



PI Web Services 2012

User Guide

OSIsoft, LLC

777 Davis St., Suite 250

San Leandro, CA 94577 USA

Tel: (01) 510-297-5800

Fax: (01) 510-357-8136

Web: <http://www.osisoft.com>

PI Web Services 2012 User Guide

Copyright © 2009-2013 OSIsoft, LLC. All rights reserved.

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

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

U.S. GOVERNMENT RIGHTS

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

Version: 1.3.0

Published: February 2013

Table of Contents

About this guide.....	1
New features in PI Web Services 2012.....	3
Product Overview.....	5
Architecture.....	5
Role of PI Data Services.....	6
Summary of features.....	6
Retrieval of data updates.....	8
Connections to the PI System.....	9
Automatic resolution of new servers.....	9
Connections to PI collectives and AF collectives.....	9
Automatic WSDL generation.....	10
Error and trace message log configuration.....	10
About the OSIsoft PI Data Access suite.....	11
PI Web Services installation.....	13
Supported operating systems.....	13
Production deployments.....	13
Test deployments.....	14
Host PI Web Services on a web server.....	14
Review PI Web Services system requirements.....	14
Install PI Web Services IIS Edition.....	21
Verify PI Web Services IIS Edition installation.....	22
PI Web Services post-installation configuration.....	29
Host PI Web Services on a PI WebParts server.....	33
Host PI Web Services with a Windows service.....	33
Install PI Web Services Standalone Edition.....	34
Verify PI Web Services Standalone Edition installation.....	35
PI Web Services silent installation.....	36
PI Web Services security.....	37
Supported security scenarios.....	37
Requirements to secure access through PI Asset Framework.....	37
User accounts required for secure connections.....	38
Verify connection identities.....	38
End user folder and registry permissions.....	39
Security settings in the PI Web Services configuration file.....	40
Binding types used by PI Web Services.....	40
Secure access to PI Server data.....	42
Configure Security for Web Service Bindings.....	44
Configure security for applications.....	46
Security binding samples.....	48
Firewall security.....	52
Configure firewall exceptions.....	53

Configure firewall exceptions on Windows Server 2008 R2.....	53
PI Web Services Standalone Edition port settings.....	55
Reserve Namespace on Windows 2003 Server.....	55
Reserve namespace on the Windows operating system.....	55
Troubleshooting.....	57
PI Web Services IIS Edition.....	57
Cannot connect to PI Web Services.....	57
Server not found.....	57
PI System not found.....	58
Remote connection failure.....	58
Data entry not allowed.....	58
Insufficient permissions.....	58
Insufficient message size.....	59
Execution of PI Performance Equations not allowed.....	59
Unsupported filters or parameters.....	59
Server error in PI Web Services application.....	60
Invalid tag name.....	60
Anonymous authentication required.....	61
PI Web Services Standalone Edition.....	62
If the service does not start.....	62
PI Web Services programmer reference.....	65
Web service inputs.....	65
PI Web Services paths.....	65
Manner.....	68
Constraints.....	68
Time stamps.....	68
IPITimeSeries interface.....	71
IPITimeSeries web methods.....	71
IPITimeSeries classes.....	81
IPISearch interface.....	91
IPISearch web methods.....	91
IPISearch classes.....	93
IPISoap interface.....	93
IPISoap web methods.....	93
IPISoap classes.....	102
Use InfoPath with PI Web Services.....	117
Configure access to PI System data.....	117
Design the form.....	120
Change the time field default.....	124
Test the form.....	126
View and configure message logs.....	129
View PI Web Services message logs.....	129
Show PI Web Services messages in PI SMT.....	129
View messages in PI SMT.....	130
Show PI Web Services messages in DebugView tool.....	130
Enable DebugView tool.....	131
Suppress duplicate messages.....	131

Change levels of messages displayed.....	132
Configure Kerberos authentication.....	133
Prerequisites for Kerberos delegation.....	133
Enable Kerberos on a SharePoint Web server.....	133
Verify that user accounts can use Kerberos delegation.....	134
Associate an SPN with the application pool domain account.....	134
Use of Basic authentication with SSL.....	135
Web sites that share an IP address and port number.....	135
Configuration that shares IP address and port number.....	136
Use of load-balanced Web farms.....	136
Troubleshooting.....	136
Verify Kerberos configuration on client computer.....	136
Verify Kerberos configuration on Web server.....	137
Resources.....	137
Technical support and other resources.....	139

About this guide

This guide includes procedures to install and configure PI Web Services on a web server, or through a Windows service. It also provides information about how to ensure your PI System data is transmitted securely and how to resolve common errors that PI Web Services users and developers might encounter. To help you get started with your security setup, PI Web Services includes configuration files you can use to modify security settings and control application behavior.

The PI Web Services Programmer Reference describes the programming components, including interfaces, web methods, classes and properties, which are required to build custom web service applications that interact with PI Systems through web services.

For more information about building and deploying web services client applications that access PI Web Services and using third-party web services clients to invoke the web methods, OSIsoft recommends that you consult programmer resources such as the [MSDN Web site](#).

For supplementary support and information about PI Web Services, see the [OSIsoft vCampus Web site](#), which provides various technical resources including a blog, a discussion forum and webinars dedicated to PI Web Services.

New features in PI Web Services 2012

PI Web Services 2012 has added the capabilities to search for PI Asset Framework (PI AF) element templates, PI AF elements and attributes, and PI event frames.

New features included in PI Web Services 2012 provide users with the ability to:

- Search for PI Asset Framework (PI AF) element templates
- Search for PI AF elements and associated attributes, improving the access to time-series data in an asset-centric way
- Search for PI event frames, including the ability to navigate to referenced elements and their attributes to obtain time-series data
- Write values to AF attribute element paths and event frame attributes



Note: InsertPIData currently only supports writing to PI Point paths

- Retrieve PI event frames statistics, such as the total time covered, average/minimum/maximum length, and the count
- Retrieve time-series data for PI event frame attributes
- Search for AF element attributes directly, without going through their parent element
- Request interpolated time-series data by time interval (StartTime, EndTime, TimeStep), in addition to requesting it by number of number of interpolations (StartTime, EndTime, NumInterval)

Product Overview

PI Web Services enables PI System data to be transmitted through standard Internet protocols. When used with the web services client application of your choice, PI Web Services allows end users to submit simple web services requests that send and receive PI System data.

The ability to call web services is common to most operating systems; therefore developers can create web service client applications on the operating system platforms of their choice. PI Web Services provides the programmatic interfaces and web methods that developers need to create client applications that use web service calls to securely pass PI System data over networks.

For example, you might use PI Web Services to build solutions that:

- Display PI data on a web site
- Integrate with line-of-business systems that support web service calls
- Allow both Windows and non-Windows environments to easily access PI data, or submit data to PI systems

Some key advantages that PI Web Services provides are:

- The ability to integrate with applications, regardless of the programming language or platform
- Broad access to PI System data and features
- Interfaces that are designed to work with client tools that require only configuration – not code – to set up web service queries
- Central administration and regulation
- Standards-based interoperability, particularly WS-Interoperability and WS-Security
- Readily available secure connections

PI Web Services is a member of the PI Data Access product suite.

Architecture

A typical scenario for using PI Web Services includes computers that host:

- PI Web Services
- PI Asset Framework (PI AF) or a PI AF collective
- PI Server, or PI collective
- Web services applications

PI Web Services is typically hosted on an Internet Information Services (IIS) Web server. You may also host PI Web Services as a Windows service. For details, see [Host PI Web Services with a Windows service](#).



Note:

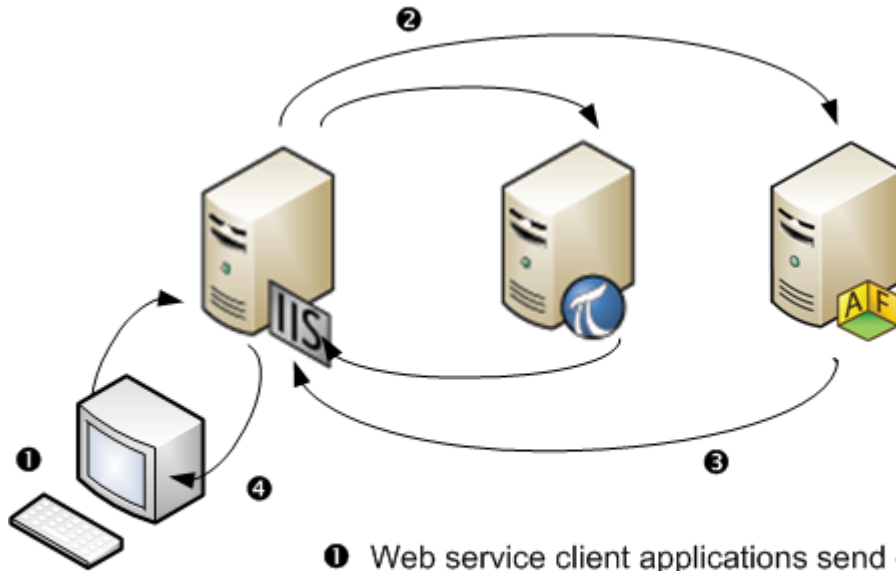
If you are using the `netTCPBinding` or `netNamedPipes` bindings and host PI Web Services with IIS, you must use IIS 7 or later. For details, see [Security settings in the PI Web Services configuration file](#).

For details about how to connect PI Web Services to a PI System, see [Verify PI Web Services IIS Edition installation](#) and [Use of endpoint and active configuration bindings](#). For an example that shows how to set up the connections to Web services applications, see [Configure Security for .NET Clients](#).



Note:

PI AF 2.4 or later is required if you plan to use PI Web Services to search for PI event frame data.



- ❶ Web service client applications send queries to Web server
- ❷ PI Web Services translates and forwards queries to PI System
- ❸ PI System data is returned to PI Web Services
- ❹ PI Web Services returns data to requesting client applications

Architecture of PI Web Services hosted on an Internet Information Services Web server

End-users enter queries into Web services applications on the client machines. The queries are sent to the Web server hosting PI Web Services, where they are checked for errors and translated into calls to PI Data Services, the data access layer that PI Web Services uses to access data from the PI System.

In response to the queries, PI System data is returned by PI Data Services to PI Web Services, where it is reformatted into the data structures defined by the PI Web Services interfaces. These structures and data representations adhere to Web service standards and are designed to provide maximum interoperability with third-party software. The reformatted data is returned to the client that hosts the Web service application, where the user can work with the data.

Role of PI Data Services

PI Web Services uses assemblies from the PI Data Services component that enable PI Web Services to retrieve and display data from PI Systems.

When PI Web Services makes a query, this underlying data engine translates the format of the data passed to the web service interface into a format that can be consumed by PI Data Services, and in turn retrieves and passes PI System data to PI Web Services.

These assemblies are installed on the server, affect required user accounts, and are referenced in some configuration files and path settings. However, PI Data Services is not visible as a separate component when you install PI Web Services. It operates in the background, and requires no administration.

Summary of features

PI Web Services consists of a Windows Communication Foundation (WCF) service that allows users to access PI System data through Simple Object Access Protocol (SOAP) compliant web services clients.

The web service provides three interfaces to provide these features:

- **IPITimeSeries**
Retrieves time series data supported by the PI System and receive updates to supported data events, including PI points and time series data that are referenced by PI AF element and event frame attributes.
- **IPISearch**
Retrieves PI System data and metadata based on search criteria.
- **IPISoap**
Searches for PI Asset Framework data references and current values of attributes for PI event frames. Searches PI systems and returns collections of PI data, including AF elements and attributes, and PI event frames.

These interfaces are optimized for client tools that enable you to set up web service calls with configuration only; no code is required. See [Use InfoPath with PI Web Services](#) for an example.

In addition, PI Web Services supports:

- Java-based applications, and web service calls from non-Windows operating systems
- These standard PI summary calculations: averages, minima, maxima, ranges, totals, values, counts, and both sample and population standard deviations
- The ability to send and receive PI Web Services calls to and from PI collectives or PI AF collectives
- Both synchronous and asynchronous calls when supported by the transport and client proxy technology. As an example, .NET and Silverlight asynchronous clients are fully supported.
- An option to host PI Web Services as a Microsoft Windows service, rather than in Internet Information Services (IIS). See [Host PI Web Services with a Windows service](#)

PI Web Services includes web methods that:

- Support entry of PI Server data
- Allow users to sign up for data updates
- Search for PI AF data references
- Search for current values of PI event frame attributes
- Search for and retrieve PI AF elements and data
- Search for and retrieve PI event frames and data

Features not supported are:

- Annotations. None of the IPITimeSeries interface methods contain a member to store annotation data, such as text, BLOB, and so on. Although the flag will show a value of **A** when the value is annotated, the annotation data is not accessible.
- Use of the InsertPIData method with a performance equation path or a PI AF data reference path.
- Writing to PI event frames through time series data

Retrieval of data updates

PI Web Services includes web methods that can retrieve changes to events that occur after a web services request is made. Changes to a PI System that make time series data, PI points, or PI AF data references more current are known as **updates**.

In PI Web Services, updates are events that reflect changes in the data for the PI points or PI AF data references that are named by the path in the web services call.

With the updates features, you can create web service clients that return to the PI Server or PI Asset Framework and retrieve updates that occur since the prior request, for a particular path or collection of paths. For example, you might create a web service client to implement updated trends or revise calculations.

Updates are supported only on PI Server paths or PI AF paths that refer to simple PI point data reference configurations, that is, data reference configurations that do not use advanced settings such as time offsets or event weights. Updates are retrieved from the PI Server, where they are stored as either snapshot values or archive events.

Updates in PI Systems typically consist of new events with time stamps later than the time stamps of snapshot events. A PI System can, of course, accept addition and deletion of older events. When this occurs, the web service client will receive these older events as updates. Updates are always snapshot or archive values, even if the sign-up occurred during a query for interpolated or plot values data.

Sign up for data updates

Web services clients must **sign up** to receive updates with either implicit sign-ups that occur through an existing query, or explicit sign-ups that pass one or more paths to the Web method `SignUpForPIUpdates`. Regardless of the method used, your application will receive a special identifier called an update ticket as the Web method response if the sign-up is successful. Your application must then pass the update ticket to the Web method `GetPIUpdates` to obtain any new or changed events. When the application is finished processing new events, you can call `CancelPIUpdates`.

There is no limit on the number of times that your Web service client application can call `GetPIUpdates`. You should be aware that sign-ups expire if the `GetPIUpdates` Web method is not called often enough for any given update ticket. Sign-ups expire after 5 minutes by default. This default can be adjusted through a setting in the `web.config` file. Your application can also set its own expiration time when it calls `SignUpForPIUpdates`. The expiration feature prevents consumption of system resources when lost or ill-behaved clients fail to request updates.

Your application may call the Web method `ListPathsByUpdateTicket` at any time to retrieve the list of paths associated with a update ticket. Calls to both `GetPIUpdates` and `ListPathsByUpdateTicket` reset the internal timer used to check for expiration of sign-ups.

Explicit sign-ups

Your application signs up explicitly for updates by passing one or more paths to the web method `SignUpForPIUpdates`. If successful, this web method returns a single update ticket that represents sign-ups for all passed paths. If any passed path cannot support updates, an error is returned for that path. If none of the passed paths can support updates, `SignUpForPIUpdates` will return a SOAP fault.

`SignUpForPIUpdates` allows you to specify the expiration time of the sign-up. It is not possible to specify an expiration time when using implicit sign-ups.

Implicit sign-ups

You can submit a data retrieval request for data and sign up for updates in a single operation. This implicit sign-up occurs if you set the Updates property of a PIArcManner or PISummaryManner object to **TRUE** for a call to GetPIArchiveData or GetPISummaryData, respectively. For paths that support updates, the web method returns the update ticket in the SeriesID property of each TimeSeries object.



Note:

Each update ticket obtained this way represents a sign-up for a single path passed to GetPIArchiveData or GetPISummaryData. To create an update ticket that represents sign-ups for multiple paths, you must use an explicit sign-up.

Implicit sign-ups for updates are supported only for data retrieval requests with time ranges that end in current time, or *. Updates always consist of all snapshot or archive values even if the sign-up occurred during a query for interpolated, plot values or summary data.

Connections to the PI System

PI Web Services retrieves PI System data using the PI SDK and AF SDK, and other data access layers, that may require pre-configuration of the server that hosts PI Web Services. In general, the PI Web Services host must be configured with connection information to the desired PI Servers and PI AF servers. In some cases, this can happen automatically; see [Automatic resolution of new servers](#). In order to forestall any possible name resolution errors, OSIsoft recommends that connections to desired PI Servers and PI AF servers be created and tested in advance whenever practical; consult the documentation for the PI SDK and the PI Asset Framework Help for more information. In addition, when retrieving data from a collective that contains multiple PI Servers or PI AF servers, it may be desirable to specify the connection priority of individual servers in the collective; see [Connections to PI collectives and AF collectives](#) for details.

Automatic resolution of new servers

Both the PI SDK (for PI Server paths and PI Performance Equation paths), and the AF SDK (for all other path types) support the automatic resolution of new servers, and their addition to the Known Servers Table (KST).

- For the PI SDK, this is a configurable setting. If the behavior is enabled (the default), the SDK will attempt to resolve the unknown server and add it to the KST automatically. If the behavior is disabled, you will get an error message.
- For the AF SDK, the SDK will always attempt to resolve the unknown server and add it to the KST automatically.

In all cases where automatic resolution is successful, PI Web Services will access the requested data normally, and the new server will be added to the KST for subsequent calls.

Connections to PI collectives and AF collectives

When PI Web Services establishes a connection to a high availability PI System to retrieve data from a collective made up of PI Servers or PI AF servers, it uses a connection type of **Any**, which means that PI Web Services will connect to either a primary or secondary member of the collective. All web service clients connected to the same web server will retrieve data from the same PI Server by default.

You can adjust of priorities of the collective members on the web server hosting PI Web Services. Higher priority members will receive client connections first, provided that the PI Server or PI AF server is available. Connection priorities within a collective of PI Servers can be adjusted using the PI Connection Manager in **PI SDK Utility**. Use **PI System Explorer** for a collective made up of PI AF servers. For details, see the Help for each application.

Automatic WSDL generation

PI Web service interfaces are communicated to client development tools through XML documents in the Web Service Description Language (WSDL). Most client programming tools read the WSDL and generate client proxy code. The WSDL document describes the methods, data structures, and any security measures used by the web service.

PI Web Services dynamically generates a WSDL document on demand.

You can view the WSDL for the IPITimeSeries interface if you enter the URL: `http://webservername/PIWebServices/PITimeSeries.svc?wsdl`

To view the WSDL for the IPISearch interface, enter: `http://webservername/PIWebServices/IPISearch.svc`

To view the WSDL for the IPISoap interface, enter: `http://webservername/PIWebServices/IPISoap.svc`

Error and trace message log configuration

PI Web Services includes an instrumentation framework that manages performance counters and message logging. By default, the instrumentation framework is configured to log error messages to the Windows event log on the Web server.

To change the reporting level of logging, enable debug trace messages or add additional destinations for messages, edit the `PIInstrumentation.config` file found in the root directory of the Web service.

Deployments that use PI Web Services can write trace and error messages to the Windows event log on the server hosting PI Web Services and the PI message log. You can also write messages to the standard debug message window. This window can be read by Microsoft Visual Studio or by the free DebugView application. Visit the Microsoft Web site to [download the DebugView](#) application.

For details, see [View and configure message logs](#).

About the OSIsoft PI Data Access suite

The OSIsoft PI Data Access product suite is designed to support implementation of custom applications on top of the PI System, as well as integration of PI System data with other applications and business systems such as Microsoft Office or SQL Server, Enterprise Resource Planning systems (ERPs), Web portals, and maintenance systems, just to name a few.

The PI Data Access suite of products covers a wide range of use cases in various environments, programming languages, operating systems and infrastructures. Product features include:

- SQL-based data access (PI OLEDB Provider, PI OLEDB Enterprise, PI JDBC driver)
- OPC specifications (PI OPC DA/HDA Server)
- Service-oriented architecture (PI Web Services)
- Programmatic access (PI SDK and AF SDK)

Licensing for the PI Data Access products is divided into development and runtime licenses. Developers and integrators obtain development licenses for most PI Data Access components through memberships to the OSIsoft Virtual Campus ([vCampus](#)) program. For details, see the OSIsoft vCampus [Frequently Asked Questions](#).

The PI System Access (PSA) license enables end users to access PI System data, including time-series data in PI Servers and asset metadata in PI AF servers. PSA is a runtime license to access PI System data using any of the programmatic access methods licensed through the PSA, including PI Web Services. For more information, see the [OSIsoft website](#) or contact [OSIsoft Technical Support](#).

PI Web Services installation

PI Web Services is typically run on top of Internet Information Services (IIS) for Windows Server, a Microsoft web server environment.

Before you run the PI Web Services setup kit, verify that the web server you use meets the system requirements and that the required pre-installation tasks have been completed.

When the default options are used, the PI Web Services IIS Edition setup kit runs and installs:

- OS/soft MS Runtime Redistributable 3.1.3
- Microsoft .NET Framework 3.5
- Microsoft .NET Framework 4
- PI SDK 2012
- PI Asset Framework 2012 client
- PI OLEDB Provider 2010 R3
- PI OLEDB Enterprise 2012
- PI Web Services 2012



Note: You can also host PI Web Services as a standalone Microsoft Windows service, independent of Microsoft IIS. See [Host PI Web Services with a Windows service](#) that is included with the setup kit.

Supported operating systems

Production deployments

OS/soft recommends that you deploy PI Web Services 2012 on the following Microsoft Windows Server operating systems, in decreasing order of recommendation:

- Windows Server 2008 R2 SP1 (or later)
All editions, in both Full and Core installations
- Windows Server 2012
All editions, in both Full and Core installations
- Windows Server 2008 SP2 (or later), 32- and 64-bits
All editions, in Full installation only



Note: Windows Server 2008 Core installation is not supported due to dependencies with the Microsoft .NET Framework runtime

- Windows Server 2003 SP2 (or later), 32- and 64-bits
All editions

Test deployments

For development, staging, or testing purposes, PI Web Services may also be deployed on the following Microsoft Windows client operating systems:

- Windows 8, 32-bit and 64-bit
- Windows 7 SP1 (or later), 32- and 64-bit

Host PI Web Services on a web server

PI Web Services IIS Edition is hosted by Microsoft Internet Information Services (IIS) for Windows Server. The web server on which you install PI Web Services must include:

- A website to host PI Web Services. The PI Web Services setup kit will create a default website in IIS. If you do not want to use the default website:
 - You must designate a website to host PI Web Services before you run the setup kit.
 - You can set up this website when you install PI Web Services by selecting the location of the website root directory and naming its virtual directory, or selecting the default.
 - To customize the root directory, use the IIS Manager **Add a New Web site** feature before you install PI Web Services.
 - Verify that the web server that hosts the website has folder and registry permissions that are compatible with PI Web Services.

When PI Web Services is hosted on a Microsoft Internet Information Services (IIS) for Windows Server, the web server that hosts PI Web Services must be configured to include the settings listed here.



Note: If the server you use does not have these IIS features activated, the setup kit will fail.

- ASP.NET
- .NET Extensibility
- ISAPI Extensions
- ISAPI Filters for .NET Framework 4
- IIS 6 Management Compatibility
- IIS 6 WMI Compatibility
- IIS 6 Management Console

If you are using Windows 7, you must also enable these settings:

- Default Document
- Static Content

Refer to these topics for details about how to configure IIS, based on the Microsoft Windows operating system version, and then install and verify that PI Web Services is ready to use.

Review PI Web Services system requirements

Before you start

Verify the system you plan to use has the user accounts required for PI Web Services.



Note:

The PI Web Services setup kit checks for the minimum system requirements. If the setup kit cannot verify the requirements described here, the installation will fail.

Procedure

1. Verify that the server that runs PI Web Services includes:

- Windows Server 2008 R2, Microsoft Windows Server 2008 with Service Pack 2, Windows Server 2003 with Service Pack 2, or Windows Server 2003 R2, Windows 7, Windows 2012, or Windows 8.



Note:

For all operating systems, you must be logged in as an administrator if you run the PI Web Services setup kit.

- Microsoft Internet Information Services (IIS) for Windows Server that is enabled to use the Windows features described in [PI Web Services IIS Edition requirements](#).



Note:

The PI Web Services setup kit contains both the 32-bit and 64-bit version of PI Web Services. The setup kit installs the appropriate version based on the operating system that you use.

2. Verify that the server that runs PI Web Services has network access to one or more of these OS/soft servers:

- PI Server 3.3 or later
- PI AF server 2.1 or later that includes PI SQL for PI AF server



Note:

To search for PI event frames, you must use PI AF server 2.4 or later.

After you finish

- If the server you use also runs PI WebParts, see [Host PI Web Services on a PI WebParts server](#).
- See [Secure access to PI Server data](#) and [Requirements to secure access through PI Asset Framework](#) for information about how to configure security settings on PI System servers.

Install and configure IIS on Windows Server 2003

These topics explain how to install and configure Internet Information Services (IIS) on Microsoft Windows Server 2003 for use with PI Web Services.

Install IIS on Windows Server 2003

Procedure

1. Click **Start**, point to **Control Panel**, and then click **Add or Remove Programs**.
2. In **Add or Remove Programs**, click **Add/Remove Windows Components**.
3. In the **Windows Components Wizard**, under **Components**, select **Application Server**.
4. Click **Next**.
5. After the wizard completes the installation, click **Finish**.
6. Verify that an ISAPI extension for .NET 4 is enabled. Change this setting if required.

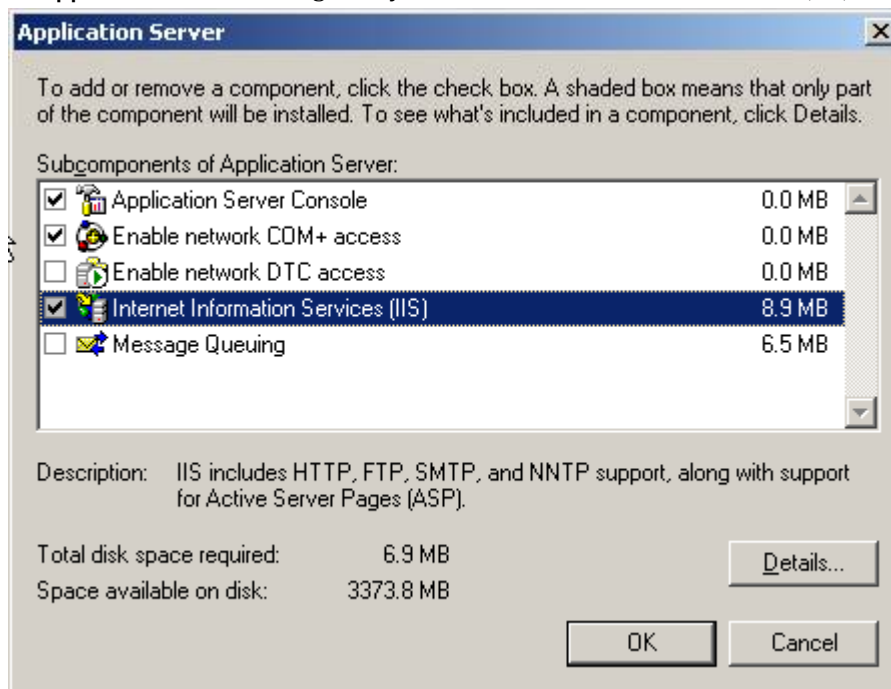
After you finish

Configure IIS Application Server settings.

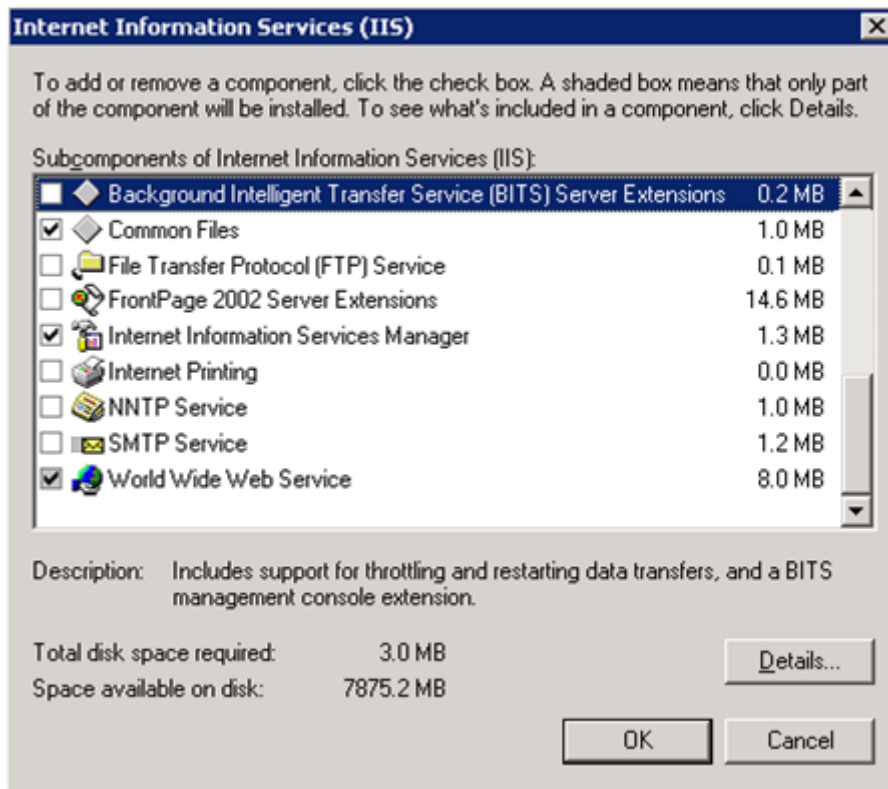
Configure IIS Application Server settings

Procedure

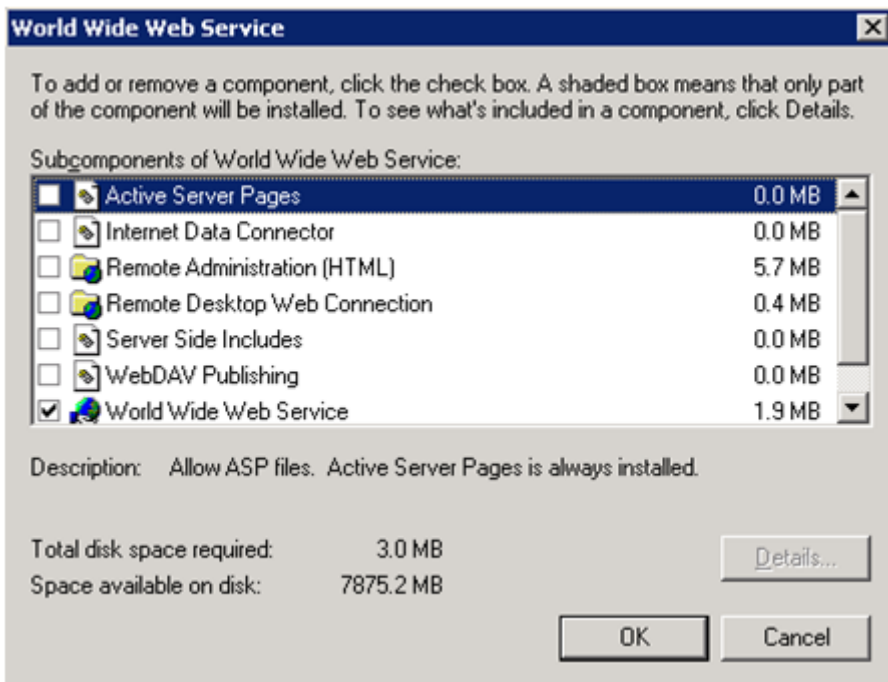
1. Choose **Start > Control Panel > Add or Remove Programs** and click **Add/Remove Windows Components**.
2. In the **Windows Component Wizard**, under **Components**, select **Application Server** and click **Details**.
3. In **Application Server** dialog, verify that **Internet Information Services (IIS)** is selected:



4. Select **Internet Information Services** in **Application Server** and click **Details**.
5. Verify that **Internet Information Services Manager** and **World Wide Web Service** are selected in **Internet Information Services (IIS)**:



6. Select **World Wide Web Service** in **Internet Information Services (IIS)** and click **Details**.
7. Verify that **World Wide Web Service** is selected in **World Wide Web Service**:



8. Click **OK** in each of the three open windows.
9. Click **Next** in the **Windows Component Wizard**.
10. Click **Finish** in the **Windows Component Wizard**.

After you finish

Run the PI Web Services setup kit.

Install and configure IIS on Windows Server 2008 R2

Before you run the PI Web Services setup kit, the Microsoft Windows Server 2008 R2 that you use to host PI Web Services must be installed and configured as described here.

Review Role Services in IIS Server Manager

Procedure

1. Choose **Start > Administrative Tools > Server Manager**.
2. Select the Roles home page under the server where you want to install PI Web Services.
3. Review the list of Role Services and verify that these services are installed: Under Application Development:
 - ASP.NET
 - .NET Extensibility
 - ISAPI Extensions
 - ISAPI FiltersUnder Management Tools:
 - IIS 6 Management Compatibility
 - IIS 6 WMI Compatibility
 - IIS 6 Management Console
4. If these role services are not installed, then add the IIS Web Server role and the required role services.

After you finish

Add web server role and services.

Add web server role and services

Before you start

Procedure

1. In **Server Manager** right-click **Roles >Add Roles**.
2. Click **Next** to use the **Add Roles Wizard** to add the web server role.
3. Click **Server Roles** and select **Web Server (IIS)** the **Roles** window.
4. Click **Next**.
5. Review the information in the **Web Server (IIS)** window and click **Next**.
6. Under Application Development in the **Role services** list:
 - a. Select ASP.NET
 - b. Click **Add Required Role Services** in the **Add Roles Wizard** pop-up. You can click pop-up links for further information.

7. Select IIS 6 Management Compatibility in the **Role services** list and then click **Next**.
 - a. Verify that these items are listed as **Installed** in the **Roles** list:
 - Under Application Development:
 - ASP.NET
 - .NET Extensibility
 - ISAPI Extensions
 - ISAPI Filters for .NET Framework 4
 - Under IIS 6 Management Compatibility:
 - IIS 6 Management Compatibility
 - IIS 6 WMI Compatibility
 - IIS 6 Management Console
8. Review your selections in the **Confirm Installation Selections** window and then click **Install**.
9. Verify that the web server was installed and click **Close**.

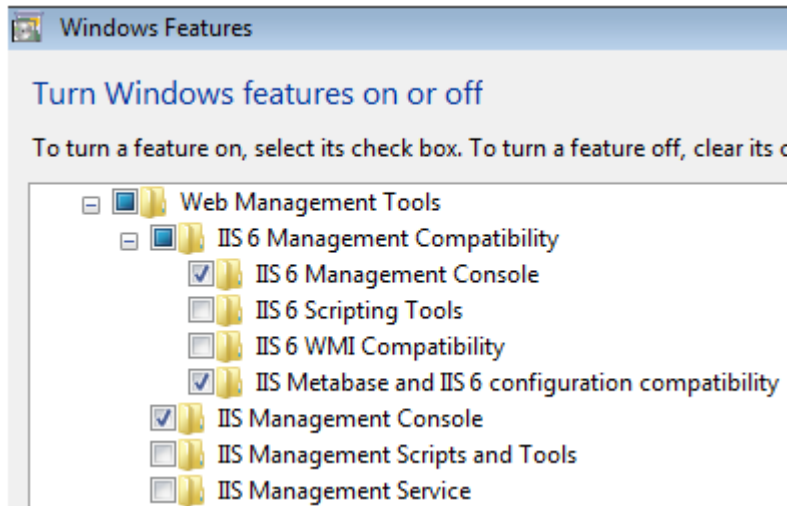
After you finish

Run the PI Web Services setup kit.

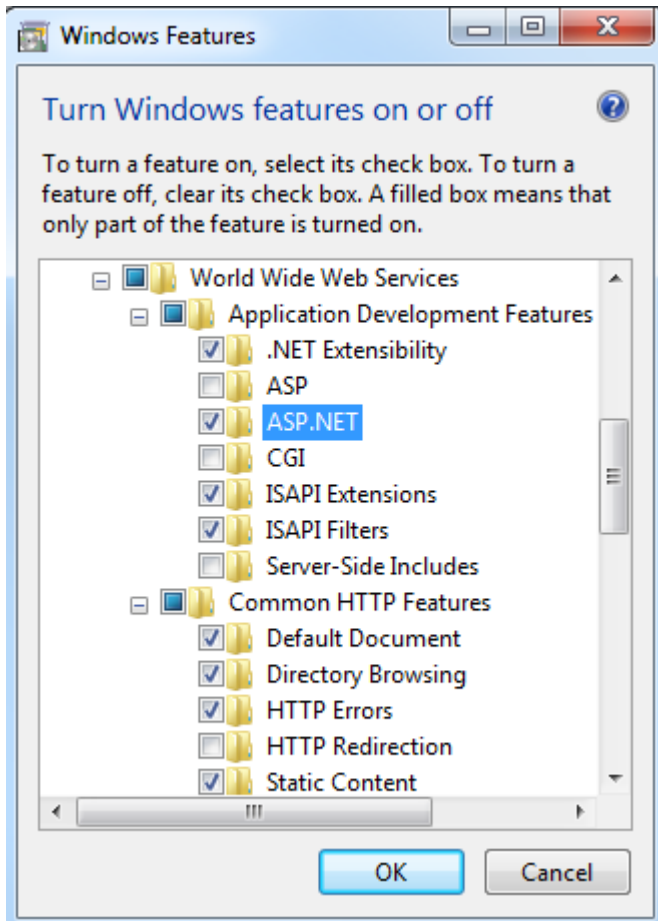
Install and configure IIS on Windows 7

Procedure

1. Choose **Control Panel > Programs** and click **Turn Windows features on or off**.
2. In **Windows Features**, click to expand **Internet Information Services**.
3. Click to expand:
 - For Vista, **Internet Information Services**, and then **Web Management Tools**, and then **IIS 6 Management Compatibility**.
 - For Windows 7, **Web Management Tools**, and then **IIS 6 Management Compatibility**.
4. Verify that:
 - a. Under **IIS 6 Management Compatibility**, these components are selected:
 - **IIS 6 Management Console**
 - **IIS Metabase and IIS 6 configuration compatibility**
 - b. **IIS Management Console** is selected.



5. Click to expand **World Wide Web Services** and then:
 - a. Click to expand **Application Development Features** and verify that these settings are selected:
 - **.NET Extensibility**
 - **ASP.NET**, if it is available in the list. If not, then verify that ASP.NET is installed after you install IIS.
 - **ISAPI Extensions**
 - **ISAPI Filters**
 - b. Click to expand **Common Http Features** and verify that these settings are selected:
 - **Default Document**
 - **HTTP Errors**
 - **Static Content**



c. Click **OK**.

After you finish

Run the PI Web Services setup kit.

Install PI Web Services IIS Edition

If you upgrade from an earlier version of PI Web Services, the PI Web Services setup kit will:

- Rename your existing `web.config` file to `old_web.config`
- Upgrade PI Web Services in the directory used for the earlier PI Web Services installation.

Before you start

Before you install PI Web Services on Microsoft Windows Server 2008, Windows 7, or later, verify that you are logged in as an administrator.

Procedure

1. Verify that the server on which you plan to install PI Web Services includes:

- All system requirements
 - Microsoft Internet Information Services (IIS) on Windows Server configured as described in [PI Web Services IIS Edition requirements](#).
2. Run the PI Web Services setup kit:
 - a. Click **Start** to start the PI SDK installation. During the PI Software Development Kit (PI SDK) installation, enter the appropriate information when you are prompted for:
 - user information
 - a path to a folder where PI SDK will be installed
 - names for a default user and a default PI Server



Note:

It is possible to buffer data sent to the PI Server to guard against loss of connection. Data is not buffered by default. To enable buffering, use the **PI SDK Utility**. See the *PI SDK Utility Help* for details.

- b. Click **Finish** to complete the PI SDK installation.



Note:

If you run the PI Web Services setup kit on a 64-bit Windows server, repeat the PI SDK installation steps to install PI SDK for 32-bit operating systems.

- c. During the PI AF client setup, enter:
 - user information
 - a path to a folder where PI AF client will be installed
 - the name of the default PI AF server to which PI Web Services will connect
 - d. Review the list of PI AF client features that will be installed and click **Next**.
 - e. Click **Finish** to complete the PI AF client installation.
 - f. During the installation of PI Web Services, use the **Select Installation Address** window to select:
 - **Site**
A website location to host the website content and web service files used by Internet Information Services (IIS) on Windows Server. By default, this file tree is stored in [wwwroot] \PIWebServices
 - **Virtual directory**
A name for the IIS virtual directory. The default value is PIWebServices.
 - **Installation directory**
You have the option to select a path to an installation folder for PI Web Services. If you are upgrading PI Web Services, the upgrade is installed in the folder used for the previous installation.
 3. Click **Next** to start the installation of PI Web Services.
 4. Click **Close** to exit the setup kit.

Verify PI Web Services IIS Edition installation

Test the web server connection

Use these steps to verify that you can connect to the web server if you use the PI Web Services IIS Edition.

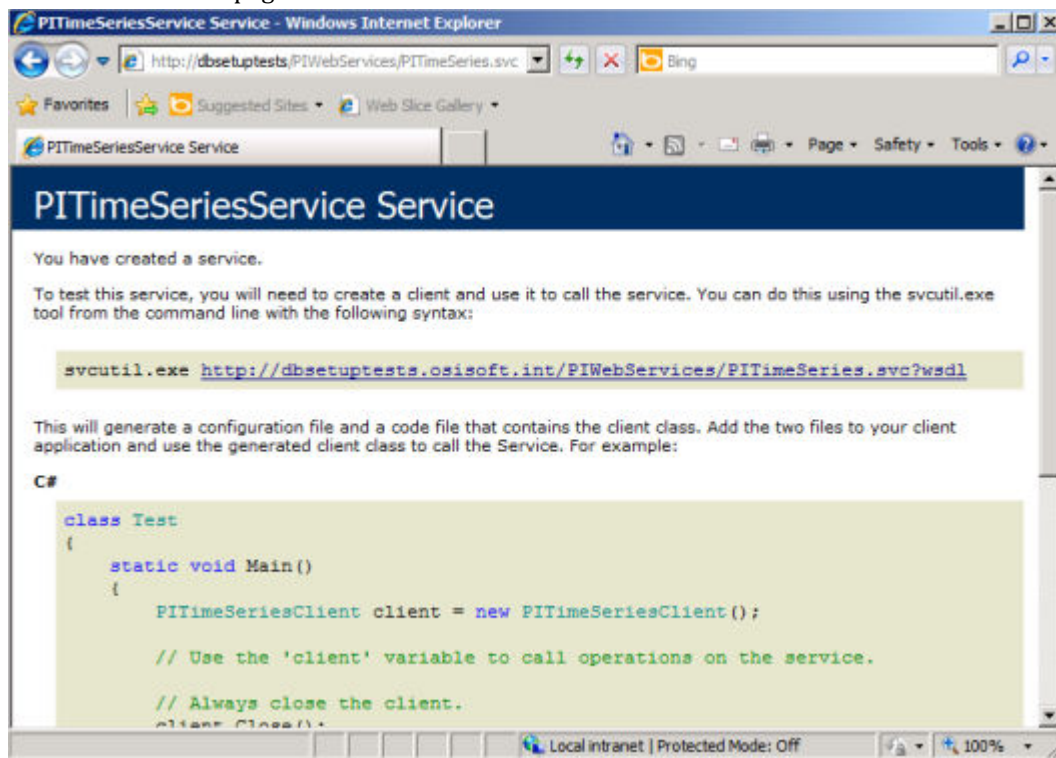


Note: To find the installation path for PI Web Services, go to the Windows Registry directory: HKLM \SOFTWARE\PISystem\PI Web Services\CurrentInstallationPath.

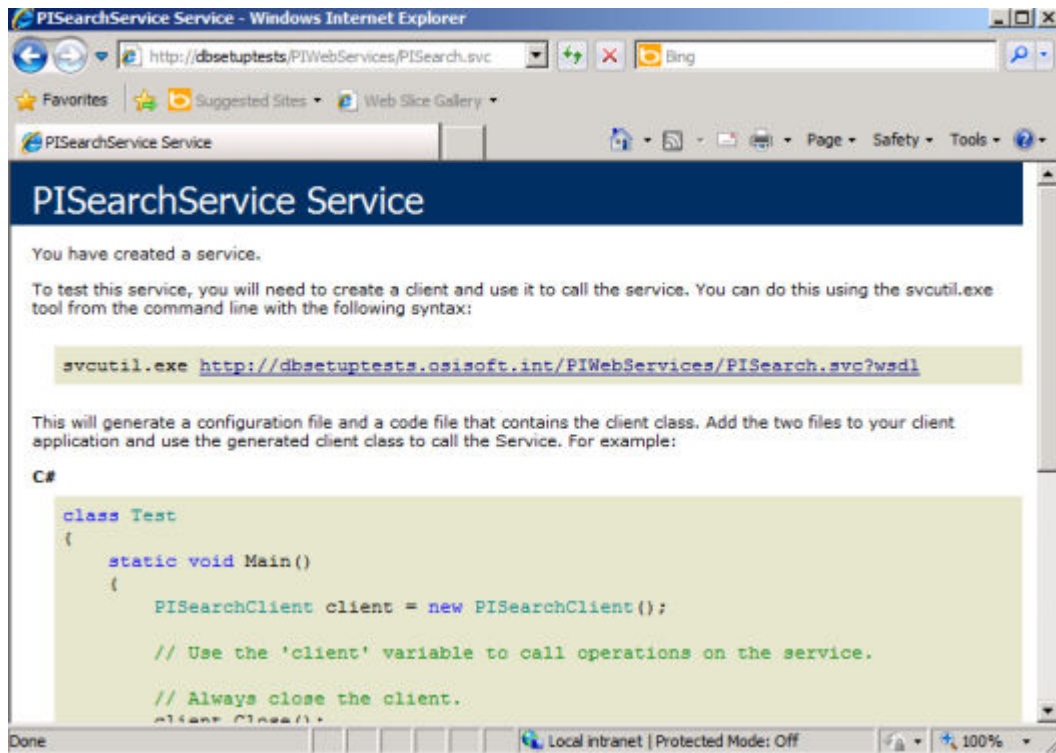
Procedure

1. Open a browser such as Windows Internet Explorer to verify your installation for data retrieval from PI server and PI Asset Framework:
 - If you used the default installation and website directories, enter:
`http://localhost/PIWebServices/PITimeSeries.svc`
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
`http://[servername]:[portnumber]/[PIWebServices directory name]/PITimeSeries.svc`

You should see this page:



2. Verify that you can use the newly installed PI Web Services to search for PI Server data:
 - If you used the default installation and website directories, enter:
`http://localhost/PIWebServices/PISearch.svc`
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
`http://[servername]/[PIWebServices directory name]/PISearch.svc`
- You should see this page:



3. Verify that you can use the newly installed PI Web Services to search for PI event frames:
 - If you used the default installation and website directories, enter:
http://localhost/PIWebServices/PISoap.svc
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
http://[servername]/[PIWebServices directory name]/PISoap.svc

Find the PI Web Services file directory

Procedure

1. Locate the directory used for your installation and review its contents to confirm that the PI Web Services files were installed correctly. This directory location will differ if:
 - PI Web Services is installed for the first time and the default directory was used, look in [PIHOME]\PIPC\PI Web Services.
 - You upgrade to PI Web Services. The PI Web Services installation path can be found in Windows Registry at HKLM\SOFTWARE\PISystem\PI Web Services\CurrentInstallationPath.

Verify data access

A web service client application is required to verify whether PI Web Services can retrieve PI System data. There are several third-party applications available that do not require programming to verify that data is being passed through PI Web Services. This section shows how to use WCFStorm, a Windows web services client application.



Note:

This example is provided to illustrate the process for configuring such tools for use with PI Web Services. OSIsoft does not endorse WCFStorm as a development tool or warrant its use or operation. To download a trial version or shareware version of this tool, see the [WCFStorm website](#).

Use the procedure here to perform a basic test to verify whether PI Web Services is retrieving snapshot data.

Download and install WCFStorm

Procedure

1. To download a trial version or shareware version of this tool, see the [WCFStorm Web site](#).
2. To install, extract the files to a local folder and launch `wcfstorm.exe`.

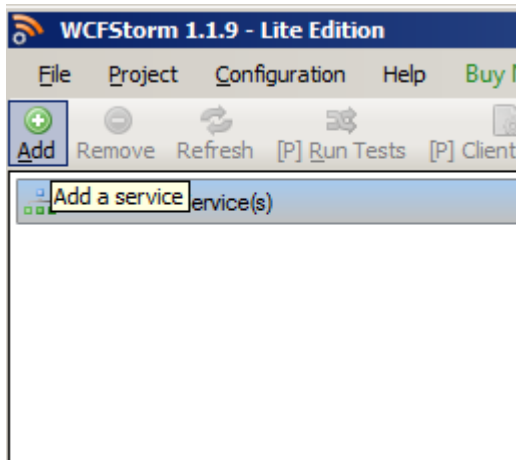
After you finish

Configure WCFStorm for PI Web Services access.

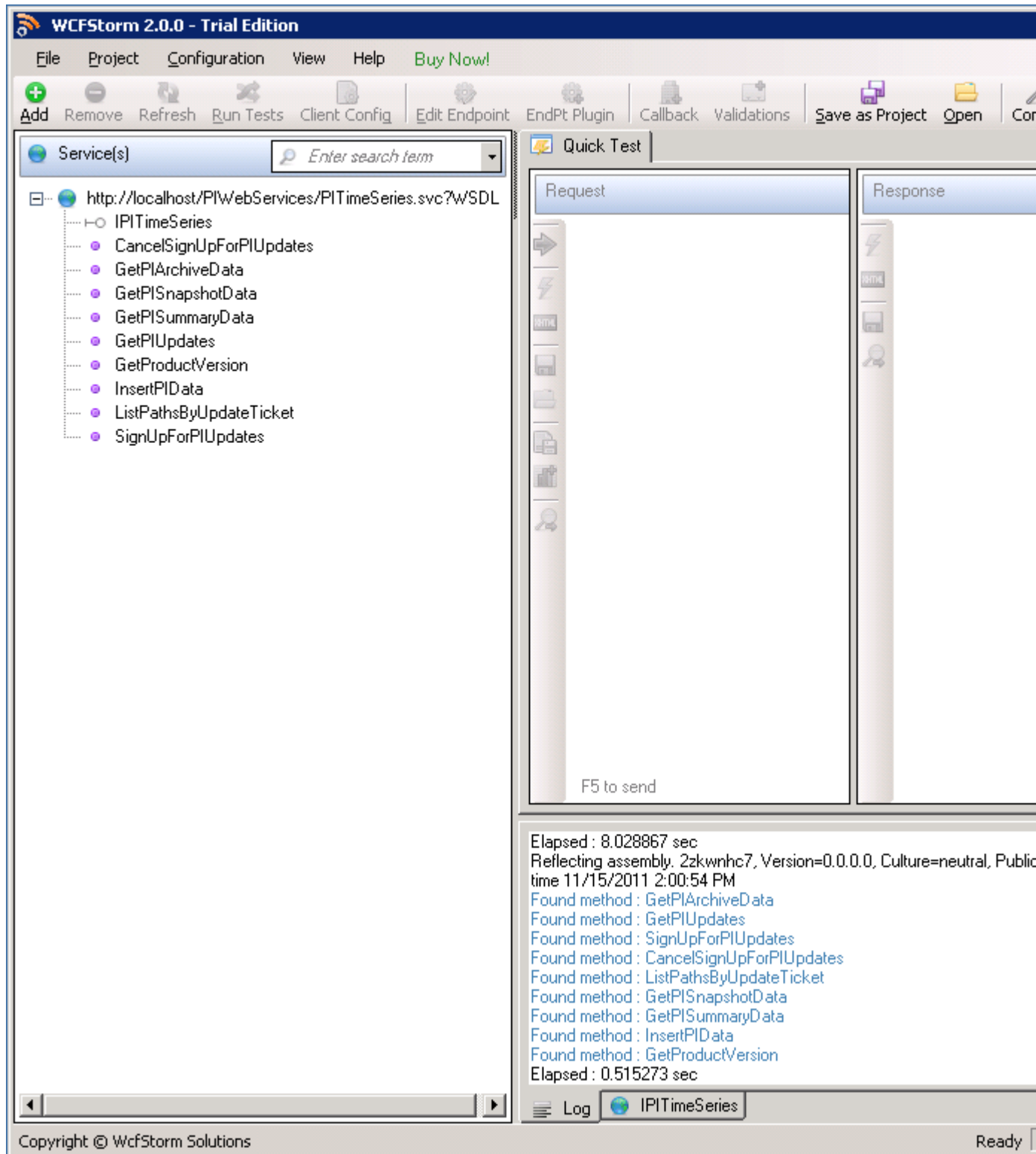
Configure WCFStorm for PI Web Services access


Procedure

1. Open WCFStorm.
2. Enter a service endpoint:
 - If you are prompted to add a service endpoint, enter the appropriate URL. The endpoint must supply the metadata that the client will need to create a request for PI Web Services:
 - For the default installation, this endpoint is:
`http://localhost/PIWebServices/PITimeSeries.svc?WSDL`
 - If you are running the client on a machine other than the Web server, change `localhost` to the name of the Web server that hosts PI Web Services.
 - If you are not prompted to add a new service endpoint when you open WCFStorm, click **Add** and then enter the service endpoint.



WCFStorm will retrieve the metadata from the Web service and display the endpoint and its methods:



3.  **Note:**

PI Web Services requires the client to authorize delegation of the user's Windows identity. By default, WCFStorm authorizes impersonation but you must change the impersonation level used by WCFStorm to Delegation.

Change the impersonation level:

- Click on **Config** on the menu bar and change the Impersonation level to **Delegation** in the **Authentication** or **Security** tab:

The screenshot shows the 'Edit Configuration Values' dialog box with the 'Authentication' tab selected. The 'Type' section has three radio buttons: 'Use Basic Authentication', 'Use Windows Authentication' (selected), and 'None'. Below this, it says 'Windows credentials will be assigned to proxy: ClientCredentials.Windows.ClientCredential'. The 'Basic Authentication' section has 'Username' and 'Password' fields. The 'Windows Authentication' section has a checked 'Use default credentials' checkbox, and 'Username', 'Password', and 'Domain' fields. The 'Impersonation' section has a 'Level' dropdown menu with 'Delegation' selected. At the bottom are 'OK' and 'Cancel' buttons.

After you finish

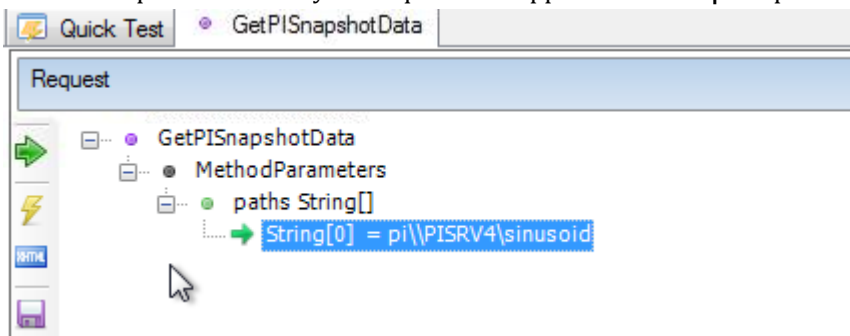
Create and execute a snapshot request.

Create and execute a snapshot request

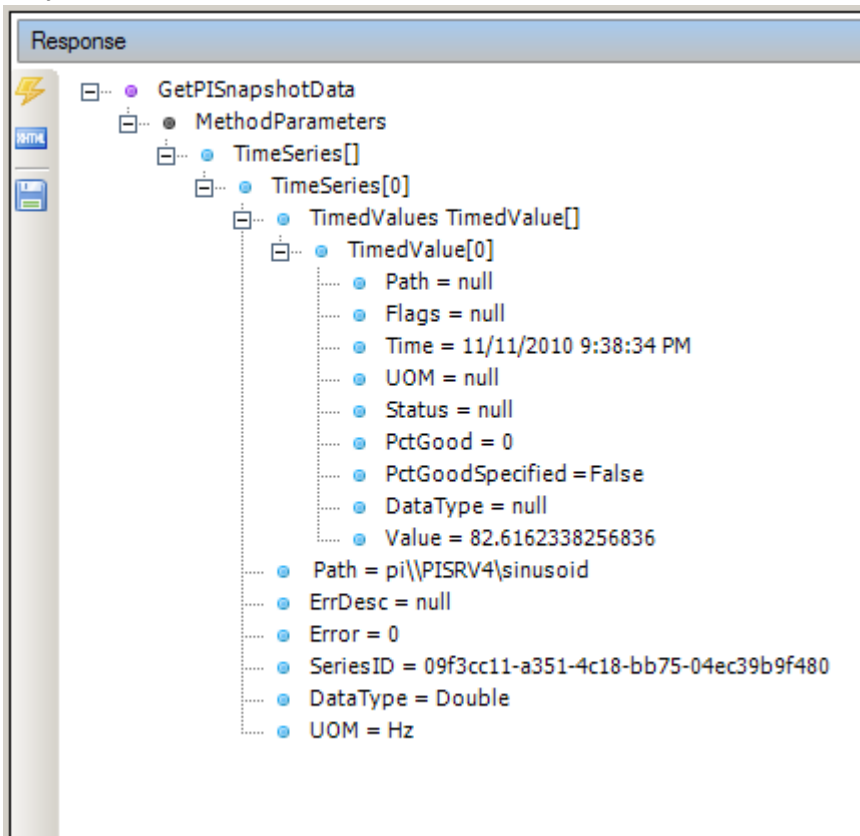
Procedure

1. Click on the entry for **GetPISnapshotData** in the left-hand pane.
2. Click on the entry under paths in the tree that appears for the request data that appears in the **Request** panel of the **Quick Test** pane.

3. Enter the path to a PI point in **Value** field of the **Edit the value** dialog. For example: pi:\PI_SRV4\sinusoid.
4. Click **OK**.
5. Review the parameters for your request that appear in the **Request** panel:



6. Click the Green arrow button to the left of the request to send the request to PI Web Services.
7. If PI System data appears in the **Response** panel, PI Web Services is correctly installed and able to retrieve PI System data:



Note:

The XML shown by WCFStorm when **View as XML** is selected does not reflect the actual structure of PI Web Services messages and should not be used as the basis for constructing messages in script clients.

Locate PI Web Services files

Procedure

1. Verify that PI Web Services was installed successfully:
 - If you install PI Web Services for the first time, check that the PI Web Services file directory contains:
 - `PIInstrumentation.config`
 - `PISearch.svc`
 - `PITimeSeries.svc`
 - `PISoap.svc`
 - `web.config`
 - `Global.asax`
 - If you upgrade from an earlier version of PI Web Services, verify that:
 - Your PI Web Services installation directory contains:
 - `Global.asax`
 - `PITimeSeries.svc`
 - `PISearch.svc`
 - `PISoap.svc`
 - `web.config`
 - `old_web.config`
 - The bin in your PI Web Services installation directory contains:
 - `OSIsoft.PIDataServices.Common.dll`
 - `OSIsoft.PIDataServices.Configuration.dll`
 - `OSIsoft.PIDataServices.DataAccess.dll`
 - `OSIsoft.PIDataServices.DataService.dll`
 - `OSIsoft.PIInstrumentation.dll`
 - `PIWebServicesInstrumentation.InstallState`
 - `OSIsoft.PIInstrumentation.Listeners.dll`
 - `PIWebServices.dll`
 - `PIWebServicesInstrumentation.dll`
 - `OSIsoft.PIInstrumentation.InstallState`

Results



Note:

PIWebServices is the default name of the IIS virtual directory. You can select another name for this directory when you install PI Web Services. If another name is selected, this virtual directory name is different.

PI Web Services post-installation configuration

Properties to control PI Web Services features

PI Web Services includes settings that you can use to control these client application features:

- Performance equation execution
- Data value displays
- PI System data insertions
- Duration of update sign ups
- Message sizes

To update these settings, use these parameters in the **PIWebServiceSettings** element of the `web.config` file on the Web server:



Note: The `web.config` file is located in the appropriate Web application folder, and depends on the location of the [PI Web Services installation files](#).

- **AllowCalculations**

Use to control whether users can execute Performance Equations:

- By default `AllowCalculations=TRUE`, which allows PE calculations.
- To prevent users from executing Performance Equations, set `AllowCalculations=False`.

- **AllowDataEntry**

Use to control whether users can insert data to a PI Server or AF Server:

- By default, `AllowDataEntry=TRUE`, which enables users to insert PI data with the `InsertPIData` method.
- To prevent user from inserting PI data with the `InsertPIData` method, set `AllowDataEntry=FALSE`.

- **FloatPrecision**

Use to control how many decimal digits are represented in the data returned by PI Web Service calls:

- By default, `FloatPrecision=Full`, which displays all digits supported by the double-precision floating point data type.
- To display the digits as configured by the `DisplayDigits` attribute of the PI point associated with the data, set `FloatPrecision=DisplayDigits`.

- **UpdatePurgeInterval**

Use to specify how many minutes updates are received before the sign-up is purged:

- By default, `UpdatePurgeInterval=5`, which causes updates to continue for 5 minutes after data for the update is not accessed.
- To specify a length in minutes for the interval that updates will continue, after data for the update is not accessed, set `UpdatePurgeInterval` parameter equal to the number of minutes desired.

- **maxReceivedMessageSize**

Use to control the size of messages sent and received between a Web client and Web server:

- PI Web Services does not alter this parameter's default value of **65,536** bytes, but your system may reach this limit and receive an error if PI Web Services is used to insert or retrieve large amounts of data.
- To specify the message size in bytes, configure this parameter in the [web.config](#) file located in the PI Web Services installation directory. See [PIHOME]\PIPC\PI Web Services.



Note: If you use .NET client applications, see [Message size in .NET client applications](#).

Examples

This element example shows how you can allow users to execute Performance Equation calculations:

```
<PIWebServiceSettings>
  <add key="AllowCalculations" value="true" />
  <add key="AllowDataEntry" value="true" />
<add key="FloatPrecision" value="Full"/>
<add key="UpdatePurgeInterval" value="5"/>
</PIWebServiceSettings>
```

This `wsHttpBinding` binding element demonstrates shows how you can add the attribute `maxReceivedMessageSize` with the default message size:

```
<wsHttpBinding>
  <binding name="wsBinding" bypassProxyOnLocal="false"
    transactionFlow="false" hostNameComparisonMode="StrongWildcard"
    maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
    messageEncoding="Text" textEncoding="utf-8"
    useDefaultWebProxy="true" allowCookies="false">
  </binding>
</wsHttpBinding>
```

Set message size received by clients

The `maxReceivedMessageSize` is unique if you use .NET applications. In this case, set `maxReceivedMessageSize` parameter in *both* the `app.config` and `web.config` files. For .NET applications, `app.config` controls the size of the message received by the client and `web.config` controls the size of the request message received by the service.

The `maxReceivedMessageSize` parameter is included in each PI Web Services sample `web.config` file and is set to the default value of **65,536 bytes (64 KB)**. If you are using one of these files, you can change the default value of the existing parameter.

To change the message size in your own `web.config` file, add the message size binding element to the `web.config` file and set the appropriate message size.



Note:

Do not set an arbitrary or inflated value for `maxReceivedMessageSize`. This setting protects against denial-of-service (DoS) attacks; if you set an excessively large value, you downgrade that protection. The impact of this setting on system security is described in this [MSDN article on security](#).

Add PI OLEDB Enterprise initialization properties

PI Web Services supports properties used by PI OLEDB Enterprise. When PI Web Services creates a new PI OLEDB connection, it honors any PI OLEDB property that is included in the `web.config` file.

To add a new property, include one or more elements to the key and value entries:

```
< add key = "Property" value="Value" />
```

Where the key and value combinations are recorded in the *PI OLEDB Enterprise User Guide*, available at the [OSIsoft Technical Support website \(http://techsupport.osisoft.com\)](http://techsupport.osisoft.com).

Examples

```
<PIWebServiceSettings>
  <add key="AllowCalculations" value="true" />
  <add key="AllowDataEntry" value="true" />
  <add key="FloatPrecision" value="Full" />
<add key="UpdatePurgeInterval" value="5" />
</PIWebServiceSettings>
<pidsSettings>
  <add key="PIInstrumentationConfigFile"
value="[Placeholder]PIInstrumentation.config" />
</pidsSettings>
<PIOLEDBEnterpriseInitializationProperties>
  <!--Refer to the PI OLEDB Enterprise users manual for more information on
  general OLEDB initialization properties, and for those specific to
  PI OLEDB Enterprise-->
  <add key="Command Timeout" value="120" />
  <!--
  <add key="Log File" value="C:\path\to\logfile.log" />
  <add key="Log Level" value="1" />
  -->
</PIOLEDBEnterpriseInitializationProperties>
```

Equation to calculate message sizes

To set the appropriate message size for PI Web Services deployments, you should be familiar with your system's typical usage and the size of the data structures passed by PI Web Services. Use this formula to calculate an approximate value for the **maxReceivedMessageSize** property for XML representations in the default UTF-8 encoding:

$$364 + (117 * \textit{number_of_paths}) + (75 * \textit{number_of_values})$$

This formula is based on these characteristics:

- Size of a GetPIArchiveData response message wrapper without any data: 364 bytes
- Size of a element without any timed values and without error data: approximately 117 bytes
- Size of a TimeSeries element. This can vary from the 117 byte value depending on the value of **Data Type**, the length of the path, and the length, if any, of the unit of measure. If your system uses long paths and routinely provides a data type and unit of measure, you may wish to increase this value.
- TimedValue element size. This varies by method. For GetPIArchiveData, the size of a single TimedValue element ranges from 69 to 77 bytes for point values of the Float data type. GetPISummaryData adds the **PctGood** property, which adds 12 to 26 bytes. The lower bound is for a value of zero, while the upper bound is a value greater than zero and less than 100, in which case a fractional component is usually reported with the value.

The request messages for GetPIArchiveData and GetPISummaryData are small in comparison to the response messages for these methods. By contrast, response messages are dominated by TimeSeries and TimedValue elements and these elements are also used in both the request and response messages for InsertPIData. As a result, although the recommended formula focuses on the response message for GetPIArchiveData, it is

representative of the response message for GetPISummaryData, and both the request and response messages for InsertPIData.

Review security configuration

PI Web Services uses configuration files that include Windows Communication Foundation (WCF) bindings to specify how data is sent across the network. To secure your Web service applications, you must edit these bindings to determine how data is transported and encoded, and specify how data is protected and represented, as well as how clients and servers communicate securely.

PI Web Services includes five sample configuration files, each with a different binding type, to help you get started with your security setup. You can use these sample bindings to set up the security for your PI Web Services deployment, or to better understand how to use your own binding files.

To determine which binding is appropriate for your PI Web Services, you must consider:

- The types of clients to be supported
- Whether you want binary or XML representation on the wire
- Security requirements

For instructions on how to use the PI Web Services sample bindings and for other security recommendations, see [PI Web Services Security](#).

Required firewall setup

The network ports that PI Web Services uses to communicate with your PI System must be open. If you have any internal firewalls between the PI Web Services server and the PI or AF Servers, the appropriate TCP ports must be open, or exceptions to open the appropriate ports must be configured.

Host PI Web Services on a PI WebParts server

If you want to host PI Web Services on a web server that has PI WebParts installed, OSIsoft recommends that you:

- Create an IIS website and associated application pool that are not under the control of Microsoft SharePoint. The website should be configured on its own port number. You must select the website as the host when you run the setup kit.
- Verify that the application pool you set up for PI Web Services runs under a Process ID that is different from the Process ID used for PI WebParts.

Host PI Web Services with a Windows service

The PI Web Services Standalone Edition provides a Windows managed service as an alternative to hosting PI Web Services with Microsoft Internet Information Services (IIS). This edition is useful for environments with security restrictions that prohibit the use of IIS. While the service edition of PI Web Services still provides services over HTTP, it uses a smaller resource footprint and has a smaller potential attack surface than that of IIS.



Note:

You may install just one edition of PI Web Services on a single server. To change from one edition to another, you must first uninstall the existing edition. If you have IIS installed on your server, you must uninstall it before you run the PI Web Services Standalone Edition setup kit.

The PI Web Services Standalone Edition is bundled with the PI Web Services setup kit and requires that the following be installed on the server that you use to install PI Web Services:

- Windows 8, Windows 2012, Windows 7, Windows Server 2008, or Windows Server 2003.

The computer must also have network access to one or more of these OS/soft servers:

- PI Server 3.3 or later
- PI AF server 2.1 or later that includes PI SQL for PI AF server



Note:

To search for PI event frames, you must use PI AF server 2.4 or later.

See [Configure Security for PI Web Services Standalone Edition](#) and [View and configure message logs](#) for information about how to configure security settings.

Install PI Web Services Standalone Edition

If you upgrade from an earlier version of PI Web Services, the PI Web Services setup kit will:

- Rename your existing web.config file to old_web.config
- Upgrade PI Web Services in the directory used for the earlier PI Web Services installation

Procedure

1. Double-click the PI Web Services setup kit.
2. De-select the option **When done unzipping open: .\Setup.exe**, click Browse to locate a folder where you want to extract the installation files for PI Web Services and click **Unzip**.
3. Open the folder that contains the extracted files and rename the file setup.ini to another name such as setup_old.ini.
4. Rename the file setup_standalone.ini to setup.ini.
5. Run the Setup.exe contained in the folder that contains the extracted files.



Note:

If you run the PI Web Services setup kit on a 64-bit Windows server, it will install the PI SDK for both the 32-bit and 64-bit operating systems.

- a. During the PI Software Development Kit (PI SDK) installation, enter the appropriate information when you are prompted for:
 - user information
 - a path to a folder where PI SDK will be installed
 - names for a default user and a default PI Server



Note:

It is possible to buffer data sent to the PI Server to guard against loss of connection. Data are not buffered by default. To enable buffering, use the **PI SDK Utility**. See the **PI SDK Utility Help** for details.

- b. Click **Start** to start the PI SDK installation.
- c. Click **Finish** to complete the PI SDK installation.

- d. During the PI AF client setup, enter the name of the AF server to which you want PI Web Services to connect.
You might be asked to provide the name of a PI Server for storage of Module Database configuration data. You can test this using PI SMT.
 - e. Click **Install** to start the PI AF client installation.
 - f. Click **Finish** to complete the PI AF client installation.
6. During the installation of PI Web Services, use the **Select Installation Address** window to select:
 - **Installation directory:** Use the default installation directory: C:\Program Files\PIPC\PI Web Services\. You have the option to select a path to another installation folder for PI Web Services.
 - **Port:** You have the option to select the port on which PI Web Services will listen. By default, the PI Web Services setup kit will use port 80.

After you finish

Configure security for the standalone edition of PI Web Services.

Verify PI Web Services Standalone Edition installation

To verify that PI Web Services Standalone Edition was installed successfully:

- Verify that the PI Web Services file directory contains:
 - OSIsoft.PIDataServices.Common.dll
 - OSIsoft.PIDataServices.Configuration.dll
 - OSIsoft.PIDataServices.DataAccess.dll
 - OSIsoft.PIDataServices.DataService.dll
 - OSIsoft.PIInstrumentation.dll
 - OSIsoft.PIInstrumentation.InstallState
 - OSIsoft.PIInstrumentation.Listeners.dll
 - PIInstrumentation.config
 - PIWebServices.dll
 - PIWebServicesIIInstrumentation.dll
 - PIWebServicesIIInstrumentation.InstallState
 - PIWebSvcHost.exe
 - PIWebSvcHost.exe.config
 - PIWebSvcHost.InstallState

Procedure

1. Start the PI Web Services service.
2. Open the `PIWebSvcHost.exe.config` that is installed with the standalone edition to see the URL for the base http site, then enter this URL into a browser such as Windows Internet Explorer.
3. To verify your installation for data retrieval from PI Server and PI AF:
 - If you used the default installation and website directories, enter:
`http://server/PIWebServices/PITimeSeries.svc`
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
`http://[servername]:[portnumber]/[virtual directory name]/PITimeSeries.svc`

4. To verify your installation for searches of PI Server data:
 - If you used the default installation and website directories, enter:
`http://server/PIWebServices/PISearch.svc`
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
`http://[servername]:[portnumber]/[PIWebServices directory name]/PISearch.svc`
5. To verify your installation for searches of PI event frame data:
 - If you used the default installation and website directories, enter:
`http://server/PIWebServices/PISoap.svc`
 - If you did not use the default installation and website directories, enter the path you used for the website relative URL:
`http://[servername]:[portnumber]/[PIWebServices directory name]/PISoap.svc.svc`

PI Web Services silent installation

If you would like to install PI Web Services without a graphical user interface (GUI), you can use the silent installer that is included with the PI Web Services setup kit:

Procedure

1. Double-click the PI Web Services setup kit.
2. De-select the option **When done unzipping open: .\Setup.exe**, click Browse to locate a folder where you want to extract the installation files for PI Web Services and click **Unzip**.
3. Open the file `setup_silent.ini` and:
 - a. Proceed to Step 5 if you do not want to create a file that specifies a list of PI Servers in the Known Servers Table (KST).
4. To create a file to specify a list of PI Servers in the Known Servers Table (KST):
 - a. Refer to the `setup_silent.ini` file for instructions on how to create a file to specify a list of PI Servers in the Known Servers Table (KST).
5. Open the folder that contains the extracted files and rename the file `setup.ini` to another name such as `setup_old.ini`.
6. Rename the file `setup_silent.ini` to `setup.ini`.
7. Run the `Setup.exe` contained in the folder that contains the extracted files.



Note:

If you run the PI Web Services setup kit on a 64-bit Windows server, it will install the PI SDK for both the 32-bit and 64-bit operating systems.

PI Web Services security

Secure connections between a PI System and a PI Web Services client will:

- Provide the identity of the client user that sends or receives data through PI Web Services
- Encrypt the data that is passed from a PI System through PI Web Services

To ensure that your PI Web Services deployment is secure, OSIsoft recommends that you configure:

- Windows-integrated security for your PI Server
- Secure access for AF Server data
- A web server that uses a secure user account
- Firewall security that enables PI Web Services to communicate with your PI System
- Security for the web service bindings
- Impersonation and delegation

Other factors to consider when you configure security for your PI Web Services deployments include:

- Local requirements for access security, data integrity, and privacy
- Level of support for WS-* standards in the client applications and programming tools that you expect to use
- Whether clients call the PI Web Services from non-Windows operating systems

For information not covered in this Help, OSIsoft recommends that you refer to security information provided by Microsoft at this website: [Windows Communication Foundation Security](#). For specific guidelines on configuration file syntax, see [Windows Communication Foundation Configuration Schema](#).

Supported security scenarios

PI Web Services supports these security configurations:

- Intranet access to PI Web Services using PI trust or Windows-integrated security
- Intranet access to PI Web Services without a domain controller
- Intranet access to PI Web Services with a non-Windows client
- Extranet access to PI Web Services using PI trust or Windows-integrated security
- Extranet access to PI Web Services using anonymous access

Requirements to secure access through PI Asset Framework

If you use AF server, PI Web Services uses the native Windows identity to authenticate the user.

When retrieving data through the PI Asset Framework (AF), you must:

- Add the Process ID account under which Internet Information Services (IIS) runs on Windows Server 2003, to the **PI SQL (AF) Trusted Users** group on the AF Servers that PI Web Services will use. To identify this account, see [Verify Identity Connections](#).
- Ensure that the required user accounts have read access to AF server.

For complete instructions on how to manage access to PI AF elements and attributes as selected data items in PI Web Services, see the PI Asset Framework (PI AF) user documentation.

User accounts required for secure connections

Security for the PI Data Access product **PI Web Services** requires two user accounts:

- Process ID
 - If you are using PI Web Services IIS Edition, the Process ID must be the Windows account under which the Internet Information Services (IIS) application pool runs
 - If you are using PI Web Services Standalone Edition, the Process ID must be the Windows account under which the PI Web Services host service runs
- User ID
Windows account of the client calling the PI Web Services, if the machine you use to run PI Web Services is configured to use Windows authentication

PI Web Services uses the **Process ID** to connect to and process updated from the PI System through PI Data Services. If PI Data Services cannot use the IIS Process ID to connect to the PI System, PI Web Services will return an error that data cannot be retrieved.

If the web services client application runs on Windows and is in the same domain as the web server or in a trusted domain and the PI Server is configured to use Windows authentication, then the web services client application will use the **User ID** to secure its connection to the web server.



Note:

On non-Windows platforms, a certificate on the web server can be configured to map to a Windows domain user. This would enable secure data access using the **User ID** from non-Windows clients.

To ensure that both of these accounts allow PI Web Services to connect with a PI System:

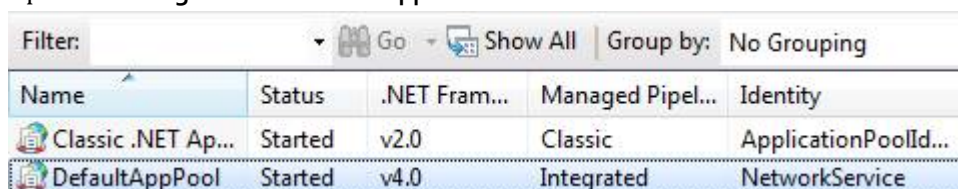
- PI Web Services will use a secure configuration if the Network Service account is used for its Process ID.
- Verify your identity connections. If the Process ID is a uses a domain account other than Network Service, you must configure Kerberos authentication.
- Connections that use Windows authentication must meet additional requirements.
- End user folder and registry permissions must be set to allow access to data stored in the PI Server, PI Module Database (MDB) and PI point database.

Verify connection identities

Identify the **User ID** and **Process ID** used in a given session.

Procedure

1. Open **IIS Manager** and select the **Application Pools** node to view the Windows account that is used:



Name	Status	.NET Fram...	Managed Pipel...	Identity
Classic .NET Ap...	Started	v2.0	Classic	ApplicationPoolId...
DefaultAppPool	Started	v4.0	Integrated	NetworkService

**Note:**

If you open IIS Manager from a remote connection, the **NetworkService** account is visible in **IIS Manager** as the machine's name.

- Use **PI SMT** to inspect the PI identity to which the Windows account is mapped.

After you finish

To verify that these identities will allow PI Web Services connections, see [Connections that use Windows Authentication](#), or [Connections that use Kerberos Authentication](#).

Connections that use Windows authentication

To take advantage of Windows-integrated security:

- Both the **User ID** and the **Process ID** need to be valid Active Directory accounts with at least one mapping between a PI identity and a Windows identity. OSIsoft recommends that you create a PI mapping to associate the **User ID** and the **Process ID** with two different PI identities, but this is not mandatory.

**Note:**

If membership in an Active Directory group used for a PI mapping is modified, or if a relevant PI mapping itself is modified, you might need to restart IIS.

- The client, Web server, PI Server and AF Server machines all belong to the same Active Directory forest.

**Caution:**

In most cases, the Web server will impersonate the **User ID** so that access to the PI System takes place using the Windows account of the client. If impersonation fails or is not configured, access to the PI System will take place using the **Process ID**. OSIsoft recommends that the **Process ID** account not be granted write access to the PI System.

Connections that use Kerberos authentication

When using Kerberos security, Kerberos delegation must be configured between the Web server and PI Server. For instructions on setting up this Kerberos environment, see [Configure Kerberos authentication](#).

End user folder and registry permissions

Location	User Account	Permissions	Notes
Web site folder and subfolders	User ID	Read & Execute, Read	The Web site folder which is typically stored in <code>[wwwroot]\PIWebServices</code>
Web site folder and subfolders	Process ID	Full Control	The Web site folder which is typically stored in <code>[wwwroot]\PIWebServices</code>
HKLM\SOFTWARE\PISystem registry key and subkeys	User ID, Process ID	Full Control	Allows PI Web Services to access the Known Servers Table and time zones in registry.

**Note:**

The permissions described here are set automatically when you enable IIS using the default settings. If you are not using the default IIS settings, you might need to change permissions to reflect these requirements.

Security settings in the PI Web Services configuration file

Security settings and other details that specify how data is transported and encoded are found in the XML element `binding` within the `web.config` file found in the PI Web Services file directory. The information contained within bindings enables web applications and web servers to communicate.

Use the security bindings to control:

- The security mode, which defines how security is applied to achieve authentication, confidentiality, and integrity. Available security modes binding elements are `Transport`, `Message`, `TransportCredentialOnly` and `TransportWithMessageCredential`.
- The client credential type
- Server credential values, which controls how the client is authenticated. For Windows authentication, set `clientCredentialType=Windows`.



Note:

Client security configuration depends on the particular client tool used. For .NET applications, this is accomplished in the `app.config` file. For an example, see [Configure Security for .NET Clients](#).

To secure your PI Web Services deployment, you must set values for security on each web service binding. Binding details must be the same for both clients and endpoints to ensure data exchanges are successful.

These security settings are pre-configured in the sample configuration files included with PI Web Services.

Binding types used by PI Web Services

Before you create web service bindings, you must select a binding type. There are four binding types that can be configured for PI Web Services. To understand more about how these files, see the sample configuration files provided with PI Web Services for each binding type.

To decide which binding to use, consider the type of security used for your PI System data.

If you use Windows-integrated security for your PI Server and AF Server, you can use any of these binding types:

- [wsHttpBinding](#)
- [netTcpBinding](#)
- [netNamedPipeBinding](#)

When configured for security, these bindings allow you to control user access to PI data through the mappings and identities; with them you can identify and control which users log in and access PI data.

A fourth option is [basicHttpBinding](#).

After you select the binding type, you can configure security for the web service bindings.

This section provides brief guidelines to help you select a binding type. For more information about which type of binding is best suited to your purposes, see the MSDN articles [Bindings and Security](#) and [Configuring System-Provided Bindings](#).

basicHttpBinding

This is the SOAP 1.1/WS-Interoperability Basic Profile binding. It is *not secure* by default. Security mode options are:

- None
- Transport
- Message
- TransportWithMessageCredential

The basicHttpBinding in WCF is provided for compatibility with WS-Interoperability Basic Profile clients and by default provides no security. WS-I Basic Profile compliant applications, such as Microsoft Office InfoPath, will require the basicHttp binding.

OSIsoft recommends **Transport** security with a Windows token as the client credential in order to protect the exchange and provide Windows credentials for impersonation.

PI Web Services includes a basicHttpBinding sample that you can use with minimal configuration.

wsHttpBinding

This is the SOAP 1.2 binding. This binding is secure by default. The PI Web Services setup kit configures the web service for this binding based on the assumption that the application pool is running under the Network Service account.

Web service clients that support profiles above the WS-I Basic level, such as custom clients written with **.NET** or **Java** frameworks should use wsHttpBinding. This binding uses impersonation and delegation by default, so only minimal configuration changes are required.

If you are using Windows authentication, and want to use impersonation and delegation, this binding requires minimal edits. PI Web Services includes a sample wsHttpBinding that you can also use with minimal configuration.

netTcpBinding

This is a non-SOAP, binary format binding. It can be used between machines. TCP/IP is the transport protocol. It is a secure binding optimized for cross-machine communication on an intranet between WCF clients and WCF web servers. The netTcpBinding sample included with PI Web Services is configured for secure operation provided you replace the server name placeholder with the actual machine name in the servicePrincipalName XML element.

netNamedPipeBinding

This is a non-SOAP, binary format binding that is restricted to use on a single machine. Named Pipes is the transport protocol. PI Web Services includes a sample netNamedPipeBinding that you can use with minimal configuration. If you use this netNamedBinding sample as is, the netNamedPipeBinding will perform Windows authentication but does not encrypt the data in transit.

Secure access to PI Server data

You can secure PI Server data for PI Web Services deployments through:

- PI identities and mappings used for Windows-integrated security
- PI trusts (PI Server 3.3 and later)
- Database security settings for the PI Server, PI modules and PI points

OSIsoft recommends that you use Windows-integrated security, rather than PI trusts alone, for web services that access PI Server 3.4.380 or later. PI Web Services connections to PI Servers that use Windows-integrated security ensure that data exchanged between a PI Server and web service client is protected through encryption and that the access to data is controlled through client user accounts.

You can configure PI identities, PI mappings, and PI trusts with the **Mappings and Trusts** tool in PI System Management Tools (PI SMT). For further details, see the *PI SMT Help* and *Configuring PI System Security*, available at the [OSIsoft Technical Support website](#).

Windows-integrated security

Windows-integrated security provides substantial advantages in security, efficiency, and flexibility, which include:

Less work for PI Server administrators. You no longer need to create and manage individual user accounts on the PI Server. When a user enters, leaves, or changes roles, you only need to modify the Windows configuration. PI Server security automatically reflects these changes. You also get complete traceability of the specific Windows account in the PI Server log and audit trail records.

Single-sign on for users. Users need only log on to their Windows account. PI clients will automatically authenticate through the PI SDK. No additional PI Server login is required.

Improved Security:

- **Secure authentication.** Connections are authenticated through Microsoft's Security Support Provider Interface (SSPI). If you're using AD, then this means the most secure Kerberos authentication, which greatly improves your PI Server security.
- **Control over server-side authentication policies.** With the new security model, you have control over the authentication protocols that client applications can use to connect to the PI Server. You can disable authentication methods that are less secure and keep only the connection methods that you need.

More control over access permissions. The security model included with PI Server 3.4.380 and later includes a much more flexible model for access permissions. In previous versions of the PI Server you could set permissions only for one owner, one user group, and for world access (everyone else). With this security model, each PI Server resource can have read and/or write permissions defined for any number of PI identities.

PI identities and mappings

PI Server 3.4.380 introduces a Windows-integrated security model that allows you to manage your PI Server authentication through Microsoft Active Directory (AD). This model improves PI Server security, reduces your management workload, and provides users a single-sign on experience. PI Server with Windows-integrated security gives you more control over authentication policies and access permissions and provides Windows account traceability in logs and audit trail records. OSIsoft recommends that you use Windows-integrated security.

To use PI Server with Windows-integrated security:

- Both the User ID and the Process ID accounts must be valid Windows users or groups with at least one PI mapping
- Each Windows User account must be mapped to a PI identity using **PI System Management Tools (PI SMT)** for PI Server 2010 or later.
- The PI identity used to install PI Web Services must have write permission on the **%OSI module** in the Module Database (MDB) prior to running the PI Web Services setup kit. Otherwise the installation will fail. This is required by PI Data Services.



Note:

OSIsoft recommends that you map both the User ID and the Process ID accounts to two distinct PI Identities.

For further details, see *Configuring PI System Security* and the *PI SMT Help*, available at the [OSIsoft Technical Support Web site](#).

PI trusts

To provide access to pre-approved network entities on PI Servers earlier than version 3.4.380, PI System applications, including those running PI Web Services, must use trusts. However, to configure access to PI Servers 3.4.380 or later, OSIsoft recommends that you instead use PI identities and mappings.

For further details, see *Configuring PI System Security* and the *PI SMT Help*, available at the [OSIsoft Technical Support Web site](#).

Summary of user permissions

Users of PI Web Services require read and write access to PI Server databases to accomplish various tasks. The permissions summarized here allow Web services users to create and edit PI points and modules, retrieve and insert PI Server data, and so on. PI MDB access is required because PI Data Services uses it to store configuration information.



Note:

Additional permissions are required to run the setup kit.

PI Server Database Security	Permission	Notes
PIBatch	r	
PIDBSEC	r	
PIHeadingSets	r	
PIModules	r,w	r for IIS application pool Process ID w for an administrator User ID to install and make configuration changes
PIPOINT	r	
PIUSER	r	

PI Point Database	Permission	Notes
PtSecurity	r	
DataSecurity	r,w	w for the User ID only if data writes are performed using the InsertPIData method

PI Module Database	Permission	Notes
Module Security*	r,w	r for IIS application pool Process ID w: for an administrator User ID to install and make configuration changes

Use **PI System Management Tools (PI SMT)** to change these permissions. For complete details, see the *PI SMT Database tool Help*.

Configure Security for Web Service Bindings

PI Web Services provides sample configuration files pre-configured with recommended settings that are intended to streamline the changes you must make to the `web.config` file in the PI Web Services file directory. If your IIS server is set up as described in [Configure the Web Server](#), you need change only the **servicePrincipalName** value to reflect the name of the server that hosts PI Web Services to use these sample configuration files.

If you use your own `web.config` file, or want to edit the sample files to meet the needs of a custom Web services deployment, refer to the topics here. To make changes to the default settings in `web.config` that are not described here, see the MSDN article [Windows Communication Foundation Security](#).



Note: There is no administration tool provided by the operating system to manage WCF settings found in `web.config`. You can use a text editor or WCF Service Configuration Editor to edit these files.

Use of endpoint and active configuration bindings

Each binding element describes some aspect of how the endpoint communicates with clients.

The **servicePrincipalName** value and other information in the `web.config` is contained within XML element `<system.serviceModel>`, which contains three major child elements: `<bindings>`, `<services>` and `<behaviors>`. To find the correct value to update, locate and edit the name of the service endpoint that is currently active. Next, locate and edit its corresponding configuration.

Edit the endpoint binding

To edit the endpoint binding:

Procedure

1. Open the PI Web Services binding configuration file and locate the `<services>` element, which has a single XML child element called `<service>`. This element will in turn have at least two children called `<endpoint>`. In general, the active `<endpoint>` is the one that has a child XML element called `<identity>`:

```
<endpoint binding="wsHttpBinding" bindingConfiguration="wsBinding"
name="BasicEndpoint"
bindingNamespace="http://xml.corporate.com/services/PIDataService"
contract="PIWebService.PIDataService.IPITimeSeries">
  <identity>
    <servicePrincipalName value="HOST/www.yourhostname.com" />
  </identity>
</endpoint>
```

2. Enter `www.yourhostname.com` to update the identity element.

**Note:**

Do not change the endpoint that begins with `<endpoint address="mex."` This is the Metadata Exchange Endpoint and is used by WCF to expose information about PI Web Services such as the WSDL to certain newer client applications.

Edit the active binding configuration

To edit the active binding configuration:

Procedure

1. Open the binding configuration file and locate the active `<endpoint>`. The active endpoint has an XML attribute called `binding` and might have an additional XML attribute called `<bindingConfiguration>`
2. To find the actual binding configuration settings, look in the `<bindings>` element within `<system.serviceModel>`. Locate an XML child element with a name that matches the binding value of the active `<endpoint>` element. In [Edit the endpoint binding](#), this is `<wsHttpBinding>`. The specific configuration for this binding can be found within its XML child element called `<binding name="wsBinding">`. Note that the XML attribute name matches the value of the `bindingConfiguration` attribute of `<endpoint>`.
3. Enter the name of your binding. In this example from the default `wsHttpBinding web.config` file, the beginning of the XML element `<bindings>` looks like this:

```
<bindings>
  <wsHttpBinding
    <binding name="wsBinding"
      <bypassProxyOnLocal="false" transactionFlow="false"
        hostnameComparisonMode="StrongWildcard" ...
```

Impersonation and delegation

WCF settings included in the Web service security bindings can be combined with Windows authentication on the data servers to impersonate the account of the client calling the PI Web Services.

All PI Web Services methods allow but do not require impersonation. When a query is executed, the service will impersonate the current Windows user. If the service and calling client are configured for secure operation, the identity used will reflect the identity used by the caller. If not, impersonation will be performed using the account under which the application pool hosting PI Web Services executes.

PI Web Services will attempt to impersonate the User ID account while accessing PI System data. If the impersonation fails, the Windows identity under which data access calls are made reverts to the Process ID account.

For further information regarding the configuration of WCF security, see the MSDN article [Delegation and Impersonation with WCF](#).

Web service bindings that use impersonation and delegation

To configure impersonation and delegation for PI Web Services deployments:

- Legacy clients which only support the WS-Interoperability (WS-I) Basic Profile require the use of the WCF `basicHttpBinding`.
- Clients that support profiles above the WS-I Basic Profile, i.e., clients that support WS-Security, should use the `wsHttpBinding`. This binding provides message security by default.
- If the server has Windows Activation Service (WAS), found on Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2, you can also use the `netTcpBinding`, or `netNamedPipeBinding`.



Note:

To use the `netTcpBinding` with PI Web Services, you must be running Internet Information Services (IIS) on Windows Server 7 or later.

Impersonation in PI Web Services sample binding files

Impersonation is enabled by default for the sample binding files included with PI Web Services but there are times when you might want to turn off impersonation. For example, you can disable impersonation if your security relies on PI trusts and you want to use a simplified IIS Web server configuration for legacy Web service clients such as Microsoft InfoPath. Access to PI System data is secure with this configuration because messages in transit might be protected by either message or transport encryption and the Web service uses the Process ID to connect. At the same time, all calls made to the Web service are granted access through the PI trust and due to the process using the IIS application pool identity.



Note:

For further information about best practices for this configuration, see the MSDN Web site for articles about [trusted subsystems](#).

Impersonation setting changes

To enable or disable impersonation in a secure configuration, update `serviceBehavior` for the Web service in the `web.config` file. Use the setting of the `impersonateCallerForAllOperations` property found in the `serviceAuthorization` element.

If you use a `web.config` file that does not include the `impersonateCallerForAllOperations` property, the default value for this property is `FALSE`. However, this property is set to `TRUE` in the `web.config` files included with PI Web Services, as shown here:

```
<serviceBehaviors>
  <behavior name="PIDataService.ServiceBehavior">
    ...
    <serviceCredentials>
      <windowsAuthentication includeWindowsGroups="true"
allowAnonymousLogons="false" />
      <issuedTokenAuthentication allowUntrustedRsaIssuers="true" />
    </serviceCredentials>
    <serviceAuthorization principalPermissionMode="UseWindowsGroups"
      impersonateCallerForAllOperations="true" />
    </behavior>
    ...
  </serviceBehaviors>
```

Configure security for applications

You can generate an `app.config` file with almost all WCF configuration required for PI Web Services applications built with .NET and Visual Studio by creating a web service reference, then providing Visual Studio

with the URL to the WSDL endpoint for the deployed web service. This URL is: `http://server_name/PIWebServices/PITimeSeries.svc?WSDL`, if you use the installation defaults.

One critical item that Visual Studio cannot generate, however, is the level of impersonation authorized by the client. This level, set by the **allowedImpersonationLevel** attribute, indicates the level of impersonation that the client authorizes the web server to use on its behalf. The possible values are:

- None
- Anonymous
- Identification
- Impersonation
- Delegation

PI Web Services requires a delegation level to function properly in a secured environment. This level only can be configured for wsHttp and netTcp bindings. To accomplish this level of authorization, .NET client developers should take these steps:

Procedure

1. Generate a web service reference as described in this MSDN article: [How to: Generate a Web Service Proxy](#).
2. Add an endpoint behavior authorizing delegation that references the new behavior from the client endpoint:
 - a. Open the newly created **app.config** file in Visual Studio and add the following section of XML as a child of the `system.serviceModel` element and as a peer of the `bindings` element:

```
<behaviors>
  <endpointBehaviors>
    <behavior name="wsImpersonationBehavior">
      <clientCredentials>
        <windows allowedImpersonationLevel="Delegation" />
        <httpDigest impersonationLevel="Delegation" />
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>
```

- b. Name the behavior as desired. This example uses the name **wsImpersonationBehavior**.
3. Edit the endpoint element to add a **behaviorConfiguration** attribute that references the name of the behavior just added:

```
<endpoint address=http://server_name/PIDataService/PITimeSeries.svc
  behaviorConfiguration="wsImpersonationBehavior"
  binding="wsHttpBinding"
  bindingConfiguration="BasicEndpoint"
  contract="PIWSTimeSeriesRef.IPITimeSeries"
  name="BasicEndpoint">
```



Note:

The allowed impersonation level can also be set in source code without changing the `app.config` file. Assuming that the proxy object generated as a Service Reference is called `proxy`, the code would be:

```
proxy.ClientCredentials.Windows.AllowedImpersonationLevel =
System.Security.Principal.TokenImpersonationLevel.Delegation;
```

Security binding samples

This section describes how to use the Web service bindings included in the sample `web.config` files to configure secure PI Web Services deployments. There is one fully configured `web.config` file for each supported WCF binding type.

You can use a text editor or WCF Service Configuration Editor to edit these files.

Sample configuration files

The directory `[PI Web Services Installation Path]\Help\Samples` contains sample `web.config` files that you can copy to your PI web Services deployments, then edit to can use to modify security settings and control application behavior through PI Web Services.

File Name	Contents
<code>web_all_bindings_basic.config</code>	Includes all binding types used by PI Web Services. The service endpoints are configured to use the <code>basicHttpBinding</code> with no security.
<code>web_config_basic_no_security.config</code>	A <code>basicHttp</code> binding file that includes no active security options. Use this binding type for best interoperability with non-Windows platforms.
<code>web_config_netTcp.config</code>	A <code>netTcp</code> binding file. The sample's default setting is secure and includes additional security options.
<code>web_config_wsHttp.config</code>	A <code>wsHttp</code> binding file suitable for cross-machine communication on an intranet between WCF clients and WCF web servers.
<code>web_config_netNamedPipe.config</code>	A <code>netNamedPipe</code> binding file suitable for high speed communications between processes on the same computer.

Use the `basicHttpBinding` sample

Use the sample `basicHttpBinding` `web.config` file provided with PI Web Services when the setting `<security mode="None">` is required to communicate with SOAP 1.1 clients, such as InfoPath.



Note:

To protect your data exchanges and provide Windows credentials for impersonation, OSIsoft recommends you use **Transport** security. In the `basicHttpBinding`, **Transport** security requires the host IIS server to be configured with a certificate and enabled for HTTPS. The sample provided with PI Web Services uses a security mode set to **None** because signed certificates are site-specific and must be generated by your Public Key Infrastructure (PKI).

The endpoint for the service should look like this:

```
<endpoint binding="basicHttpBinding" bindingConfiguration="basicBindingConfig"
name="BasicEndpoint"
bindingNamespace="http://xml.osisoft.com/services/PIDataService"
contract="PIWebService.PIDataService.IPITimeSeries" />
```

To configure the binding configuration element:

```
<bindings>
  <basicHttpBinding>
    <binding name="basicBindingConfig">
      <security mode="None">
```

```

        <transport clientCredentialType="Windows" />
    </security>
</binding>
</basicHttpBinding>
</bindings>

```



Note:

The name of the binding – `basicBindingConfig` – is used as the value of the endpoint's `bindingConfiguration` attribute to link this configuration to the endpoint.

For secure communications, the `web.config` file for the service must authenticate the client using a certificate under this mode. A service behavior that accomplishes this is as follows:

```

<serviceBehaviors>
  <behavior name="BasicSecuredSvcBehavior" >
    <serviceMetadata httpGetEnabled="true" httpsGetEnabled="false" />
    <serviceCredentials>
      <serviceCertificate findValue="d5 04 7e e0 c9 30 fb 53 50 26 0b 74 84 e1 35 aa 56
c6 5f a1"
                                storeLocation="LocalMachine"
                                x509FindByType="FindByThumbprint" />
      <windowsAuthentication includeWindowsGroups="true" allowAnonymousLogons="false" />
      <issuedTokenAuthentication allowUntrustedRsaIssuers="false" />
    </serviceCredentials>
    <serviceAuthorization principalPermissionMode="UseWindowsGroups"
impersonateCallerForAllOperations="true" />
  </behavior>
</serviceBehaviors>

```

For additional security, you can require client applications to supply a certificate by adding a `clientCredentials` section to the service behavior, and thereby provide for mutual authentication.



Note:

This example includes a `findValue` setting that uses the thumb print of a certificate. For more information about the certificates used in this security setting, see this MSDN article on [service credential certificates](#).

Use the `wsHttpBinding` sample

Configure the `wsHttpBinding` sample to provide a secure connection suitable for user impersonation and delegation over the HTTP protocol.

Procedure

1. Add a section for `wsHttpBinding`. While most of the settings shown here are defaults, ensure security mode is **Message** and message `clientCredentialType` is **Windows**:

```

<wsHttpBinding>
  <binding name="wsBinding"
    bypassProxyOnLocal="false"
    transactionFlow="false"
    hostNameComparisonMode="StrongWildcard"
    maxBufferPoolSize="524288"
    maxReceivedMessageSize="65536"
    messageEncoding="Text"
    textEncoding="utf-8"
    useDefaultWebProxy="true"
    allowCookies="false">
    <reliableSession ordered="true"
    inactivityTimeout="00:10:00"
    enabled="false" />
  <security mode="Message">

```

```

<message clientCredentialType="Windows"
    negotiateServiceCredential="true"
    algorithmSuite="Default"
    establishSecurityContext="true" />
</security>
</binding>
</wsHttpBinding>

```

2. Create an endpoint where:

- The binding attribute denotes that wsHttpBinding is the **binding** type.
- The bindingConfiguration attribute that names the binding configuration. In this example, bindingConfiguration="wsBinding".
- The **servicePrincipalName** under identity refers to the name of the machine hosting the Web service:

```

<endpoint binding="wsHttpBinding"
    bindingConfiguration="wsBinding"
    name="BasicEndpoint"
    bindingNamespace=http://xml.corporate.com/services/PIDataService
    contract="PIWebService.PIDataService.IPITimeSeries">
    <identity>
        <servicePrincipalName value="HOST/server_machine_name" />
    </identity>
</endpoint>

```

3. In the serviceBehaviors section, ensure the behavior configuration used by the Web service has **TRUE** for the value of impersonateCallerForAllOperations.

```

<serviceBehaviors>
    <behavior name="PIDataService.Service1Behavior">
        <serviceMetadata httpGetEnabled="true" />
        <serviceDebug includeExceptionDetailInFaults="true" />
        <serviceCredentials>
            <windowsAuthentication includeWindowsGroups="true"
                allowAnonymousLogons="false" />
            <issuedTokenAuthentication allowUntrustedRsaIssuers="true" />
        </serviceCredentials>
        <serviceAuthorization principalPermissionMode="UseWindowsGroups"
            impersonateCallerForAllOperations="true" />
    </behavior>
</serviceBehaviors>

```

Use the netTcpBinding sample

Before you start

To use the netTcpBinding with PI Web Services, you must be running Internet Information Services (IIS) on Windows Server 7 or later and the Windows Activation Service (WAS) feature must be installed for use with non-HTTP requests.

Procedure

1. Verify that Windows Activation Service (WAS) for use with non-HTTP requests is installed:
In the Control Panel click **Programs and Features > Turn Windows Features on and off > Microsoft .NET Framework 3.x** and verify that **Windows Communication Foundation Non-HTTP Activation** is selected.
2. Configure the Web site that hosts PI Web Services to listen for netTcp requests:
Log in as administrator then open a command prompt and enter:
%windir%\system32\inetsrv\appcmd.exe set app "Default Web Site/PIWebServices" /
enabledProtocols:http,net.tcp

**Note:**

The command above assumes the Web site hosting PI Web Services is the default Web site.

3. Add an endpoint configuration to the `web.config` file:

```
<netTcpBinding>
  <binding name="netTcpConfig">
    <security mode="Message" />
  </binding>
</netTcpBinding>
```

4. Create an endpoint to specify the `netTcpBinding` and the binding configuration created above:

```
<endpoint binding="netTcpBinding"
  bindingConfiguration="netTcpConfig"
  name="BasicEndpoint"
  bindingNamespace="http://xml.corporate.com/services/PIDataService"
  contract="PIWebService.PIDataService.IPITimeSeries">
  <identity>
    <servicePrincipalName value="HOST/server_machine_name" />
  </identity>
</endpoint>
```

5. Modify the `servicePrincipalName` to reflect the machine name of the Web server hosting PI Web Services.
6. Ensure the service behavior enables impersonation for all operations.
7. If you are using IIS 7.0, you can use **IIS Manager** to configure the `netTcpBinding`:
 - a. Select the Web site that hosts PI Web Services.
 - b. Right-click and select **Edit Binding... Add netTcp**.
 - c. Add the port you want to use for `netTcp` and `*`, for example, `"808: *"`.
 - d. Under the PI Web Services application, right-click the application, then select **Advanced Settings....** Under the **Behavior** section, ensure `netTcp` is listed under **Enabled Protocols**.

**Note:**

If more than one protocol is enabled, such as the case when metadata is exposed over HTTP, enabled protocols are in a comma-delimited list with no spaces between the comma delimiter and the protocol. For example, `http,netTcp`. Verify that no spaces appear in the list.

Use the `netNamedPipeBinding` sample

Use the `netNamedPipeBinding` sample to configure the Web site hosting PI Web Services to listen for `netNamedPipe` requests.

Before you start

To use the `netNamedPipeBinding` with PI Web Services, you must be running Internet Information Services (IIS) on Windows Server 7 or later and the Windows Activation Service (WAS) feature must be installed for use with non-HTTP requests. To check this setting, go to **Windows Features > Microsoft .NET Framework 3.0**, and verify that **Windows Communication Foundation Non-HTTP Activation** is selected.

Procedure

1. From a command prompt with administrator privileges, run the command:
`%windir%\system32\inetsrv\appcmd.exe set app "Default Web Site/PIWebServices" /enabledProtocols:http,net.pipe`

**Note:**

The command line above assumes that the Web server uses the default IIS Web site to host PI Web Services.

2. Add an endpoint configuration to the `web.config` file:

```
<netNamedPipeBinding>
  <binding name="netPipeBinding" >
    <security mode="Transport">
      <transport protectionLevel="None"/>
    </security>
  </binding>
</netNamedPipeBinding>
```

3. Create an endpoint that specifies the `netTcpBinding` and the binding configuration created above:

```
<endpoint address="net.pipe://localhost/PIWebServices/PITimeSeries.svc"
binding="netNamedPipeBinding"
bindingConfiguration="netPipeBinding" name="BasicEndpoint"
bindingNamespace="http://xml.osisoft.com/services/PIDataService"
contract="PIWebService.PIDataService.IPITimeSeries"/>
```

4. Repeat this process to create an endpoint for the Search service, specifying an address of `net.pipe://localhost/PIWebServices/PISearch.svc` and a contract of `PIWebServices.PISearchService.IPISearch`.
5. Ensure the service behavior specifies impersonation for all operations.
6. If you are using IIS 7.0, you can use **IIS Manager** to configure `netNamedPipes`:
 - a. Right-click the Web site hosting PI Web Services and select **Edit Binding...**
 - b. Add **net.pipe**, then ***** for the binding information.
 - c. Under the PI Web Services application, right-click the application, then select **Advanced Settings...**
 - d. Under the Behavior section, ensure **net.pipe** is listed under **Enabled Protocols**.
 - e. If more than one protocol is enabled, such as the case when metadata is exposed over HTTP, enabled protocols are in a comma-delimited list with no spaces between the comma delimiter and the protocol. For example, `http,netpipe`. Verify that no spaces appear in the list.

WCF Service Configuration Editor

You can create and edit WCF configuration files with WCF Service Configuration Editor, `SvcConfigEditor.exe`, a Microsoft tool that ships with Windows SDK. The editor provides an XML authoring environment in which you can open and modify existing configuration files and a wizard to create new configuration files. It requires knowledge of services, binding, endpoints, service behaviors, and binding configurations.

For details, see the MSDN article [Configuration Editor Tool \(SvcConfigEditor.exe\)](#), or the Microsoft documentation for the Windows SDK version you are using.

Firewall security

Firewalls within your LAN, whether dedicated hardware devices or software firewalls such as the one implemented in the Windows operating system, must be configured properly to pass data through the PI Web Services server and your PI Servers and AF Servers.

If you have any internal firewalls between the PI Web Services server and the PI or AF Servers, verify that the appropriate TCP ports are open, or configure exceptions to open the appropriate ports.

Whenever possible, limit firewall exceptions that allow data passage to known networks and IP addresses.

Configure firewall exceptions

Because of the underlying technologies that PI Data Services uses to communicate with PI System servers, some ports must be open on the firewall to allow incoming TCP connections.

Machine	Open these Firewall Ports:
IIS Web Server hosting PI Web Services	Any TCP ports used by applications that transmit data across the Internet; PI Web Services supports these protocols: <ul style="list-style-type: none">• HTTP, typically port 80• HTTPS, typically port 443• netTcp, typically site-specific
PI Server	5450
PI AF Server, version 1.x	5454 and 5455
PI AF Server, version 2.x, 2010 or later	5457 and 5459

To verify whether there are additional ports were added since this version of PI Web Services was released, see *What Ports need to remain open in a firewall for a PI Server and clients to communicate?* on the [OSIsoft Technical Support Web site](#).

Configure firewall exceptions on Windows Server 2008 R2

Procedure

1. Open **Control Panel > Security > Allow a program through Windows Firewall**.
2. Select the **Exceptions** tab and click **Add Port**.
3. Enter a name and port number in the appropriate fields and select **TCP** for the protocol.
4. Verify that the port you add is visible and selected on the **Exceptions** tab.
5. Click **OK**.

After you finish

For more details, see the Microsoft Web page [Firewall: frequently asked questions](#).

PI Web Services Standalone Edition port settings

When PI Web Services is hosted within a Windows Service, a reservation must be made in the operating system to grant the service permission to listen at the specified URL. In most cases, the PI Web Services installer handles this for you automatically.

If the port is later changed from the option selected at the time of installation, or if the service is run as a different user than the default (Network Service), a new URL reservation will need to be made manually using a command line tool provided by Microsoft.

Reserve Namespace on Windows 2003 Server

The utility for Windows Server 2003 is `HttpCfg.exe`. The command line provides a base URL and associates it with an access control list (ACL). To create this association, use this command:

```
HttpCfg.exe set urlacl /u url /a acl
```

where *url* is the URL and *acl* is a Security Descriptor Definition Language (SDDL) string that provides owner and access specifications. A sample command line reserving a URL for PI Web Services on port 8000 and giving the Network Service generic access is:

```
HttpCfg.exe set urlacl /u http://myserver:80/PIWebServices /a D:(A;;GA;;;NS)
```

For details on the SDDL format, see [Security Descriptor String Format](#).

Reserve namespace on the Windows operating system

The utility for Windows is `netsh.exe`, accessible from the Windows command prompt. To reserve a namespace, use this command:

```
netsh.exe http add urlacl url=url user=user sddl=acl
```

where *url* is the base URL, *user* is the user account to associate with the URL, and *acl* is the SDDL string providing access control. For example:

```
netsh.exe http add urlacl url=http://myserver:80/PIWebServices user=domain  
\machine_name$ sddl=D:(A;;GA;;;NS)
```

where *domain* is the Windows domain for the server and *machine_name* is the name of the physical host machine.

Troubleshooting

Information to troubleshoot conditions that prevent PI Web Services from executing correctly are described here. To verify that PI Web Services software is installed correctly, see the procedures included in [PI Web Services installation](#).

If you use the default installation of PI Web Services, errors are written to the Windows application log on the server hosting the web service.

PI Web Services offers further error and trace message logging options to debug web service deployments. You can also record errors on the PI Server in a PI message log, or log trace messages in normal operation. For configuration details, see [View and configure message logs](#).

To identify and fix common error messages that can occur when calling PI Web Services, select the Help topic that most closely describes the error.

PI Web Services IIS Edition

Cannot connect to PI Web Services

```
Could not connect to "http://<PIWebServiceHostMachineName>/PIWebServices/PITimeSeries.svc"  
PI Web Server Name/PIWebServices/PITimeSeries.svc. TCP error code 10061: No connection  
could  
be made because the target machine actively refused it  
<PIWebServiceHostMachineIPAddress>.
```

This error indicates:	To resolve this error:
For PI Web Services 2012 IIS Edition, that Microsoft Internet Information Services (IIS) on Windows Server is not running if a user receives this error when they enter the URL of the Web service into the Internet Explorer browser.	Verify that Microsoft Internet Information Services (IIS) is running.
For PI Web Services 2012 Standalone Edition, that the Windows service is not started.	Verify that the Windows service is started.

Server not found

The requested server was not found in the known servers table PI Server name.

This error indicates:	To resolve this error:
If a SOAP fault with this message is received in response to a PI Web Services call, the PI Web Services call includes a path to a PI Server that is not in the Known Servers Table (KST), although that server might be running.	Verify that the path includes a PI Server that exists in the Known Servers Table (KST). If it does not, add the PI Server to the KST. See the PI SDK Utility Help for details.
The PI SDK feature that automatically adds servers to the KST is <i>disabled</i> .	Enable this feature as described in the PI SDK Utility Help to avoid this error.

PI System not found

PI System not found

This error indicates:	To resolve this error:
PI AF server is not running if a SOAP fault with this message is received when a PI Web Services call is sent.	Verify that the PI AF server that is included in the path of the web services call is running.

Remote connection failure

Unable to open a session on a server. [-10758] Failed to create remote connection.: bad PI Server name.

This error indicates:	To resolve this error:
<ul style="list-style-type: none">• A problem with the PI Server connection.• A PI Server that is not running.• A <i>server</i> name in the path that is not a valid PI Server.• Impersonation is not properly configured. See Configure Security for .NET Clients.	<ul style="list-style-type: none">• Use the PI SDK Connection Manager to test the connection between the PI Server and the machine on which PI Web Services is installed.• Verify that your PI Server is running.• Review and verify that the correct name of the PI Server is used in the path used to connect to PI Web Services.• Refer to Configure Security for .NET Clients.


Data entry not allowed

Data entry is disallowed by system configuration. from <user domain>\<user>

This error indicates:	To resolve this error:
The InsertPIData method is disabled, if the error is received with attempts to use InsertPIData where the identity of the calling user is reflected in <user domain>\<user>.	Review the AllowDataEntry key in the PIWebServiceSettings section of web.config and set the AllowDataEntry key to TRUE.

Insufficient permissions

1000: Insufficient permission to access or complete operation. [-10401] No Write Access - Secure Object

This error indicates:	To resolve this error:
The PI Web Services user does not have write access to the PI point included in the path of the InsertPIData call, and does not have the permissions required to complete PI System data insertions.	Verify that users of PI Web Services have write access to the PI points into which they insert data.  Note: Failed events are flagged in the TimedValue Status property. To determine which insertions failed, examine the time and the path associated with TimedValue Status property.

Insufficient message size

The maximum message size quota for incoming messages (configured_limit_or_65536) has been exceeded.

This error indicates:	To resolve this error:
That data exceeds the size limit that the client is prepared to accept.	Increase the maximum message quota on the web server: <ul style="list-style-type: none"> Set the <code>maxReceivedMessageSize</code> property in the appropriate binding element of the <code>web.config</code> file. For a WCF client, set <code>maxReceivedMessageSize</code> property in the <code>app.config</code> file. If the property does not appear in the <code>web.config</code> or <code>app.config</code> files, the default message size is 65,536 bytes. See Control message size for details about how to estimate and configure message sizes.

Execution of PI Performance Equations not allowed

Calculations are disallowed, path = `pe:\\server\pe_expression`

where the `path = string` shows the path that the client passed to the web service.

This error indicates:	To resolve this error:
That the <code>web.config</code> setting to allow PI Performance Equation calculations is disabled if the error occurs during attempts to retrieve data through a call that contains a <code>pe</code> designator.	<ul style="list-style-type: none"> Use the <code>path = string</code> in the error message to identify which request failed. To enable this setting, set AllowCalculations to TRUE in the PIWebServiceSettings section of the <code>web.config</code> file.

Unsupported filters or parameters

Filters are not supported for `pi` paths for mode: `interpolated` Parameter name: `PIArcManner`


This error indicates:	To resolve this error:
The use of web method parameter values, or combinations of parameter values, that are not currently supported by PI Web Services. If unsupported parameters or parameter values are used, a SOAP fault containing this message or a similar one is sent.	Review the setting of the named parameter and use the information here to identify the parameter values or combinations of parameter values that are not supported.

Source Designator	Operation	Parameter	Unsupported Value
<code>pi</code>	<code>GetPIArchiveData</code>	<code>PIArcManner.Filter</code> and <code>RetrievalType</code>	Any non-empty filter value for a <code>RetrievalType</code> of <code>PlotValues</code>
<code>af</code>	<code>GetPIArchiveData</code>	<code>PIArcManner.Filter</code> and <code>RetrievalType</code>	Any non-empty filter value for a <code>RetrievalType</code> of <code>PlotValues</code>
<code>pe</code>	<code>GetPIArchiveData</code>	<code>PIArcManner.Filter</code>	Any non-empty filter value
<code>pi, pe, af</code>	<code>GetPISummaryData</code>	<code>PISummaryManner.Filter</code>	Any non-empty filter value
<code>pi, pe, af</code>	<code>GetPIArchiveData</code> , <code>GetPISummaryData</code>	<code>TimeRange</code>	Mismatch in time zones between Start and End ; Start and End must have the same time zone

Source Designator	Operation	Parameter	Unsupported Value
pe	GetPIArchiveData, GetPISummaryData	Attempting retrieval when AllowCalculations is FALSE in web.config	Retrievals do not occur if performance equations are disabled; default AllowCalculations setting is TRUE, which allows PI performance equation calculations
pi	InsertPIData	events	Data inserts must contain events; inserts with no events are not allowed
pe, af	InsertPIData	all	Data entry is not permitted for pe and af paths
pi, pe, af	InsertPIData	Attempting insertion when AllowDataEntry is FALSE in web.config	No insertions are performed if AllowDataEntry is FALSE; by default data entry is enabled, or TRUE
pe	GetPIArchiveData, GetPISummaryData	path	No server token when using the pe source designator

Server error in PI Web Services application

Server Error in '/PIWebServices' Application

This error indicates:	To resolve this error:
Permissions on the file path specified in the error message are restricted if received when a user enters a URL in Windows Internet Explorer browser to verify the PI Services installation.	<p>Verify that the directory in the message has read and write access for the Process ID account:</p> <ul style="list-style-type: none"> The folder Temporary ASP.NET Files, typically found in C:\Windows\ <p> Note: Microsoft.NET\Framework\v2.0.50727, is used by CLR to store compiled code for ASP.NET applications. PI Web Services does not modify this in any way. The Network Service account requires write permission to this folder and its subfolders.</p> <ul style="list-style-type: none"> You can also use the aspnet_regiis.exe utility to access these permissions. The utility can be found in C:\Windows\Microsoft.NET\Framework\v2.0.50727. Or, to open in a command prompt window, enter: aspnet_regiis -ga "NT AUTHORITY\NETWORK SERVICE"

Invalid tag name

2147220432 - Invalid tag name

This error occurs:	To resolve this error:
If the syntax for the path that designates the source of your PI System data is not correct.	Review the syntax of the path included in the source designator and correct as required.
With programming languages that use the backslash character, \, to represent special characters are used, as is the case with C#.	Use two backslashes to represent one backslash. For example, use pi: \\pserver1\sine to represent the path pi: \pserver1\sine.

Anonymous authentication required

If this error occurs under one of these conditions:

- The URL `http://localhost/PIWebServices?PITimeSeries.svc` is entered into a browser
- PI Web Services methods are accessed through WCFStorm

```
Security settings for this service require 'Anonymous' Authentication but it is not
enabled
for the IIS application that hosts this service. Exception Details:
System.NotSupportedException:
Security settings for this service require 'Anonymous' Authentication but it is not
enabled for the
IIS application that hosts this service
```

This error indicates:	To resolve this error:
Security settings for Microsoft Windows and Internet Information Services (IIS) for Windows Server are mismatched, when PI Web Services is installed for the first time on an IIS host machine that was previously configured.	One strategy to resolve this conflict is to configure the security settings that are associated with PI Web Services to use only bindings that are set to all <code>basicHttpBinding</code> , or <code>wsHttpBinding</code> , depending on the desired level of SOAP compliance. These bindings should be configured with a security mode of either <code>transport</code> for use with <code>https</code> , or <code>TransportCredentialOnly</code> for use with Kerberos/SPNEGO over <code>http</code> . For further information, see the example below or: <ul style="list-style-type: none">• The MSDN article <i>wsHttpBinding Security Class</i>, available on the MSDN website.

Examples

Change security mode of the `basicHttpBinding` web service binding:

1. Ensure that `http` bindings are enabled for the Internet Information Services site that hosts PI Web Services.



Note: This setting should be enabled by default.

2. Open the PI Web Services `web.config` file and change the security mode to `TransportCredentialOnly`:

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="basicBinding" >
        <security mode="TransportCredentialOnly">
          <transport clientCredentialType="Windows"></transport>
        </security>
      </binding>
    </basicHttpBinding>
```

Remove the `mexHttpBinding` in the PI Web Services `web.config` file:

```
<services>
  <service behaviorConfiguration="PIDataService.PITimeSeriesServiceBehavior"
    name="PIWebServices.PIDataService.PITimeSeriesSvcImpl">
    <!--endpoint address="mex"
    binding="mexHttpBinding"
    name="mexBasicEndpoint"
    contract="IMetadataExchange" /-->
    <endpoint binding="wsHttpBinding" bindingConfiguration="wsBinding_2011
    name="BasicEndpointbindingNamespace="
```

```

contract=PIWebService.PIDataService.IPITimeSeries">
  <identity>
    <servicePrincipalName value="Host/web_server_machine_name" >
    </servicePrincipalName>
  </identity>
</endpoint>
</service>

```

Change security mode of the wsHttpBinding web service binding:

1. Ensure that http bindings are enabled for the Internet Information Services site that hosts PI Web Services.



Note: This setting should be enabled by default.

2. Open the PI Web Services web.config file and change the security mode to TransportCredentialOnly:

```

<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="wsBinding_2011" >
        <security mode="Transport">
          <transport clientCredentialType="Windows"></transport>
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
</system.serviceModel>

```

Remove the mexHttpBinding in the PI Web Services web.config file:

```

<services>
  <service behaviorConfiguration="PIDataService.PITimeSeriesServiceBehavior"
    name="PIWebServices.PIDataService.PITimeSeriesSvcImpl">
    <!--endpoint address="mex"
    binding="mexHttpBinding"
    name="mexBasicEndpoint"
    contract="IMetadataExchange" /-->
    <endpoint binding="wsHttpBinding" bindingConfiguration="wsBinding_2011"
    name="BasicEndpointbindingNamespace="
    contract=PIWebService.PIDataService.IPITimeSeries">
      <identity>
        <servicePrincipalName value="Host/web_server_machine_name" >
        </servicePrincipalName>
      </identity>
    </endpoint>
  </service>

```

PI Web Services Standalone Edition

If the service does not start

```

Service cannot be started. System.Runtime.CallbackException: A user callback threw an
exception.
Check the exception stack and inner exception to determine the callback that failed.
---> System.InvalidOperationException: Service PIWebServicesHost was not found on
computer '.'.
---> System.ComponentModel.Win32Exception: The specified service does not exist as an
installed service

```

This error indicates:	To resolve this error:
Sufficient rights have not been reserved through the security configuration, and therefore the service host does not operate properly.	If the Windows service starts and immediately shuts down upon an attempted startup, and entries in the Windows event log do not indicate a security issue, review your security configuration. If necessary, add the appropriate permissions through a <code>urlacl</code> reservation as described in Configure security for PI Web Services Standalone Edition .

PI Web Services programmer reference

The PI Web Services programmer reference contains information required to develop applications that consume data from PI Web Services, and describes how such applications retrieve data from and search for data within PI Systems.

PI Web Services is divided into interfaces based on the type of PI System data that is returned. Each interface contains the individual Web methods and classes that allow you to build Web service applications that access PI System data.

Web service inputs

In SOAP web service implementations like PI Web Services, inputs and outputs are specified by means of a WSDL document and XML schema, which make method signatures and data structures accessible to clients. In all major programming platforms, including Microsoft .NET, Java, Ruby, Python, C, C++, and Objective-C, client libraries capable of consuming a WSDL and translating the native XML/SOAP calls of PI Web Services into an object-oriented paradigm are widely available. For more information about how to access the WSDL document, see [Automatic WSDL generation](#).

Before programming, you should also familiarize yourself with the three common data concepts in PI Web Services that are used throughout the product: [paths](#), [constraints](#) and [manner](#).

PI Web Services paths

A PI Web Services path designates a unique source of PI System data, such as a PI point, or a PI AF attribute. There are currently six types of PI Web Services paths.

PI Server paths

PI Server paths refer to a single PI point on a single PI Server, and take the form `pi:\\server\pipoint`

- Where *server* is the name of a PI Server that is resolvable on the server hosting PI Web Services
- Where *pipoint* is the name of a PI point on that server



Note: In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Alternatively, the form `pi:pipoint` may be used. In this construction, where the name of the server is omitted, the default PI Server configured on the PI Web Services server is queried.

Examples

- `pi:\\piServer1\\sinusoid`
- `pi:sinusoid`

PI Performance Equation paths

PI Performance Equation paths refer to a resolvable Performance Equation paths that take the form `pe:\\server\\performance-equation`

- Where *server* is the name of a PI Server that is resolvable on the server hosting PI Web Services
- Where *performance-equation* is a PI Performance Equation that can be executed on the referenced PI Server



Note: In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Examples

- `pe:\\piServer1\'sinusoid\'*2`

PI AF element paths

PI AF element paths take the form `af:\\server\database\path\to\element`

- ◦ Where *server* is the name of a PI AF server that is resolvable on the server hosting PI Web Services
- ◦ Where *database* is the name of a PI AF database on that PI AF server
- ◦ Where *path\to\element* represents a resolvable hierarchical path to an AF element in that AF database



Note: In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Examples

- `af:\\afServer1\database1\RootElement`
- `af:\\afServer1\database1\RootElement\Child1\Child2`

PI AF element attribute paths

PI AF Element Attribute paths take the form `af:\\server\database\path\to\element|attribute;UOM Conversion`

- Where *server* is the name of a PI AF server that is resolvable on the server hosting PI Web Services
- Where *database* is the name of a PI AF database on that PI AF server
- Where *path\to\element* represents a resolvable hierarchical path to an AF element in that AF database
- Where *attribute* represents an attribute or child attribute of that element
- Where *UOM Conversion* represents an optional conversion of the Unit of Measure of the AF attribute. This conversion applies when accessing data from a PI AF element attribute with a defined native Unit of Measure, or when the AF extended settings override the native Unit of Measure. The conversion is only applied in cases where such conversion is useful, that is, when data values are being retrieved directly.



Note: In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Examples

- `af:\\afServer1\database1\Element|Attribute`
- `af:\\afServer1\database1\Element|Attribute|ChildAttribute1|ChildAttribute2`
- `af:\\afServer1\database1\Element\Child1\Child2|BoilingPoint; degree Fahrenheit 1`

PI event frame paths

PI Event Frame paths take the form `ef:\\server\database\EventFrames[{id}]`

- Where `server` is the name of a PI AF server that is resolvable on the server hosting PI Web Services
- Where `database` is the name of a PI AF database on that PI AF server
- Where `id` is the globally unique identifier (GUID) of a PI event frame in that AF database



Note: In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Examples

```
ef:\\afServer1\database1\EventFrames[{12345678-abcd-ef01-2345-6789abcd0123}]
```

There also exists a second form of PI event frame path, of the form `ef:\\server\database\EventFrames[name]`

- Where `server` is the name of a PI AF server that is resolvable on the server hosting PI Web Services
- Where `database` is the name of a PI AF database on that PI AF server
- Where `name` is the name of a PI event frame in that AF database

Name-based event frame paths are not guaranteed to be unique. In methods where uniqueness is required, such as in [GetEventFrameTimeSeries method](#) or [GetEventFrameValues](#), name-based event frame paths are not permitted. In other cases, their use is discouraged, as name-based event frame paths may result in undesirable ambiguity when attempting to resolve such paths.

Examples

- `ef:\\afServer1\database1\EventFrames[EventFrameA]`
- `ef:\\afServer1\database1\EventFrames[EventFrameA]\ChildEventFrame`

PI event frame attribute paths

PI Event Frame paths take the form `ef:\\server\database\EventFrames[{id}]| attribute`

Notes

In some cases new PI Servers may be resolvable automatically, see [Automatic resolution of new servers](#).

Name-based Event Frame paths, while supported in the PI AF SDK, are not supported by PI Web Services. Name-based Event Frame paths are not guaranteed to be unique, which leaves undesirable ambiguity when attempting to resolve such paths.

Examples

- `ef:\\afServer1\database1\EventFrames[{12345678-abcd-ef01-2345-6789abcd0123}]|Attribute1`
- `ef:\\afServer1\database1\EventFrames[{12345678-abcd-ef01-2345-6789abcd0123}]|Attribute1|Attribute2`

Manner

Manner determines how data results are presented for results that are matched internally within PI Web Services. Unlike constraints, items specified in the manner do not influence whether a particular data point is matched by your search; instead they determine how such data points are accessible in the results.

Examples

- For retrievals of time series data from a PI System, use manner to determine the maximum number of data values that are returned and whether results should be compressed or interpolated.
- When searching for AF element data, use manner to specify a maximum number of levels of child elements to return and whether template names are included.

Constraints

Constraints define the filters of a PI Web Services data query and control how the results are matched internally within PI Web Services. In any given call, constraints determine which data points may be returned, subject to conditions specified in the manner.

Examples

- Retrievals of time series data from a PI System can use constraints to specify the time boundaries including StartTime and EndTime.
- Searches of AF element data can use constraints to specify a NameMask that will apply to matched elements or screen for elements with attributes with defined values that apply to a specific search mask.

Time stamps

This section describes the rules and formats that apply to time stamps used by PI Web Services.

Time input translation rules

In general, use strings to define time constraints in a data retrieval Web method. For details about the syntax that you can use for time stamps and time intervals, see PI time.

With one exception, fixed time strings are translated to PI System internal time on the PI Web Services server, rather than the PI Server. As a result, the translation might not occur as expected when:

- You assume that the PI Web Services server and the PI Server are in the same time zone, but they are instead located in different time zones
- The transition to or from daylight savings time occurs

To avoid ambiguity, use ISO 8601 time string format and include the time zone offset.

Exception to time input translation

There is an exception to the time input translation rule: when InsertPIData is used, the time stamps of the data are used as the time inserted into the PI Server. InsertPIData takes a TimeSeries array as input. All time stamps

in this array are defined as **fixed times** stamps in UTC format which translates to ISO 8601 UTC when represented in XML as a string.

Returned time stamps

Time stamps returned by all PI Web Services methods are defined as **fixed** UTC time stamps in ISO 8601 UTC format. As a result, clients will typically determine the local time zone and perform the desired conversion and will adjust for variations such as DST.

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.

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 <i>DD</i>	december <i>DD</i>	00:00:00 (midnight) on the <i>DD</i> th day of december in the current year
<i>YYYY</i>		00:00:00 (midnight) on the current day and month in year <i>YYYY</i>
<i>M-D</i> or <i>M/D</i>		00:00:00 (midnight) on the <i>D</i> th day of month <i>M</i> in the current year
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.		

Abbreviation	Full version	Reference time
DD		00:00:00 (midnight) on the DDth day of the current month
<p>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.</p>		

PI time expressions

PI time expressions can include a reference time and a time offset, indicated by a direction (either + or -) and a time unit with a value. PI time expressions might include:

- Only a reference time, such as t
- Only a time offset, such as +3h
- A reference time with a time offset, such as t+3h

A reference time can be a fixed time, such as 24-aug-2012 09:50:00, or a valid reference-time abbreviation, such as t.

You can only include 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 or fixed times with a time offset

When included with a fixed time or 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
1-jan-11-1d	00:00:00 (12:00 a.m.) December 31, 2010

- 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 time 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

IPITimeSeries interface

The IPITimeSeries interface contains methods that retrieve PI System data as collections of [TimeSeries](#) objects. [TimeSeries](#) data consists of time stamped data values as stored in a PI Server. [TimeSeries](#) data are identified by a [PI Server path](#), [PI Performance Equation path](#) or [PI AF element attribute path](#).

IPITimeSeries web methods

CancelPIUpdates method

CancelPIUpdates will cancel sign-ups for paths associated to an update ticket, and therefore free resources. After this method is called, the update ticket passed is no longer valid when used in [GetPIUpdates method](#).

Syntax

```
void CancelPIUpdates (Guid updateTicket)
```

Arguments

- **updateTicket**

A unique identifier that denotes a collection of paths that is currently signed up for updates.

Notes

If an update ticket is found in the list of update sign-ups, the sign-up is removed. If an update ticket is not found in the list of sign-ups, no error is returned. When an update ticket is cancelled, updates are no longer available for any of the paths in the collection referenced by the update ticket and the update ticket is invalid.

If a given path is signed up for updates through another update ticket, that update ticket can still be used to get updates for the path.

GetPIArchiveData method

GetPIArchiveData retrieves data from the PI Server as Compressed, Interpolated and Plot values, and returns the data as an array of [TimeSeries](#) objects. This method also makes it possible to sign up for data updates.

GetPIArchiveData will generate one update ticket for every path for which updates are requested. Use the [SignUpForPIUpdates method](#) to sign up group of paths at one time and receive a single update ticket for the group.

Syntax

```
TimeSeries[] GetPIArchiveData (PIArcDataRequest[] requests)
```

See [TimeSeries](#) and [PIArcDataRequest](#).

Arguments

Array of [PIArcDataRequest](#) objects.

Returns

Array of [TimeSeries](#) objects.

The returned [TimeSeries](#) array has the same number of elements as the input [PIArcDataRequest](#) array.

Errors

A data request that includes a sign-up for data updates, but cannot honor the sign-up, will fail and report an error condition even if the request itself can be satisfied.

Do not use an event frame path that contains either:

- a retrieval type of Interpolated is used and an empty start or end time is passed
- a [reference time](#) offset is passed without a known PI time token, such as * or T
Such event frame paths will fail with the message:

```
Interpolation interval cannot be correctly calculated when relying
on an event frame time range, path is path_string.
```

Condition	Error Property Value	ErrDesc Property Value
Updates=true for af or pe path	4000	Updates are not supported for this type of path: [path].
Updates=true and end time is not reference time	4001	The end time of the request is not reference time and updates cannot be performed.
Updates=true, af path, af DR has an extended definition	1005	Updates are not supported for PI AF data references with extended properties.

Notes

The behavior during retrieval of Compressed values depends on whether the PI Server contains events for the specified time range. If the time range specified in the request contains no events, the [TimeSeries](#) object returns an empty [TimedValue](#) array. If query [paths](#) contain stale data, PI Web Services retrieves data only if the time range includes the time of the most recent snapshot. To prevent retrieval of empty arrays, set the **Boundaries** property of the [PIArcManner](#) object of the [PIArcDataRequest](#) object to **Outside**.

This is important for queries of snapshot values that specify a [TimeRange](#) for the current time; these queries must specify **StartTime=*** and **EndTime=*** since the snapshot value is usually before or at current time.

The [TimeRange](#) portion of the [PIArcDataRequest](#) has unique handling when used with PI event frame attribute paths that point to attribute data references. [Fixed time](#) and [reference time](#) times may be used for the start and end times of the time range, as is true for any other path type. If an empty string "" is passed for the start and end times, the time range of the event frame is used for the time range. When using this feature to retrieve interpolated values, the [Duration](#) property of [PIArcManner](#) object must be used or an error will result. If an offset without a known PI System time token, such as -2H, is used, the offset is applied to the event frame time. For example, if the path `ef:\\server\database\EventFrames[S21Process]|OpTemp` is used with the time range of -2H, -1H, data will be returned for the time range with a start time of S21Process, which is 2H, end time of S21Process, which is 1H.

Calls to this method can also implicitly sign up one or more PI point or PI AF paths to receive data updates, if the [Updates](#) property of the object is set to **TRUE**. When the registration succeeds, the [SeriesID](#) property of the returned [TimeSeries](#) object for that path is a string representation of the ticket that can be used with [GetPIUpdates method](#), [ListPathsByUpdateTicket method](#), and [CancelPIUpdates method](#). Each path registered will receive its own unique ticket in the [SeriesID](#) of the [TimeSeries](#) object returned.

Client applications can sign up for updates from multiple paths in a group operation if you register the paths using [SignUpForPIUpdates method](#).

GetPISnapshotData method

GetPISnapshotData returns the current snapshot for each [path](#) that is passed to this method.

Syntax

```
TimeSeries [] GetPISnapshotData(string[] paths)
```

See [TimeSeries](#) and [GetPISnapshotData method](#).

Errors

The returned [TimeSeries](#) array has the same number of elements as the input string array. If an error occurs retrieving data for an individual path, the error code and description will be reflected in the corresponding element of the [TimeSeries](#) array.

Notes

The returned [TimeSeries](#) array has the same number of elements as the input string array.

This method is equivalent to [GetPIArchiveData method](#) with a [PIArcMannerRetrievalType](#) of **Compressed** and [PIArcMannerBoundaries](#) of **Outside** for the [TimeRange](#) that denotes current time; these queries must specify **StartTime=*** and **EndTime=*** since the snapshot value is usually before or at current time.

Each successfully returned [TimeSeries](#) array contains a [TimedValue](#) collection with a single [TimedValue](#) object that represents the snapshot value.

GetPISummaryData method

[GetPISummaryData](#) returns summaries of PI archive data. This method can return averages, means, minima, maxima, ranges, totals, values, counts, and both sample and population standard deviations.

Calls to this method may also implicitly register one or more [PI Server path](#) to receive data updates if the [Updates](#) property of the [PISummaryManner](#) object is **TRUE**. [GetPISummaryData](#) will generate one update ticket for every path for which updates are requested. To sign up group of paths at one time and receive a single update ticket for the group, see [SignUpForPIUpdates method](#).

Syntax

```
TimeSeries[] GetPISummaryData(PISummaryDataRequest[] requests);
```

See [TimeSeries](#) and [PISummaryDataRequest](#).

Arguments

Array of [PISummaryDataRequest](#) objects

Returns

Array of [TimeSeries](#) objects

Errors

Calls to the [GetPISummaryData](#) method will report an error if:

- The web service endpoint cannot be reached and no data are returned.
- A data request that involves a sign-up for data updates cannot honor the sign-up. Such a call will fail even if the data request can be satisfied.
- It contains an empty start or end time, or an offset without a known PI time token; such calls will result in an exception with the message:

```
Summary interval cannot be correctly calculated when relying  
on a PI event frame time range, path is path_string.
```

Data retrievals that fail for a [PISummaryDataRequest](#) object, can result in various errors. These error codes originate with PI Web Services:

Condition	Error Value	ErrDesc Value
Updates=true for pe path	4000	Updates are not supported for this type of path: [path].
Updates=true and end time is not a PIreference times	4001	The end time of the request is not a PI reference times and updates cannot be performed.
Updates=true, af path, af DR has an extended definition	1005	Updates are not supported for PI AF data references with extended properties.
PI point or a PI AF data reference to a PI point is not numeric data type, such as digital point, string point, timestamp point, and so on.	3000	Updates are not supported for summary calculations on non-numeric data.

Notes

Each path for which Updates is **TRUE** is:

- signed up for updates separately
- receives its own unique update ticket in the SeriesID property of the returned [TimeSeries](#) object

The TimeRange portion of the [PIArcDataRequest](#) has unique handling when used with [PI event frame attribute paths](#) that point to attribute data references. [PI fixed times](#) and [reference times](#) may be used for the start and end times of the time range, as is true for any other path type. If an empty string "" is passed for the start and end times, the time range of the PI event frame is used for the time range. When using this feature, the TimeStep property of the PISummaryManner object must be used to specify the calculation interval for the interpolated values. If an offset without a known PI System time token, such as -2H, is used, the offset is applied to the PI event frame time. For example, if the path ef: \\server\database\EventFrames[S21Process] | OpTemp is used with the time range of -2H, -1H, data will be returned for the time range with a start time of S21Process, which is 2H, end time of S21Process, which is 1H.

When the registration succeeds, the SeriesID property of the returned [TimeSeries](#) object for that path is a string representation of an update ticket that may be used with [GetPIUpdates method](#), [ListPathsByUpdateTicket method](#) and [CancelPIUpdates method](#). When updates are later retrieved using the update ticket, the updates represent new or changed values to the recorded values. They do not represent changes to the summary.

Summary data may be retrieved for [PI event frame path](#) pointing to attributes configured as PI point data references provided start and end times are specified with [PI fixed](#) or [PI reference times](#).

When using this feature, the Duration property of the [PISummaryManner](#) does not specify the summary calculation interval.

GetPIUpdates method

Client applications call the GetPIUpdates method to retrieve data updates for paths that were previously signed up to receive data updates through [GetPIArchiveData method](#), [GetPISnapshotData method](#), or [SignUpForPIUpdates method](#).

Syntax

```
TimeSeriesUpdates[] GetPIUpdates(Guid updateTicket, ushort maxWaitForUpdates, UpdateFilterType evtFilter)
```

See [TimeSeriesUpdates](#) and [GetPIUpdates method](#).

Arguments

- **updateTicket**

A unique identifier that denotes a collection of paths that is currently signed up for updates.

- **maxWaitForUpdates**

Maximum number of seconds to wait during a call for updates. If this value is greater than zero, the call will block until updates are available or this number of seconds elapses without updates. If the number is zero, the call makes one attempt to retrieve updates and returns.

- **evtFilter**

Enumerated value that indicates whether to return snapshots (Snapshot), archive events (Archive), or both snapshot and archive event updates (SnapshotAndArchive).

Returns

An array of [TimeSeriesUpdates](#) objects, is returned for each path associated with the specified update ticket. If no data updates occurred for a registered path since the last call for updates, a [TimeSeriesUpdates](#) object is returned with an Updates property that has no [TimedValueUpdate](#) objects, and the Error property of the [TimeSeriesUpdates](#) object has the value **o**. The SeriesID property of all [TimeSeriesUpdates](#) objects is the same as that included in the updateTicket.

**Note:**

The [TimeSeriesUpdates](#) object has a [TimedValue](#) member which is always null. The collection of updates is called Updates.

All data updates indicate the type of action that caused the update in the enumerated UpdateType property of the returned [TimeSeriesUpdates](#):

Action	UpdateType Value
New snapshot value	Snapshot
New archive event	Archive
Deletion of existing event	Delete
Change to existing event	Edit
User inserts event into the PI Server with a time stamp that matches the time stamp of an existing event	AddNoReplace

Errors

If the update ticket does not reference an existing sign-up, an Error 4002 and an ErrDesc value The update ticket was not found in the list of tickets registered for updates is returned in an otherwise empty [TimeSeries](#).

Notes

When maxWaitForUpdates is greater than 0, the Web method checks if events occur in any of the paths associated with the update ticket. If no events occur for any of the paths, the Web service waits until any updates are available or the specified number of seconds has elapsed. When maxWaitForUpdates equals 0, PI Web Services returns all available updates and does not wait.

Use maxWaitForUpdates when the frequency of events for a given PI point is close to the interval at which the application makes calls to the GetPIUpdates method. For such a case, this parameter provides a grace period to avoid another call.

**Note:**

Performance can be negatively impacted if `maxWaitForUpdates` is set to a non-zero value. OSIsoft recommends that you call this Web method asynchronously to ensure that the client application remains responsive.

Sign-ups for data updates expire after the interval is set by the `UpdatePurgeInterval` property in the `web.config` file. This feature prevents clients from failing or terminating before a sign-up is cancelled. The Web service will return a [TimeSeries](#) object with the Error 4002 if a sign-up has expired, was cancelled by a [CancelPIUpdates method](#) call, or used an incorrect update ticket.

GetProductVersion method

`GetProductVersion` reports the version of the assembly that implements the interface as viewed in Windows Explorer.

Syntax

```
string GetProductVersion()
```

Returns

A string representing the version of the `PIWebServices.dll` assembly.

Errors

None

Notes

The version is determined when the assembly is loaded and stored for subsequent use and is displayed as a string value.

Example

```
1.3.0.0
```

InsertPIData method

The `InsertPIData` method allows a client to insert values into one or more locations in the PI System by passing an array of [TimeSeries](#) objects, each of which contains an array of [TimedValue](#) objects.

Syntax

```
TimeSeries InsertPIData(TimeSeries [] events, PIInsertDuplicateHandling duplicateSwitch)
```

See [TimeSeries](#).

Arguments

- **Events**
An array of [TimeSeries](#) objects that contain the data to be inserted into the PI System.
- **duplicateSwitch**

An enumerated type that controls how duplicate values are processed:

Enumeration Value	Usage
InsertDuplicate	Always insert a new value. This can result in the creation of duplicate values, that is, multiple values with the same time stamp.
ReplaceDuplicate	Always insert the value, and replace an existing event in the archive that has an identical time stamp as the submitted value. If multiple duplicate events exist in the archive, the first duplicate encountered is updated with the submitted value.
ReplaceOnlyDuplicate	Insert the timed value only if it replaces an existing value with the same time stamp. If there is no value at the passed time stamp, the insertion does not occur and an error is returned.
ErrorDuplicate	Insert a timed value if there is no existing value at the passed time stamp. Return an error if there is already a value in the PI Server with the same time stamp.
ErrorDuplicatesSilent	Write an error to the PI Server log when an event with the same time stamp exists but does not return an error.
ReplaceOnlyDuplicatesSilent	Write an error to the PI Server log when there is no pre-existing event with the given time stamp but no error is returned to the client.

Properties

Each **TimedValue** object must contain either the **Value** or **Status** to determine what is inserted into the PI System. The value's time stamp comes from the **TimedValue** Time property. The target path comes from the **TimeSeries** object's path member. If the **TimedValue** path member is populated, its path will override the **TimeSeries** object's path for that **TimedValue** only.

- **Status**

Status must be null or blank or 0 if you intend to insert good data. Do not pass strings for **Status**, such as Good data.

To use the **TimedValue Status** property setting, it is important to note that:

- When writing to a PI point, if the **Status** property is set with a negative number, it is passed as is. However, the **fixed** value of the property is interpreted as the number of a **system digital state** and the string value of that digital state is inserted in place of the value passed. For example, if Status is set to **-254**, the system digital value **Shutdown** is inserted.
- When writing to a PI point, if **Status** is a string that contains the text of a system digital state but the PI point is not a string tag, the PI Server will reflect a system digital state. For example, if **Status** is set to **Shutdown** and the PI point data type is Float16, the Value of the PI point is the digital state **Shutdown**.
- When writing to a PI point, if **Status** is a string and the PI point is a string, the PI Server is unable to determine whether you are inserting a string or a digital state. In such cases, it is recommended that the **Status** property be set to a negative number.

- **Value**

In most cases, the value of the **Value** property should be set to the data value that will be inserted. This is true for all data types, including digital state PI points.

Returns

If all data values are successfully inserted into the PI System, this web method returns a [TimeSeries](#) object with no [TimedValue](#). The value of the **SeriesID** property of the first [TimeSeries](#) passed into the method is copied to the **SeriesID** property of the [TimeSeries](#) returned to the client. Client applications can use this to keep track of asynchronous calls to `InsertPIData`.

If any [TimedValue](#) objects are returned, they represent the objects that could not be inserted due to errors. The value and time stamp of the failed value will be found in the members of [TimedValue](#). The [TimedValue](#) object's properties will contain the values the caller tried to insert, with the exception of the **Status** property, which records the error message.

Errors

Condition	Error Property Value	ErrDesc Property Value
If duplicateSwitch is ReplaceOnlyDuplicate and no value exists in the PI Server at the submitted time stamp, the Status property of the returned TimedValue is:	-2147219630	No value exists at specified timestamp.
If the client cannot reach the web service end point, the method returns a fault and no data are returned		
If duplicateSwitch is ErrorDuplicate and a value exists in the PI Server with the submitted time stamp	-2147219631	Value already exists at specified timestamp.
If the Status property is set to a negative integer whose fixed timevalue does not correspond to a system digital state string, this error is set in the Status property of the TimedValue object returned when the insertion is rejected	-2147220396	Digital State not found. <i>value</i>
If a string is passed in the Status property that does not correspond to a system digital state value and the path refers to a non-string type tag	-2147219633	Server returned write error: : [-15013] PIvalue Type or PIstring is Not <i>tag_datatype</i>

Notes

When data is inserted into the PI System, one of the following paths can be used:

- [PI Server path](#)
- [PI AF element attribute path](#)
- PI event frame attribute path

When using the `InsertPIData` method, the path cannot refer to:

- a Performance Equation
- a PI AF element
- an event frame

The calling user must have the necessary privileges to write events to the PI System. The `web.config` file for the web service must also have its **AllowDataEntry** key set to the default value **TRUE**.

Since the Time property of [TimedValue](#) has a **DateTime** data type, the client must generate an [fixed](#) UTC time stamp for data to be inserted into the PI System. Fixed and reference time strings, including current time as `*`, cannot be used.

When failed events appear, the time stamps of these failed events are in the ISO 8601 UTC format. That is, they will end in **Z** regardless of how they were passed in.

When failed events are returned, the **Status** property of the **TimedValue** contains an error message; this is the only instance in which the **TimedValue Status** property returned by PI Web Services does not contain a digital state.

When writing to the location with a timestamp data type, the Value property of the **TimedValue** should be an ISO 8601 UTC date time string. If local PI time strings are passed, they are converted to UTC time stamps on the web server. When writing to such locations, the **DataType** property should be set to **DateTime**. This **DataType** value may be set for either the **TimeSeries** object or the individual **TimedValue** objects.



Note:

When used for entry to a PI point, the UOM for the data to be inserted must be in units that match the PI point; no UOM conversions are performed. For example, if a PI point uses gallons per minute, the UOM for the inserted data must also use gallons per minute.

ListPathsByUpdateTicket method

ListPathsByUpdateTicket retrieves the **path** associated with a specified update ticket so that a user can see which PI **paths** were signed up for data updates for the given update ticket.

Syntax

```
string[] ListPathsByUpdateTicket(Guid updateTicket)
```

Arguments

- **updateTicket**
A unique identifier that denotes a collection of **paths** that is currently signed up for updates.

Errors

If the update ticket is not found in the list of updates, a SOAP fault is returned with the message:

```
The update ticket was not found in the list of tickets registered for updates.
```

Notes

All paths are returned in the format in which they were originally submitted. The update ticket provided must reference an active (that is, unexpired) sign-up for data updates.

SignUpForPIUpdates method

SignUpForPIUpdates signs up for data updates for one or more paths without retrieving an initial set of data. An update ticket is used to represent that entity.

Syntax

```
SignUpResults SignUpForPIUpdates(string[] paths, ushort expiration)
```

See [SignUpResults](#).

Arguments

- **paths**
An array of strings that denotes a path to the items for which updates are desired that identifies a supported PI System data sources; it can be a [PI Server path](#), a [PI AF element attribute path](#), or a [PI event frame path](#).
- **expiration**
Duration, in minutes, of the sign-up before it expires and is removed from the list of sign-ups.



Note: The minimum expiration setting is 5 minutes.

If expiration is 0, the expiration value is taken from the value of the UpdatePurgeInterval property in the Web service's `web.config` file. If that setting does not appear, expiration is set to 5 minutes. Any use of a given ticket in [GetPIUpdates method](#) or [ListPathsByUpdateTicket method](#) resets the expiration timer.

Returns

- **SignUpResults**
If the sign-up was successful, the ErrorCount property of the object is 0 and the updateTicket property of that object is a valid GUID that can be passed without change into the [GetPIUpdates method](#). If all paths in the specified array fail sign-up, the updateTicket value is 00000000-0000-0000-0000-000000000000 .

Errors

If at least one path can be registered for updates but one or more paths fail, a valid update ticket is returned, but PI Web Services returns an Error of 4000 and the description: Updates are not supported for this type of path: [path]. [PI Data Services](#) may also report errors that range from 1000 to 3099, PI SDK reports errors with a negative **Status** value, such as `point not found`.

IPITimeSeries classes

PIArcDataRequest

Represents a query to a PI System for Compressed, Interpolated or Plot Values from a single [path](#), subject to a single [TimeRange](#) that uses a single [manner](#). Since an array of [PIArcDataRequest](#) objects can be passed in a single call to [GetPIArchiveData method](#), any number of independent data requests can be placed immediately.



Note: PI Web Services does not support using [PIManner](#) directly. Any implementation that uses [PIManner](#) must use either [PIArcManner](#) or [PISummaryManner](#).

Applies to

[GetPIArchiveData method](#)

Members

- **Path**
An array of strings that denotes a path to the items for which updates are desired that identifies a supported PI System data sources; it can be a [PI Server path](#), a [PI Performance Equation path](#), a [PI AF element attribute path](#), or a [PI event frame path](#).

- [PIArcManner](#)
[PIArcManner](#)
- [TimeRange](#)
[TimeRange](#)

PIArcManner

This class contains properties needed to define retrieval behavior for PI archive data. Use it to specify Compressed, Interpolated or Plot Values retrieval, boundary handling, and data limits.

- Member of
[PIArcDataRequest](#)
- Applies to
[GetPIArchiveData method](#)

Members

PIArcManner contains these properties and parameters:

- **RetrievalType**
Use a value from the supplied `PIArcMannerRetrievalType` enumeration that defines the type of Archive data retrieved. You can use these enumeration values:
 - **Compressed (default)**
The values that are recorded in the PI Server.
 - **Interpolated**
Generates interpolated values at intervals equal to the time range duration divided by the **NumValues** property.
 - **Plot Values**
Generates an array of values suitable for trending. This retrieval type takes into account the number of pixels on a display and generates the most significant values for each.

For example, `arcmnr.RetrievalType = PIArcMannerRetrievalType.Compressed`

- **Boundaries**
Use a value from the supplied `PIArcMannerBoundaries` enumeration that defines how Compressed events are retrieved, relative to the time range boundaries. You can specify the boundaries as:
 - **Inside**
Returns only events that fall within the time range or exactly at the boundaries.

- Outside
Returns events within the time range plus the first value before the first value and after the time range.
- Interpolated
Returns events within the time range plus interpolated values at the start and end of the time range.

For example, `arcmnr.Boundaries = PIArcMannerBoundaries.Outside;`

- **NumValues**

Use a **NumValues** integer value to specify how values are retrieved from the PI Server:

- For Compressed values, **NumValues** sets the maximum number of values to return.
- For interpolated values, **NumValues** sets the number of interpolation intervals into which the time range is evenly divided.
- For Plot Values, **NumValues** sets the number of screen pixels in the representation of a trend.
- When retrieving Plot Values, no truncation is performed and the number of timed values returned can be up to five times the value of **NumValues**. The minimum value is **1**. The default value is 400.

- **TimeStep**

Defines the length of the calculation interval for the interpolated values. The string must consist of an integer followed by one of the PI time units:

`s|S|m|M|d|D|w|W|mo|MO|Mo|y|Y`

Any string not in this format will be ignored. When **TimeStep** is specified, it will be used in preference to **NumValues** for establishing the interpolation interval.

- **Updates**

Use the **Updates** property to designate a path to sign up for data updates:

- Set to **TRUE** to enable data updates.
- Default is **FALSE**.

Data updates are supported for `pi` paths and `af` paths that specify a simple PI AF data reference.

- **Filter**

A string expression in performance equation (PE) syntax used to filter values returned for PI points in Compressed mode, for example `'sinusoid' > 25`.



Note:

Filters are only supported for PI point retrieval in Compressed or Interpolated mode, or PI AF data references in Compressed mode. The PI point name in the filter expression must be the point's tag name. Do not use the name of the PI AF data reference attribute.

Example

```
PIArcManner myPIArcManner = new PIArcManner();
myPIArcManner.RetrievalType= PIArcMannerRetrievalType.Compressed;
myPIArcManner.Boundaries= PIArcMannerBoundaries.Outside;
myPIArcManner.NumValues = 100;
```

PISummaryDataRequest

Represents a query for summary data from a single [paths](#) for a given time range subject to certain parameters. Since an array of PISummaryDataRequest objects can be passed in a single call to the [GetPISummaryData method](#), any number of independent summary data requests can be placed at once.



Note: PI Web Services does not support using PIManner directly. Any implementation that uses PIManner must use either PIArcManner or PISummaryManner.

- Applies to [GetPISummaryData method](#)
- Members
- Path
A path to one of these supported PI System data sources: a [PI Server path](#), a [PI AF element attribute path](#), or a [PI event frame path](#).
- [TimeRange](#)
- [PISummaryManner](#)

PISummaryManner

- Member of [PISummaryDataRequest](#)
- Applies to [GetPISummaryData method](#)

Members

- [PISummaryMannerValue](#)
Use these enumerations to specify the type of a summary calculation:
 - Average (default)
 - Count
 - Minimum
 - Maximum
 - PStdDev (Population Standard Deviation)
 - Range
 - StdDev
 - Total
- [PISummaryMannerWeightType](#)
Use this enumeration to set the calculation basis for summary calculations:

- **TimeWeighted** - (default) Weight the values in the calculation by the time over which they apply
- **EventWeighted** - Evaluate values with equal weighting for each event

- **UseStart**

Use to indicate whether the Start time of each interval is reported as the time stamp for the interval's summary value:

- If **TRUE**, the time of the start of the time interval over which the summary is calculated is reported as the time for the resulting [TimedValue](#) object.
- If **FALSE**, the end time of the interval is reported.

When updates are later retrieved using the update ticket, the updates represent new or changed values to the recorded values. They do not represent changes to the summary.

- **Updates**

Use to enable updates:

- If set to **TRUE**, updates for pi paths and some af paths that specify a PI AF data reference. Paths that specify extended PI AF data references are not supported.
- By default updates is disabled, or **FALSE**.

- **Filter**

This property is not supported.

- **TimeStep**

Defines the length of the calculation interval for the interpolated values. The string must consist of an integer followed by one of the PI time units:

s | S | m | M | d | D | w | W | mo | MO | Mo | y | Y

Any string not in this format will be ignored. When TimeStep is specified, it will be used in preference to NumValues for establishing the interpolation interval.

Example

```
mySummaryManner.SummaryManner =
PISummaryMannerSummaryValue.Maximum;
```

TimeRange

Represents the start and end times of a time range.

This class models the traditional PI time range construct. The start and end times can be represented as either [PI reference times](#), or as PI or ISO 8601 [fixed times](#).

- Applies to

[GetPIArchiveData method](#)

[GetPISummaryData method](#)

Members

- **StartTime**
Use a string to represent the start time of a time range.
- **EndTime**
Use a string to represent the end time of a time range.

If both **StartTime** and **EndTime** are [fixed times](#), both must have the same offset. For example, **Start = 2010-11-16T08:0:00Z** and **End = 2010-11-16T10:00:00Z** or **Start = 2010-11-16T03:00:00-05:00** and **End = 2010-11-16T05:00:00-05:00**. If either **StartTime** or **EndTime** is a [reference times](#), the local time zone of the web server hosting PI Web Services is used for both **StartTime** and **EndTime**.

- **IsStartAbsolute**
Use to return a boolean value:
 - Set to **TRUE** if the respective property is an [fixed times](#).
 - Set to **FALSE** if it is a [reference times](#).
- **IsEndAbsolute**
Use to return a boolean value:
 - Set to **TRUE** if the respective property is an [fixed times](#).
 - Set to **FALSE** if it is a [reference times](#).

Transition To or From Daylight Savings Time

ISO 8601 time strings provide a Web-standard representation of a [PI time expression](#). As used in PI Web Services, ISO 8601 absolute time strings include the date in **YYYY-MM-DD** format, followed by the character **T**, and then by the time in **HH:MM:SS** format. Times can be either UTC or use a specific time zone.

UTC times are denoted by the character **Z** at the end of the time. Thus, **2010-11-22T08:00:00Z** is **22 November 2010** at **08:00:00 UTC**. To specify a time zone other than UTC, an offset from UTC must be passed in the form **[+]-JHH:MM**. If the time zone in question is eight hours earlier than UTC, as with the Pacific time zone in the United States time zone during standard time, the offset **-08:00** is appended to the end of the time string: **2010-11-22T00:00:00-08:00** represents the same data and time as **2010-11-22T08:00:00Z**.

A problem arises during the transition to or from daylight savings time (DST). A time zone is the combination of the UTC offset and the rules that specify which offset applies and when it changes. ISO 8601 has no mechanism for communicating the time zone, only the offset at the moment in question. In transitioning from DST to standard time in the Pacific time zone, the offset changes from seven hours to eight. The data layer on which PI Web Services relies, however, requires that a single time zone be used for any given query. PI Web Services selects a time zone with the same offset, which might not match that of the time zone where the user is located. Furthermore, PI Web Services cannot fix times passed because it does not know which time zone rules to apply, and the user cannot specify two offsets in a single query.

OSIsoft recommends that times be supplied to PI Web Services in UTC. Queries that use UTC yield correct results even if queries are submitted during the hour in which DST changes to, or from, standard time.

TimeSeries

- Applies to
 - [GetPIArchiveData method](#)
 - [GetPISummaryData method](#)

This class represents a time series of timed values. It consists of a set of properties that pertain to the time series and an array of [TimedValue](#) objects that contain the actual time stamped data.

Some properties in the TimeSeries class pertain to the entire time series. The array of [TimedValue](#) objects is the most significant member of this class; it represents actual time-stamped data values.

Some members within TimeSeries and [TimedValue](#) objects have identical names and meanings. This allows for efficient data transfer, and also supports overriding TimeSeries properties for individual timed values.

Properties

These properties reflect the data type that best matches the underlying PI System data type. The data received by PI Web Services hides the difference between sizes of types. Details about how data types are presented to the Web service are described here.



Note: When using [InsertPIData method](#), you do not need to include the data type.

- Path
Use a string value to identify the name of a supported PI System data source such as a PI point, PI Performance Equation, PI AF element attribute, or PI event frame. Every element of the [TimedValue](#) array normally comes from a [PI Server path](#), a [PI Performance path](#), a [PI AF element attribute path](#), or a [PI event frame path](#). [TimedValue](#) does have its own Path member which is normally blank but can be used to override this path. This can be done by client software when calling [InsertPIData](#) where it is possible to send data values to many PI paths in a single [TimedValue](#) array.
- UOM
Use a string value to indicate the unit of measure (UOM) of all [TimedValue](#) objects. The [TimedValue](#) class also has a UOM field which can be used to override this value.



Note:

When used with [InsertPIData method](#), the UOM for the data to be inserted must match the UOM of the data into which the insert takes place. For example, if a PI point uses gallons per minute, the UOM for the inserted data must also use gallons per minute.

- Error
A non-zero integer indicates an Error in a TimeSeries retrieval. Applications should use this property to test for the success or failure of an operation.
- ErrDesc
This optional property is an error description for a failure that involves the TimeSeries object. If the operation is successful, ErrDesc is null.

- **DataType**

Use these XSD standard names of the data type of the values in the [TimedValue](#) array:

This list summarizes how common data types are represented as a value of the `DataType` property, as performed by the Web service:

- Boolean
Maps to a boolean value in the `DataType` property
- Decimal
Maps to a decimal value in the `DataType` property
- single, double precision floating point
Map to a double value in the `DataType` property
- int16, int32
Map to an integer value in the `DataType` property
- time, date time
Maps to a `DateTime` value in the `DataType` property
- BLOB
Maps to a `Byte[]` value in the `DataType` property



Note:

BLOB data is not currently available through PI Web Services. BLOB data is represented as a data type of byte. The `Value` property of a [TimedValue](#) object whose `DataType` value is `Byte` will be `null`.

- PI Digital, string
Map to a string value in the `DataType` property



Note: Digital tag data values are reported using the string representation of the digital state, not the numeric code.

The [TimedValue](#) class also has a `DataType` field which can be used to override this value.

- **SeriesID**

Use a string to represent a GUID that identifies the original query. The string includes both a path and manner.

- **TimedValues**

An array of time-stamped data values:

- `TimedValue[]`

These properties reflect the data type that best matches the underlying PI System data type. The data received by the PI Web Services hides the difference between different sizes of types. For example, single and double precision floating point types are presented to the Web service as double precision.

Notes

Some properties in the `TimeSeries` pertain to the entire time series. The array of `TimedValue` objects is the most significant member of this class; it represents actual time-stamped data values.

Some members within `TimeSeries` and `TimedValue` have identical names and meanings. This allows for efficient data transfer, and also supports overriding `TimeSeries` properties for individual timed values.

TimeSeriesUpdates

This object is derived from `TimeSeries`. In addition to the properties of that object, `TimeSeriesUpdates` adds the property `Updates`, which is an array of `TimedValueUpdate` objects.

- Applies to `GetPIUpdates` method

TimedValueUpdate

This object is derived from `TimedValue`. In addition to the properties of that object, `TimedValueUpdate` adds an enumerated property `UpdateType`. The enumeration, also named `UpdateType`, permits the values **Archive**, **Snapshot**, **Delete**, **Edit**, and **AddNoReplace**.

Instances of this object are returned by `GetPIUpdates` method.

When you call `GetPIUpdates`, the `TimedValues` property is null and `TimeSeriesUpdates` is an instantiated object. `TimedValue` is used for the regular data methods such as `GetPIArchiveData` method; `TimeSeriesUpdates` is only used for `GetPIUpdates` method.

TimedValue

This class represents a single time-stamped data value retrieved from or sent to the PI System. The `TimeSeries` class contains an array of `TimedValue` objects.

Properties

Name	Data Type	Description
Path	String	(Optional) One of the paths to a supported PI System data source: a PI Server path , a PI AF element attribute path , or a PI event frame path . <code>TimeSeries</code> object is returned from a Web method call, Path is reflected in the <code>TimeSeries</code> class for every element of the <code>TimedValue</code> array; This field is used primarily in <code>InsertPIData</code> method; where it is possible to send data values to many paths in a single <code>TimedValue</code> array
Time	XSD dateTime, .NET DateTime	Time stamp of the value in UTC
UOM	String	(Optional) Unit of measure of the value; This is usually blank because the UOM is usually the same for the entire <code>TimeSeries</code> and is specified there; If this field is not blank, the UOM overrides this value only

Name	Data Type	Description
Flags	String	(Optional) Denotes whether the value is Questionable (Q), Substituted (S), or Annotated (A) in the PI Server; Possible values are any combination of Q , S , and A
Status	String	(Optional) If a value is non-zero, this property contains a Status string indicating what is wrong with the value. When entering data through InsertPIData method , Status should be populated with the digital state value being inserted. If Status is populated, the value is ignored.
PctGood	Double	(Optional) This property is only used when summary values are retrieved through GetPISummaryData method . In that operation, the percent of data in the calculation interval with good values is reported if it is less than 100 percent; intervals with 100 percent good data do not report this value; that way, response message size remains minimal.
DataType	String	(Optional) XSD standard name of the value's data type; This is usually blank because the data type is usually the same for the entire TimeSeries and is specified there; If this field is not blank, the data type overrides this value only
Value	String	Data value

- Applies to [TimeSeries](#)

SignUpResults

Class used to communicate the results of explicit registration for data updates through the [SignUpForPIUpdates method](#).

- Applies to [SignUpForPIUpdates method](#)

Properties

- **UpdateTicket**
A unique identifier that denotes a collection of paths that is currently signed up for updates. Paths which are successfully signed up for updates return the value of path as a valid GUID. If all sign-ups fail, the path value is an empty GUID: 00000000-0000-0000-0000-000000000000.
- **updateTicket**
Guid. Paths which are successfully signed up for updates return the value of path as a valid GUID. If all sign-ups fail, the path value is an empty GUID: 00000000-0000-0000-0000-000000000000.

- Errors

`int[]`. Array of error codes. When a sign-up is successful, the corresponding value is 0.

- ErrDescs

`string[]`. When a sign-up is successful, the corresponding value is an empty string. When the error entry is non-zero, the value in ErrDescs describes the error.

- ErrorCount

Integer count of the number of errors. That is, the number of entries in Errors with a non-zero value. If ErrorCount is greater than 0, the client should iterate through the array and determine which path(s) failed sign-up.

Notes

Errors, and ErrDescs will always have the same number of entries. This number will match the number of paths passed into [SignUpForPIUpdates method](#), and the order of entries will match. For any entry `i`, `Errors[i]`, and `ErrDescs[i]` will completely record the results of the sign-up for `paths[i]`. If ErrorCount is 0, clients may pass the updateTicket into [GetPIUpdates method](#) to retrieve data updates.

If ErrorCount is greater than zero, the client should iterate through the array and determine which path(s) failed sign-up. Empty GUID values for updateTicket `00000000-0000-0000-0000-000000000000` result from unsuccessful sign-ups for all paths.

IPISearch interface

The IPISearch interface contains methods that search PI Systems and collections of [paths](#) or path-related metadata. Returned paths may then be used by methods of the IPITimeSeries interface without change. The returned paths and metadata are intended to be used to locate items of interest for use with other interfaces. For example, a client application can use the FindPIPathsBasic method to locate PI points for data that is subsequently retrieved with an IPITimeSeries method, such as GetPIArchiveData.

IPISearch web methods

FindPIPathsBasic method

This method performs a simple tag search, similar to that used in the **Basic Search** tab of the **PI SDK Tag Search** and the PI SDK **IGetPoints2.GetPoints** method. It returns the tags for PI points that meet the specified criteria as an array of [paths](#), in a syntax suitable for use with methods from the [IPISoap](#) interface. Prior to PI Web Services 2012, this method returned paths for use with the [IPITimeSeries interface](#), which is now deprecated.

Syntax

```
string[] FindPIPathsBasic(string server, string tagMask, string pointtype, string pointsource, string classname, string descriptor, string UOM, int numValues)
```

The search returns an array of [paths](#) in a syntax suitable for use with the with methods from the IPITimeSeries interface.

Arguments

- **server**
String that denotes the full name of the PI Server to search without leading slashes. This search is not valid for PI Asset Framework (PI AF) servers. Wildcard characters are not permitted.
- **tagMask**
String that contains the PI tag mask, including the optional wildcard characters * and ?. For example, a PI point named **FIC*.PV**.
- **pointtype**
String that consists of one of the supported PI Web Services, PI tag types, or the wildcard character *.
- **PI Types**
 - Float16
 - Float32
 - Float64
 - Digital
 - Int16
 - Int32
 - Timestamp
 - Blob
 - String
- **Web Service Types**

Type	Maps to PI Type
Decimal	Int32
Double	Any Float
Float	Any Float
Int	Any Int
DateTime	Timestamp
Byte[]	Blob

- **pointsource**
Point source, including optional wildcards.
- **classname**
Point class, including optional wildcards. Examples include Base and Classic.
- **descriptor**
PI point description, including optional wildcards.

- UOM

Units of measure, that is, PI point engineering units, including optional wildcards.

- numValues

Maximum number of [paths](#) to return. Default is **100**. If the number of paths matching the search criteria exceeds numValues, the web service returns the first numValues paths returned from the PI Server.

Returns

A string array that contain the tags for PI points that meet the specified criteria, in a format that is suitable for use as [paths](#) in methods of the [IPITimeSeries interface](#). For example, the tag for a PI point named **sinusoid** on a PI Server named **piserver01**, is returned as **pi:\piserver01\sinusoid**.

Notes

If tagMask, pointtype, pointsource, classname, descriptor or UOM are passed as empty or null strings, it means that no particular value of these parameters is required. It is as if a wildcard character * is passed.

The pointtype parameter is examined without regard to case.

This method does not include search by Value or Status.

The paths returned are suitable for use with any of the methods in [IPITimeSeries interface](#). However, the retrieval of BLOB type data for PI tags is not supported.

IPISearch classes

PIPointType

Enumeration denoting a PI point type:

- Float16
- Float32
- Float64
- Int16
- Int32
- Digital
- String
- Timestamp
- Any - represents wildcard *

IPISoap interface

The IPISoap interface retrieves PI Asset Framework data references and current values of attributes for PI event frames.

IPISoap web methods

FindAttributePaths

This method searches the child PI Asset Framework (PI AF) attributes of an AF element or AF attribute, and returns the paths of the attributes matching the search constraints. FindAttributePaths accepts search criteria and returns an array of AF attribute paths.

Syntax

```
String[] FindAttributePaths(AttributeSearchConstraints Constraints,  
AttributeSearchManner Manner)
```

See [AttributeSearchConstraints](#) and [AttributeSearchManner](#).

Arguments

- **Constraints**
Search constraints including information about the search root path, search level, name mask, and so on. See [AttributeSearchConstraints](#) for more details.
- **Manner**
Search manner including information about how the search is conducted. See [AttributeSearchManner](#) for more details.

Returns

Array of the paths of the AF attributes matching the search constraints.

Errors

Errors in validation or processing will result in the return of a SOAP fault:

Code	Message
WA051	AF Path could not be resolved
WA052	AF Path could not be used to build type to an AF (Element Attribute)
WA054	Path cannot be null, empty or white space string

FindElements method

This method accepts search criteria as defined in constraints and returns an array of Elements containing the information specified in the manner.

Syntax

```
Element [] FindElements(ElementSearchConstraints constraints, ElementSearchManner  
manner)
```

Errors

Errors in validation or processing will result in the return of a SOAP fault:

- WA004
Child Element depth must be an integer ≥ 0 . See
ElementSearchManner .ChildElementDepth
- WA051
Root Path could not be resolved.
- WA052
Root Path did not resolve to an AF server, database, or element. See
ElementSearchConstraints.RootPath.
- WA053
The supplied template name does not exist. See ElementSearchConstraints.TemplateNames
- WA055
The supplied category name does not exist. See ElementSearchConstraints.CategoryNames
- WC101
Multiple datatypes were provided for Attribute value when only one is allowed. See
ElementSearchConstraints. AttributeFilters . Value
- WC102
Value(s) provided for attribute filter when Operation does not permit any value. See
ElementSearchConstraints. AttributeFilters . Operator
- WC103
Value(s) provided for attribute filter when Operation requires a Value argument.
ElementSearchConstraints. AttributeFilters . Operator
- WC104
Multiple values provided attribute filter when Operation permits at most one
value. See ElementSearchConstraints. AttributeFilters . Operator
- WC105
Multiple values provided attribute filter when Operation permits at most one value.
Comparison value provided in AttributeFilter without specifying an Operator.
- WC106
TemplateAttributeOnly was specified in AttributeFilter, but no comparison value was
provided.
- WC107
A comparison operator was specified in AttributeFilter but no Attribute Name was
provided.

FindElementTemplates method

Syntax

```
string[] FindElementTemplates(string Server, string Database, string NameMask, bool IncludeDerivedTemplates)
```

Arguments

- **Server**
A string value that is the name of the AF server to which PI Web Services will connect.
- **Database**
A string value that is the name of the PI AF database to which PI Web Services will connect.
- **NameMask**
A string that contains optional wildcards to specify which returns are included in the results list:
 - *****
A wildcard that matches 0 or more characters.
 - **?**
A wildcard that matches exactly 1 character.
- **IncludeDerivedTemplates**
A boolean indicator that specifies whether templates that inherit from matched results are included in the results list.

Returns

An array of the PI AF element template names that match the search criteria.

Errors

Errors in validation or processing will result in the return of a SOAP fault:

Code	Message
WA001	No server name specified
WA002	No database name specified

Use of wildcard to search for PI AF template names

template NameMask	Results
foo*	foo, foobar
ba	foobar, bar, baz
ba?	bar, baz
foo?	<none>

FindEventFrames method

This method accepts search criteria and returns an array of [EventFrame](#) objects. It performs a search from the specified root within the event frames subtree on a specified server and database based on the criteria specified. An array of PI event frames matching the criteria returned.

Syntax

```
EventFrame [ ] FindEventFrames ( EventFrameSearchConstraints Constraints, EventFrameSearchManner )
```

See [EventFrame](#).

Constraints

Use an [EventFrameSearchConstraints](#) object that provides search criteria.

Manner

Use an [EventFrameSearchManner](#) object that specifies parameters to control how search results are returned.

Returns

An array of PI event frames that match the search criteria.

Errors

Errors in validation or processing will result in the return of a SOAP fault:

Code	Message
4003	Invalid event frame path syntax
2501	Root Path Event Frame not found
2502	Root Path badly formed
2503	Root Path had bad PISystem name
2504	Root Path had bad database name
2505	Root Path no privilege to read database
2506	Root Path point to more than one Event Frame
2507	Category name not found
2508	Template name not found
2509	No Privilege to read Database
2510	bad attribute template name
2511	must have at least one attribute
2512	Bad Element Template Name

Notes

The root path parameter specifies an event frame at the top level of the event frame hierarchy. It may be named or specify a GUID identifier.

Examples

- ef:\\afserver\db\EventFrame[NamedEF]
- ef:\\afserver\db\EventFrames[{12345678-1234-1234-1234-123456789012}]
- ef:\\afserver\afdb\

FindEventFramePaths method

FindPIEventFramePaths searches within the event frames subtree from a root subtrees from a root server and database that is specified. The search is based on specified criteria. An array of [paths](#) to event frames matching the criteria are returned.

Syntax

```
string[] FindEventFramePaths( EventFrameSearchConstraints Constraints,  
EventFrameSearchManner Manner)
```

Constraints

Use an [EventFrameSearchConstraints](#) object that provides the search criteria.

Manner

Use an [EventFrameSearchManner](#) object to specify parameters that control how search results are returned.

Returns

An array of paths to PI event frames. The array will contain up to the maximum number of paths indicated by the *Manner.MaxCount*, and the returned paths will use either:

- a single unique ID for the matching event frame and be set as `Manner.UseName = false`.
- a name-based path to the matching event frame and be set as `Manner.UseName = true`.

Descendant event frames may be returned based on the values of `Manner.FromLevel` and `Manner.ToLevel`. If `FromLevel` and `ToLevel` are both 0, only event frames matching the search criteria will be returned.

Errors

Errors in validation or processing will result in the return of a SOAP fault:

Code	Message
4003	Invalid event frame path syntax
2501	Root Path Event Frame not found
2502	Root Path badly formed
2503	Root Path had bad PISystem name
2504	Root Path had bad database name
2505	Root Path no privilege to read database
2506	Root Path point to more than one Event Frame
2507	Category name not found
2508	Template name not found
2509	No Privilege to read Database
2510	bad attribute template name
2511	must have at least one attribute
2512	Bad Element Template Name

FindPIPathsBasic method

This method performs a simple tag search, similar to that used in the **Basic Search** tab of the **PI SDK Tag Search** and the PI SDK **IGetPoints2.GetPoints** method. It returns the tags for PI points that meet the specified criteria as an array of [paths](#), in a syntax suitable for use with methods from the [IPISoap](#) interface. Prior to PI Web Services 2012, this method returned paths for use with the [IPITimeSeries interface](#), which is now deprecated.

Syntax

```
string[] FindPIPathsBasic(string server, string tagMask, string pointtype, string pointsource, string classname, string descriptor, string UOM, int numValues)
```

The search returns an array of [paths](#) in a syntax suitable for use with the with methods from the [IPITimeSeries interface](#).

Arguments

- **server**
String that denotes the full name of the PI Server to search without leading slashes. This search is not valid for PI Asset Framework (PI AF) servers. Wildcard characters are not permitted.
- **tagMask**
String that contains the PI tag mask, including the optional wildcard characters * and ?. For example, a PI point named **FIC*.PV**.
- **pointtype**
String that consists of one of the supported PI Web Services, PI tag types, or the wildcard character *.
- **PI Types**
 - Float16
 - Float32
 - Float64
 - Digital
 - Int16
 - Int32
 - Timestamp
 - Blob
 - String
- **Web Service Types**

Type	Maps to PI Type
Decimal	Int32
Double	Any Float
Float	Any Float

Type	Maps to PI Type
Int	Any Int
DateTime	Timestamp
Byte[]	Blob

- **pointsource**
Point source, including optional wildcards.
- **classname**
Point class, including optional wildcards. Examples include Base and Classic.
- **descriptor**
PI point description, including optional wildcards.
- **UOM**
Units of measure, that is, PI point engineering units, including optional wildcards.
- **numValues**
Maximum number of [paths](#) to return. Default is **100**. If the number of paths matching the search criteria exceeds numValues, the web service returns the first numValues paths returned from the PI Server.

Returns

A string array that contain the tags for PI points that meet the specified criteria, in a format that is suitable for use as [paths](#) in methods of the [IPITimeSeries interface](#). For example, the tag for a PI point named **sinusoid** on a PI Server named **piserver01**, is returned as **pi:\\piserver01\sinusoid**.

Notes

If tagMask, pointtype, pointsource, classname, descriptor or UOM are passed as empty or null strings, it means that no particular value of these parameters is required. It is as if a wildcard character * is passed.

The pointtype parameter is examined without regard to case.

This method does not include search by Value or Status.

The paths returned are suitable for use with any of the methods in [IPITimeSeries interface](#). However, the retrieval of BLOB type data for PI tags is not supported.

GetElementsByPaths

This method accepts a list of paths and a field selection, and returns an array of Elements.

Syntax

```
Element[] GetElementsByPaths(String[] Paths, ElementSearchManner Manner)
```

Arguments

- **Paths**

A `String []` that contains a list of paths to the PI AF elements that should be returned.

- **Selection**

An optional `ElementSearchFieldSelection` specification that indicates which fields of match elements are included in the results.

Returns

An array of `Elements` that match the provided paths.

Errors

Errors in validation or processing will result in the return of these SOAP faults:

Code	Message
WA003	An AF element path must be specified
2501	Path could not be resolved
2502	Path did not resolve to an AF element

GetEventFrameTimeSeries method

This method returns an array of `EventFrame` objects that includes time series data for object attributes, with data references of `PIPoint`. The time series data's start time and end time will be the parent event frames' start time and end time. If the event frame is still running, that is, its end time is null or empty, the time series data's end time will be a time stamp of Current when the method is executed.

Syntax

```
EventFrame[] GetEventFrameTimeSeries(String[] Paths)
```

See `EventFrame`.

Arguments

- Array of event frame paths.

Returns

Array of `EventFrame` classes.

Errors

Condition	Error Property Value
Array of event frames paths cannot be null or empty End of added content	WA301
Failed to generate data queries to retrieve event frames. Ensure that all paths are in correct format: <code>ef: \\AFServerName\AFDatabaseName\EventFrames[{GUID}]</code>	WA302

Notes

The event frame path must use the format as shown here: `ef: \\server\database\EventFrames[{GUID}]`.

Examples

- `ef:\\afserver1\TestDatabase\EventFrames[{A231E534-31B2-493E-AA38-1A78F7FE8E96}]`

GetEventFrameValues method

GetPIEventFrameValues method retrieves current values for the attributes of the specified event frame.

Syntax

```
EventFrame [] GetEventFrameValues(string[] path)
```

See [EventFrameValues](#).

Argument

paths

An array of paths to the desired event frames.

Returns

An array of [EventFrame](#) objects corresponding to the array of paths passed into the method. Each object contains data about the event frame, paths to parent event frames and referenced elements, and a list of current attribute values.

Errors

Condition	Error Property Value	ErrDesc Property Value
Invalid event frame paths	4003	Malformed path: <i>path_string</i> in the returned EventFrameValues object

Notes

Event frames in the paths must be specified by GUID, to uniquely match an event frame. PI Asset Framework does not enforce a unique name for event frames.





IPISoap classes

AttributeFilter

This class specifies tests to determine whether the attribute of a PI AF element or PI event frame should be matched. In general, OSIsoft recommends that attributes be indexed if those attributes are intended to the subject of an [AttributeFilter](#).

- Applies to [EventFrameSearchConstraints](#)

Members

- **Name**
A string that is the name of the attribute to compare.
- **Operator**
An optional operation for testing the value of the PI AF attribute.
 - **EqualTo**
=
 - **NotEqualTo**
 - **GreaterThanOrEqualTo**
>=
 - **LessThan**
<
 - **LessThanOrEqualTo**
<=
 - **In**
 **Note:** Searches for PI event frames do not support this property.
 - **Not In**
 **Note:** Searches for PI event frames do not support this property.
 - **IsEmpty**
 **Note:** Searches for PI event frames do not support this property.
 - **IsNotEmpty**
 **Note:** Searches for PI event frames do not support this property.
- **IsTemplateAttribute**
A boolean value that specifies whether the attribute is part of a template. If so, performance will be improved.
- **Value**

A list of allowed values relative to the Operator specified. Note that for unary operators `IsEmpty` and `IsNotEmpty`, a runtime error (WC102) will be thrown if values are provided. Note that for list comparison operators `In` and `NotIn`, a runtime error (WC103) will be thrown if no values are provided. Note that for all binary operators, a runtime error (WC103) will be thrown if no value is provided, and a runtime error (WC104) will be thrown if more than one value is provided.

Example


```
Name = "Temperature", Operator = Operators.GreaterThanOrEqualTo, Value = "100" will match any event frame which has an attribute named "Temperature" whose value is >= 100.
```

AttributeSearchConstraints

This class specifies the constraints of a [FindAttributePaths](#) search.

- Applies to [FindAttributePaths](#)

Members

- **RootPath**
A string for a required root path of the AF element or attribute, which is the search root. Any valid AF format for AF element or attribute is allowed.
- **NameMask**
An optional string that uses wildcards to determine if the name of an AF attribute or element will be included in the results. When this member is null or empty string, the name mask search criterium will be ignored.
- **MaxElementSearchLevel**
An optional `Nullable<UInt16>` that sets the maximum search level for child elements from the search root. Value 0 means that only the child attributes (and the grandchildren elements) of the root will be searched. If this member is not specified (that is, is equal to null), the maximum element search level criterium will be ignored.
 **Note:** This member only takes effect when the search root is an AF element, because an AF attribute cannot have an AF element as a child.
- **MaxAttributeSearchLevel**
An optional `Nullable<UInt16>` that sets the maximum search level for child attributes from any level's elements. Value 0 means that only the direct attributes of an AF element will be searched. If this member is not specified (i.e. equals to null), the maximum attribute search level criterium will be ignored.
- **AllowedAttributeCategoryNames**
An optional string [] of the array of the names of allowed attribute categories. An AF attribute will be returned as matched only if its category presents in this array when this array is not empty. When the array is empty, the category search criterium will be ignored.

- Wildcards
 - *
 - A wildcard that matches 0 or more characters
 - ?
 - A wildcard that matches exactly 1 character

Use of wildcards to search for PI AF elements or attributes

NameMask	Results
foo*	foo, foobar
ba	foobar, bar, baz
ba?	bar, baz
foo?	<none>

AttributeSearchManner

This class specifies the search manner of an [FindAttributePaths](#) search. Currently, this class is empty but reserved for future requirements.

- Applies to [FindAttributePaths](#)

AttributeValue

- Applies to [Attribute Filter](#)

Use this class to specify data that pertains to the current value of a PI AF attribute value. Only one of the five properties may be populated.

Members

- **StringValue**
A string value. The supplied value will be compared to the actual current value of the attribute with respect to the specified operator. To a Boolean attribute, you should use a 1 for true, and 0 for false.
- **GuidValue**
A globally unique identifier. The supplied value will be compared to the actual current value of the attribute with respect to the specified operator. To a Boolean attribute, you should use a 1 for true, and 0 for false.
- **LongValue**

A numeric value, expressed in signed 64-bit integral format. The supplied value will be compared to the actual current value of the attribute with respect to the specified operator. To a Boolean attribute, you should use a 1 for true, and 0 for false.

- **DoubleValue**

A numeric value, expressed in signed 64-bit floating-point decimal format. The supplied value will be compared to the actual current value of the attribute with respect to the specified operator. To a Boolean attribute, you should use a 1 for true, and 0 for false.

- **DateTimeValue**

A time value, expressed in ISO 8601 format. The supplied value will be compared to the actual current value of the attribute with respect to the specified operator. To a Boolean attribute, you should use a 1 for true, and 0 for false.

DurationFilter

This class represents a search criterion which must match the duration of a PI event frame in order to satisfy an event frame search. The duration of an event frame will be compared to the Value property and use the Operation property to control the comparison.

- Applies to
[EventFrameSearchConstraints](#)

Members

- **Operation**
String from Operator to use in comparison.
 - **Operator**
 - **EqualTo**
=
 - **NotEqualTo**
 - **GreaterThanOrEqualTo**
>=
 - **LessThan**
<
 - **LessThanOrEqualTo**
<=

- In



Note: Searches for PI event frames do not support this property.

- Not In



Note: Searches for PI event frames do not support this property.

- IsEmpty



Note: Searches for PI event frames do not support this property.

- IsNotEmpty



Note: Searches for PI event frames do not support this property.

- Value

Value that is compared to the duration of the event frame duration that takes the form {numeric_duration_value}{s|m|h|d|w|mo|y} where:

- numeric_duration_value
a non-negative double value
- s
seconds (case-insensitive)
- m
minutes
- h
hours
- d
days
- w
weeks
- mo
months defined as 30 days
- y

years defined as 365 days

Due to the need to fix the value of months and years, these units of measure for a duration are not recommended.

Example

```
Operation = Operators.GreaterThan, Value = "3600s" will match any event frame whose duration is >= 3600 seconds (i.e., one hour)
```

ElementSearchConstraints

This class specifies the criteria for matching elements to be returned in a search for PI AF elements.

- Applies to
 - [FindElements](#)
 - [FindEventFramePaths method](#)
 - [FindEventFrames method](#)

Members

- **RootPath**
A string for an optional root path for which all results must be descendants. If no path is provided, the default PI AF server and database are used. If no database is provided in the path, the default database of the matched AF server is used.
- **NameMask**
An optional string that uses wildcards to determine if the name of an element will be included in the results.
- **TemplateName**
An optional string [] to determine whether a PI AF element that is used in the element template will be included in the results. If the element's template matches one of the name masks provided, it will be included in the results.



Note: To include elements that match any template derived from the list of names provided, set `AllowMatchOnDerivedTemplates`.

- **AllowMatchOnDerivedTemplates**
A boolean value that determines whether templates derived from the list provided in `TemplateName` should be considered a match.
- **CatetoryNames**
An optional string [] that determines whether a PI AF element is included in results based on the element's category. If the element is in one or more of the categories provided, the element is included in the results.
- **AttributeFilters**

An optional list of tests on the attributes of this element.

- **Wildcards**
 - *
A wildcard that matches 0 or more characters
 - ?
A wildcard that matches exactly 1 character

Use of wildcard to search for PI AF template names

template NameMask	Results
foo*	foo, foobar
ba	foobar, bar, baz
ba?	bar, baz
foo?	<none>

ElementSearchManner

- Applies to
[FindElements](#)

Use this class to specify what should be returned from matched PI AF elements in a search for elements.

Members

- **IncludeTemplateName**
A boolean value to determine whether the name of the element template should be nested within the element.
- **ChildElementDepth**
An integer value for the depth to which to child elements that are retrieved are nested within the parent element.
- **FlatHierarchy**
A boolean value that determines whether instead of nesting child elements, child elements should be retrieved and returned at the same level of the hierarchy as their parents. OSIsoft recommends that this only be set to true in environments that have trouble with nested hierarchies, such as SAP.

EventFrame

- Applies to

[PISoap interface](#)

[PISearch interface](#)

This class models a PI event frame object.

Members

- **Name**
Use a string value that names the PI event frame.
- **Path**
Use a string value that identifies the path to the PI event frame.
- **ID**
Enter a string representation of a GUID
- **Description**
Use a string that describes the PI event frame.
- **ParentEventFrames**
A `string[]` that represents the paths of the parent of the PI event frame.



Note: AF SDK currently has a limit of two parents; PI Web Services does not impose a limit in order to accommodate future growth.

- **Start**
Use `anxsd:dateTime` of the absolute start time of the PI event frame.
- **End**
Use `anxsd:dateTime` of the absolute end time of the PI event frame, or null if a PI event frame is in progress. The default value is null.
- **ReferencedElements**
Enter a `string[]` of paths to referenced elements
- **Attributes**
Use an `EFAAttribute[]` of attributes to the PI event frame.

EventFrameSearchConstraints

- Applies to
[FindEventFramePaths method](#)
[FindEventFrames method](#)

Members

EventFrameSearchConstraints specifies constraint criteria for a PI event frame search:

- **RootPath**
A string that contains the ef format path to locate the event frame tree to be searched.
- **SearchRootLevelOnly**
If false, this boolean value searches the entire event frame tree rooted by the root property in the EventFrameSearchItem. Otherwise, only the immediate children of the root property are examined. By default, SearchRootLevelOnly=False.
- **NameMask**
A string that contains the filter to match event frame names. This follows the PI Asset Framework (PI AF) name matching patterns.
- **Template**
The name of the event frame template that is used to create that event frames used in a search. Wild cards are not permitted.
- **Category**
The optional name of a category to which a PI event frame must belong in order match a search for event frames that are included in the specified category.
- **RefElementNameMask**
A string that contains the filter for matching names of AF elements that are referenced by an event frame. Like NameMask, this argument follows the PI Asset Framework (PI AF) name matching patterns. If this argument is null or empty, referenced elements are not used as search criteria.
- **Start**
A string that contains the start time of the search interval. It can be any time string supported by PI Web Services.
- **End**
A string that contains the end time of the search interval. It can be any time string that is supported by PI Web Services.
- **TimeMode**
A string from an EventFrameSearchTimeMode enumeration that controls how Start and End times are used in the search:
 - **Overlapped**
The PI event frame must fall within the range specified by the Start and End times used in the search but may also extend outside of the specified range. By default, TimeMode=Overlapped.

- None
A synonym for *Overlapped'*
- StartInclusive
A match will occur if the start time of a PI event frame must falls within the range specified by the Start and End times used in the search.
- EndInclusive
A match will occur if the end time of a PI event frame must falls within the range specified by the Start and End times used in the search.
- Inclusive
A match will occur if both the start and end times of a PI event frame falls within the range specified by the Start and End times used in the search.
- DurationFilters
A DurationFilter array that consists of 0 or more duration queries AND'ed together to constrain the search.
- AttributeFilters
An AttributeFilter array that consists of 0 to 5 attribute queries AND'ed together to constrain the search. If attribute queries are specified, a template must be specified.

Notes

PI Asset Framework (PI AF) name matching patterns

Match patterns can include regular characters and wildcard characters. Regular characters must match exactly the characters specified in the query string. Wildcard characters can be matched with arbitrary fragments of the query string. Wildcard characters can be escaped using the single backslash (\) character. Use a double backslash (\\) to match a single backslash. The syntax of the query string has the following rules:

- If null (Nothing in Visual Basic) or empty string, then everything will be matched.
- If no wildcards, then an exact match on the query string is performed.
- Wildcard * can be placed anywhere in the query string and matches zero or more characters.
- Wildcard ? can be placed anywhere in the query string and matches exactly one character.
- One character in a set of characters are matched by placing them within []. For example, **a[bc]** would match 'ab' or 'ac', but it would not match 'ad' or 'abd'.
- One character in a set of characters are not matched by placing them within [!]. For example, **a[!bc]** would match 'ad', but it would not match 'ab', 'ac', or 'abd'.

Rules to create a root path

- The root path parameter specifies the top point in the hierarchy from which to begin the search. The root path may be:
 - Null or empty, in which case the default database on the default PI Asset Framework server that is configured on the PI Web Services server is used
 - An PI AF server, in which case the default database on the specified PI AF server is used

- An PI AF database, in which case the event frame hierarchy in the specified database is used
- An event frame, specified in name- or GUID-format, in which case the specified event frame is used

Examples

- `ef:\\server`
- `ef:\\server\db`
- `ef:\\server\db\EventFrames[{12345678-1234-1234-1234-123456789012}]`
- `ef:\\server\db\EventFrame[SearchRootEF]`
- `ef:\\server\db\EventFrame[ParentEF]\ChildEF\SearchRootEF`

EventFrameSearchManner

- Applies to [FindEventFramePaths method](#) and [FindEventFrames method](#)

Members

- **UserName**
Set to True to use names for all PI event frames in a returned path. Set to False to return a GUID of the event frame that matches the criteria returned in the form `ef:\\server\database\EventFrames[ID]`. The default value is True.
- **ChildFrameDepth**
Use to specify the depth to which child event frames should be returned. A value of 0 displays no children event frames; 1 displays children of the given event frame only, with no grandchildren event frames; 2 displays children event frames and the children of those event frames, and so on. By default, `ChildFrameDepth=0`.
- **ToLevel**
Use to specify the child level at which results stop being returned. If `ToLevel=0`, only the matching event frames are returned. By default, `ToLevel=0`.
- **AttributeDepth**
Use to specify the depth to which attributes should be returned. A value of 0 displays no attributes; 1 displays attributes of the given event frame only, with no children; 2 displays the attributes of the given event frame and the children of those event frames, and so on. By default, `AttributeDepth=0`.
- **AttrReturnEnd**
Use to specify the end generation for returning attributes. By default, `AttrReturnEnd=0`.
- **MaxCount**

Use to indicate the maximum number of paths or PI event frames to return to the client. By default, `MaxCount=100`.

- **SearchAll**
Set to true, to search the entire event frame tree rooted by the `Root` property in the `EventFrameSearchItem`. If `SearchAll=False`, only the immediate children of `Root` are examined. By default, `SearchAll=False`.


EventFrameValues

This class is used to contain attribute value data for an entire event frame. It is the top-level object returned by the [GetEventFrameValues method](#) and it exposes data pertaining to the event frame query as well as an array of [AttributeValue](#) objects.

[Applies to](#)

[GetEventFrameValues method](#)

Members

- **Name**
A string that names the event frame.
- **Path**
A string that identifies the path to the event frame in the format submitted to the retrieval method.
- **ID**
A string representation of a GUID.
 **Note:** This is rendered as string for compatibility with XSD types.
- **Start**
An `xsd:dateTime` that represents the [fixed](#) start time of the event frame in UTC.
- **End**
Absolute end time of the event frame, or null if the event frame is running. If the `End` value is not null, the time will be UTC.
- **Error**
A non-zero integer value that indicates an error occurred as a result of the attempted retrieval of an event frame.
- **ErrDesc**
An optional error description string for a failure to retrieve a event frame. If `Error = 0`, this property will be null.
- **AttributeValues**

An `AttributeValue []` array of the current values for the attributes of the event frame.

Use InfoPath with PI Web Services

To access data through PI Web Services, you must use a compatible Web service client application. You can use any development tool capable of creating code or a user interface from a WSDL file with PI Web Services.

This section demonstrates how you can use Microsoft Office InfoPath 2007 to develop a client form for the IPTimeServices interface. This code-free solution requires configuration only.



Note:

A basicHttpBinding is required when you use InfoPath.

The form created here retrieves time series data with the **GetPIArchiveData** method:

The screenshot shows the Microsoft Office InfoPath 2007 interface. The title bar reads "(Preview) Template1 - Microsoft Office InfoPath". The menu bar includes File, Edit, View, Insert, Format, Tools, Table, and Help. The toolbar contains various icons for document operations. The main content area displays a form titled "OSisoft UC 2010: Retrieve PI Archive Data". The form has several input fields: "Path" (containing "pi:\\philly\\sinusoid"), "Start" (containing "-4H"), "End" (empty), "Retrieval Type" (a dropdown menu set to "compressed"), "Num Values" (a text box containing "400"), "Boundaries" (a dropdown menu set to "inside"), and "Updates" (a checkbox that is unchecked). There is an "Insert item" button and a "Run Query" button. Below the form, there are more input fields: "Path" (empty), "Error" (empty), "Error Description" (empty), "Unit of measure" (empty), and "Datatype" (empty). At the bottom, there is a table with four columns: "Flags", "Time", "Status", and "Value". The "Status" column has a red asterisk. Below the table is another "Insert item" button. The status bar at the bottom shows the form template's location: "Form template's location: C:\Users\smohr\AppData\Local\Microsoft\InfoPath\Designer2\5e1da6cdf3134c4f\manifest.xsf".

Configure access to PI System data

Procedure

1. Open Microsoft InfoPath client.
2. Select **File > Design a Form Template**.

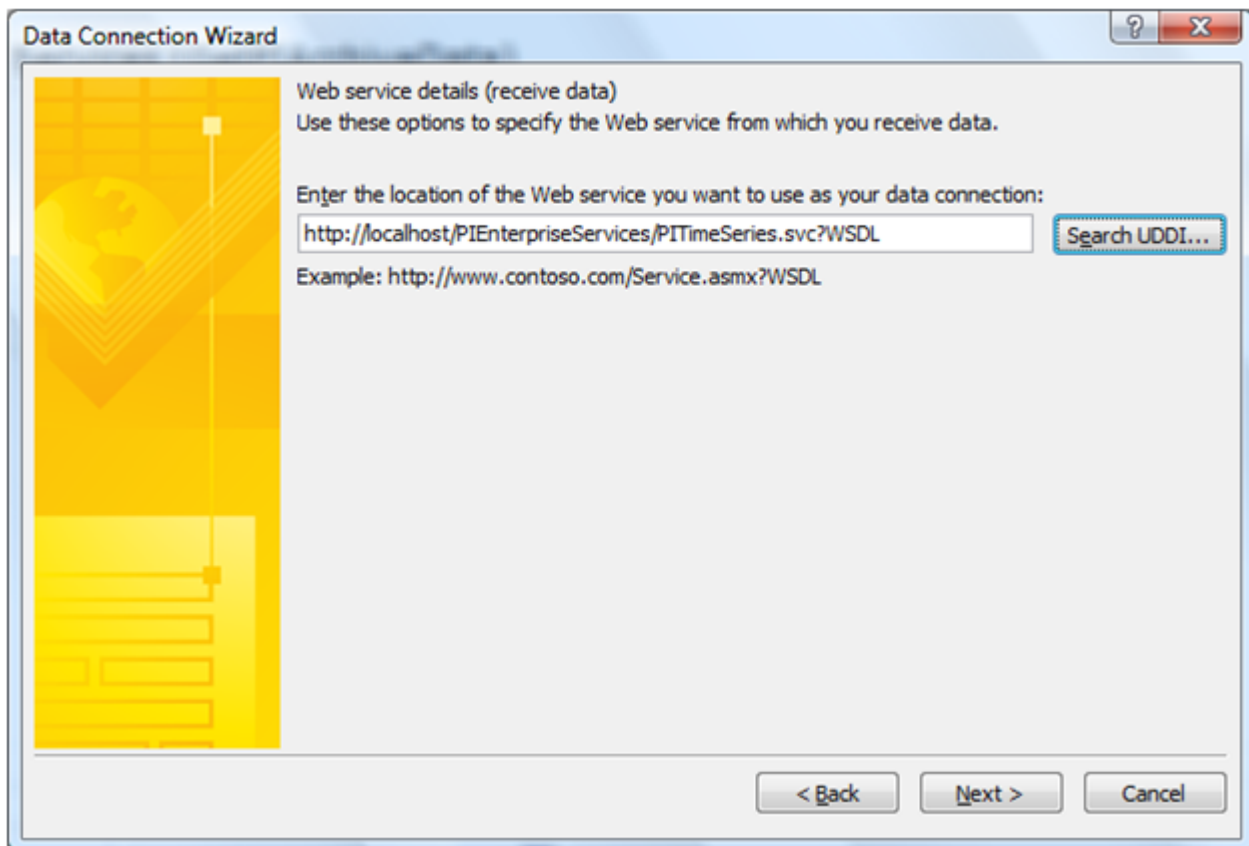


Note:

Alternatively, use the **Design a Form Template** option from the **Getting Started** dialog.

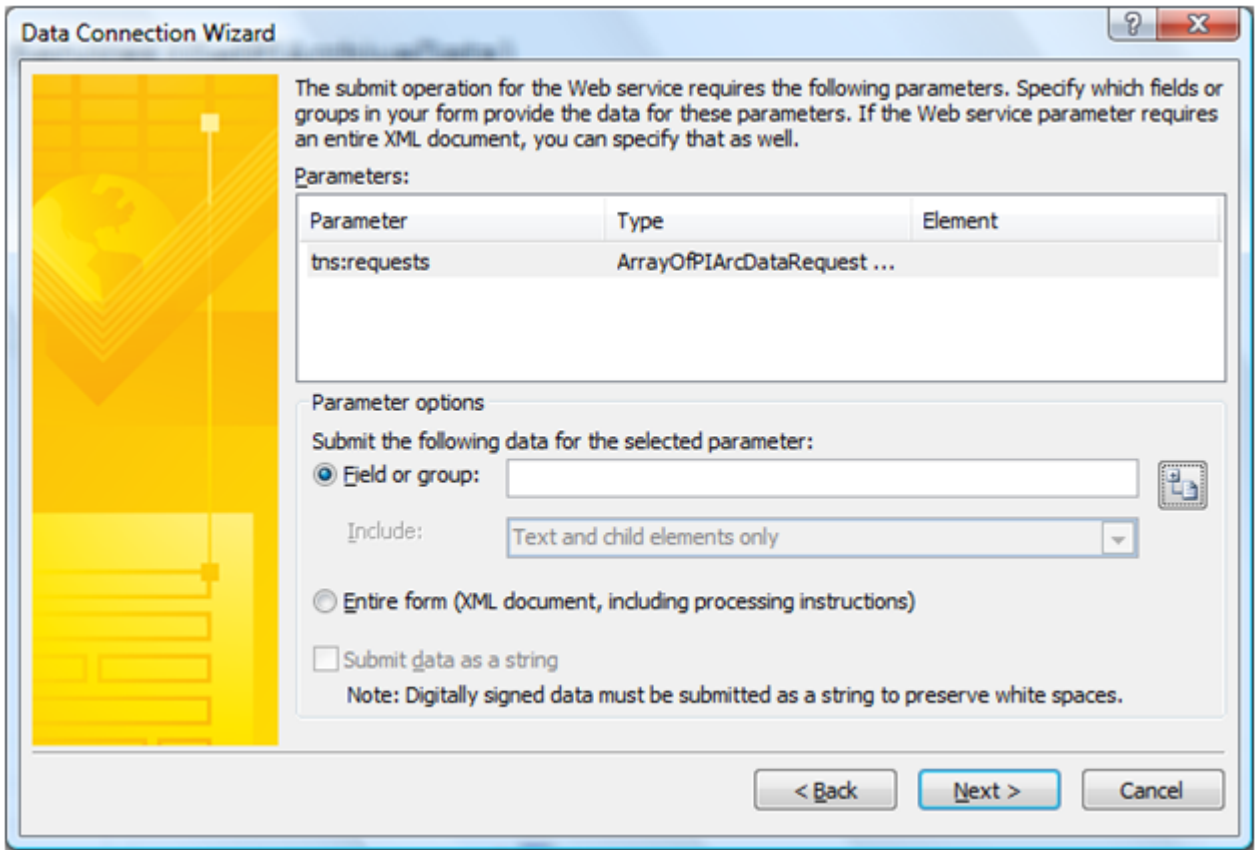
3. Select **Web Service** from the **Based On** options in **Design a Form Template** and click **OK**.
4. Click **Next** to accept the default option in the **Data Connection Wizard–Receive and submit data**.
5. Use the **Data Connection Wizard** to configure the form to receive data from PI Web Services:

- a. Enter the path to PI Web Services that you will use for data connection:

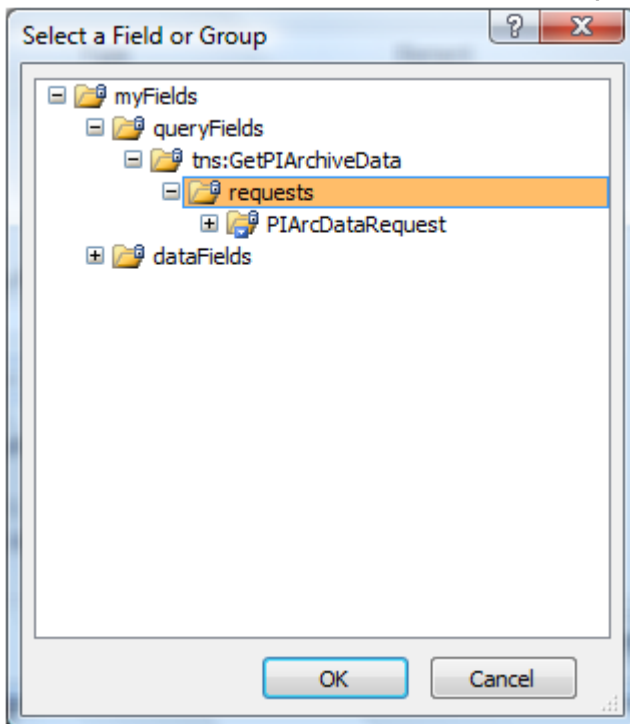


InfoPath queries PI Web Services for metadata in a process that might take several seconds. When the process is complete, the **Data Connection Wizard** gives you the option to select a method supported by the PI Web Services 2010 R3:

- InsertPIData
 - GetPISummaryData
 - GetPIArchiveData
- b. Select **GetPIArchiveData** and click **Next**.
- c. Accept the default name of **Main query** or provide another name to identify the data connection that will receive data entered into the form and click **Next**.
6. Configure the InfoPath form to use parameters that capture PI System data: InfoPath uses the WSDL file to create empty fields in the form. To ensure the fields capture the correct data, you must associate, or bind a Web services request parameter to each field. Binding these parameters is like labeling a bucket; only data that matches the label can be added to the bucket, or in this case, the form field.
- a. Select a parameter in **Parameters**.



- b. Select **Field or Group** in **Parameter options** and click the button to the right of the field.
- c. Select the requests element in **Select a Field or Group**:



- d. Click **OK**.
- e. Click **Next**.
- f. Click **Finish**.

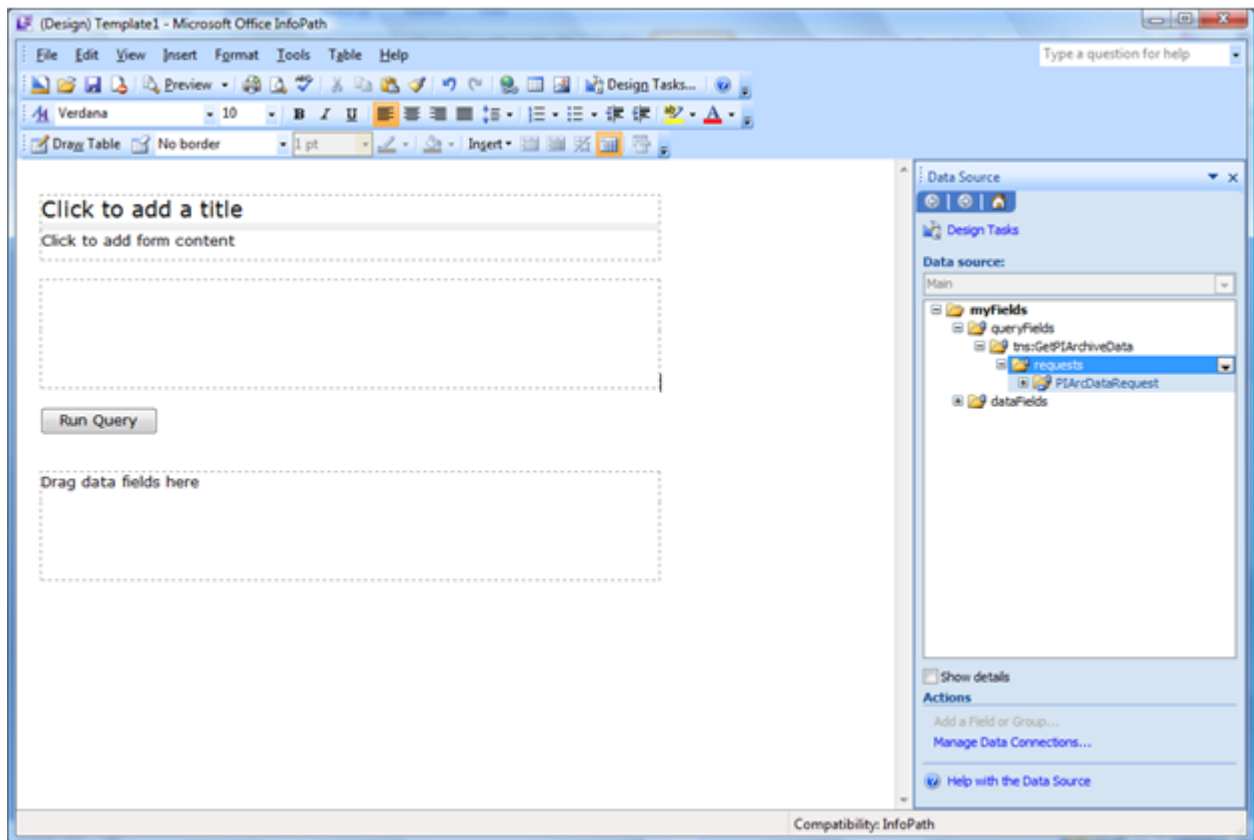
After you finish

Design the layout for the form template.

Design the form


Now you can create the template for the form that users will see when they enter requests for, and receive, data through PI Web Services.

The form template is configured to make Web service requests and receive data responses. When the **Data Connection Wizard** closes, an empty form template is displayed:



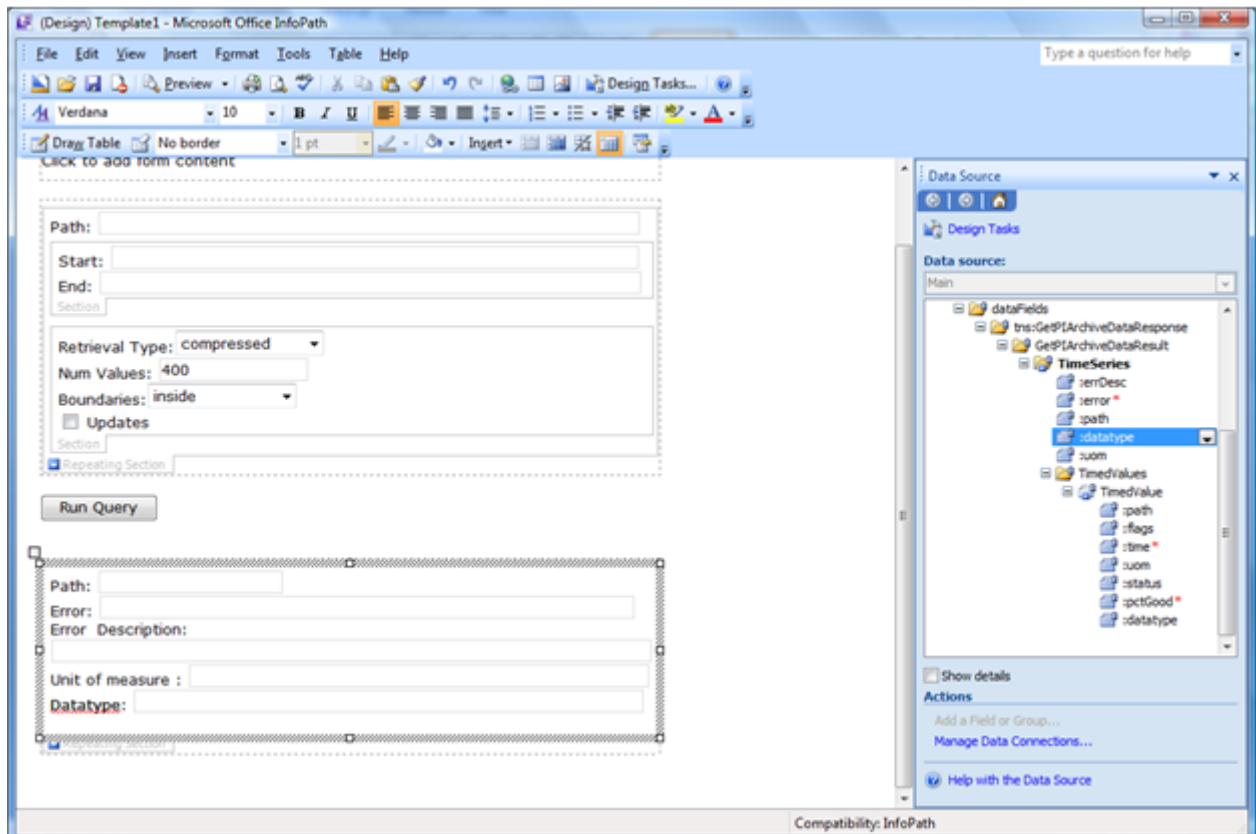
Procedure

1. Lay out fields on the form that will elicit the request parameter values and display the response values.
 - a. Drag the **Path** element onto the form to create a text field within a repeating section.
 - b. Drag the **TimeRange** element onto the repeating section under **Path**.

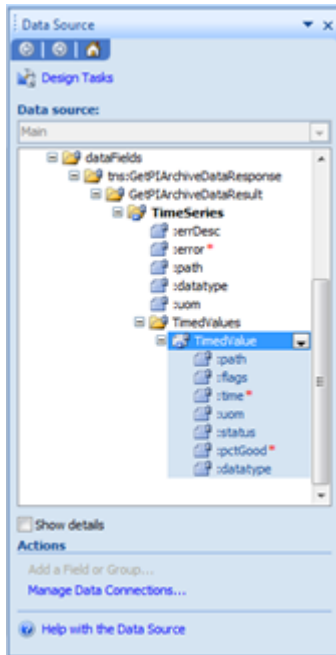
 **Note:**

You can change the width of the text fields. However, do not change the control type from text fields. Date picker controls cannot be used to submit PI [reference times](#).

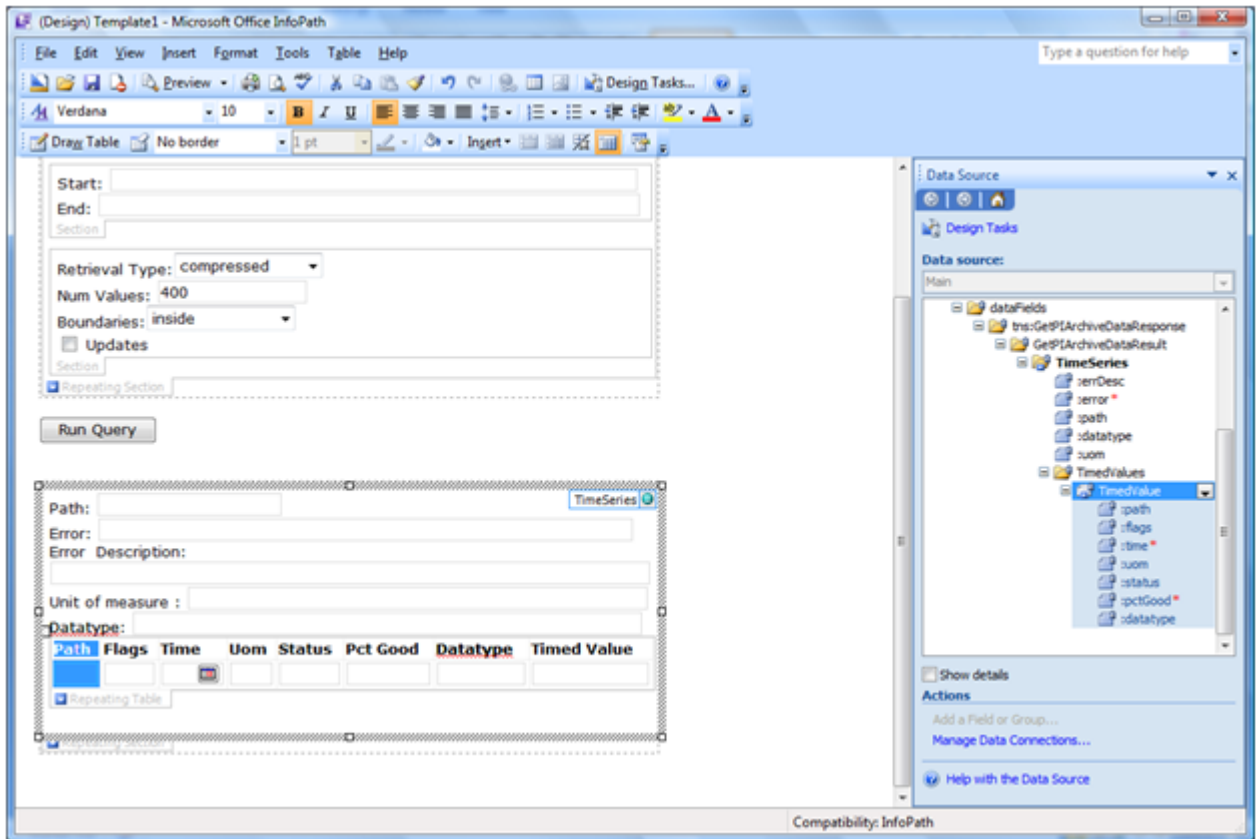
- c. Drag **PIArcManner** onto the form. Place it within the repeating section created by dropping Path onto the form, but outside the section that encloses **TimeRange**. Select **repeating section with controls**.
 2. Drag the Path from TimeSeries onto the lower portion of the form to create a repeating section. The data returned by the PI Web Services is an array of TimeSeries objects. Each such object consists of some data pertaining to the time series as a whole, followed by an array of TimedValue objects representing events returned by the PI Server.
 3. Drag **Error**, **ErrDesc**, **UOM**, and **DataType** onto the form, in succession, in the repeating section under **Path**.
 4. Edit the labels as desired:
 - a. The **Error** field is a numeric error code returned if the retrieval as a whole failed.
 - b. If it is non-zero, an error occurred and **ErrDesc** will contain a descriptive string.
 - c. The **UOM** field contains the units of measure for the PI point or performance equation.
 - d. **DataType** is the XML Schema data type equivalent to the type of the PI Point or performance equation.
- The form should look like this:



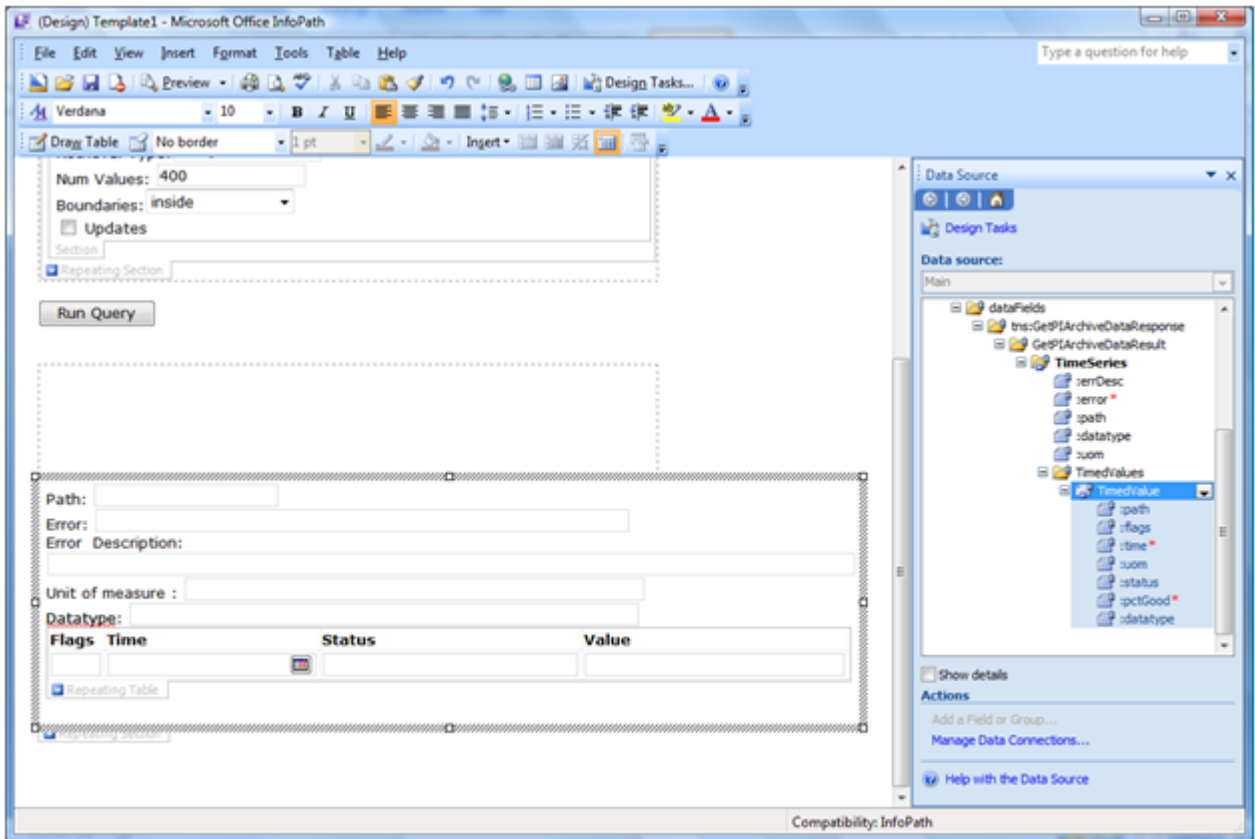
5. Next, create form fields for the individual timed values returned. If you examine the **TimedValues** field in the tree on the right of the form, you will note that there is nothing listed for value. This occurs because the value itself is returned as the textual content of an XML element in the Web service reply. In order to pick this up, drag the entire **TimedValue** object onto the repeating section created above, then edit the table based on what the Web service actually returns for this method.
 - a. Select the **TimedValue** field and drag it into the **TimeSeries** repeating section, placing it directly beneath the **Datatype** field:



- b. Select **repeating table** from the options. Note that each row is crowded. You can edit this to make it more presentable. Much of the data fields defined for the TimedValue object overlaps the TimeSeries object, or are used in different Web service methods. For example, since every TimedValue returned in this method will have the same path as the parent TimeSeries, the Web service does not repeat this information. This reduces the amount of data transmitted over the network for each call. The **pctGood** property, moreover, is used when performing summary retrievals using the **GetPISummaryData** method.
- c. Select the column of the table that contains Path, then right-click and select **Delete > Columns**:



- d. Select the **UOM** field and drag it into the **TimeSeries** repeating section, placing it directly beneath the **Datatype** field.
- e. Select **repeating table** from the options. Note that each row is crowded. You can edit this to make it more presentable.
- f. Select the column of the table that contains Path, then right-click and select **Delete > Columns**:
- g. Select the **PctGood** field and drag it into the **PctGood** repeating section, placing it directly beneath the **Datatype** field: Image doesn't match here
- h. Select **repeating table** from the options. Note that each row is crowded. You can edit this to make it more presentable.
- i. Select the column of the table that contains Path, then right-click and select **Delete > Columns**:
- j. Select the **DataTyp** field and drag it into the **DataTyp** repeating section, placing it directly beneath the **Datatype** field.
- k. Select **repeating table** from the options. Note that each row is crowded. You can edit this to make it more presentable.
- l. Select the column of the table that contains Path, then right-click and select **Delete > Columns**:
The results should be similar to this:



6. Decide whether to change the default setting of the time field control.

After you finish

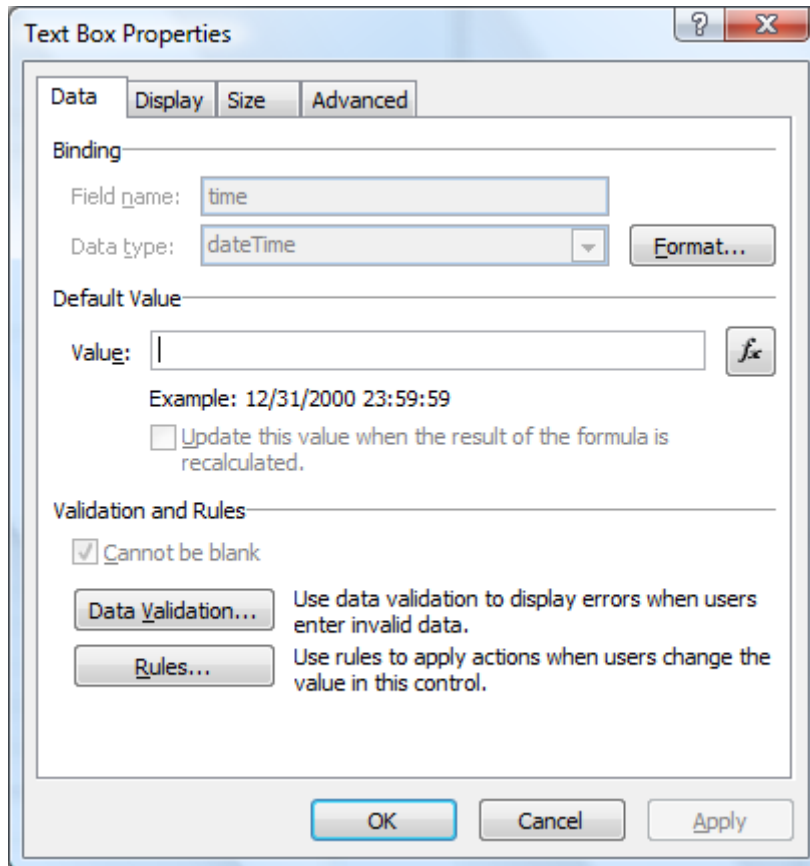
Test the form.

Change the time field default

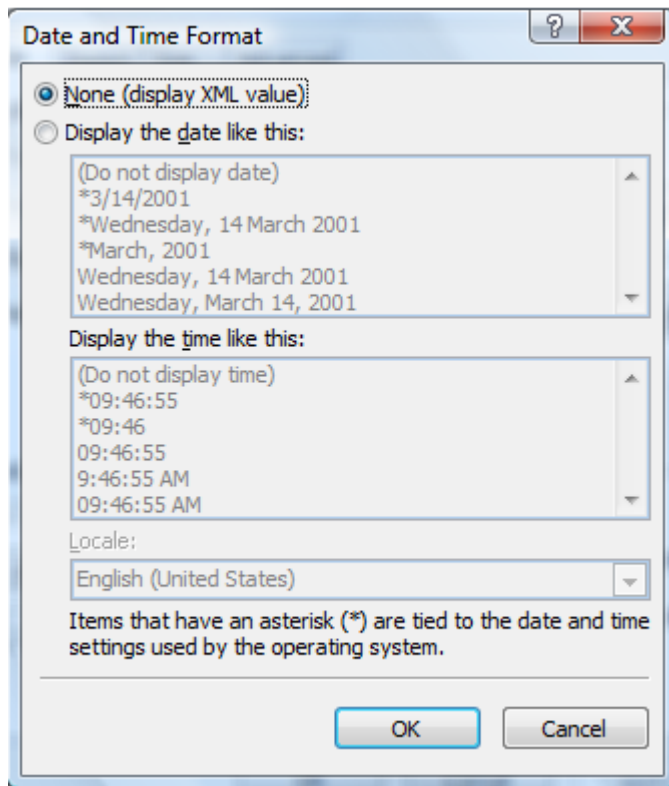
By default, InfoPath configures the time field, which is indicated by `dateTime` in the WSDL, to use a date picker control. Change this default setting if you plan to use PI System data that uses time stamps submitted as PI reference times.

Procedure

1. Right-click on the date picker and select **Change To > Text box**, then double-click on the text box to display the field properties:



2. Select the **Data** tab and click **Format**.
3. Select the data and time formatting as desired:



After you finish

Test the form

Test the form

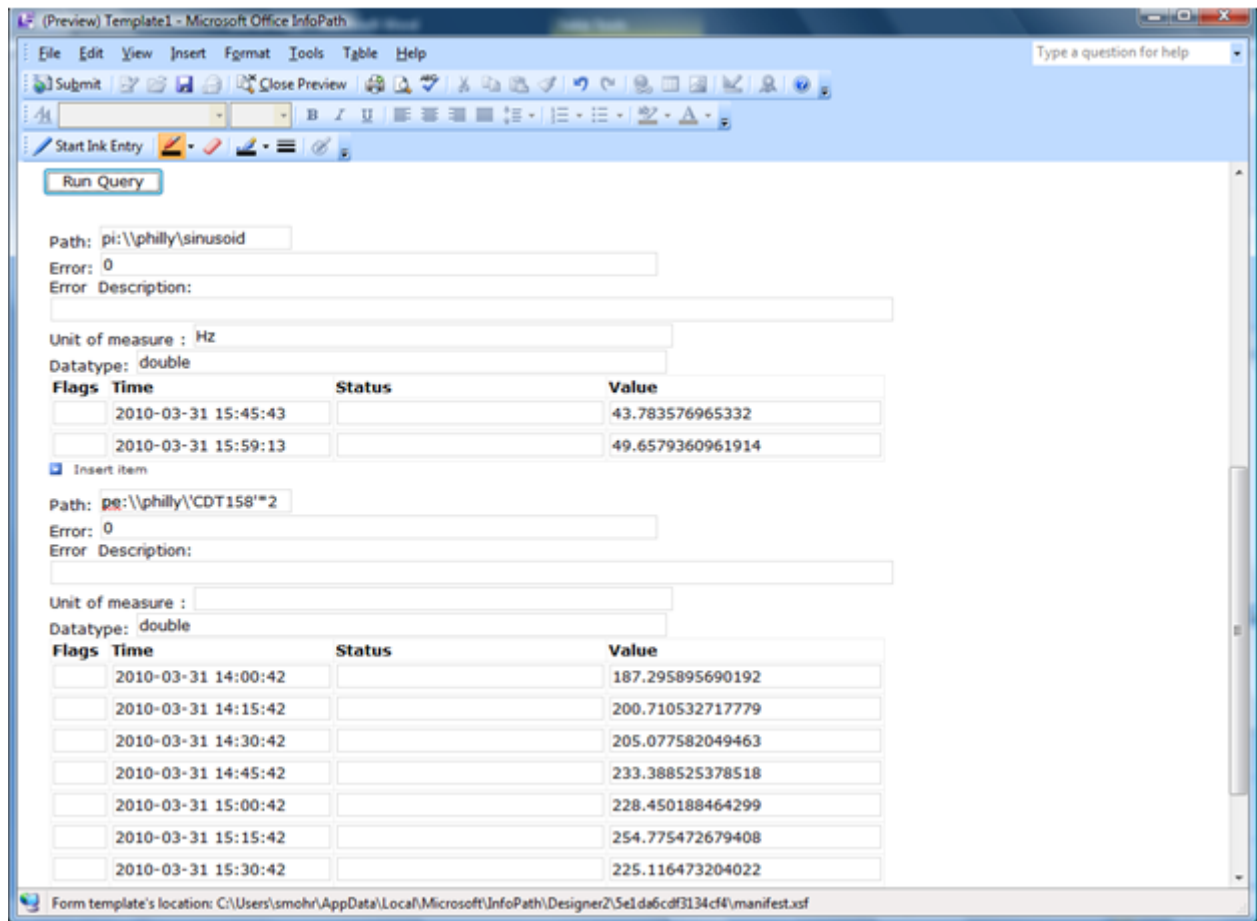
Use PI Web Services and test a form based on the template.

Procedure

1. Click **Preview** on the InfoPath control ribbon.
2. Use this data to enter two requests into the form preview, using the **Insert item** to add fields for the second request:

Path	Start	End	Retrieval Type	Num Values	Boundaries
pi:\\server \sinusoid	*-1H	*	compressed	400	inside
pe:\\server \'CDT158'*2	*-2H	*	interpolated	9	interpolate

3. Click **Run Query**, then **OK** on the security notice.
4. Review the results:



The InfoPath form fields should contain time series data retrieved with PI Web Services.

View and configure message logs

PI Web Services can be configured to display messages and message logs in the tools of your choice. By default, PI Web Services messages appear in the Windows Event Viewer tool on the server on which PI Web Services is installed.

You can change the level of messages shown and the location where messages are stored and viewed. Configure these settings in the `PIInstrumentation.config` file, located in the `PIPC\DAT` directory.

View PI Web Services message logs

Before you start

Procedure

1. Open the Windows Event Viewer tool on the server on which PI Web Services is installed.
 - a. Under Window Logs, click **Application**.
 - b. To find PI Web Services messages, sort on the **Source** column. Look for the source:
 - PI Web Services
2. Select an row in the list of errors to get more information about each error.

Show PI Web Services messages in PI SMT

PI Web Services sends messages to the Windows Event log on the web server. You can configure PI Web Services to also send the messages to the PI message log. This allows you to view PI Web Services messages in PI SMT.



Note: If you opt to send messages to multiple tools (PI SMT, DebugView, Windows Event Viewer) this might result in some performance loss on your server.

Before you start

Procedure

1. On the PI Web Services web server, open the `PIInstrumentation.config` file, which is located in the `PIPC\dat` directory.
2. In the file, locate the `<listener>` parameters. *Listener* refers to listening for log messages. These parameters are in a top-level section called `<listeners>`. By default, three listeners are defined.
3. Edit the listener with the name `PISDKLogEventListener`. It looks like this:

```
<listener xsi:type="PISDKListener"
name="PISDKLogEventListener"
description="PISDK Message Log "
type="OSIsoft.PIInstrumentation.Listener.PISDKLogEventListener"
listenerDataType="OSIsoft.PIInstrumentation.Listener.PISDKLogEventListener,
OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=53b77d1d3d7a979b" formatter="General" PIServer="&lt;your-PI-server-
name-here&gt;" />
```

4. Change the value of the `PIServer` parameter, to the name of the PI server where you want the messages to go. For example, if your PI server is named, `PISRV1`, then the `PIServer` parameter would look like this: `PIServer="PISRV1"`
5. In the `<logFilters>` section, locate `<listeners>` and add the `PISDKListener` to the list of listeners for logging. . By default only the Windows Event log listener is specified here. Add the `PISDKLogEventListener`. The section should look like this:

```
<listeners>
  <listener>WmiTraceListener</listener>
  <listener>PISDKLogEventListener</listener>
</listeners>
```

You can, if you choose, remove the `WmiTraceListener` from the list. Then PI Web Services messages will no longer appear in the Windows Event log.

6. Save the file.



Note: You do not need to restart your application after you modify the `PIInstrumentation.config` file.

View messages in PI SMT

Before you start

Procedure

1. Open PI System Management Tools (PI SMT).
2. In the Collectives and Servers pane, select the PI server that you specified in the `PISDKLogEventListener` section of the `PIInstrumentation.config` file.
3. Open the Message Logs tool (Operation > Message Logs).

Results

PI Web Services messages appear in the PI Message Log tab. The PI SMT help explains how to search and apply filters to the message logs.

Show PI Web Services messages in DebugView tool

You can configure PI Web Services to monitor messages in Microsoft DebugView tool. This tool allows you to save error messages in a text file that can then be sent to the OSISOFT Technical Support team for additional troubleshooting.

Download DebugView tool from:

<http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx>



Note: If you opt to send messages to multiple tools (PI SMT, DebugView, Windows Event Viewer) you might experience some performance loss on your server.

Enable DebugView tool

PI Web Services sends messages to the Windows Event log on the server on which PI Web Services is installed. You can configure PI Web Services to also send the messages to the PI message log. This allows you to view PI Web Services messages in PI SMT.



Note: If you opt to send messages to multiple tools (PI SMT, DebugView, Windows Event Viewer) this might result in some performance loss on your server.

Before you start

Procedure

1. On the server on which PI Web Services is installed, open the `PIInstrumentation.config` file, which is located in the `PIPC\dat` directory.
2. In the file, locate the `<listener>` parameters. *Listener* refers to listening for log messages. These parameters are in a top-level section called `<listeners>`. By default, three listeners are defined.
3. Edit the listener with the name `PISDKLogEventListener`. It looks like this:

```
<listener xsi:type="PISDKListener"
name="PISDKLogEventListener"
descriptions="PISDK Message Log "
type="OSisoft.PIInstrumentation.Listener.PISDKLogEventListener"
listenerDataType="OSisoft.PIInstrumentation.Listener.PISDKLogEventListener,
OSisoft.PIInstrumentation.Listeners, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=53b77d1d3d7a979b" formatter="General" PIServer="&lt;your-PI-server-
name-here&gt;" />
```

4. Change the value of the `PIServer` parameter, to the name of the PI server where you want the messages to go. For example, if your PI server is named, `PISRV1`, then the `PIServer` parameter would look like this: `PIServer="PISRV1"`
5. In the `<logFilters>` section, locate `<listeners>` and add the `PISDKListener` to the list of listeners for logging. . By default only the Windows Event log listener is specified here. Add the `PISDKLogEventListener`. The section should look like this:

```
<listeners>
  <listener>WmiTraceListener</listener>
  <listener>PISDKLogEventListener</listener>
</listeners>
```

You can, if you choose, remove the `WmiTraceListener` from the list. Then PI Web Services messages will no longer appear in the Windows Event log.

6. Save the file.



Note: You do not need to restart your application after you modify the `PIInstrumentation.config` file.

Suppress duplicate messages

You can prevent client applications from flooding logs with duplicate messages when there is a recurring problem. The default value is 5 minutes, meaning that if the same message repeats within 5 minutes, logging applications will not log that message.

To modify this setting, edit the `web.config` file located on your client machine. Edit the following entry under `<appSettings>` to adjust throttling settings:

```
<add key="ErrorSuppressionTime" value="" />
```

The value is a time in minutes. So the following line would set message throttling to ten minutes:

```
<add key="ErrorSuppressionTime" value="10" />
```

If the value is zero, then message throttling is disabled. OSIsoft recommends that you do not disable message throttling.

Change levels of messages displayed

By default, PI Web Services logs only at the **Warnings** level. To help troubleshoot, you might need to increase the level of reporting. Similarly, you might want to screen for error messages only, without displaying warnings.

There are three possible levels of messages that you can log:

- **Warnings**, which is the default level of logging that PI Web Services is installed with, logs error and warning messages
- **Errors** logs only the error messages
- **All** logs all the messages, including informational messages

To change the level of messages displayed:

Before you start

Procedure

1. Open the `PIInstrumentation.config` file, which is located in the `PIPC\dat` directory on the web server where PI Web Services is installed.
2. Search for the following string: `<logFilter logMode=`
3. Edit the string to change the level of reporting. For example, to enable trace messages, you would type: `<logFilter logMode="All"` To send only error messages to the log, you would type: `<logFilter logMode="Errors"`

Results

Open the Windows Event Viewer tool to review the updated level of messages logs.

Configure Kerberos authentication

This appendix describes how to set up Kerberos authentication for products that use PI Data Services, including PI Web Services and PI WebParts.

**Note:**

The security configuration changes described in this appendix will affect all applications running in the Microsoft Internet Information Services (IIS) Web application.

PI Web Services and PI WebParts use delegation to perform operations on behalf of the calling user. The underlying PI Data Services data layer, also uses delegation. This results in a double-hop situation that requires proper configuration.

The `web.config` file that is shipped with OSISOFT product is properly configured for the double-hop situation if the Web service is running on the IIS Web server under the **NETWORK SERVICE** account. If an application pool uses another identity, you must configure an SPN. To create an SPN, you must have domain administrator privileges.

If the Internet Information Services (IIS) application pool runs under a domain account, you must complete these additional steps to configure Kerberos delegation:

- Associate an SPN with the domain account
- Mark the domain account as trusted for delegation in Active Directory
- Modify the `web.config` file

Prerequisites for Kerberos delegation

To use Kerberos delegation, the Web server that hosts the product must be configured to accept the user's login credentials and relay the credentials to the remote server.

To achieve this configuration:

- All servers are Windows 2003 or Windows 2008
- All user accounts, service accounts and computers are members of the same Active Directory forest
- The installing user is able to select the application pool under which the Web service will run

Enable Kerberos on a SharePoint Web server

If you are using PI Web Services on a server that is also running PI WebParts, you can find procedures to enable Kerberos delegation on a SharePoint server in these Microsoft articles:

Microsoft Windows SharePoint Server version	Location
SharePoint Server 2010	http://technet.microsoft.com/en-us/library/ee806870.aspx
SharePoint Foundation 2010	http://technet.microsoft.com/en-us/library/ff607695.aspx
Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007	http://support.microsoft.com/?id=832769

Verify that user accounts can use Kerberos delegation

Generally, there are no changes required to user accounts or Active Directory computers. In **Active Directory Users and Computers**, open properties for the end user's account and go to the **Account Options** tab. Ensure that the option **Account is sensitive and cannot be delegated is not selected** is enabled.

Associate an SPN with the application pool domain account

The Kerberos protocol relies on the use of a Service Principal Name (SPN) to associate services with domain accounts. If the application pool is configured to run under a domain account, an SPN must be established to bind the domain account to the HTTP service.



Note:

The application pool can be associated with a local machine account or a domain account, depending on your site's security policies.

SPNs are administered with the command line tool SETSPN available from the Microsoft Web site or the Windows operating system:

- SETSPN is available in Windows Server 2008 when the Active Directory Domain Services role is added.
- SETSPN for Windows 2003 is part of support tools included with Windows Server 2003 Service Pack 1 (SP1); See: <http://support.microsoft.com/kb/892777>.

Create an SPN

- Use the SETSPN utility to create an HTTP SPN for a Web site.
 - For example, use this command to create an SPN for a Web service that runs under the account **piwp123\1ws** that is accessed at <http://corporatews.piwp123.com/ws.aspx>:
`SETSPN -A HTTP/corporatews.piwp123.com piwp123\1ws`

After you finish

If an application pool uses an identity that does not run under the **NETWORK SERVICE** account, associate an SPN with the domain account.

Associate an SPN with the domain account

Normally, the Service Principal Name (SPN) is based on the machine name and SETSPN is run twice, once with the NETBIOS name of the server and once with the fully qualified domain name.

- Use SETSPN with the -A option for both the NETBIOS name and the fully-qualified domain name of the Web server. :

- For example, with a machine named `piwp123\corporate1web` and an account called `1web`: `SETSPN -A HTTP/CORPORATE1WEB piwp123\1webSETSPN -A HTTP/corporate1web.piwp123.com 1web`
- For a machine named `wdomain\piwserver1` and an account `piws1acct` on the same domain: `SETSPN -A HTTP/piwserver1 wdomain\piws1acctSETSPN -A HTTP/piwserver1.wdomain.com piws1acct`

After you finish

Set the domain account as trusted for delegation.

Set the domain account as trusted for delegation

To perform this procedure, you must have permission to modify the permissions of the account in Active Directory.

In Active Directory **Users and Computers**, trust the domain account using delegation.

Procedure

1. From the account properties dialog, click the **Account** tab.
2. Under **Account Options**, select the option **Account is trusted for delegation**.
3. Click **OK**.
4. Open the `web.config` file and locate the `servicePrincipalName` binding element. Change `HOST/machine_name` to the value of the SPN established in the first step.
For the example above, the value would be set to `HTTP/piwserver1`.



Note:

- Use of Alternate port numbers
Web sites running under port numbers other than the default port 80 do not require any additional configuration. The Kerberos HTTP SPN enables the IIS W3WP process to delegate credentials on all ports, including 443 for SSL.

After you finish

Modify the `webconfig` file.

Use of Basic authentication with SSL

Network authentication through the Internet is often configured on a Web server that is located behind a firewall in a DMZ. Browsers connect using Basic authentication, with the password encrypted through SSL on port 443. A successful login establishes the user's Windows credentials for the duration of the HTTP session.

With this scenario, the domain authentication occurs on the Web server, so access to data sources has only one hop. Kerberos delegation is not necessary, and there is no configuration required beyond the default installation.

Web sites that share an IP address and port number

Within Internet Information Services (IIS), Web sites can be configured to share a single IP address and a port number if they are accessed through different host headers. For clients to access the Web site using a host

header, a DNS entry must be configured to resolve the IP address correctly. In such cases, the fully-qualified domain name that is used to access the Web site is not same as the machine name. To enable Kerberos delegation to work as expected, a new SPN must be established for the HTTP service and the fully-qualified domain name.

Configuration that shares IP address and port number

For example, a Web server name *CORPORATE1WEB* has two Web sites:

Site	Port	Header	Application Pool Identity	FQDN
Site1	80	sample	Network Services	sample.piwp123.com
Site2	80	example	piwp123\1web	example.piwp123.com

To establish Kerberos SPNs for the two sites:

```
SETSPN -A HTTP/sample.piwp123.com piwp123\corporate1web
```

```
SETSPN -A HTTP/example.piwp123.com piwp123\1web
```

Use of load-balanced Web farms

Supported versions of Microsoft SharePoint can be installed in a Web farm consisting of multiple front-end Web servers sharing a single configuration database. The Web farm is accessed through a single URL established in DNS that points to the Web farm cluster address. As with host headers, a Kerberos SPN must be established for both the NETBIOS name and the domain name of the farm.

For example, a Web farm at `http://piwpfarm.piwp123.com` is configured with application pools running under a domain account `piwp123\1web`. To add the SPN:

```
SETSPN -A HTTP/PIWPFARM piwp123\1web
```

```
SETSPN -A HTTP/piwpfarm.piwp123.com 1web
```

Troubleshooting

Establishing Kerberos delegation in an existing distributed environment can be challenging, particularly if Service Principal Names must be established or if the domain policies are restricted. As with other networking tasks, OS/soft recommends that you confirm one hop at a time to verify the configuration.

Verify Kerberos configuration on client computer

Procedure

1. Log out of the client machine and log back in with the appropriate user account, and then browse to the page with double-hop data access.
2. Review the security event log on the Web server. The security event log records events that reflect the user account activity appear.
3. Determine which protocol is used for authentication. If the protocol is NTLM instead of Kerberos, a double hop cannot succeed. This occurs if the Web server is not configured to accept Kerberos or the client account cannot use Kerberos.

Verify Kerberos configuration on Web server

Procedure

1. Open the security event log on the target machine where the data source resides. The data source resides on the file server, the Web service machine or the database server. The security event log records events that reflect the user account activity appear.
2. Review the log for events that verify whether the network logon is successful, and list the user name and the authentication protocol that was used.
 - If the protocol is NTLM and the user name is blank, the Web server is not using Kerberos to communicate with the target machine.
 - If the client uses Kerberos for a successful connection to the Web server, this hop can fail if the target machine is not configured to use Kerberos.

Resources

- KerbTray
A utility that helps diagnose Kerberos issues for Windows Server 2003. You can use it to view Kerberos tickets and purge credentials without logging out. **KerbTray** is available at the Microsoft Web site.
- Troubleshooting Kerberos
See this [MSDN article \(http://technet.microsoft.com/en-us/library/cc728430.aspx\)](http://technet.microsoft.com/en-us/library/cc728430.aspx) for a comprehensive discussion of Kerberos troubleshooting.

Technical support and other resources

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

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

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

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

Index

AttributeFilter, 102
AttributeSearchConstraints, 104
AttributeSearchManner, 105
AttributeValue, 105
CancelPIUpdates, 72
DurationFilter, 106
ElementSearchConstraints, 108
ElementSearchManner, 109
EventFrame, 109
EventFrameSearchConstraints, 110
EventFrameSearchManner, 113
EventFrameValues, 114
FindAttributePaths, 94
FindElements, 94
FindElementTemplates, 96
FindEventFramePaths, 98
FindEventFrames, 97
FindPIPathsBasic, 91,99
GetElementsByPaths, 100
GetEventFrameTimeSeries, 101
GetEventFrameValues, 102
GetPIArchiveData, 72
GetPISnapshotData, 73
GetPISummaryData, 74
GetPIUpdates, 75
GetProductVersion, 77
InsertPIData, 77
IPISearch, 91
IPISoap, 93
IPITimeSeries, 71
ListPathsByUpdateTicket, 80
PIArcDataRequest, 81
PIArcManner, 82
PISummaryDataRequest, 84
PISummaryManner, 84
SignUpForPIUpdates, 80
SignUpResults, 90
TimedValue, 89
TimeRange, 85
TimeSeries, 87
TimeSeriesUpdates, 89

T

technical support, 139

