and end time, and includes the start time but excludes the end time. The offline archive utility discards events outside this range. The usage is:

  `-filter_ex` *`starttime endtime`*

  Start time must be before end time.

Specify a start time for the output file

Use the`-ost` flag to specify the start time for the output file. Type:

```
-ost option
```

Where *option* is one of the following:

`input`	Sets the start time to the start time of input. This is the default behavior.
`event`	Sets the start time to time of first event in input.

`time` (where *time* is specified in absolute PI time format)	Sets the start time to the specified time string. Times are specified in absolute PI time format. Relative times are not supported. Times must be enclosed in double quotes when containing spaces. If only date is specified the time defaults to 00:00:00 (midnight) For example:
	`"22-JAN-02 23:59:59"`
	`23-JAN-02`
	`21-Feb`
	Output file start and end times must differ by at least one second.

`NFE`	Sets the start time to time of first event which passes the time filter.

Specify an end time for the output file

Use the `-oet` flag to specify the end time for the output file. Type:

```
-oet option
```

where `option` is one of the following:

`input`	Sets the end time to the end time of input file. This is the default behavior.
`event`	Sets the end time to the time of last event in input file.

`time` (where *time* is specified in absolute PI time format)	Sets the end time to the specified time string. Times are specified in absolute PI time format. Relative times are not supported. Times must be enclosed in double quotes when containing spaces. If only date is specified the time defaults to 00:00:00 (midnight). For example:
	`“22-JAN-02 23:59:59”`
	`23-JAN-02`
	`21-Feb`
	Output file start and end times must differ by at least one second.

`NFE`	Sets the end time to time of last event which passes the time filter.
`primary`	Sets the end time to indicate the archive is a primary archive.
`NoChange`	End time is not altered.

Correcting event timestamps (-tfix -tfixend)

Offsets, as a function of time, are defined in the time conversion file. This can be used to apply corrections to times on some systems that had incorrect timestamps due to run-time library problems, or non-standard DST setting.

This adds a given time offset to every event:

```
-tfix conversion_file
```

The optional `-tfixend` parameter allows you to specify a timestamp after which no timestamp corrections are performed. For example, the following combination of `-tfix` and `-tfixend` specifies to perform the timestamp corrections specified in the file `conversion_file.txt` and to not modify any event timestamps at or after 13:00:00 January 1, 2010:

```
-tfix conversion_file.txt -tfixend "1-jan-2010 13:00:00"
```

Topics in this section

- Time conversion file format
- Time conversion file examples

Time conversion file format

Lines starting with # are comments.

Empty lines and white spaces are ignored.

Data lines have the format:

```
StartTime, offset
```

StartTime may be expressed as UTC - seconds since 1/1/70 or as PI local timestamp string:

UTC timestamps and strings cannot be intermingled, the first format is assumed for all entries.

Offset is a number of seconds added to the timestamp of every event within the time range. Fractional seconds are not supported. Offset applies from timestamp up to, but not including the next timestamp.

Time conversion file examples

Move entire archive ahead by 1 hour:

```
0,3600
2000000000,3600
```

Move entire archive ahead by 1 hour (another format):

```
01-Jan-70 00:00:00,3600
01-Jan-10 00:00:00,3600
```

Apply a missed DST conversion to an archive that covers the summer of 2002:

```
01-Jan-02 00:00:00,0
06-Apr-02 02:00:00,3600
26-Oct-02 02:00:00,0
31-Dec-02 23:59:59,0
```

Apply the time adjustments for each time period as a series of UTC values and offsets:

```
814953600,-61200
828871200,-57600
846403200,-61200
860320800,-57600
```

Combine or divide archives

Archive files are organized according to the time ranges that they span. Occasionally, you might find it useful to change the organization of your archive files. Use the offline archive utility to:

- Combine archive files with overlapping dates into one archive file.
- Combine archive files with adjacent time ranges into one archive file.
- Divide an archive file into smaller archive files, each covering a portion of the original time span.

Topics in this section

- Combine multiple archives into a single archive
- Divide an archive into smaller archives

Combine multiple archives into a single archive

To combine several archives, invoke the offline archive utility (piarchss) once for each input file, using the same output file for all the input files. Start from the oldest input file and continue in ascending time order (the offline archive utility will not work in descending or random time order). For example:

```
piarchss -if D:\pi\dat\oldest.dat -of D:\pi\dat\bigfile.dat
piarchss -if D:\pi\dat\newer.dat -of D:\pi\dat\bigfile.dat
piarchss -if D:\pi\dat\newest.dat -of D:\pi\dat\bigfile.dat
```

In this example, `bigfile.dat` does not exist prior to the operation. The first session creates the file and the second and third sessions add events to the file. By default, the utility creates the file as a dynamic archive. After you create the file, you can register the archive, and PI Snapshot Subsystem can add events to the archive file.

Any of the three input files that were registered prior to the operation are unregistered during the operation. When the operation is complete, you can register them again. Dynamic archives, which is the default type created by the offline utility, are not shiftable.

The end-time of the output file can be moved forward as required, but the start-time cannot be changed after creation.

Archives with an unknown end time should be processed into a new archive to determine the actual end time. The resulting archive can then be merged chronologically. Merging a series of archives with overlapping dates requires processing the archive with the oldest start time first, then process the remaining in chronological order based on the archive end times.

Divide an archive into smaller archives

To break a single archive into smaller archives, invoke the offline archive utility once for each output file and use the same input file for all the outputs. Each invocation, specify a different start and end time in absolute PI time format.

For example, the following statements divide an archive into two smaller archives:

```
piarchss -if D:\pi\dat\bigfile.dat -of D:\pi\dat\january.dat -filter "1-jan"
"31-jan-02 23:59:59" -ost "1-jan" -oet "31-jan-02 23:59:59"
piarchss -if D:\pi\dat\bigfile.dat -of D:\pi\dat\february.dat -filter "1-feb"
"28-feb-02 23:59:59" -ost "1-feb" -oet "28-feb-02 23:59:59"
```

In this example, `january.dat` and `february.dat` do not exist prior to the operation. The offline archive utility creates them as dynamic archives by default. After the offline archive utility creates the files, you can register them with **`piartool -ar`**, and then add events to the archive files in the usual way. Because these output archives are dynamic archives, they are not shiftable.

The filter start time of `january.dat` is specified as 1-jan. This defaults to 1-jan, of the current year, at 00:00:00. The filter end time is enclosed in double quotes because of the embedded space character. The output archive start and end times are explicitly specified. Excluding the `-ost` and `-oet` flags results in the default behavior. For more details, see Specify a start time for the output file and Specify an end time for the output file.

If the input file was registered prior to the operation, it will be unregistered during the operation. When the operation is complete, you can use **`piartool -ar`** to register the file again.

PI interfaces overview

PI interfaces are the software applications that take the data from your data source and send them to PI Server. Most interfaces can also send data in the reverse direction. Typical data sources are Distributed Control Systems (DCSs), Programmable Logic Controllers (PLCs), OPC Servers, lab systems, and process models. However, the data source can be as simple as a text file.

There are hundreds of different PI interfaces and each interface is fully documented in its own dedicated manual. However, most interfaces share a common set of features.

For most interfaces, it is important to configure buffering on the interface node. This prevents loss of data when the PI Server is not available for some reason (such as an upgrade on the Server). The exceptions are:

- Some interfaces do not require buffering because the data source itself is buffered. For example, the UFL interface and batch interfaces such as the Emerson DeltaV Batch interface do not require buffering.
- There are a very few interfaces that should not be run with buffering. Consult the documentation for your interface.

The majority of OSIsoft interfaces use the PI Application Programming Interface (PI API) to retrieve configuration information from the PI Server and to write data to the PI Server. A few non-batch interfaces also use the PI Software Development Kit (PI SDK) to retrieve configuration information from the PI Server and to create PI points, digital states, and so on. Almost all batch interfaces use the PI SDK to write batch data to PI. The PI API and the PI SDK are described in more detail below.

PI interfaces typically run on a dedicated system, called an interface node, which connects both to the data source and to PI Server. The interface node might be a machine that is a part of the foreign data system, or a stand-alone dedicated interface machine, or even a PI Server (PI to PI). Interface nodes can run multiple interfaces and connect to multiple PI Servers.

Note:

Interface nodes are sometimes referred to as API nodes or data source nodes.

Since data sources run without human intervention, user logins are not applicable. To allow PI interfaces to connect to PI Server, you typically configure a *PI trust*. This is sometimes called a *trust login*.

Topics in this section

- UniInt interface framework
- PI API
- PI SDK
- Interfaces included with PI Server
- Interface installation checklist
- Additional documentation on interfaces

- Interfaces Quickstart Series

UniInt interface framework

UniInt is an OSIsoft framework for interface development. UniInt stands for "Universal Interface." The purpose of UniInt is to provide interfaces with a consistent feature set and behavior. UniInt performs many tasks that need to be performed by most interfaces, such as loading points, parsing command line, arguments, and scheduling scans for data. UniInt-based interfaces use some of the UniInt-supplied configuration parameters and some interface-specific parameters. UniInt uses the PI API to write and read data from the PI Server.

PI API

The PI Application Programming Interface (PI API) is a library of functions that read and write values from the PI Server, and also let you retrieve point configuration information. OSIsoft uses the API to create interfaces that run on a variety of platforms.

The PI API also provides the ability to buffer data that is being sent to PI, enabling the PI Server be taken offline for software or hardware upgrades without compromising data collection. When the server becomes available again, the buffered data are forwarded to it.

API Nodes are workstations that run programs such as interfaces that are based on the PI API. In practice, the term *API Node* is sometimes used as a synonym for *interface node*, because historically, most interfaces are API-based.

You can call the PI API from C, C++, Visual Basic, or other languages. For more information on the PI API, refer to the *PI API Programmer's Help*. You can access this from *PI SDK Help*.

PI SDK

The PI Software Development Kit, (PI SDK), is an ActiveX in-process server that provides COM access to OSIsoft historians. The product provides an object-oriented approach to interacting with PI Systems in contrast to the procedural methods used in the PI API.

The PI SDK can be installed only on Windows. Only interfaces that run on Windows can take advantage of the functionality provided by the PI SDK. All interfaces written for UNIX or VMS use the PI API exclusively for all communication with the PI Server.

Some interfaces use the PI SDK because certain functionality is not provided by the PI API. For example, the PI SDK enables interfaces to create points, digital sets. Also, any interface that writes batch data to PI, such as the PI Batch Generator Interface (PIBaGen), must use the PI SDK to write data.

Any data that is written to PI by the PI SDK is not buffered by the BufServ service. For this reason, all interfaces write time-series data to the PI Server using the PI API.

Interfaces that connect to the PI Server using both the PI SDK and the PI API maintain two separate connections to the PI Server.

Interfaces included with PI Server

The PI server installation includes versions of several useful interfaces. These basic versions have limited point counts and other restrictions. You can download and install full versions of these interfaces. These interfaces include:

Interface	Description	Location
PI Batch Generator (PIBaGen)	Collects data from the PI Server, generates batch data and writes the batch data to the PI Server in the PI Batch Database. PIBaGen is used when there is no native interface to generate and store batch data in the PI System. PIBaGen should be run directly on the PI Server.	`pipc\Interfaces\PIBaGen`
PerfMon	Reads Windows performance counters and stores the values in PI points.	`pipc\Interfaces\PIPerfMon_Basic`
PI Interface Status interface	a utility that determines whether an interface program is writing new values to its points. PI Interface Status periodically checks whether a particular PI point, known as the watchdog tag, is receiving new values.	`pipc\Interfaces\PIIntStatus`
Ping	Monitors the network availability of computer systems by pinging them and stores the response times in PI points.	`pipc\Interfaces\PING_Basic`
Random and Ramp Soak	Two simulator interfaces that can simulate random, sinusoidal, and batch data. You can run them remotely or directly on the PI server.	`pipc\Interfaces\Random` `pipc\Interfaces\rmp_sk`
SNMP	Collects performance data from computer systems and network routers using the *Simple Network Management Protocol*, and stores the values in PI points.	`pipc\Interfaces\SNMP_Basic`

Ramp Soak interface point configuration

The default point source for this interface is "9". To list the points that the PI Server installer creates and to create points for the Ramp Soak interface, use PI System Management Tool's Point Builder option.

Set point attributes as follows:

- Point Type (Data Type): Float16, float32, int16, int32, and digital point types are supported. For data, only numeric types are supported. For active and phase tags, the data type can be numeric or digital.
- Location1: Time in minutes to wait before cycle start at simulator startup.
- Location2: Time in minutes between cycles.
- Location3: Maximum signal noise specified as percentage * 10. For example, for 10 percent noise, enter 100.
- Location4 (Scan Class): Determines the frequency at which new values are generated.
- Location5: Phase end time noise specified as percentage * 10. For example, for 10 percent, enter 100.
- InstrumentTag: Configures the optional batch active tag and phase tags. Specify the tag names separated by a space. At the start of a cycle (the beginning of the first phase), the interface writes a value of 1 to the batch active tag and 0 to the phase tag. The phase tag is incremented by 1 at the start of each subsequent phase. At the end of the last phase, the interface writes to the batch active tag. This feature allows simulated batches to be tracked using PI-Batch.

  > **Note:**
  > Do not configure more than one such tag for a simulated unit. If you are simulating batches, you might have more than one ramp soak point for the simulated unit (for example, temperature and pressure). Only one of those points can specify active and/or phase tags, otherwise both will write to the active and phase tags, most likely at different times.

- ExDesc: Specifies phase information that is required to simulate ramp-ups, ramp-downs and soaks.

More about ExDesc: Each phase requires an entry formatted as follows: `(period, end_value)`

- period: Time in minutes
- end_value: The value for the tag at the end of the phase period. By default, the values are reported in engineering units. To report values as a percentage of range, specify a percent sign (%) after the end value parameter.

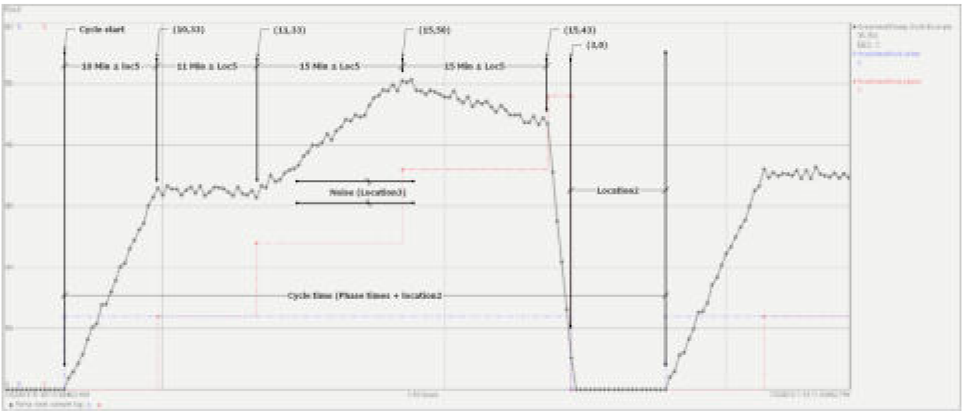
For example, to configure a cycle with five phases, specify the following:

```
(10,33) (11,33) (15,50) (15,43) (3,0)
```

The preceding configures five phases for a total cycle time of 54 minutes, as follows:

- Phase 1: 10-minute ramp from the tag's zero to 33.
- Phase 2: 11 minute period remain at 33.
- Phase 3: 15-minute ramp from 33 to 50
- Phase 4: 15-minute ramp down to 43
- Phase 5: 3-minute ramp down to 0

The resulting trend for the tag will look like this:



If signal phase noise is specified (Location5), the actual phase periods vary within the specified range. The lengths of the remaining phases are adjusted so each cycle has the same overall time. This approach synchronizes several tags to simulate batch data.

Random interface point configuration

The default point source for this interface is "R". To list the points that the PI Server installer creates, and to create points for the Random interface, use PI System Management Tool's Point Builder option. To configure the type of data generated for a point, set `Location5`. The interface can generate the following kinds of data:

- **Sinusoid (Location5=0)**

  Sine wave data that oscillates over a specified amplitude and period.

- **Random (Location5=1)**

  A signal similar to white noise, values range from 0 to 1.

- **Station (Location5=2)**

  Random data changes around a specified midpoint.

- **Fractal (Location5=3)**

  Mandelbrot Sets, which resemble process data and pink noise.

- **Profile model (Location5=4)**

  Data based on a Profile model for use with the PI-Profile client application.

- **String (Location5=5)**

  Random-length string data.

Set point attributes according to the type of data you want to generate, as described below. The following point data types are supported:

- float16
- float32
- int16

- int32
- string
- digital

Sinusoid Data

To configure how the sine-wave data is to be generated, set tag attributes as follows:

- Location2: The number of cycles per day for the sine wave. For example, setting location2 equal to 2 sets a period of 12 hours for the wave.
- Location3: Phase offset from midnight (in seconds). The time can be local time or UTC-based. To specify a UTC based time, Location2 is set to a negative number whose absolute value signifies cycles per day.
- Span: The maximum value to be generated
- Zero: The minimum value of the wave

Random Data

To configure the random data to be generated, set tag attributes as follows:

- Location2: Limits the amount of change between each value, specified as thousandths of Span
- Span: The range of values to be generated
- Scale: Optional scale factor by which the generated value is multiplied

Station Data

Stationary random values are useful for experimenting with SQC (statistical quality control) analysis. By default, the midpoint is (Zero + Span)/2. To periodically reset the midpoint, set Location3 to the desired frequency of reset.

- Span: Maximum amount of excursion
- Zero: Minimum value
- Location3: (Optional) Specifies the number of values to be generated before the midpoint is reset to the current value.

Fractal Data

Mandelbrot sets resemble pink noise and are useful for simulating process trends. Set Location2 to the scale factor for change size (rate) in units of 0.001 of Span. The default scale factor is 25.

Profile Model Data

To simulate data based on a Profile model for use with the PI-Profile client application, set tag attributes as follows.

- Location1: Profile model number (0,1,2,...,9). This attribute corresponds to a paper machine.
- Location2: Attribute array number (0,1,2,...,9). This attribute corresponds to an attribute of a paper machine.
- Location3: The model point type. The following types are supported:

  - 1 Reel Number Tag (integer)
  - 2 Elapsed Distance Tag (real)
  - 3 Grade Tag (digital)
  - 4 HiHi Limit Tag (integer or real)
  - 5 Hi Limit Tag (integer or real)
  - 6 Target Limit Tag (integer or real)
  - 7 Lo Limit Tag (integer or real)
  - 8 LoLo Limit Tag (ineger or real)
  - 9 Attribute Array Tag (integer or real)
  - 10 Defect Type (digital)
  - 11 Defect MD start (real)
  - 12 Defect MD length (real)
  - 13 Defect CD start (real)
  - 14 Defect CD width (real)

- ExDesc: In conjunction with the Location3 attribute, configures characteristics of the model point type.
  - Loc3=1: The MinReelTime,MaxReelTime, and ResetTime (in seconds). Default values are 1800,3600,23400.
  - Loc3=2: Machine Speed (in distance/minute). The default value is 50.0.
  - Loc3=3: Not used
  - Loc3=4-8: LoValue,HiValue
  - Loc3=9: If ExDesc is not blank, the tag will always have bad data.
  - Loc3=10: Defect frequency per scan (0.0 – 1.0). The default value is 0.02.
  - Loc3=11: Not used
  - Loc3=12: DefectMinLen,DefectMaxLen (in % of machine speed). Default values are 0.5 and 10.0.
  - Loc3=13: Machine Width. Default value is 30.0
  - Loc3=1:4 DefectMinWidth,DefectMaxWidth (in % of width). Default values are 0.01 and 0.05.

String

The interface generate strings with lengths that vary between the minimum and the maximum configured for the tag. Zero-length strings are supported. The string contains ASCII alphabetical characters in order from A-Y followed by a-y, repeating as necessary. For example:

```
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmopqrstuvwxyABCDEFGHIJ
```

Set tag attributes as follows:

- Location2: Minimum length of the string, 0 or greater.
- Location3: Maximum length of the string (max 976).

Interface installation checklist

This section provides some general information for installing interfaces. It does not include interface-specific information. Consult the documentation for your interface for specific information.

These steps rely on PI Interface Configuration Utility (ICU). If the interface is configured by PI ICU, then the batch file of the interface is maintained by the PI ICU and all configuration changes are kept in that file and the module database. The PI ICU must be installed on an interface node in order to use it to configure an interface.

If you are familiar with running PI data collection interface programs, this checklist helps you get the interface running.

Topics in this section

- Interface installation
- Advanced interface features (optional)
- Interface diagnostics (optional)

Interface installation

This topic provides some general information for installing interfaces. It does not include interface-specific information. Consult the interface documentation for specific information.

In most cases, you should not run interfaces on the PI Server computer. Running interfaces on a separate computer allows the PI Server to be taken down for maintenance while the interface continues to collect and buffer data. Also, you do not want interfaces competing for computer resources with the PI Server.

These steps rely on PI Interface Configuration Utility (ICU). If the interface is configured by PI ICU, then the batch file of the interface is maintained by the PI ICU and all configuration changes are kept in that file and the module database. PI ICU must be installed on the computer running the interface that you are configuring.

Procedure

1. Confirm that you can use PI SMT to configure the PI Server. 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 Server's Trust table to allow the interface to write data.
3. Run the installation kit for 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 Server 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
   - Interface ID
   - PI Server
   - Scan Class
7. Use the Connection tool to confirm connection between the interface node and the device.
8. If you will use digital points, define the appropriate digital state sets.
9. Add the X, Y, and Z states to the System State Set.
10. Build input tags and, if desired, output tags for this interface.
11. Start the interface interactively and confirm its successful connection to the PI Server without buffering.
12. Confirm that the interface collects data successfully.
13. Stop the interface and configure a buffering application (either Bufserv or PIBufss).
14. Start the buffering application and the interface. Confirm that the interface works together with the buffering application by either physically removing the connection between the interface node and the PI Server node or by stopping the PI Server.
15. Configure the interface to run as a service. Confirm that the interface runs properly as a service.
16. Restart the interface node and confirm that the interface and the buffering application restart.

Advanced interface features (optional)

Consider configuring the interface for disconnected startup. Similarly, consider configuring UniInt Failover. See *UniInt Interface User Manual* for details.

Configure the PI Interface Status Utility

The PI Interface Status Utility provides a convenient means to distinguish true flatlines in data from disruptions in data collection. That is, the utility provides a means of indicating to a user that data from a given interface is stale. Data becomes stale when no fresh data is sent from the interface to the PI Server. For example, stale data can occur under the following scenarios.

- The interface is running but the PI API node cannot send data to the PI Server.
- The interface is not running and a system digital state was not written to indicate that the interface has been shut down. This could happen if the interface crashes.
- One PI Interface Status tag is configured per monitored interface, each tag monitors a watchdog tag collecting data from the interface. If the watchdog tag's value on the PI Server

hasn't updated after a user specified amount of time, the PI Interface Status tag's status changes to a bad digital state status.

The PI Interface Status Utility is always configured to run on the PI server computer. For more information see the *PI Interface Status Utility (ISU) User Manual*.

Configure automatic point synchronization (APS)

Many interfaces, such as the PI OPC interface, support automatic point synchronization (APS). For example, PI points on the PI Server can be automatically created based on the points in the OPC server using a configurable set of rules. You must consult your interface-specific documentation to determine whether your interface supports APS.

If the interface of interest has an APS Connector, consult the interface's *PI AutoPointSync Connector* manual for configuration steps. You can also refer to the *PI AutoPointSync* user manual and *OSI PI COM Connector* user manual for additional information.

Interface diagnostics (optional)

Procedure

1. Configure Scan Class Performance points.
2. Install the PI Performance Monitor interface (full version only) on the interface node.
3. Configure Performance Counter points.
4. Configure UniInt Health Monitoring points.
5. Configure the I/O Rate point.
6. Install and configure the Interface Status Utility on the PI Server node.
7. Configure the Interface Status point.

Additional documentation on interfaces

In addition to this document, the following OSIsoft manuals provide general information with regard to interface configuration and management. They are available on the OSIsoft Technical Support Website.

Manual Name	Notes
PI API Installation Instructions	On Windows, this manual is installed into the `pipc\bin` directory by the PI SDK installation kit. The manual provides several important post-installation details for configuring the PI API and buffering.
PI Buffering User Guide	Explains how to configure buffering for interfaces.

Manual Name	Notes
UniInt Interface User Manual	Most interfaces are based on the OSIsoft Universal Interface (UniInt) and therefore share a common set of features. Certain UniInt features may be described in more detail in the *UniInt Interface User Manual* document than in interface-specific documentation. However, not all features that are described in *UniInt Interface User Manual* are supported by all UniInt interfaces.
PI Interface Configuration Utility User Manual	PI Interface Configuration Utility provides a graphical user interface for configuring the interface command line, interface services, and various PI points that are useful for monitoring interface performance.
PI Performance Monitor Interface to the PI System	The PI Performance Monitor interface, PIPerfMon, obtains Microsoft Windows performance counter data and sends it to the PI System.
Interface Status Interface to the PI System	The PI Interface Status Utility is an interface that runs on the PI Server node. The utility writes events such as "ISU Saw No Data" to PI Points that have not received values for a configurable period of time from a particular interface.
PI AutoPointSync for Interfaces and PI COM Connectors	Some interfaces (such as the OPC interface) support auto-point synchronization. PIAutoPointSync (PIAPS) is a utility that synchronizes PIServer points for an interface using tag definitions on the interface's data source.
PI Server System Management Guide	This document some general information for installing interfaces. It does not include interface-specific information. Consult the documentation for your interface for specific information.

Interfaces Quickstart Series

The Quickstart Series is meant as a tool for those who are familiar with the basics of PI interfaces (configuring trusts, creating services) and wish to see a specific tag configuration example. Those who are newer to interface configuration should consult the complete interface documentation for full instructions.

The following Quickstart Series links take you to the OSIsoft Tech Support website.

- PI OPC Interface: Advise Tag Example (https://techsupport.osisoft.com/Troubleshooting/KB/KB00772)
- PI UFL Interface: Configuration Example (https://techsupport.osisoft.com/Troubleshooting/KB/KB00946)
- PI RDBMS Interface: Output Tag Example (https://techsupport.osisoft.com/Troubleshooting/KB/KB00773)

Monitor the PI Server

Basic PI Server monitoring is discussed in the *Introduction to PI Server System Management* guide. This chapter does not include those basics.

PI System messages

All PI server processes send messages to PI Message Subsystem, which writes messages to the PI message log. The message source is generally a PI server subsystem. However, the message may originate from a source within a subsystem, such as the PI point database.

In PI Server versions earlier than 3.4.380, system messages are plain text. PI Server 3.4.380 introduced the PI message-definition file, which contains message IDs and definitions.

View PI System messages with PI SMT

You can view PI System messages with PI System Management tools (PI SMT). Use this tool to search for messages on all selected PI servers and collectives. Consult the PI SMT Help for more information about using this tool.

Procedure

1. Run PI SMT.
2. Select **Operation** > **Message Logs.**

View messages in the Windows event log

When PI Message Subsystem is not available, PI server components write messages to the Windows event log. You can view these messages with Windows Event Viewer if you are running PI server as a service, or with the command windows if running PI Server interactively. When PI Message Subsystem starts, it merges messages from the Windows event log into the PI Server message log.

Note:

During startup, some components might write messages to the Windows event log before PI Message Subsystem starts.

Message structure

Messages contain the following information:

- Severity — There are five severity levels (listed by level of severity):
  - C — Critical
  - E — Error
  - W — Warning

  - I — Information
  - D — Debug
- Timestamp — Time when component wrote message.
- Source — Component that wrote message.
- Message ID
- Text — Message text, describes event.

For example:

```
C 23-Jul-08 09:27:46.243 piarchss:piarcmgr                                (2050)
 >> Primary archive file failed to load or has invalid dates. Archiving will be
turned off.
```

Severity levels

Messages with an ID of 0-5 are considered *free form*. This means that they don't require a definition in the message definition file to be read. All messages generated prior to the introduction of the message database have an ID 0. The message database introduces five new IDs for free-form messages(1-5), one for each severity level.

ID	Severity Level	Description
1	critical	Loss of System Functionality. Requires immediate attention.
2	error	Action failed.
3	warning	An anomaly has occurred that does not impact the user.
4	information	Action succeeded.
5	debug	Debug/Tracing message

In messages, the leading character indicates the severity level.

- Critical. Requires immediate attention.

  ```
  C 23-Jul-08 09:27:46.243 piarchss:piarcmgr                                (2050)
   >> Primary archive file failed to load or has invalid dates. Archiving will
  be turned off.
  ```

  Note the first character, C, which indicates that this is a critical error.
- Error

  ```
  E 23-Jul-08 09:27:46.243 piarchss:piarcmgr                                (2049)
   >> Failed to load archive file F:\PI archives\8-Sep-04_14-Sep-04: [3] The
  system cannot find the path specified.
  ```
- Warning

  ```
  W 23-Jul-08 09:41:32.733 piarchss:pievqreader                             (6012)
   >> Invalid queue creation path "E:\PI\dat\", using default location
  ```
- Information

  ```
  I 29-Jul-08 18:31:53.211 pinetmgr                                         (7039)
   >> Connection accepted:     Process name:  pigetmsg(6260) ID: 3
  ```
- Debug

  ```
  D 29-Jul-08 15:22:59.421 pibackup                                         (5136)
   >> PIvsswriter::OnFreeze. succeeded
  ```

PI message definition file

The message definition file stores information about the messages. This database associates a message ID with each message, along with the message text, parameters, severity, and other information. The message definition file is called `pimdf.dat` and it is stored in the `PI\dat` directory.

Every product installation may update the definition with new messages. If the message definition file is not present, or is an old version, then some messages may not be able to be rendered into readable text. In this case the message will read something like:

```
E 29-Jul-08 17:44:17 pibasess                                              (8020)
 >> Unknown Message # 8020
```

To see what version of the message definition file is installed, type:

```
pidiag -mdfv
```

Interpret error codes

Use the **`pidiag`** utility to interpret any error codes included in the message logs.

To display the error message, enter:

```
pidiag -e errorcode
```

where *errorcode* is the error number displayed in the message log. Error code values may be positive or negative.

For example, if the error code is -10722, enter:

```
pidiag -e -10722
[-10722] PINET: Timeout on PI RPC or System Call
```

You can also use the **`pidiag`** utility to translate operating system error codes, which are always positive numbers.

Subsystem health checks (RPC resolver error messages)

Every few minutes, **`pinetmgr`** sends a health check message to each of the PI subsystems. If **`pinetmgr`** does not receive a response within a given amount of time, it generates the following message and the subsystem (RPC resolver) is marked offline:

```
>> Deleting connection: pisnapss, Subsystem Healthcheck failed.
```

If an RPC is made to a subsystem that is marked offline, you will see this message:

```
[-10733] PINET: RPC Resolver is Off-Line
```

The output will include details if only the first part of a message was retrieved. In this example, the message contains the message length. However, a timeout occurred when **`pinetmgr`** attempted to retrieve the rest of the message:

```
>> Deleting connection: pisnapss, Connection lost(1),
[-10731] PINET: incomplete message
```

Log file naming convention

During normal operation, PI Message Subsystem maintains central log files for messages from all PI Server subsystems. PI Server creates a new log file every day, on universal time coordinate (UTC) time, puts the log files in the `PI\log` directory and names each log according to the date. The file names use the format, `pimsg_YYYMMDD.dat`, where:

- *YYY* is years since 1900. For example, if the year is 2007, *YYY* is 107.
- *MM* is the month. For example, if the month is January, *MM* is 01.
- *DD* is the day. For example, if it is the fifth of the month, *DD* is 05.

For example, the log file for January 5, 2007 is named `pimsg_1070105.dat`.

You can use the PI SMT Message Logs tool or the **pigetmsg** utility to view these PI message log files and search for messages by time, subsystem, source, severity, or message ID. PI Server must be running to view messages.

The PI Server creates a new log file every day and stores them for 35 days, before it automatically purges log files. To keep log files beyond 35 days, you must back up the log files before the PI Server deletes them. Then, if necessary, you may restore and investigate the backed up files later.

PI Server daily health check

OSIsoft recommends that you regularly check key elements of your PI Server. This process has traditionally been called the *Daily Health Check* even though most of the checks do not need to be performed daily. These checks allow you to identify and resolve potential problems before they become serious. You can also determine how your PI Server operates under normal conditions, and establish a baseline to use when you set up automatic monitoring.

OSIsoft recommends that you schedule the tasks listed here to monitor PI Server:

Frequency	Component	Task	Tools	Automation Method
Daily:				
	Archives	Verify daily that the expected archives are registered and that you have prepared for the next archive shift.	PI SMT **piartool -al**	See "Set up automatic archive creation" in *PI Server Introduction to System Management.*
	Backups	Verify daily that PI System backups were run for the previous day and that you have sufficient disk space for future backups.	PI SMT **piartool -al**	See Back up PI Server .
Weekly, or more often:				
	PI System messages log	Review the system message log and interface logs at least once a week to determine if unusual events occurred.	PI SMT **pigetmsg**	N/A
	Interface logs	Check logs to see whether unusual events have occurred	PI SMT	N/A

Frequency	Component	Task	Tools	Automation Method
	Snapshot data flow	Determine if snapshot data flow is normal.	PI SMT **`piartool -ss`**	
	IO rate counters	I/O rate points monitor the flow of data from an interface. Every 10 minutes each IO-rate point registers the 10-minute average data transfer rate to PI Server in events/ second. If the value stops updating in PI then this is an indication that the interface has stopped collecting data.	PI DataLink or PIProcessBook	
Monthly:				
	License limits and usage	Perform monthly usage statistics and point count checks for your PI System to anticipate license increase needs.	PI SMT **`piartool -lic`**	N/A
Upon initial setup:				
	Performance counters (Windows)	Use key Windows performance counters to review statistics about all subsystems.	Windows Performance Monitor, PI Performance Monitor (PerfMon) Interface	
As needed:				
	Tag data	Review archive data reference tags.	PI SMT `pisnap.bat` or `pisnap.sh`	
	Update Manager	Verify client connections to the PIServer.	**`pilistupd`**	

Windows performance counters

You can use key Windows performance counters to review statistics about all PI subsystems. These counters may be viewed with the Windows Performance Monitor or recorded to PI Server with the OSIsoft Performance Monitor interfaces.

The Performance Monitor Interface can access and record any published counter, including Windows counters such as CPU utilization.

PI Server backup performance counters

OSIsoft recommends that you monitor the following Windows performance counters for the PI Backup subsystem:

- Last Backup Failed will be 1 if the last backup failed; otherwise it will be 0.
- Backups Started should increase by 1 every night if you have a nightly backup task installed.
- Failed Backups will increase by 1 for every failed backup.

All of these counters are reset to 0 when the PI Backup subsystem is restarted. The values for these performance counters can be stored into PI points with the PI Performance Monitor interface, the basic version of which is installed by default on the PI Server node.

If you have PI Notifications, you can configure e-mail alerts based on Last Backup Failed or Failed Backups. Otherwise, you can view the values of these performance counters with PI ProcessBook.

If the `pisitebackup.bat` fails or if a third-party backup of your backup directory fails, this is not reflected in Last Backup Failed or Failed Backups.

PI Server tuning parameters

The Tuning Parameters tool in PI SMT allows users to view all the settings that have been correctly set and a list of settings that are commonly altered. (This tool was formerly known as the PI Timeout Table Editor.)

Most PI Servers require no tuning and work well with the default settings. PI Server tuning parameters allow you to adjust default settings if needed. Tuning parameter values are preserved during PI Server upgrades. When you upgrade the PI Server, you should review any values that you changed from the default.

Procedure

1. Configurable tuning parameters.
2. Edit tuning parameters.
3. Add a tuning parameter to the list.
4. Pending update limit.

Configurable tuning parameters

By default, each tab in the Tuning Parameters tool provides a list of the most commonly used server settings for each category. Settings are displayed on these tabs:

- **General:** Command line tool and server application settings.
- **Archive:** PI Archive Subsystem settings.
- **Backup:** PI Backup Subsystem settings.
- **Base:** PI Base Subsystem settings, including module database parameters.
- **Net Manager:** PI Network Manager settings.
- **Snapshot:** PI Snapshot Subsystem settings, including event queue settings for buffered values that are not yet archived.
- **Update Manager:** Update Manager settings that affect programs that sign up for point or data updates, including ProcessBook and most interfaces.
- **Security:** Security settings that affect server authentication, trust configurations and properties of PI identities and mappings.

If the tuning parameter that you want is not displayed in these lists, then you can add it.

Edit tuning parameters

Changes to tuning parameters do not take effect until you stop and restart PI Server or the subsystem associated with the updated parameter. Applications that were already connected to PI Server will not reflect tuning parameter edits until you reconnect.

Procedure

1. Select **Start** > **PI System Management Tools** > **Operation** > **Tuning Parameters.**
2. In the **Collectives and Servers** box, select the check box of the PI server on which you want to edit the parameter.
3. Clear the check boxes for all other PI servers.
4. Click the tab for the subsystem where you want to edit a tuning parameter value.
5. If necessary, Add a tuning parameter to the list.
6. Right-click an existing parameter in the list, and select **Edit**.
7. Enter a **Value** and click **OK.**

Add a tuning parameter to the list

By default, only the most commonly-used tuning parameters are displayed in the PI SMT Tuning Parameters tool. Not all tuning parameters are displayed in the list.

Procedure

1. In the **Collectives and Servers** box, select the check box of the PI Server on which you want to add the parameter.
2. Clear the check boxes for all other PI Servers.
3. Click the **New Parameter** button 🗋 on the toolbar to open the server dialog box for parameter properties.
4. Verify that the correct PI Server is displayed on the title bar. If not, close the dialog box, and select the correct PI Server under **Collectives and Servers.**
5. In **Parameter name**, choose the name of the parameter that you want to add to the list. (If you know the name, you can type it in exactly. The setting menu field remains yellow until a recognized parameter name is entered.)
6. Click **OK** to add the parameter to the parameter list.

   You can type in a value for the tuning parameter before you click **OK.** You will then need to stop and restart the PI Server or the subsystem associated with the updated parameter, for the changes to take effect.

Pending update limit

The MaxUpdateQueue parameter sets the maximum number of events per consumer. The default is 50,000 pending updates per consumer. Similarly, the TotalUpdateQueue parameter sets the maximum events queued in the entire Update Manager database. The default number of maximum events is 1,000,000.

If either of these limits is reached, a message is sent to the PImessage l. Another message is sent when the level goes back below 99 percent of the limit and queuing is resumed. Messages for consumers using less than 0.1 percent of the TotalUpdateQueue limit (100 for the default) are still queued even when the total limit is reached.

Procedure

1. When to change the pending update limit.

When to change the pending update limit

The default is suitable for most systems, with the following exceptions:

- The number should be increased on systems with one or more of the following:
  - Very large physical memory
  - High frequency of updates (normally snapshots)
  - Applications that might retrieve update limits slowly
- If a PINet node or PItoPI interface is connected to the PI Server, the default MaxUpdateQueue value is likely to be too small. It should be increased to at least the point count of the PI Server. This value ensures that all point updates requested by PINet can be queued on the PI Server if a system manager performs an edit operation on every point.

PI SQL Subsystem

PI SQL Subsystem (pisqlss) prepares and executes SQL statements directed at the PI Server. The primary users of this subsystem are the PI ODBC driver and the PI SDK. This driver conforms to the ODBC API standards and makes PI data appear to be organized into data tables. PIODBC1.1.2 or later is required to connect to PI Server.

OSIsoft's implementation of SQL gives applications access to the PI Point Database, snapshot, and archive. For supporting information, such as details of OSIsoft's implementation of SQL, see the *PI ODBC Client User Manual*.

SQL processing capability is also implemented in the PI System for OpenVMS. Differences between the two are discussed in this chapter.

Procedure

1. PI SQL Subsystem architecture.
2. PI SQL Subsystem configuration.
3. PI SQL Subsystem troubleshooting.

PI SQL Subsystem architecture

Procedure

1. Statement handles.
2. Concurrency.

Statement handles

Most interactions between PI clients and the PI Server do not require the Server to maintain any context, that is, any record of the client's operation. Any request for point information or archive data can be serviced using only the information provided by the client in the request itself.

The processing of SQL statements is different. When an SQL statement is processed, the Server must maintain a record of the statement's status in order to be able to perform subsequent operations.

This is done by having PI SQL Subsystem allocate a statement handle when a client initiates the processing of an SQL statement. The client retains the handle's identifier and provides it to the server with every request.

The details of handle allocation and de-allocation are managed internally by a PI API based client application, such as the PI ODBC Driver.

If connection between the client and Server is lost, PI SQL Subsystem retains any statement handles allocated by the client. These handles become orphaned and cannot be accessed again. The handles are de-allocated when PI SQL Subsystem shuts down. During shutdown, pisqlss will report the total number of handles allocated during its run, and the number of orphaned handles that were aborted.

Concurrency

PI SQL Subsystem handles SQL processing for all client connections to the PI Server.

Operations such as parsing an SQL statement and fetching results are relatively inexpensive. Execution, however, can potentially take time and system resources as data are obtained from other subsystems.

PI SQL Subsystem configuration

No advance configuration is necessary to start the PI SQL Subsystem (**`pisqlss`**). It is started and stopped exactly the same way as other subsystems. On Windows, **`pisqlss`** can be run as a service.

Some tuning of **`pisqlss`** performance is possible. Settings can be changed using an initialization file, **`pisqlss`** command-line parameters, and through settings on a client product, such as the PI ODBC Driver.

Note:

The use of an initialization file may change in a later release to an alternate method. At that time, any site-specific settings found in the initialization file are migrated.

See your client product documentation for instructions on changing SQL processing settings from the client application.

Procedure

1. Hierarchy of PI SQL Subsystem settings.
2. Initialization file settings.
3. pisqlss.

Hierarchy of PI SQL Subsystem settings

Since it is possible to set parameters using more than one technique, some of the settings may be in conflict. The actual value of the settings employed follows this priority scheme:

- Initialization file settings
- PI SQL Subsystem (**`pisqlss`**) command-line arguments
- Client product settings

Settings made lower in the list override settings higher up. For example, if the SQL query timeout is set to 45 seconds in the initialization file and to 60 seconds on the pisqlss command-line, the value used is 60 seconds.

Initialization file settings

The initialization file is called `pisql.ini` and can be found in the `PI\dat` directory of your PI Server. The file contains defaults for all settings. You may change the settings by editing the file.

The initialization file settings are read when a new statement is allocated. Any change to this file is reflected in the next new statement.

> **Note:**
>
> On a PI System for OpenVMS, the initialization file is `PISysDat:pisql.ini`. The interpretation of the file settings is exactly the same for both PI Servers.

For details on the settings, see the *PI ODBC Client User Manual*.

pisqlss

This section outlines the PI SQL Subsystem (**`pisqlss`**) settings that can be altered using command line arguments. The mechanism for specifying command-line arguments differs between supported platforms. This section outlines the techniques.

pisqlss command line options

In general, an command line option is specified by an argument token, one or more spaces, and then the argument value. The argument token always begins with a leading dash (`-`). For example:

```
pisqlss -t 60 -ts 7200 -o trace,aggrtimestart
```

In this example, SQL query timeout is set to 60 seconds, the default time step (for queries against the piinterp table) is set to 7200 seconds (that is, 2 hours) and the `trace` and`aggrtimestart` options have been set.

PI SQL Subsystem (pisqlss) supports the following command-line arguments:

Argument	Description
`-o`	Options (letter "o", not zero). The options are specified in a comma-separated list of tokens. The interpretation of the tokens is not case-sensitive. See the following table for the list of supported options.
`-t`	SQL query timeout, in seconds. If this time expires, PI SQL Subsystem will cause the query to return either a timeout error, or a subset of the actual results, if the `SUBSET` option is set. See the table of options below.
`-ts`	Default value of the timestep column in the PIINTERP table. This value can be overridden for any query by specifying a timestep constraint in your SELECT statement.

The `-o` argument is followed by a comma-separated list of option tokens with no space between the tokens. The supported options are:

Option Token	Description
`AGGRTIMESTART`	Causes all queries of the aggregate tables to use the time at which the interval starts to identify the aggregate; the default is to use the time at which the aggregate period ends.

Option Token	Description
EXECSAFE	If set, the query does not execute if the PI SQL determines that a query is too unspecific and would take too long to execute.
LOG	Writes a summary of every operation by pisqlss on a statement handle to the file `sqltrace.log` in your `pi\log` directory. See also the TRACE option. This file is generated in all PI Server configurations, except while not running as a service on Windows. In this case, output is directed to standard output, which is the command window.
QEP	Causes the gateway to dump a query execution plan to a file called `pisql_n.qep` in the `pi\log` directory on the PI Server, where *n* is the handle number.
SUBSET	If a query times out while this option is in effect, PI SQL Subsystem returns a subset of the actual results with no error. If this option is in effect, the results returned do not represent the actual final results of the query. When query development is complete, remove this option.
TRACE	Writes a summary of every Prepare, that is, query parsing, and Execute operation by PI SQL Subsystem to the file `sqltrace.log` in your `pi\log` directory. See also the LOG option.

Start pisqlss as a Windows service

You can start subsystems by running **Services** in Control Panel, or by using the `pisrvstart.bat` file in your `PI\adm` directory.

To pass command-line arguments to PI SQL Subsystem running as a Windows service: in Control Panel open **Administrative Tools.** Open **Services**, select **PI SQL Subsystem**, right-click and choose **Properties.** Stop the service, and enter the arguments into the Start parameters dialog box. Click the **Start** button to restart PI SQL Subsystem.

This example shows a system manager about to restart PI SQL Subsystem while setting the default `timestep` to 7200 seconds and turning on TRACE mode.

Note:

This works one time only. If you close the **Services**, then reopen and reselect your service, you will not see your command-line arguments from the last run. This method cannot be used to provide command-line parameters to services started automatically when your Windows system boots.

Start pisqlss in a command window

Command-line arguments can be provided to a Windows program by listing them after the program name. You can edit the file `pistart.bat` to include command-line arguments when starting subsystems.

Starting the PI Server this way results in a command window for every subsystem. You cannot log off Windows in this configuration and leave the system running.

Use caution when editing `pistart.bat`. This file is overwritten at every PI Server upgrade.

PI SQL Subsystem troubleshooting

Unexpected errors may be generated when using an ODBC application to communicate with the PI Server. This section outlines techniques to help you validate the operation of PI SQL Subsystem and to log its processing steps.

Procedure

1. PI SQL Subsystem trace files.
2. Clear expensive query problems.
3. Performance counters.
4. PI System unexpected shutdown or power outage recovery plan.
5. Technical support and problem reports.

PI SQL Subsystem trace files

You can generate a trace file by starting PI SQL Subsystem with the `LOG` or `TRACE` options.

The options `LOG` and `TRACE` are similar. Both generate information in the `sqltrace.log` file in the `PI\log\` directory. The `LOG` option provides more detail. The options can be used together. Output from the two is interspersed.

Procedure

1. Output from the TRACE option.
2. Output from the LOG option.

Output from the TRACE option

Invoking the TRACE option shows a summary of SQL statement preparation and execution only.

Procedure

1. Statement preparation output.
2. Statement execution output.

Statement preparation output

Output lines are of the form:

```
Prepare[HandleNum]>[ErrorCode][ElapsedTime] query_strng
```

Elapsed time is given in seconds. For example:

```
Prepare[1]>[0][0.012s] select * from picomp
where tag = sinusoid and time > ?
```

This statement contains one parameter, identified by a question mark (?), whose value is provided at run time. Run-time parameters are discussed in detail in the *PI ODBC Driver Manual*.

Statement execution output

Output lines are of the form:

```
Execute[HandleNum]>[ErrorCode][ElapsedTime] Parameters: NParams Columns: Ncols
Rows: Nrows
```

If the number of run-time parameters is non-zero, this message is followed by one line for every provided parameter:

```
Pnn [DataType Length] parameter_value
```

where *nn* is the run-time parameter number, starting with 0.

For example, the Execution message following the above Prepare message might read:

```
Execute[1]>[0][0.041s] Parameters: 1 Columns: 4 Rows: 16
P00 [time32]    21-Jul-97 00:00:00
```

The query in this example returned 16 rows of 4 columns. The query was provided with one run-time parameter: a timestamp.

Output from the LOG option

Output from the `LOG` option is more detailed. It reflects directly the argument list of the Remote Procedure Calls (RPCs) implemented by PI SQL Subsystem. Most of the information generated should be forwarded to OSIsoft in the event of a query processing problem.

In general, the first argument of each RPC is the handle ID. The only exception is the `newstatement` function, which is the routine that generates the handle ID. In this case, the returned handle ID is the second argument.

Procedure

1. Function summary format.

Function summary format

The LOG option generates output that looks like this:

```
FunctionName(arg1, arg2, ...) [ErrorCode]
```

During query execution, progress messages are written to the log file. Each progress message is of the form:

```
(HandleId): Calls: n PctDone: n Etime: n Limit: n
```

The items reported are:

- Number of calls to get PI data from other subsystems.
- Percent complete, based on an initial estimate.
- Elapsed time since the start of execution, in seconds.
- Timeout (Limit) in seconds. If this number is 0, no timeout limit has been set.

For example:

```
newstatement(8,21) [0]
clear(21,1) [0]
clear(21,0) [0]
Prepare[21]>[0][0.431s] select * from picomp
where tag = "sinusoid" and time > "y"
execute(21,&params) begins...
callback(21): Calls: 1 PctDone: 0 Etime: 1 Limit: 0
fetch(21,*results) [0]
clear(21,1) [0]
```

Clear expensive query problems

It is possible that an ODBC client application sends an incomplete query, or a query that returns too many results, to PI Server. When a query is timed out, it may or may not hold on to the server resource, mainly the virtual memory. If the timeout occurs during the query execution, the statement handle and its resource are freed. If the timeout occurs during the fetch, the statement handle is not freed. To clear the statement handle and its resource, shut down and restart the PI SQL Subsystem.

To do this, send a **`stop`** command to PI SQL Subsystem using one of the following methods:

- From the **Control Panel** > **Administrative Tools**, run **Services.** Select PISQL Subsystem from the list and click **Stop.**
- From a command prompt, enter:
  ```
  net stop pisqlss
  ```
- From a command prompt, enter:
  ```
  \pi\bin\pisqlss -stop
  ```

A message is written to the message log indicating that PI SQL Subsystem has been stopped. Another message indicates the number of handles allocated and the number of handles aborted during the shutdown.

To restart PI SQL Subsystem and resume normal operation, use one of the following methods:

- From the **Control Panel** > **Administrative Tools**, run **Services.** Select PISQL Subsystem from the list and click **Start.**
- From a command-line prompt, enter:
  ```
  net start pisqlss
  ```
- From a command prompt, enter:
  ```
  \pi\bin\pisqlss -start
  ```

A message is written to the message log indicating that PI SQL Subsystem has been continued.

Shutting down and restarting the subsystem can be done at any time and is equally effective. This is the only option available when running PI SQL Subsystem on Windows interactively.

Performance counters

In PI Server for Windows, you can monitor several aspects of PI SQL subsystem processing continuously with the Performance Monitor application.

PI System unexpected shutdown or power outage recovery plan

In the event of an unexpected shutdown of the PI System, see this recovery plan: http://techsupport.osisoft.com/Support+Solution/10/KB00669.htm (http://techsupport.osisoft.com/Support+Solution/10/KB00669.htm)

Technical support and problem reports

If PI SQL Subsystem consistently returns an error when processing SQL statements, or appears to generate incorrect results, you should stop the PI SQL subsystem and then restart with the TRACE and LOG options enabled. Send the resulting `sqltrace.log` to OSIsoft Technical Support.

PI data retrieval with foreign data sources

Data is sometimes not stored in PI Archive Subsystem or PI Snapshot Subsystem. Data may be stored in an external or foreign data source. PI Base Subsystem, PI Archive Subsystem, and PI Snapshot Subsystem can request data from foreign data storage systems through modules called COM connectors. A separate COM connector must be installed to communicate with each specific foreign data system.

A PI Server system may have any number of COM connectors installed. Since the identity of the COM connector to use is determined on a point-by-point basis, a single PI Server can access any number of foreign data systems.

The core subsystems of the PI Server do not communicate directly with COM connectors. Instead, the subsystems send requests to PI Redirector, which acts as a request broker. PI Redirector loads one or more COM connectors and forwards the requests to them.

PI Redirector and the COM connectors are COM objects, implemented using Microsoft Component Object Model (COM) technology. PI Redirector is installed as part of the PI Server. COM connectors are installed separately.

COM connectors are installed on the PI Server, but are not loaded into the server's memory until needed. When PI Server shuts down, PI Redirector and all COM connectors are automatically unloaded from memory.

COM connectors may be in-process or out-of-process COM objects. In-process COM objects are `.dll` files, while out-of-process COM objects are `.exe` files. For a list of available COM connectors, see the COM Connectors page on the OSIsoft Technical Support website. If the existing COM connectors do not fit your needs, contact OSIsoft Technical Support.

PI Redirector is an out-of-process COM object. It does not run as a service, which means it is not found in **Services** in Control Panel. When PI Redirector runs, system managers can see a process called `piudsrdr.exe` in the **Processes** tab of the Windows Task Manager.

Client applications are not aware of the difference between data retrieval from the PI archive and data retrieval from a foreign data storage system using a COM connector. In all cases, the application connects to PI Network Manager. Each point that data is retrieved from is identified by a tag, and has attributes stored in the PI point database, regardless of the source of the data. PI Snapshot Subsystem and PI Archive Subsystem implement the differences in data flow. For details, see Retrieval of snapshot data and Retrieval of archive data.

PI Server sends data to client applications in exactly the same way, regardless of whether the data is stored in PI Archive Subsystem or in a foreign data source. The same is true of data requests from PI Server subsystems, such as PI Totalizer Subsystem, PI Alarm Subsystem, and PI Performance Equation Scheduler.

PI Server can write data into a foreign data system if it is supported by the COM connector for the foreign data system.

Procedure

1. Point configuration.
2. Retrieval of snapshot data.
3. Retrieval of archive data.
4. Archive files.

5. Snapshot updates.
6. Compression.
7. Point security.

Point configuration

To interact with a point on a foreign system, a corresponding point, called a mapped point or COM connector point, must be created in the PI point database. A mapped point in the PI point database is one that links to data in a foreign system.

To build a mapped point, select a point class that includes the following point attributes, as well as the normal attributes of a point class:

COM connector point attributes

Attribute	Description
ctr_progid	COM program ID, as stored in the Windows registry. This name is used to invoke the COM connector object when needed.
ctr_lmap	Longword mapping parameter. The ctr_lmap and ctr_strmap attributes are passed to the COM connector so that it can locate the appropriate foreign system point.
ctr_strmap	String mapping parameter. The ctr_lmap and ctr_strmap attributes are passed to the COM connector so that it can locate the appropriate foreign system point.

PI Server includes the classicctr point class, which contains these point attributes as well as the base and classic attribute sets. To create this point class, run the script `PI\adm\classicctr.dif` using the **`piconfig`** utility.

Construct points according to the specifications of the point class. For details, see PI point classes and attributes. Points are created and maintained using PI Tag Configurator, a Microsoft Excel spreadsheet-based tool, or the **`piconfig`** script-based tool.

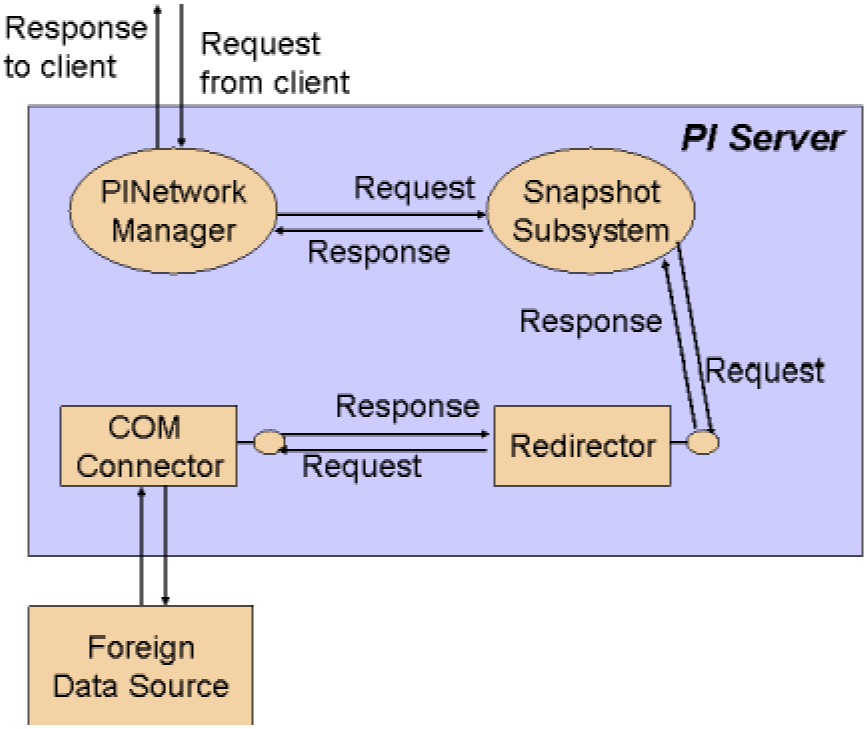
Whenever the point information indicates that the requested point is a mapped point, the Redirector obtains data values from the corresponding foreign system point.

Retrieval of snapshot data

When PI Snapshot Subsystem starts, PI Base Subsystem notifies it of the presence of mapped points. When a client application requests a snapshot value, PI Network Manager routes the request to PI Snapshot Subsystem.

If the point's data is normally stored in the PI archive, the snapshot value is retrieved from PI Snapshot Subsystem and then returned to PI Network Manager.

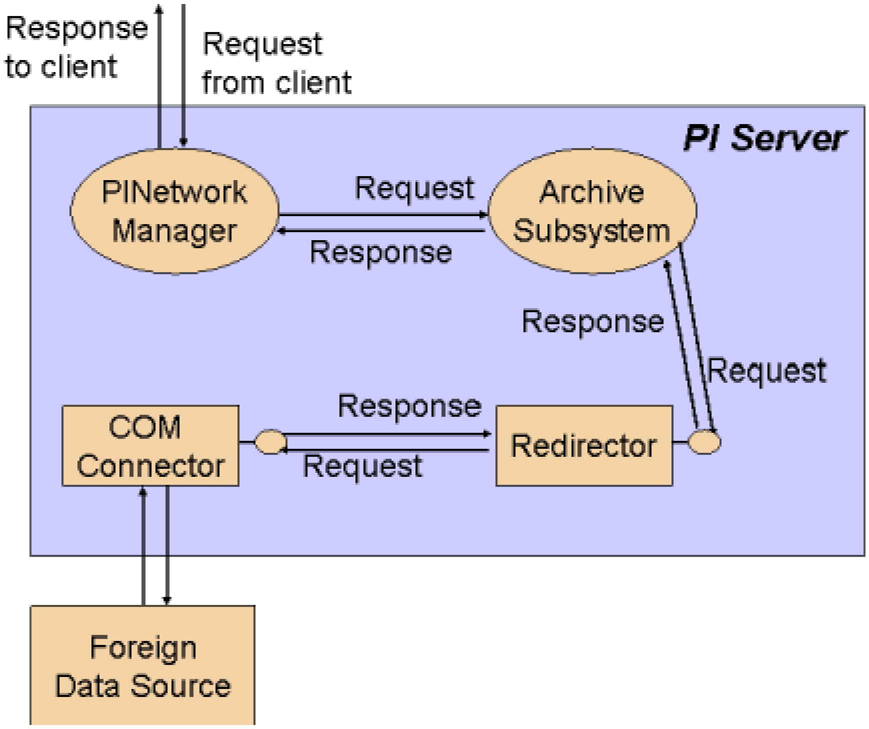
If a snapshot value for a mapped point is requested, the data path is different. In this case, PI Snapshot Subsystem requests the value from PI Redirector, which in turn requests the value from the appropriate COM connector. The COM connector obtains the value from the foreign data storage system and returns it to PI Redirector, which sends it to PI Snapshot Subsystem. It is then routed to PI Network Manager for transmission to the client.



Retrieval of archive data

The retrieval of archive data from the COM connector by PI Archive Subsystem is similar to snapshot retrieval by PI Snapshot Subsystem. When PI Archive Subsystem starts, PI Base Subsystem notifies it of the presence of mapped points.

If archive values for a mapped point are requested, PI Archive Subsystem requests the value from PI Redirector, which in turn obtains the value from the appropriate COM connector.



Archive files

Even though archive files are not used if data is retrieved using COM connectors, the files must exist and must be sized as if all points on the PI Server were PI archive points. Each COM connector point consumes a primary record in the archive file even though it will never be used for data storage or retrieval. Normal maintenance and backup procedures apply to the archive files.

Snapshot updates

The COM connection mechanism includes support for exception reporting. PI Snapshot Subsystem calls a sign-up method in the COM connector if a client signs up for exceptions on a mapped point. PI Snapshot Subsystem obtains exception values from the COM connector by way of PI Redirector.

If the foreign system does not support exception handling, it may be coded to return a standard COM error code indicating that the method is not implemented.

Compression

PI Server does not apply the data compression algorithm to mapped foreign points. If the COM connector supports putting new data values into the foreign system, then that system is responsible for their storage. The foreign system may or may not support compression.

Point security

Data retrieved from a foreign data system is subject to the same security as data values retrieved from storage within PI Server. Every PI point, whether mapped or not, carries a security descriptor that defines the access that PI users may have to data. For details about how to set point security, see the *Configuring PI Server Security* guide.

Troubleshooting

Data passes through many steps on the way into and out of the archive. When troubleshooting, it is important to identify both data path malfunctions, and data paths that function correctly.

Topics in this section

- Troubleshooting checklist
- General troubleshooting tasks
- Specific problems

Troubleshooting checklist

OSIsoft recommends that you complete the steps in this checklist to troubleshoot PI Server problems. If you have not resolved the problem when you finish these steps, contact OSIsoft Technical Support (http://techsupport.osisoft.com).

Procedure

1. Look for Error Messages. If you know the specific error message, search the Technical Support site for that error. If you do not yet have a specific error message, look at the message logs on the server and on the client node. For server messages, you can use the Message Log tool in PI SMT. Filter messages for a severity of `Error` and greater (see Severity levels).

   You can't look at a client node message log remotely. You can run PI SMT directly on the client node or you can use **`pigetmsg`**. For interface problems, you can examine the `pipc\dat\pipc.log` files directly on the interface node. If this is a setup problem, look at the setup logs in the 32-bit `pipc\dat` directory.

   To get a text description for an error number, use:

   ```
   pidiag -e errornumber
   ```

   For more information on error messages, see PI System messages. For information about **`pigetmsg`**, see the *PI Server Reference Guide*.

2. Determine which computers exhibit the problem:

   - Client computer(s)
   - Server computer(s)
   - Interface computer(s)

   To isolate the computers, run the questionable system against a system that is functioning correctly and review the results.

   - A network problem is likely if all computers exhibit the malfunction.
   - A server problem is indicated if the malfunction occurs on all clients.
   - A hardware or networking problem is likely if the applications that malfunction do not use PI Server. Run **`telnet`** to further isolate the problem. If **`telnet`** works, then the network is not likely the problem, although it might be a network issue such as DNS or firewall blocking.

3. If this is a client problem:
    - Check security. Log onto Windows using an account that has a Mapping to piadmin.
    - Check the Update Manager to make sure that the client is signed up for and receiving updates. Use the **pilistupd** command utility to check for updates.
    - If a trend in PI ProcessBook flatlines, see PI ProcessBook trend flatlines.
4. If this is a server problem:
    - Verify that all PI processes are running. You can use the PI Services tool in PI SMT to see the status of all processes running as services.

      Note:

      PI Systems may take several minutes to start; loading of the Point Database, snapshot and archives takes most of the time. Utilities, such as **piartool** and **piconfig**, are not fully operational until startup has completed.

    - Even if a process is listed as running, it might be in a state where it is not communicating with PI Network Manager (PINet Manager). Here are some things to check:
        - In the SMT PI Services tool, select the service and check its details under **Thread Details for Selection.** Look at the **State** column and the `CurTime` (milliseconds) column. If ***State*** is **InUse** and **CurTime** is unexpectedly large, the thread might be hung.
        - Use the SMT Network Manager Statistics tool to get more information (see Connection checking).
        - Use the utilities listed in Verification that PI Server processes are running to verify that individual PI Server processes are communicating. For information about connections, look at the Network Manager statistics.
        - Use `piartool -block` *`<subsystem>`* `-verbose` to check whether a subsystem is responsive.
    - Try running the PI Server with `pistart.bat`, rather than as services. The interactive command windows may display additional status messages.
5. If a subsystem crashes, there may be additional information that can be useful to our developers. Configure Dr. Watson (see Configure Dr. Watson for Windows) or another application debugger to generate a crash dump file.
6. Use **`netstat -a`** to verify whether other processes are communicating on port 5450; if so, PINet Manager communicated successfully at one time.
7. If you have an archive or snapshot problem, use the **`piartool -as`** and **`piartool -ss`** utilities to gather more information about the data flow (see Verification that PI Server processes are running.
    - Try retrieving a snapshot three different ways; the combined results of all three tests helps pinpoint the source of the problem:
        - **`apisnap`** from a remote node (uses API + network)
        - **`apisnap`** from the home node (uses API)
        - **`piconfig < pisnap.dif`** from the home node (uses internal communication)
    - Do a snapshot dump with **`piartool -sd`**. Run this a few times to determine if the snapshot is changing for the tags you are interested in.

- To determine if the archive is corrupt, use **`piartool -aw`** (see List archive record details (archive walk)
- If this is a PI Update Manager problem, use the **`pilistupd`** utility to see which processes are signed up for events. Use **`pilistupd -cs`** to see the list of consumers.

8. Each PI System is distributed with a standard set of points including sinusoid, cdep158, and cdm158. Verify that the points work properly. The RampSoak and Random interfaces must be running, otherwise these points are not updated.
9. For troubleshooting backups, see Back up PI Server.
10. For troubleshooting PI collectives, first use Collective Manager to check the status of all members. Next use **`piconfig < pisysdump.dif`**:
    - `isavailable` should be 1 for all members
    - `lastsynctime` indicates the last successful communication
    - `role` should be 1 for a primary and 2 for a secondary
11. If the problem is with interfaces, try the following:
    - Try running an interface with only one point.
    - Run the interface interactively.
    - Run the interface without buffering. When running interactively you will most likely be using a different account, so security can affect your results.
    - Determine if the problem is with all points on all interfaces or just a few points on some interfaces.
    - Verify that a PI trust exists for that node or specifically for that interface. If you are using buffering, a trust must exist for the buffering process if you are not using a machine trust.
    - Check the PI Firewall database.
    - Check the individual interface log files, if any; also check the PI Message Log on the interface node. Use the **`pigetmsg`** utility, located in the `pipc\adm` folder, to check messages in this file.
    - If an API interface is not able to connect, try to connect with **`apisnap`**.
    - Make sure the SDK can connect using the **`AboutPI-SDK`** utility.
    - Try running as Administrator. If the problem goes away when you run as the System Administrator, then you have a permission problem.
    - If this is the OPC interface, check DCOM settings. The settings are documented in the *OPC Interface Manual*.

General troubleshooting tasks

Topics in this section

- Verification that PI Server processes are running
- Connection checking
- Update Manager checking
- Configure Dr. Watson for Windows

Verification that PI Server processes are running

When PI Server is running, all its processes should be running. When PI Server is stopped, all PI Server processes should be stopped. The exception is `pishutev`, which only runs briefly at PI Server startup.

You can use the PI SMT PI Services tool to view the status of PI services. (Under **System Management Tools**, select **Operation** > **PI Services.**) Even when the tool lists a process as running, the process might not be communication with PI Network Manager. You can use **`piartool`** to verify that each process communicates properly:

```
piartool -thread subsystem -info
```

To list all processes that run as services, enter at the command prompt:

```
net start
```

If you are running a PI Server process or interface interactively, that process will have a separate command window labeled with the process name.

Topics in this section

- PI Archive Subsystem
- PI Base Subsystem
- PI License Manager
- PI Snapshot Subsystem
- PI Update Manager
- Independent operation of PI Server processes

PI Archive Subsystem

Run the **`piartool -al`** (archive listing), **`piartool -as`** (archive statistics), and **`pisnap`** commands to test PI Archive Subsystem. If PI Archive Subsystem (piarchss) is not working correctly, you will see:

```
C:\PI\adm>piartool -al
Getarchivefilelist Failed: [-10733] PINET: RPC Resolver is Off-Line.
C:\PI\adm>piartool -as
Getarcmemtablestatistics Failed: [-10733] PINET: RPC Resolver is Off-Line.
C:\PI\adm>pisnap
C:\PI\adm>apisnap localhost:5450

PI-API version 1.6.1.5
 Attempting connection to localhost:5450

Enter tagname:  sinusoid

Error:  piar_getarcvaluex -10733
Error:  piar_getarcvaluesx 100

  Tag = SINUSOID   Point Number = 1   Type = Real-32
  12 Hour Sine Wave

              Snapshot value
  Value = ERROR ERROR
  Status = ERROR
```

```
              Latest archive value
  Value = ERROR ERROR
  Status = ERROR
```

Archive events are queued when PI Archive Subsystem is not operating correctly. Use the **piartool -qs** command to view the event queue count.

PI Base Subsystem

Run the **pisnap** and **piconfig** utilities to test PI Base Subsystem. If PI Base Subsystem is not working correctly, you will see:

```
C:\PI\adm>apisnap localhost:5450

PI-API version 1.6.1.5
 Attempting connection to localhost:5450
 Error -994, connecting to localhost:5450
C:\PI\adm> piconfig
* (Ls - ) PIconfig> @tabl pipoint
*PIconfig Err> Table initialization error (PIPOINT
*@tabl pipoint
*[-10733] PINET: RPC Resolver is Off-Line.
```

PI License Manager

PI License Manager, **pilicmgr**, provides license services for PI programs including subsystems, client applications, and interfaces. For example, PI Archive Subsystem registers with PI License Manager to obtain a valid license. If it fails to get its license, it may not operate properly.

Run the **piartool -lic usage** command to test PI License Manager. If **pilicmgr** is not working correctly, you will see:

```
C:\PI\adm>piartool -lic usage
Continue after failure to register with License Manager. [-10733] PINET: RPC
Resolver is Off-Line.
```

PI Snapshot Subsystem

Run the **piartool -ss** and **pisnap** commands to test PI Snapshot Subsystem. If PI Snapshot Subsystem is not working correctly, you will see:

```
C:\PI\adm>piartool -ss
Getsnaptablestatistics Failed: [-10733] PINET: RPC Resolver is Off-Line.
C:\PI\adm>pisnap

C:\PI\adm>apisnap localhost:5450

PI-API version 1.6.1.5
 Attempting connection to localhost:5450

Enter tagname:  sinusoid

Error:  pisn_getsnapshotsx -10733
Error:  piar_getarcvaluex -10733
Error:  piar_getarcvaluesx 100

  Tag = SINUSOID   Point Number = 1   Type = Real-32
  12 Hour Sine Wave
```

```
              Snapshot value
Value = ERROR ERROR
Status = ERROR

            Latest archive value
Value = ERROR ERROR
Status = ERROR
```

PI Update Manager

Run the **`pilistupd -ss`** command to test PI Update Manager (**`piupdmgr`**). If PI Update Manager is not working correctly, you will see:

```
C:\PI\adm>pilistupd -ss
pilistupd -h      for help
[-10727] PINET: RPC is Non-Existent
Producer   Consumer   Qual. Flags Pending
---------  ---------  ------ ------ --------
status: [-12150] not registered in updmgr
```

Independent operation of PI Server processes

Under normal operation, all of the PI Server processes are started up together using the `pisrvstart` script, and are stopped together using the `pisrvstop` script. It is sometimes useful in troubleshooting to run a subset of the PI Server processes. On Windows, `pisrvstart.bat` starts each subsystem interactively in its own command window.

There are inter-process dependencies with the PI System. For example, all PI Server subsystems rely on PINet Manager. Most subsystems require PI License Manager, which provides license services, and PI Update Manager, which provides key services, to be running. Also, PI Archive Subsystem requires PI Snapshot Subsystem for normal startup. When troubleshooting, the following subsystems should generally be started in the order listed:

1. `pinetmgr`
2. `pimsgss`
3. `pilicmgr`
4. `piupdmgr`
5. `pibasess`
6. `pisnapss`
7. `piarchss`

Connection checking

For information about connections, look at the Network Manager statistics. You can see these in the PI SMT Network Manager Statistics tool (select **Operation** > **Network Manager Statistics**). The `pinetmgr` process manages the remote procedure calls (RPCs) that PI Server subsystems and processes use to communicate with each other.

For example, if PI Snapshot Subsystem (`pisnapss`) sends an event to PI Archive Subsystem (`piarchss`) for storage, the communication flows from `pisnapss` to `pinetmgr` to`piarchss`.

If PI Archive Subsystem writes a message to the System Message Log, the communication flows from `piarchss` to `pinetmgr` to `pimsgss`.

The Network Manager Statistics tool shows a lot of information. Some things to check for:

- Use the **BytesSent** and **BytesRecv** columns to identify applications that are requesting or sending unusual amounts of data. If the value for an application or interface is large compared to other applications or interfaces of that type, then that might point to the problem connection. (Values in the **BytesSent** and **BytesRecv** columns from PINetMgr will always be the highest.)
- What client processes are connected from which nodes.
- How long clients have been connected.
- You can tell what type of application is connecting by looking at the **RegAppType** column.
  - `OSISDKApp` indicates an SDK application.
  - `OSIinterface` indicates an interface.
- Use the **ProtocolVersion** column to tell whether the connection is from an SDK or API application. A version number of 1.x indicates an API application. A version number of 3.x indicates an SDK application.

Update Manager checking

Update Manager keeps track of *producers* of updates (such as the snapshot, the archive, and so on) and *consumers* of updates (such as interfaces, PI ProcessBook, or PI ACE). It lets clients and other consumers know when a value that they are interested in is updated.

If a client is not getting updates, check the Update Manager statistics for consumer and producer statistics with the **`pilistupd`** utility. The **`pilistupd -cs`** command gives consumer statistics and the **`pilistupd -ps`** command gives producer statistics. The **`pilistupd -h`** command gives information on usage. For more information on **`pilistupd`**, see the *PI Server Reference Guide*.

- Consumers

  Look for the following:

  - Is it the consumer registered?
  - Is the consumer timed out?
  - Is the consumer signed up?
  - When was the last time the consumer retrieved an update?
- Producers

  Look for the following:

  - Is the producer connected?
  - Is the producer sending updates?
  - When is the last time it sent an update?
  - How many sign-ups does it have?

For more on a specific producer, you can use **piartool -upd**. The syntax is:

```
piartool -upd subsystem producer
```

- See Producers and associated subsystems for a list of producers and associated subsystems.

  Look for:

  - Send failures
  - Retrying
  - Is the producer responsive?

If all the consumers and producers are communicating correctly, but no events are coming through, check the producer of the data (for example, snapshot, archive, or MDB).

Producers and associated subsystems

Producer	Description	Subsystem
Snapshots	Snapshot	Snapshot
Archive	Archive	Archive
PtUpdates	Point updates	Base
MDBUpdates	Module database	Base
PIChangeRecordUpdates	Changes for PI Server replication	Base
DigitalSets	Digital sets	Base
BDBUpdates	Batch database updates	Archive
PIBatchUpdates	Batch updates	Archive
PIUnitBatchUpdates	Unit batches	Archive
PIUnitBatchOnUnitUpdates	Unit batch updates for a specific unit	Archive
PICampaignUpdates	Campaigns	Archive
PITransferRecordUpdates	Transfer records	Archive

Configure Dr. Watson for Windows

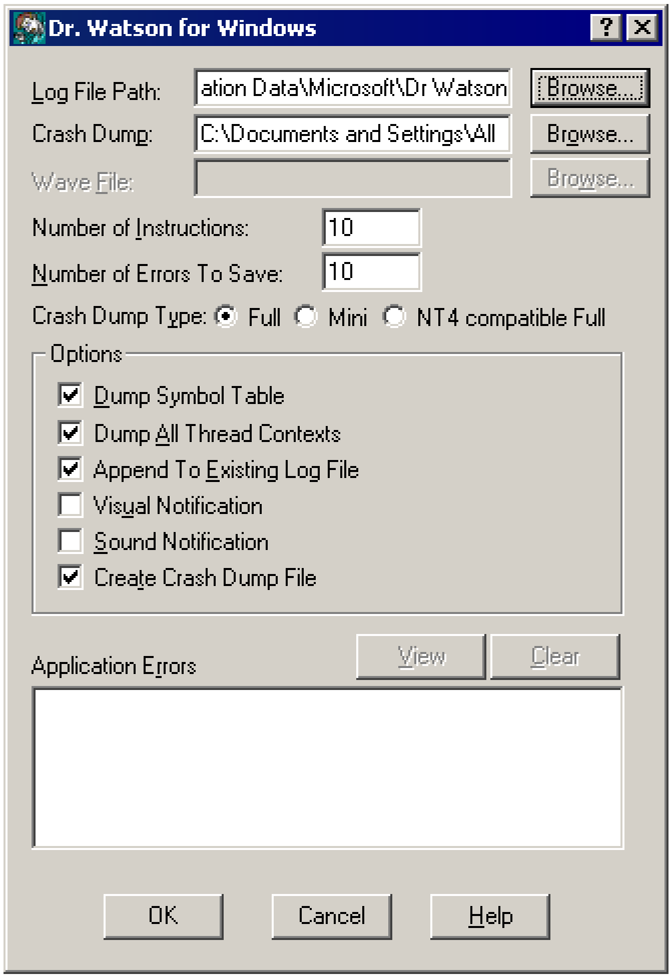
Dr. Watson is an application debugger included with some Microsoft operating systems. If your operating system includes Dr. Watson, you can configure it to be the default debugger and have it generate a crash file dump if your system experiences an error causing a computer crash. This file can provide useful data to OSIsoft Technical Support.

Procedure

1. At a command prompt, type:

   ```
   drwtsn32.exe -i
   ```

2. Enter `drwtsn32.exe` again, without the `-i` parameter:



Dr. Watson for Windows

Log File Path: ation Data\Microsoft\Dr Watson

Crash Dump: C:\Documents and Settings\All

Wave File:

Number of Instructions: 10

Number of Errors To Save: 10

Crash Dump Type: ● Full ○ Mini ○ NT4 compatible Full

Options

- ☑ Dump Symbol Table
- ☑ Dump All Thread Contexts
- ☑ Append To Existing Log File
- □ Visual Notification
- □ Sound Notification
- ☑ Create Crash Dump File

Application Errors

3. Use the window to specify these recommended settings:

    - **Log File Path**

      Use the default location.

    - **Crash Dump**

      Use the default name. This file may be overwritten; after a process crash, copy this file to a safe location.

    - **Number of Instructions**

      25

    - **Number of errors to save**

      25

    - **Crash dump type**

      `Full`

4. Select the check boxes next to these options:

    - **Dump symbol table**
    - **Dump all thread contacts**

- **Append to existing log file**
- **No visual notification**
- **No sound notification**
- **Create crash dump file**

5. Click **OK** to close the window.

After you finish

To test your selections, type `pidiag -crash` at the command prompt and examine the generated log files.

Specific problems

Topics in this section

- Slow connections and connection failures
- PI ProcessBook trend flatlines
- Abusive usage
- Problems with COM connectors

Slow connections and connection failures

PI Network Manager looks up the host name of all computers connecting with PI API. The name lookup is used to identify the connecting computer for trust login and PI Firewall access. A Reverse Name Lookup requests the Domain Name Server (DNS) to translate an IP address to host name. This request must complete in a reasonable amount of time for PI Server to function correctly.

Network systems with malfunctioning Reverse Name Lookup will experience slow connections or failure to connect to PI Server. Often the symptoms may be isolated to a subnet or computers connecting from outside the LAN. The standard TCP/IP utility, **`nslookup`**, can be used to check Reverse Name Lookup. If the **`nslookup`** utility reports a delay when resolving an IP address to name, the network has DNS problems that should be addressed. Resolving this problem is a system network configuration task and beyond the scope of this document.

If the problem cannot be resolved in a timely manner, you can temporarily disable the reverse name lookup feature. To do this, set the ReverseNameLookupFlag tuning parameter to zero (0). You can do this in PI SMT:

1. Select **Operation** > **Tuning Parameters**.
2. Click **New parameter** 🗋.
3. In **Parameter Name**, type:
   ```
   ReverseNameLookupFlag
   ```
4. In **Value**, type:
   ```
   0
   ```

5. Click **OK**.
6. Restart PI Server.

Note:

This setting (like any tuning parameter) is not replicated.

To re-enable reverse name lookup, set the value to a nonzero entry.

PI ProcessBook trend flatlines

If a PI ProcessBook trend flatlines, you may have a problem with PI ProcessBook, with the PI Server, with network performance/connectivity/configuration, or with the data source. Here are some possible diagnostic steps to take:

Procedure

1. Determine whether only one tag is affected or several are affected. Check another trend in ProcessBook to see if the problem is limited to only some points. If the problem involves multiple points, go to step 2. If the problem involves only one point, go to step 4.
2. Try retrieving the data from the PI archive by closing and reopening the trend window. If the trends appear normal, the problem may be in the update sign-up. Go to step 3. If the trends still show no data, go to step 4.
3. If no tags are producing trends, run the PI SMT tool **Operation** > **Update Manager** on the PI Server to see if the flat line is due to PI ProcessBook not being signed up for updates.
4. To determine whether the problem is with the PI Server or with the client application, view the pending numbers in the results. Pending numbers represent the number of events queued and not yet retrieved by the client such as PI ProcessBook. If data is not arriving from the data source, the pending number remains at 0. If the client PI ProcessBook is not retrieving the updates, the pending number continually grows and does not shrink.
5. If the pending updates are growing, try restarting the PI System. If this does not fix the problem, contact OSIsoft Technical Support (http://techsupport.osisoft.com) for additional assistance. If the pending updates remain zero, go to step 4.
6. If all the points are signed up for updates, and refreshing the data from the archive still yields a flat-line trend, the problem could be in the archive, in the data source, or in communications to the data source.
7. To determine if the server is working, create a trend for a point with data generated on the server, such as sinusoid, which is generated by the Random interface on the server.
8. If the trend for sinusoid appears correctly, it means that the archive is working and communication between Server and client is working. Then go to step 6.
9. If the trend for sinusoid does not appear correctly, go to step 5.
10. Use the **`piartool -as`** or **`piartool -al`** command to verify that the archive is functioning properly.
11. Use the **`piconfig`** utility, PI DataLink, or the PI SMT Archive Editor tool to try inserting data into a test point trend the point. If this is successful, go to step 6. If not, contact OSIsoft Technical Support (http://techsupport.osisoft.com) for additional assistance.

12. If all these tests are successful, the data source for the flat-lined points may not be working. Examine the source point attribute of the point to find out which interface is feeding data for the point in question.
13. Check the OSIsoft Technical Support (http://techsupport.osisoft.com) website to verify that the interface program is running and connected to the server.
14. Verify that the interface program is communicating with the external data source (DCS system, RDB system, and so on). See the documentation for the specific interface for details.
15. If the data source is running successfully, go to step 16.
16. Security may be preventing the process at some point. Examine the interface log files and the PI Server message log files. Verify that both the data source interface and PI ProcessBook have the correct access to the system. Examine all messages about trusts granted or refused.
17. If the points in question have some access restrictions, there must be established trust logins. The interface must have access as a PI user group, or PI identity with write access to the points. PI ProcessBook must have read access to all these points.

After you finish

If none of the above steps have resolved the problem, contact OSIsoft Technical Support (http://techsupport.osisoft.com) for additional assistance.

Abusive usage

If PI Server uses most of the CPU needed to pinpoint a problem, it may be due to improper sizing; that is, the configuration may be set too large for available hardware components, such as CPU, memory or disk.

The introduction of threading in release 3.4 solves many past performance issues; however, very large archive queries can still affect performance. The total number and size of queries can be monitored with the **`piartool -as`** command.

If the amount of read access and number of events retrieved seem excessive, use the activity grid. See Activity grid.

Also, the PINetMgrStats table in the **`piconfig`** utility or the PI SMT Network Manager Statistics tool can help identify network connections with the highest traffic.

Activity grid

PI Archive Subsystem provides a tool to monitor read-access to the archive. This tool creates, over a finite time period, a grid of activity. The grid provides an account of connections and point activity.

Start the activity grid to temporarily identify the connections that present the greatest load on the system and the points that are being queried most often.

This monitoring requires significant computing resources and therefore is normally turned off. Once the load on the system is identified, OSIsoft recommends that you turn off the activity grid.

The syntax for the activity grid is:

```
piartool -aag <start|stop|pause|point|cnxn> <event|access> -min n -max n
```

Note:

Earlier versions of PI Server collected the **aag** statistics by PI user instead of by connection (cnxn). The cnxn option replaces the option user, which is no longer supported.

Set up the activity grid to view activity per point or per connection. You can query by

- Event — How many events were read for this point or connection.
- Access — How many read calls were made to this point or by this connection. For example, one trend display call is counted as one access but can retrieve 100s of events.

To start the activity grid:

```
piartool -aag start
```

To stop it, and remove all its memory:

```
piartool -aag stop
```

To temporarily stop the accounting yet allow querying of the current statistics:

```
piartool -aag pause
```

Each query requests the number of events retrieved or the number of retrieval calls made (<event|access>). These can be arranged by points or by connection ID (<point|cnxn>):

```
piartool -aag <point|cnxn> <event|access>
```

The following gets the number of events retrieved by point, from the time the activity grid was started:

```
piartool -aag point event
```

It would return something that looks like this:

```
Count       ID-Name
--------------------------------------
        19        1-sinusoid
      2982        4-CDM158
         6       43-BaGen:bid.2
         6       57-BaGen:PIBatchIndex.1
        12       64-batchid-1
         6      336-BaGen:proc.2
         6      338-BaGen:prod.2
        12      350-productid-1
        40      368-piba1
     13544      388-BAE9CF24-C8B3-46c0-AD5D-A64DF2174ED9
```

In this mode, the ID-Name column corresponds to the point ID and tag name.

The following gets the number of retrieval calls, by connection ID:

```
piartool -aag cnxn access
```

Here is an example of what would be returned:

```
 Count       ID-Name
--------------------------------------
       131        7-PIBaGen.exe -
         4        9-pibatch -
        79       29-PIPESCHD -
         3      659-snapE - ::1
        18     9916-ptmoE - 10.0.0.6
        34    10248-SMTHost.exe(988):remote - 10.0.0.5
       120    10253-Procbook.exe(4848):remote - 10.0.0.5
        52    10255-EXCEL.EXE(4296):remote - 10.0.0.5
```

In this mode, the ID-Name column corresponds to the ID, Name, and PeerAddress as displayed in the PI SMT Network Manager Statistics tool (**Operation** > **Network Manager**). For local connections, the PeerAddress is `::1` or `127.0.0.1` or blank.

The following gets the number of events for each connection:

```
piartool -aag cnxn event
```

Here is an example of what would be returned:

```
      Count       ID-Name
--------------------------------------
     13875        7-PIBaGen.exe -
        44        9-pibatch -
      3045       29-PIPESCHD -
        80     9915-ptmon.exe -
```

Use `min/max` to limit the display to only the cases of interest (abusers). For example, to display only the connections that made more than 1,000 access calls, use the following:

```
piartool -aag cnxn access -min 1000
```

To display only the points that had more than 10,000 events read from since the activity grid was turned on, use the following:

```
piartool -aag cnxn event -min 10000
```

Problems with COM connectors

Occasionally, errors are observed when OSIsoft client applications fail to obtain process data. If the errors are related to a foreign data historian, the applications generally receive error codes in the range -11200 to -11209, instead of returning data. As usual, you can use the **`pidiag -e`** command to translate these errors to text.

The source of the error can be the Redirector or the specific COM connector in use. Before you troubleshoot a COM connection problem, you should verify that the Redirector is operating correctly. Errors may be logged in either the Windows event log or the PI message log. In general, the distinction is this:

- The Redirector logs information about its own activities to the Windows Event Application Log. This includes startup, shutdown, and loading of COM connectors.
- The PI Server subsystems record errors in foreign system point lookup and data retrieval in the PI message log.

This section gives some guidelines for troubleshooting data retrieval problems from COM connectors. As new techniques become available, they are posted on the OSIsoft Technical Support COM Connector page.

Topics in this section

- Check for mapped points in the point database
- Check for the PI Redirector process
- PI Redirector dump script
- Verify PI Redirector installation
- COM connectors

Check for mapped points in the point database

Mapped points should be correctly defined in the PI point database. A mapped point is one that has the three identifying point attributes: ctr_progid, ctr_strmap, and ctr_lmap.

Procedure

1. At a command prompt, go to the `PI\adm` directory
2. Type the following command:

   `piconfig`
3. At the **`piconfig`** prompt, type:

   ```
   @table pipoint,classicctr
   @mode list
   @ostructure tag, ctr_progid, ctr_strmap, ctr_lmap
   @select ptclassname=classicctr
   @ends
   ```

   Note that although this example uses classicctr point class, COM connectors are not limited to using this point class as long as the three identifying point attributes are present in their point class. The point class classicctr can be created using the **`piconfig`** file `classicctr.dif` provided with the PI Server installation kit. You can create your own point classes for PI Server mapped points.

Check for the PI Redirector process

If the PI Server mapped points are defined, a process called `piudsrdr.exe` should be running.

Procedure

1. Open Windows Task Manager.
2. On the **Processes** tab, confirm that the `piudsrdr.exe` process is running.

   > **Note:**
   >
   > After the piudsrdr service has been started, it remains in the list of running processes, even if all mapped points are subsequently deleted.

Topics in this section

- Check the PI message log
- Check the Windows Event Viewer
- Review Application Properties settings

Check the PI message log

If PI Redirector is not running, check the PI message log using the **`pigetmsg`** utility. Check for any messages related to PI Redirector or a COM connector. If the following message appears, it means that PI Redirector is not installed correctly:

```
0 pipoints 23-Jun-03 16:07:25
>> Error getting UDS Point interface. [-2147467261] Invalid pointer
```

Attempt to reinstall by opening a command window, setting your default directory to the `PI\bin` directory and issuing the command:

```
piudsrdr /RegServer
```

A Windows message is displayed in an alert box if the registration fails. Issuing this command if PI Redirector is already correctly installed has no effect.

Check the Windows Event Viewer

Use Event Viewer to troubleshoot a PI Redirector failure.

PI Redirector will also fail to start if the Windows event log exceeds the maximum log size. The default setting is to overwrite events as needed when the log file exceeds the pre-configured maximum size.

Procedure

1. Run Event Viewer.
2. Select **Application**.
3. Look for a message that indicates PI Redirector can start but cannot keep running.

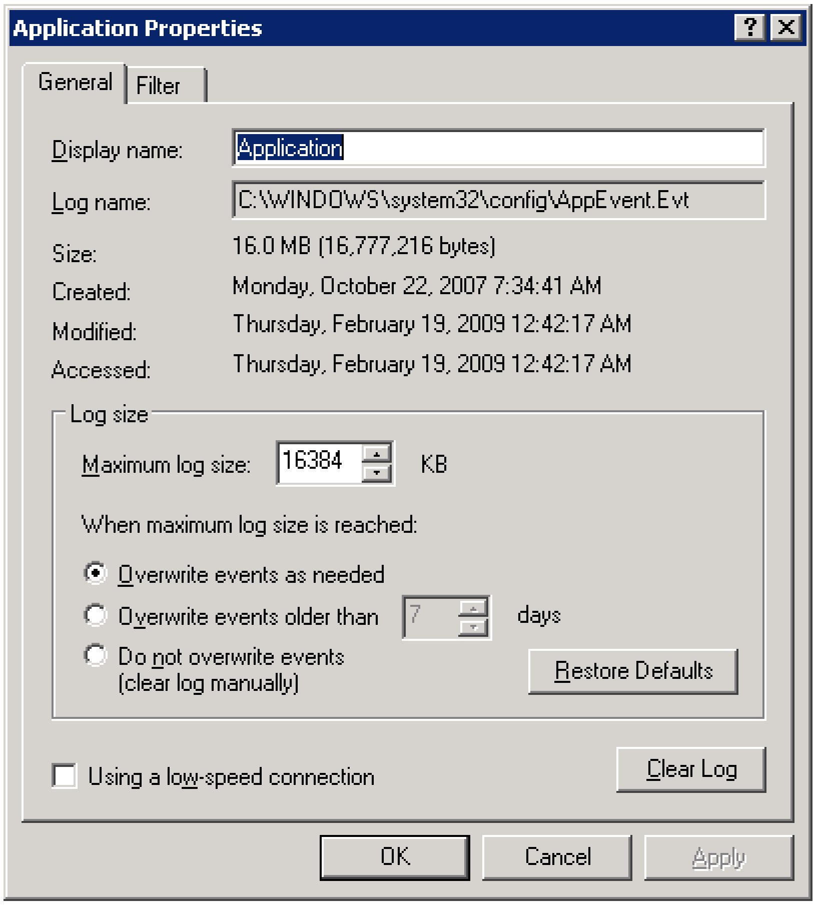
Review Application Properties settings

You should adjust the settings according to your site's policy regarding logs and your disk capacity.

Procedure

1. Run the Windows Event Viewer.
2. Select **Application** > **Properties.**

   The default settings are:



Application Properties

Display name: Application

Log name: C:\WINDOWS\system32\config\AppEvent.Evt

Size: 16.0 MB (16,777,216 bytes)

Created: Monday, October 22, 2007 7:34:41 AM

Modified: Thursday, February 19, 2009 12:42:17 AM

Accessed: Thursday, February 19, 2009 12:42:17 AM

Log size

Maximum log size: 16384 KB

When maximum log size is reached:

● Overwrite events as needed

○ Overwrite events older than 7 days

○ Do not overwrite events (clear log manually)

□ Using a low-speed connection

3. You can manually clear the log by clicking **Clear Log** to make room for more log events.

PI Redirector dump script

Every COM connector implements a method for obtaining information on its mapped points. You can obtain a script for requesting this information from the OSIsoft Technical Support website.

Use these guidelines to ensure this script works correctly:

- Set the identity of PI Redirector to `This User` in **`dcomcnfg`**; that is, some domain user with administrative privileges on the hosting machine.
- Restart PI Base Subsystem, PI Snapshot Subsystem, and PI Archive Subsystem *only* if the logged in user account is different from the account those subsystems are running under.
- If the identity is set to `The launching user`, any logged in user who runs the script is likely to start another instance of the Redirector. Such an instance of Redirector will not share information with the one started by PI Base Subsystem which contains the mapped point information.
- If a change is made to the identity setting, restart the Redirector by restarting PI Base Subsystem, PI Snapshot Subsystem, and PI Archive Subsystem.
- If the identity of PI Redirector is set to a specific user, you should make sure that all out-of-process COM connectors can be started and accessed by this user.

To find more information for troubleshooting PI Redirector and COM connectors, go to the OSIsoft Technical Support website and select **Products** > **COM Connectors** > **PI COM Connectors.**

Verify PI Redirector installation

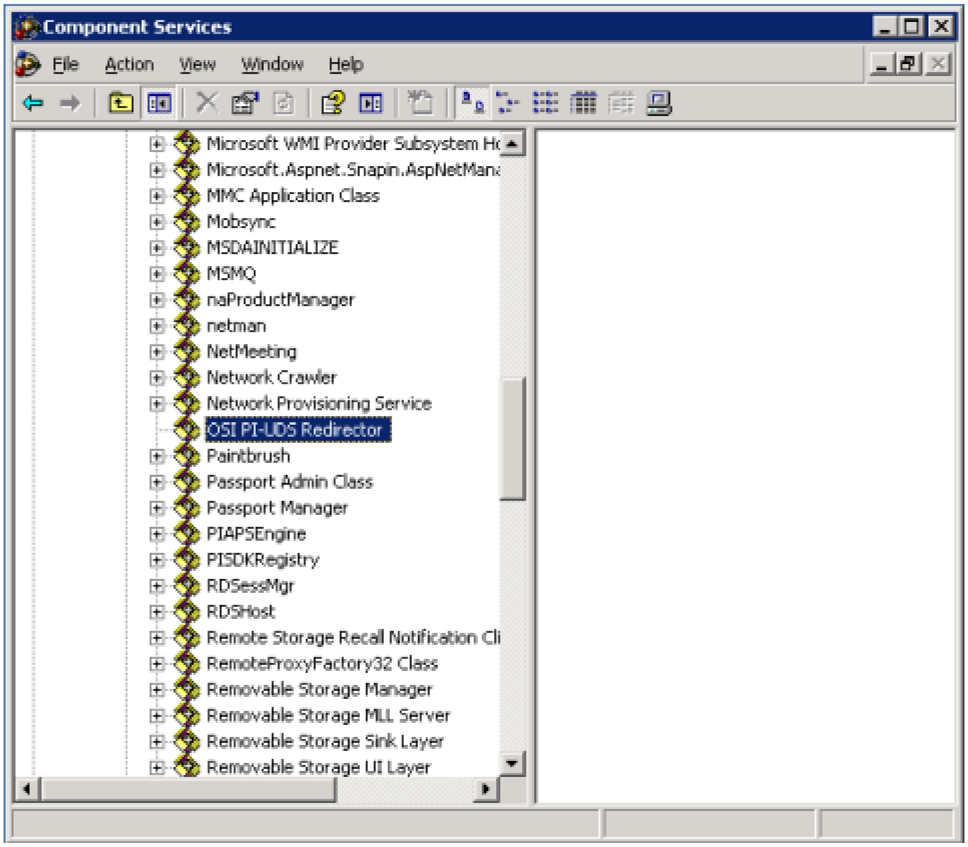
PI Redirector is not installed separately; it is installed as part of PI Server. To verify that PI Redirector was installed correctly, check the PI Server installation log file (`piudsinst.log`) located in the root of your system disk. Look for two lines reading:

```
Registering PI UDS Redirector.
Creating EventLog registry key for piudsrdr
```

You can use the Windows utility **`dcomcnfg`** to confirm installation and check the PI Redirector properties.

Procedure

1. At a command prompt, type `dcomcnfg` to open the Component Services window.
2. Under **Console Root**, expand **Component Services** > **Computers** > **My Computer** > **DCOM Config.**
3. Verify that **OSI PI-UDS Redirector** is included in the list of DCOM applications.


Component Services
File
Action
View
Window
Help
Microsoft WMI Provider Subsystem Hc
Microsoft.Aspnet.Snapin.AspNetMan
MMC Application Class
Mobsync
MSDAINITIALIZE
MSMQ
naProductManager
netman
NetMeeting
Network Crawler
Network Provisioning Service
OSI PI-UDS Redirector
Paintbrush
Passport Admin Class
Passport Manager
PIAPSEngine
PISDKRegistry
RDSessMgr
RDSHost
Remote Storage Recall Notification Cli
RemoteProxyFactory32 Class
Removable Storage Manager
Removable Storage MLL Server
Removable Storage Sink Layer
Removable Storage UI Layer

After you finish

After you have confirmed that OSI PI-UDS Redirector is installed, continue troubleshooting until the problem is solved.

After you fix the problem, you must restart PI Redirector. To restart PI Redictor, shut down PI Base Subsystem, PI Snapshot Subsystem, and PI Archive Subsystem, and then restarting them. There is no method to restart PI Redirector alone. PI Redirector is not started if there are no mapped points configured on the system.

COM connectors

If a COM connector is successfully loaded by PI Redirector, a message like this is written to the Windows event log:

```
Successfully registered PI Universal Data Server COM Connector
pipi.pipiconnector.
```

If the COM connector cannot be loaded, a message to this effect is logged. Troubleshooting steps depend on how the COM connector is implemented. For details, see the manual for the individual COM connector.

COM connectors are of two different types: in-process or out-of-process. The manual for the specific COM connector you are using will tell you the connector type.

Topics in this section

- In-process COM connector
- Out-of-process COM connector

In-process COM connector

An in-process COM connector is implemented as a DLL. When PI Base Subsystem loads a point that references the COM connector, this DLL is loaded into the process space of PI Redirector. You will not see a separate process running.

Check the Windows event log. If the COM connector is not registered, the message will say this.

In this case, attempt to re-register the COM connector by opening a Windows command window, setting your default directory to the location of the COM connector DLL and running the **regsvr32** utility. For example, if the COM connector DLL were named myconn.dll, enter:

```
regsvr32 myconn.dll
```

An alert box is displayed if the COM object implemented by the DLL cannot be registered. A common cause of inability to register is DLLs upon which the COM object DLL depend are not installed. The missing DLLs may be those provided by the foreign data historian vendor.

Note:

In-process COM objects are not visible to the **dcomcnfg** utility. One way of seeing the DLLs loaded by PI Redirector is to use the **ListDLLs** utility available from Microsoft.

Out-of-process COM connector

This type of COM connector will appear as a separate process in Windows Task Manager.

You should also check the COM object properties using **`dcomcnfg`**. The Properties and Identity should match those of PI Redirector, unless the COM connector's manual says otherwise.

If the COM connector does not appear in the **`dcomcnfg`** list, it has not been successfully registered. Attempt to re-register the COM connector by opening a Windows command window, setting your default directory to the location of the COM connector executable, and running it with the `/RegServer` switch. For example, if the COM connector executable were named `myconn.exe`, you would enter:

```
myconn.exe /RegServer
```

An alert box is displayed if dependent DLLs are missing. Other errors are not displayed.

Repairs

Topics in this section

- Repair the archive manager data file
- Archive file corruption
- Correction of archive event time stamps
- Snapshot recovery
- Database repair
- Diagnose and repair PI Server database files
- Recover from accidental system time change
- Excessive CPU usage by utilities

Repair the archive manager data file

The archive manager data file, `piarstat.dat`, contains the list of archive files known (registered) to PI Server. If this file is corrupted, you can use the **`pidiag`** utility to rebuild it.

pidiag option	Description
`-ad`	Dumps the current `piarstat.dat` file. This is used to review the data in the file.
`-ar`	Provides an interactive recovery utility for renaming the old piarstat.dat to piarstat.old and generating a new one with a single entry – the primary archive – provided by the user.
`-ara`	Provides an automated recovery utility that renames the old `piarstat.dat` to `piarstat.old` and generates a new one with all of the entries found in the original file. Any errors will cause the automated version to abort, and the user should resort to the interactive version.

Procedure

1. Copy the `piarstat.dat` file to a temporary location. Do not overwrite the original file. Rename the file, in case you need it later.
2. Stop PI Server.
3. Run the **`pidiag -ad`** command, collect the dump of the `piarstat.dat` file, and verify the output.
4. If the results of the dump look normal, attempt the automated recovery. Otherwise, use the interactive one.
5. Restart PI Server.

6. Check the message log to see if all archives are loaded. If the interactive version is used, only the primary archive is loaded.
7. Register any remaining archives. If the interactive version was used, all other archives must be registered.

Archive file corruption

Archive files have a header and a record structure. The records include data and auxiliary information to index the records and to link the records together for fast data access.

All data is susceptible to corruption if a system failure such as a power outage occurs. When archive file corruption occurs and the file becomes unreadable, it is important to recover the file to the most complete state possible.

You can use the offline archive utility, **`piarchss`**, to recover the data and rebuild the archive header and its associated metadata.

Topics in this section

- Recovery of a non-primary archive
- Recover a primary archive

Recovery of a non-primary archive

To recover the data from a corrupted non-primary archive, run **`piarchss`**, specifying the corrupted archive as the input file and a non-existing file as the output file. By default, the start and end times of the input archive are used as the start and end times of the newly created output archive.

It is possible to recover the data in a non-primary archive while the PI Server is still archiving data. The offline archive utility unregisters the archive during the recovery operation.

Here is an example command to recover a non-primary archive:

```
$ ../bin/piarchss -if /export/PI/dat/piarch.001 -of piarch1.fix -f 0
...First pass...
...Sorting input archive...
...Output pass...
676 Loaded in 2( 1 + 1 ) Seconds 338 Event/Sec.
739 Archive Total seconds - ratio: 369
```

In this example, the `piarch1.fix` file does not exist prior to the operation. It is created as a fixed archive the same size as the input archive because the `-f 0` option was specified. After it is created, it may be registered using the **`piartool -ar`** command, and then data events may be added to the archive in the usual way.

If the input file is registered prior to the recovery, it will be unregistered during the operation. You need to register the input file when the recovery is complete.

Recover a primary archive

PI Server cannot archive data during the recovery process if the corrupted archive is the primary archive. Because a primary archive cannot be unregistered, to recover it you must either:

- Stop PI Archive Subsystem
- Force a shift so that the archive is no longer the primary archive

To force a shift, use the **`piartool -fs`** command.

Procedure

1. Stop PI Archive Subsystem.
2. Run **`piarchss`** and specify the parameters:
   - Output archive is fixed size (`-f 0`)
   - End time left open (`-oet Primary`)
3. After recovery:
   a. Rename the old primary archive.
   b. Rename the output file to the same path and filename of the original primary archive.
   c. Restart PI Archive Subsystem.

Example of primary archive recovery

In this example, the `Failed to unregister input archive` message may be ignored. It occurs because PI archive subsystem was stopped prior to recovery.

```
$ ../bin/piarchss -if /export/PI/dat/piarch.005 -of piarch.005.fix
 -f 0 -oet 0
...First pass...
...Sorting input archive...
 Failed to unregister input archive: [-10733] PINET: RPC Resolver is Off-Line
 Archive utility not running - or archive not registered
 Continue processing...
...Output pass...
1084 Loaded in 2( 0 + 2 ) Seconds 542 Event/Sec.
1038 Archive Total seconds - ratio: 519
```

In this example, the `piarch.005.fix` file does not exist prior to the operation. It is created as a fixed archive the same size as the input archive because the `-f 0` parameter was specified. The end time of the output archive is left open because the `-oet 0` parameter was specified.

After you finish

Every archive file has a parallel annotation file, with an extension `.ann`. In PI Server version 3.3.361.43, the annotation file is identified incorrectly after renaming its associated archive file. Since renaming is necessary in this case, unregister the renamed file after initial registration, and re-register it.

Correction of archive event time stamps

Offline archive processing with time transformation differs little from standard offline archive processing. All arguments, such as input file and output file, must be specified. Additional arguments specify time transformation behavior. The additional arguments are:

```
-tfix time_conversion_file [-tfixend time -oeendtime time]
```

The argument `-tfix` followed by full file specification is required. The arguments `tfixend` and `oeendtime` are optional.

The first option, `-tfixend`, followed by a time stamp specifies the time to perform no transformations. All events with time stamps greater than or equal to this time will not be transformed.

This option is used when only a portion of the archive has incorrect event time stamps. For example, if a PI System was run for a period with incorrect system clock setting, then the clock was set correctly and run for some period before applying a time transformation fix.

The second option, `-oeendtime`, followed by a time stamp specifies a time stamp to set as the archive end time when conversion is complete. The archive end time is set to the passed value if all events are older than this time; otherwise, the end time is set to the time of the oldest archive event.

Time conversion file format

The time conversion file is an ASCII file containing a list of time-stamp/offset pairs. The time stamp and offset are separated with a comma. Lines beginning with #, and empty lines and white spaces are ignored.

The time stamp may be a local time string in PI time format; either a fixed time in the format `dd-mmm-yy hh:mm:ss` or a reference time, such as `*-300d` or `*`. Only one time-stamp format can be used in a given file. The first format encountered is assumed for all time stamps.

The offset is the number of seconds to add to the event time stamps. Sub-second precision of the time shift is not supported. The offset is applied to all events with time stamps greater than or equal to specified time stamp but less than next time stamp in the conversion file.

Example 1

The following example uses UTC seconds time format. The time stamp 0 is January 1, 1970, and the time stamp 2000000000 is well into the 21st century. The offset is a positive 3600 (one hour).

Therefore this data file will simply move all events ahead by one hour.

```
# Example 1, Moves entire archive ahead by 1 hour
0,3600
2000000000,3600
```

Example 2

Similar to the first example, this example uses local time stamps to specify a suitably large time range to cover all events. The offset is -3600. This data file will move all events back by one hour.

```
# Example 2, Also moves entire archive back by 1 hour
01-Jan-70 00:00:00,-3600
01-Jan-30 00:00:00,-3600
```

Example 3

This example applies a missed DST conversion for the Northern Hemisphere summer of 2003. The first time stamp is set at 01-Jan-03 to include all events up to the DST transition; no offset is applied up to, but not including 06-Apr-03 02:00:00. From 06-Apr-03 02:00:00 up to, but not including, 26-Oct-03 02:00:00 one hour is added to all events. No offset is applied from 26-Oct-03 02:00:00 to current time.

```
# Example 3, Applies a missed dst conversion to an
# archive that covers summer of 2003
```

```
01-Jan-03 00:00:00,0
06-Apr-03 02:00:00,3600
26-Oct-03 02:00:00,0
31-Dec-03 23:59:59,0
```

Snapshot recovery

There are two types of possible damage to the snapshot from which PI Server can recover:

- Snapshot times in the future. Accidentally moving the PI Server system time into the future can cause this; at a minimum all points collected locally will be in the future. Snapshot events are replaced when a newer value is received; therefore these events remain in the snapshot until actual time catches up. Events earlier than snapshot time bypass compression. Events that bypass compression can put a large load on your PI Server.
- Damaged or corrupted snapshot file (`piarcmem.dat`). Corruption may be caused by disk or power failures.

Topics in this section

- Recover from future times in the snapshot
- Rebuild the snapshot file
- Removal of future-time snapshots

Recover from future times in the snapshot

Procedure

1. Stop PI Snapshot Subsystem (the **`pisnapss`** process) on a running PI Server.
2. Restart PI Snapshot Subsystem from a command prompt and pass the `-f` argument:

   ```
   pisnapss -f
   ```

   This must be done interactively, not as a Windows service. On startup, PI Snapshot Subsystem looks for all snapshots more than 20 minutes in the future. These future snapshots are overwritten with a NULL value. PI Snapshot Subsystem reports the number of future events detected to the message log. If no future snapshots were detected, no fix messages are written to the message log. New incoming data immediately overwrites the NULL snapshot, even if the incoming value is out of order.

   PI Snapshot Subsystem continues to run normally after the fix.
3. Press Ctrl+C in the interactive **`pisnapss`** process and restart it as a service.

> **Note:**
>
> Snapshots fixed by this procedure remain set to NULL until a new snapshot event arrives. A NULL snapshot value is replaced by any new event that is received for a point, even if the event is an out-of-order event.

Rebuild the snapshot file

If you receive a message that indicates that PI Server is unable to start because the snapshot file, `piarcmem.dat`, cannot be loaded, it is necessary to generate a new snapshot file. A rebuilt snapshot file contains events based on the point database and, in some cases, digital states of SnapFix. The rebuilt snapshot is updated as the PI System receives new data.

Note:

If the `piarcmem.dat` file is damaged, OSIsoft recommends that you contact OSIsoft Technical Support for assistance with this procedure.

Procedure

1. Shut down PI Base Subsystem, PI Archive Subsystem, and PI Snapshot Subsystem.
2. At a command prompt, go to the `PI\bin` directory and type:

   ```
   pibasess -snapfix -file
   ```

3. As needed, enter additional parameters:

Parameters	Purpose
filename -ds	Provides the full file name of an optional input file to use in place of the `piarcmem.dat` file.
-ds *state*	Specifies a system digital state that will be inserted in the new file.
-maxtime	Sets the latest time stamp allowed in the snapshot; `pisnapss -f` truncates at `*+20` minutes.
time	Indicates a time limit beyond which the prior digital state of events will be replaced with SnapFix, or the digital state you specify with the state parameter.

   a. Enter a *filename* only if you have a file that you want your new `piarcmem.dat` file to be built from. If you want to rebuild the snapshot file based on the most current values in the snapshot, do not enter a *filename*.

      If a `piarcmem.dat` file exists, PI Base Subsystem will rename the `piarcmem.dat` file that contains the current snapshot values to `DD_MON_YY_piarcmem.dat`; then, it will build a new `piarcmem.dat` file from the renamed `piarcmem.dat` file. If there is no `piarcmem.dat` file, a new file is created.

   b. If you enter a *state*, you must use an existing digital state.

4. If duplicate points are found, use the `ptpurge` parameter to remove duplicate points:

   ```
   pibasess -snapfix -ptpurgepointtopurge
   ```

   or

   ```
   pibasess -snapfix -ptpurge filelist
   ```

   where *pointtopurge* is a single point, for example `mypoint`. If you want to use a file that contains the names of multiple points, use *filelist*. For example, `pointstodelete.dat`.

   When recovery is complete, PI Base Subsystem will write a final message to the screen and exit.

5. Restart PI Base Subsystem, PI Archive Subsystem, and PI Snapshot Subsystem.

Current snapshot values are preserved in the rebuilt `piarcmem.dat` file. Points that have no previous data will use the SnapFix digital state, or the digital state you specify with the state parameter, until the state is replaced by new snapshot values. Any snapshot record that does not have a matching point is removed.

> **Tip:**
>
> This snapshot recovery command can be run with the entire PIServer shut down.

Removal of future-time snapshots

The **`piconfig`** utility can be used to remove all or selected snapshot events. When the snapshot event is removed, PI Snapshot Subsystem attempts to retrieve the latest archived event from the archive and replace the snapshot event with it. That event is removed from the archive. If there are no events for the point in the archive, the snapshot is deleted and remains uninitialized until a new snapshot event is sent.

The following **`piconfig`** script shows how to do that:

```
piconfig table pisnap
* (Ls - PISNAP) piconfig> @sele tag=*,time>"*+10m"
* (Ls - PISNAP) piconfig> @ostru tag,value,time
* (Ls - PISNAP) piconfig> @sigd 8
* (Ls - PISNAP) piconfig> @output deletesnap.dat
@endsection
@output
* (Ls - PISNAP) piconfig> @table piarc
* (Ls - PIARC) piconfig> @mode ed,t
* (Ed - PIARC) piconfig> @modify mode=remove
* (Ed - PIARC) piconfig> @istru tag,value,time
* (Ed - PIARC) piconfig> @echo v
* (Ed - PIARC) piconfig> @input deletesnap.dat
```

The first part extracts all the events that are later than 10 minutes past the current time into a file. The second part (using the PIARC table) removes all these events from the snapshot. The last archive event for each tag replaces the snapshot.

Any combination of conditions can be used to select the events to be deleted, for example all tags of a specific interface.

> **Note:**
>
> The significant digit command, `@sigd 8`, ensures that rounding errors do not cause values to be missed.

Database repair

If PI Base Subsystem does not start due to a corrupted database, the log shows a message indicating this. If there is no such message, start PI Base Subsystem in interactive mode and observe the errors in the window. Database corruption is a relatively rare event. It is usually due to hardware failure or improper shutdown. To repair all databases:

```
 pibasess -copydb path
```

For example:

```
pibasess -copydb \pi\recovereddb
```

Following this command, the target path contains recovered copies of all the configuration databases. Corrupted records are eliminated and related messages displayed.

Be sure to make a backup copy of the `\pi\dat\` directory before you copy the recovered database files back into this directory.

After that, PI Base Subsystem should load all databases and work normally. The resulting files are slightly smaller than the originals as they are compacted in the process.

Repair the PI Module Database

Procedure

1. At a command prompt, go to the `PI\bin` directory.
2. Type the following command:

   ```
   pibasess -mdbfix
   ```

   PI Base Subsystem completes several tasks:

   - Checks table and index entry count size. The entry counts should be the same.
   - Removes modules that have a record size of zero. These modules would be unrecoverable.
   - Removes parent and children references to non-existent modules.

Diagnose and repair PI Server database files

The PI Server stores most of its internal data in files that have a common internal structure, called a file-base structure. These database files store data in indexed records. PI archive files are not file-base files, though the corresponding annotation files are file-base files.

This section discusses the tools that you can use to diagnose and repair file-base database files.

Topics in this section

- List the header and index
- Compress a file base file
- Recover file base file

List the header and index

Before you start

To run this command, you must shut down the subsystem that owns the file.

Procedure

1. In a command line window, change to the `PI\adm` directory.
2. Type:

   ```
   pidiag -fb path [-header | -dv]
   ```

   where *path* is the complete path of the file.

   Use the `-header` option to list only the header (no index). Use the `-dv` option to display only the file's version.

If the command returns an error, see Recover file base file to fix the error.

With the header information, OSIsoft Technical Support can determine if there are any errors in the structure of the file:

```
D:\PI\adm>pidiag -fb d:\pi\dat\pidigst.dat -header
        PIfilebaseheader[$Workfile: pifile.cxx $ $Revision: 125 $]::
          File Name:          D:\PI\dat\pidigst.dat
          Major Version:      4
          Minor Version:      0
          Byte Alignment:     1
          Directory Location: 1024
          Directory Size:     1024
          Record Count:       18
          Last Recno:         0
          Maximum Recno:      128
          User Block Size:    512
          Data Location:      2048
          Data Size:          23325
          Auto Compact %:     0
          Last Modified:      10-Sep-09 09:45:11
          Backup Time:        25-Aug-09 14:26:11
          PIsecureobject[$Workfile: pisecobj.cxx $ $Revision: 46 $]::
            ACL ID: 1 [ 1:A(r,w)|5:A(r)|2:A(r) ]
        % unused:   0
```

Compress a file base file

If a file has more than 10 percent of unused space, you can compress the file to save disk space.

Before you start

To run this command, you must shut down the subsystem that owns the file.

Procedure

1. Find the percentage of unused space in a file. See List the header and index.
2. In a command line window, change to the `PI\adm` directory
3. Type:

   `pidiag -fbc`*`path`* `[-header]`

   where `path` is the complete path of the file.

   Use the `-header` option to list only the header.

Recover file base file

When a file-base file has a corrupted index or inaccessible or corrupted records, you can recover the file. Use this procedure to recover readable data records from a file-base file and rebuild the index.

Note:

When recovering files, this command discards unreadable records and prints an error message. Only use this tool at the direction of OSIsoft Technical Support.

Procedure

1. In a command prompt window, change to the `PI\adm` directory
2. Type:

   ```
   pidiag -fbf inpath outpath [alignment][-compress][-header]
   ```

   where the options are as described in the following table:

Option	Description
inpath	Path and name of input file.
outpath	Path and name of output file.
alignment	*Optional*. Sets the byte alignment in the output file. A value of 0 or 1 lets the file grow to 2GB. Higher values let the file grow to 2GB times the specified value. For example, if you specify an alignment of 2, the file can grow to 4GB, or if you specify an alignment of 4, the file can grow to 8GB.
-compress	*Optional*. Removes the unused records at the end of the file and allocates unused disk space.
-header	*Optional*. Displays only the header of the new file rather than both the header and index of the new file after recovery.

Results

In some cases **`pidiag -fbf`** will report the following:

```
Error reading input record # nn [-10466] No Record Available for Passed recno
```

This is normal for records between the actual last record and the maximum allocated record. The warning disappears if you run the utility a second time.

Recover from accidental system time change

PI Server handles automatically all changes to system time. Thus we recommend that you *never* manually change the system time. On Windows, always use the automatic `DST` option.

However, occasionally such changes are required, and unfortunately, from time to time this change leads to human errors. For example instead of moving the clock to 2 a.m. it is moved to 2 p.m. Time synchronization software, designed to keep computer clocks accurate without error-prone human intervention, have also been implicated in moving system clocks erroneously. As a result, the events are recorded in the future. Usually this is discovered after many of these events were already stored in the archive.

Procedure

1. Stop the PI System.
2. Correct the system time and the time on all connected nodes.

   > **Note:**
   >
   > If you are using PI Buffer Subsystem to buffer data from PI interfaces, see Recover from accidental time change at interface node that uses PI Buffer Subsystem.

3. Isolate the PI Server from interface nodes. The best technique is to disconnect the server from the network. While fixing the PI Server, it is best to allow the data to buffer until the system is verified up and running normally.

4. Rename the event queue file, `pimapevq.dat`, for later processing. The event queue may contain many future events. Rename the following files located in the `dat` directory:
   - `pilastsnap.dat`
   - `pilasttot_T.dat`
   - `pilastalarm.dat`
5. Create an empty archive file using PI SMT or the **`piarcreate`**utility.
6. Type `pidiag -ar` and register only the new empty archive.
7. There are two options for fixing the snapshot:
   - If the erroneous future data can be discarded, start PI Snapshot Subsystem with `-f` flag as described in Recover from future times in the snapshot.
   - Otherwise, keep the current file, and after the system startup, delete or edit individual values using the **`piconfig`** utility, as explained above.
8. Start the PI Server in base mode:

   ```
   pisrvstart -base
   ```

   This starts only the minimum required subsystems: PI Network Manager, PI Message Subsystem, PI License Manager, PI Update Manager, PI Snapshot Subsystem, PI Archive Subsystem, and PI Base Subsystem.
9. Register all the old archive files except for the previous primary, which contains future data.
10. Examine the unregistered archive file header to confirm the time boundaries of the various archives involved before using offline archive processing to merge archives:
    a. To look at the header of an unregistered archive:

    ```
    pidiag -ahd
    ```

    b. To look at registered archives:

    ```
    piartool -al
    ```
11. Create a new primary archive using the **`piartool -ac`** command.
    a. Specify a start time before any events that might be coming in. Specify the end-time as *.

    This instructs PI Archive Subsystem to register the new archive as primary archive. The start time specified must account for all buffered data. If you are unsure, set the start time well before the time the problem was first encountered.

    b. If necessary, use offline archive processing later to merge this data with existing archives.
12. Verify that the PI System is running correctly. Reconnect the server to the network.
13. Reprocess the old primary archive using the offline tool to either filter out the future data, or correct its time by the required difference.
14. Reprocess the event queue into an archive file and correct time stamps as required.
15. If desired, combine two archives: the old primary and the result of the event queue.
16. Register the corrected archive file.

Recover from accidental time change at interface node that uses PI Buffer Subsystem

If the time stamp on an interface node were changed to a future time stamp and that node uses PI Buffer Subsystem, you must complete additional steps.

Procedure

1. Stop PI Buffer Subsystem.
2. Stop the interface nodes.
3. Delete the `pibufmem.dat` file.
4. Restart PI Buffer Subsystem.
5. Restart the interface nodes.

Excessive CPU usage by utilities

The utilities **piconfig**, **pigetmsg**, **pilistupd** and **piartool** may use excessive CPU. You can fix this problem by increasing the timeout values for these utilities. This prevents the utilities from using when listening to messages.

Use the **piconfig** utility to increase the values:

```
piconfig> @table pitimeout
piconfig> @mode edit
piconfig> @istructure name, value
piconfig> piartool, 100
piconfig> piconfig,1000
piconfig> pigetmsg,1000
piconfig> pilistupd,1000
piconfig> @endsection
```

This message would typically be viewed with the **pigetmsg** utility or in the `log/pinetmgr.log` file. This error is due to insufficient resources available to complete the transfer of a large message. The fix is to increase the default timeout and the number of retries PI Network Manager uses for message transfer. Read and write timeouts default to 50,000 microseconds, read and write retries default to 250. We recommend increasing the timeouts in increments of approximately 25 percent until the errors disappear. If the errors persist when the timeout values are over 150,000 microseconds, contact OSIsoft Technical Support (http://techsupport.osisoft.com).

Use the **piconfig** utility to increase the timeouts:

```
piconfig> @table pitimeout
piconfig> @mode edit
piconfig> @istructure name, value
piconfig> readtimeout,62500
piconfig> writetimeout,62500
piconfig> readretry,350
piconfig> writeretry,350
piconfig> @endsection
```

Note:

PI Server installation sets all timeout values to well-tested initial values. Changes to these values should be done under the advice of OSIsoft Technical Support. Very short timeout values may cause specific utilities to spin faster and thus use more CPU. Before making changes based on CPU consumption, isolate the CPU to the offending processes. Use available tools to analyze each process. For example, if the pisnapss process is in a high CPU state, use the **`piartool -ss`** command to look at snapshot read and write rates because excessive rates may be the true source of CPU load.

Technical support and other resources

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

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

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

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