PI Web Services 2010 R3

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

OSIsoft Australia • Perth, Australia

OSIsoft Europe GmbH • Frankfurt, Germany

OSIsoft Asia Pte Ltd. • Singapore

OSIsoft Canada ULC • Montreal & Calgary, Canada

OSIsoft, LLC Representative Office • Shanghai, People's Republic of China

OSIsoft Japan KK • Tokyo, Japan

OSIsoft Mexico S. De R.L. De C.V. • Mexico City, Mexico

OSIsoft do Brasil Sistemas Ltda. • Sao Paulo, Brazil

OSIsoft France EURL • Paris, France

PI Web Services 2010 R3 User Guide

Version: 1.2.8.0

Published: January 2012

Table of Contents

Chapter 1 Introduction 1

In this Guide 1
About the OSIsoft PI Data Access Suite 2
Architecture 3
Summary of Features 4
Role of PI Data Services 6
Web Service Inputs 7
PI System Data Types 9
Error and Trace Message Logging 9
Connections to PI and AF Collectives 10
Time Stamps 10
Automatic WSDL Generation 13

Chapter 2 Install PI Web Services 15

Before Installation 15
Run Setup Kit 27
After Installation 28
Host PI Web Services with a Windows service 40
Silent Installation of PI Web Services 45

Chapter 3 PI Web Services Security 47

Supported Security Scenarios 48
Secure Access to PI Server Data 48
Secure Access to Data through PI Asset Framework 51
Configure Required User Accounts 51
Configure Secure Web Service Access 53
Firewall Security 68

Chapter 4 Troubleshooting 71

IIS is Not Running 71
Server Not Found 72
PI System Not Found 72
Remote Connection Failure 72
Data Entry Disallowed 73
Insufficient Permissions 73
Insufficient Message Size 73
Execution of Performance Equations Disallowed 74
Filters or Parameters Not Supported 74
Server Error in PI Web Services Application 75

Invalid Tag Name 76

Chapter 5 PI Web Services Programmer Reference 77

PI Web Services Interfaces 77

Chapter 6 Use InfoPath with PI Web Services 99

Configure the Data Connection 100
Select a Method 101
Configure Field Input 101
Design the Form 102
Design Returned Data Display 103
Change the Date Picker Control 107
Test the Form 108

Appendix A Configure PI Web Services to use Kerberos Authentication 111

Prerequisites 111
Configure the User Account 112
Use of Alternate port numbers 115
Use of Basic Authentication with SSL 115
Use of Host Headers 115
Use of Load-Balanced Web Farms 116
Troubleshooting 116

Appendix B Logging and Instrumentation 119

Initial Settings 119
Modify Configuration 123
Message Throttling 125

Appendix C Technical Support and Resources 127

Index 131

Chapter 1

Introduction

PI Web Services 2010 R3 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 calling Web services
- Allow both Windows and non-Windows environments to easily access PI data, or submit data to PI systems

These examples illustrate some of the key advantages that PI Web Services provides, including:

- Ability to integrate with applications, regardless of the programming language or platform
- Broad access to PI System data and features
- Interfaces 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* (page 2).

In 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* (page 77) 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* (*http://msdn.microsoft.com/*).

For supplementary support and information about PI Web Services, see the *OSIsoft vCampus Web site* (*http://vCampus.osisoft.com*), which provides various technical resources including a blog, discussion forum and Webinars dedicated to PI Web Services.

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. Products 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 their individual membership to the OSIsoft Virtual Campus (*vCampus* (*http://vCampus.osisoft.com*)) program. For details, see the OSIsoft vCampus *Frequently Asked Questions http://vCampus.osisoft.com/content/FAQ.aspx*.

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 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 Web site* (*http://www.osisoft.com*) or contact *OSIsoft Technical Support* (page 127).

Architecture

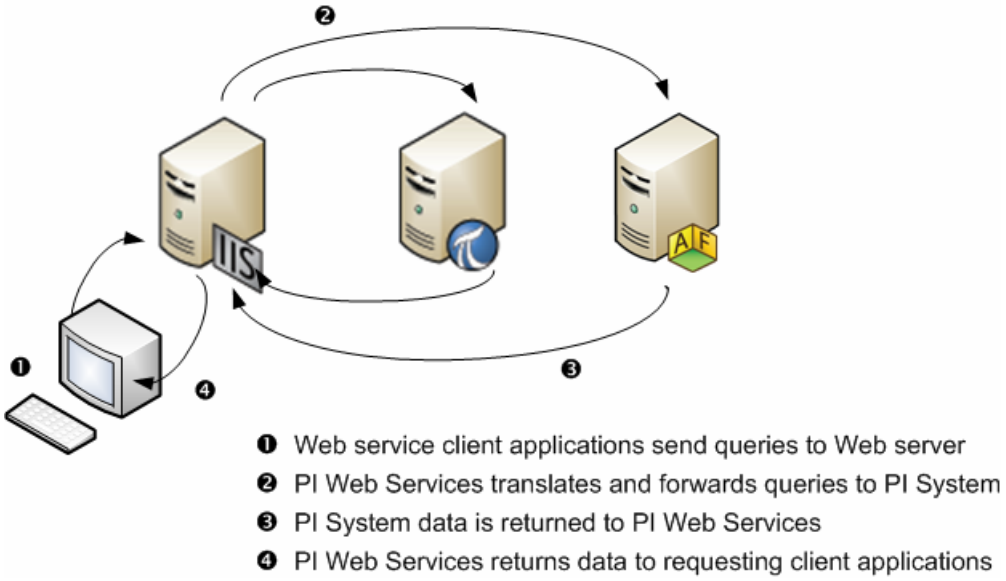
A typical scenario for using PI Web Services includes computers that host:

- 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. For details about how to connect PI Web Services to a PI System, see *Install PI Web Services* (page 15) and *Change the Endpoint and Active Configuration Bindings* (page 56). For an example that shows how to set up the connections to Web services applications, see *Configure Security for .NET Clients* (page 61).

You may also host PI Web Services as a *Microsoft Windows service* (page 40).

Note: If you are using the netTCPBinding or netNamedPipes bindings, you must use IIS 7 or later. For details, see *Configure Secure Web Service Access* (page 53).



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* (page 6), 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* (page 6) to PI Web Services, where it is reformatted into the data structures defined by the *PI Web Services interfaces* (page 77). 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.

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 two interfaces:
  - **IPITimeSeries** – To retrieve time series data supported by the PI System and receive updates to supported data events. This includes PI points and time series data that are referenced by PI AF attributes. Future releases will add support for the other major *PI System data types* (page 9).
  - **IPISearch** – for searches the PI System that yield search results consisting of paths for use with IPITimeSeries.
- Support of these standard PI summary calculations: averages, minima, maxima, ranges, totals, values, counts, and both sample and population standard deviations.
- Interfaces 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* (page 99) for an example.
- A Web method that supports entry of PI Server data.
- Web methods that allow users to sign up for *data updates* (page 5).
- Support for Java-based applications, and Web service calls from non-Windows operating systems.
- The ability to send and receive PI Web Services calls to and from *PI or PI AF collectives* (page 10).
- Support for 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* (page 40), rather than in Internet Information Services (IIS).

Features not supported are:

- Annotations. None of the *IPITimeSeries Interface* (page 77) 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* (page 80) method with a performance equation path or a PI AF data reference path.

Retrieval of Data Updates

PI Web Services 2010 R3 introduces 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 named by the path included 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 configuration, that is, data reference configurations that do not used 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, either *implicitly* (page 6) through an existing query, or *explicitly* (page 6) by passing one or more paths to the Web method *SignUpForPIUpdates* (page 83). 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* (page 84) to obtain any new or changed events. When the application is finished processing new events, you can call *CancelPIUpdates* (page 84).

There is no limit on the number of times that your Web service client application calls *GetPIUpdates* (page 84). You should be aware that signups expire if the *GetPIUpdates* (page 84) 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* (page 83). The expiration feature prevents consumption of system resources when lost or ill-behaved clients fail to request updates.

Your application may call *ListPathsByUpdateTicket* (page 86) at any time to retrieve the list of paths associated with a update ticket. Calls to both *GetPIUpdates* (page 84) and *ListPathsByUpdateTicket* (page 86) reset the internal timer used to check for expiration of signups.

Explicit Sign-ups

Your application signs up explicitly for updates by passing one or more paths to the web method *SignUpForPIUpdates* (page 83). 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* (page 83) will return a SOAP fault.

SignUpForPIUpdates (page 83) allows you to specify the expiration time of the signup. It is not possible to specify an expiration time when using implicit signups.

Implicit Sign-ups

You can submit a data retrieval request for data and sign up for updates in a single operation. This implicit signup occurs if you set the Updates property of a PIArcManner (page 87) or PISummaryManner (page 89) object to **TRUE** for a call to GetPIArchiveData (page 77) or GetPISummaryData (page 79), respectively. For paths that support updates, the Web method returns the update ticket in the SeriesID property of each TimeSeries (page 92) object.

Note: Each update ticket obtained this way represents a signup for a single path passed to GetPIArchiveData (page 77) or GetPISummaryData (page 79). To create an update ticket that represents signups for multiple paths, you must use an *explicit* (page 6) signup.

Implicit sign-ups for updates are supported only for data retrieval requests whose time range ends in current time, or *****. Updates always consist of all snapshot or archive values even if the signup occurred during a query for interpolated, plot values or summary 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 and affect *security requirements* (page 51) 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.

Web Service Inputs

In general, PI Web Services requires three inputs to specify the data to be retrieved from the PI System: *Path* (page 7), *Constraint* (page 9), and *Manner* (page 9).

Path

A *Path* (page 8) designates a unique source of PI System data, such as a PI point or a PI AF attribute. It is comprised of three tokens assembled in this format:

```
<Source Designator> (page 7):\\<Server> (page 7)\<Path to Item>
(page 8)
```

PI Server paths are honored in all methods of *IPITimeSeries* (page 77).

PE paths and PI AF paths work only with the *GetPIArchiveData* (page 77) and *GetPISummaryData* (page 79) methods.

PI Server paths are returned by methods of the IPISearch interface.

<Source Designator>

A Web Service *path* (page 7) begins with one of three source designators to indicate what type of data is used. The source designator of a path can be **pi**, **pe**, or **af**. Use **pi** to access PI archive data, **pe** for PI performance equations, or **af** for data stored in PI Asset Framework.

Note: If you do not specify a server, but include **pi** only, the path will point to the default PI Server for the machine hosting the Web service. For such a path to be valid, you must include the <Source Designator> token. For example, `pi:sinusoid`.

<Server>

This second token of a Web Service Input path designates the server that contains the data source and is:

- the server hosting the PI archive for **pi**
- the PI Server evaluating the performance equation for **pe**
- the PI AF database server for **af**

If there is no server token, queries that use the pi: *designator* (page 7) are sent to the PI Server configured as the default server to which PI Web Services will connect. You can use the **PI Connection Manager** to verify this setting.

The server used in the path must be valid: if the path contains the name of a PI Server that exists and is running but is not in the Known Servers Table (KST), there are two possible outcomes:

- If the PI SDK is configured to automatically add the names of valid PI Servers not already in the KST (the default setting), PI Web Services will access data normally and the PI Server name will be added to the KST for subsequent calls.
- If the PI SDK is not configured to automatically add the names of valid PI Servers not already in the KST, you will get an *error message* (page 72).

<Path to Item>

The third token of a Web Service Input path is <**Path to Item**> which represents the type of object that will be used, depending on which source designator is specified:

Source Designator	Path to item format
pi:	name of PI point
pe:	performance equation that uses PE format
af:	`<database>\{<element>\}+|<attribute>[;uom]`, where the attribute token is a PI AF data reference to a PI point or a scalar type and the optional UOM token preceded by a semicolon is the string representing the unit of measure in which to return the data. If the attribute (or referenced PI point data) is stored in another unit of measure, conversion will be attempted.

Examples

Some examples of PI Web Services input paths are:

`pi:\\piserver1\sinusoid`

`pi:sinusoid`

`pe:\\piserver1\'sinusoid'*2`

`af:\\afserver1\PIDS01\PIWS01|piref1`, where `PIDS01` is a PI AF database and `piref1` identifies a PI AF attribute that is a reference to a PI point

`af:\\afserver1\PIDS01\PIWS01|pi`, where `pi` is an attribute whose value is the constant **3.1415926535**

`af:\\afserver1\PIDS01\PIWS01|BoilingPoint; degree Fahrenheit`, where `BoilingPoint` is an attribute with a constant value of **100 ° C**

Constraint

Constraints define the data to be extracted from a data source that is referenced in a *path* (page 7). Since the current implementation addresses only retrieval of time series data, the only constraint type is the *TimeRange* (page 91).

PI Web Service calls can reference a single instance of time, provided a *TimeRange* (page 91) with identical start and end times is used. For example, to reference current time, enter **StartTime=*** and **EndTime=*** for the time range.

Note: GetSnapshotData is equivalent to **GetPIArchiveData** with a time range that refers to current time, that is **StartTime=*** and **EndTime=***.

Times supported are *ISO 8601 absolute times* (page 12), and PI *relative* (page 11) and *absolute time* (page 12) strings.

Manner

The Manner of data retrieval is a concept that can apply to any kind of data source. In general, Manner is the way you provide details about how you want your data to be retrieved and returned from PI Web Services, aside from *Constraints* (page 9). For example, when retrieving Archive data, you need to specify the maximum number of data values to return and whether results should be compressed or interpolated. When retrieving summary values, you need to specify the summary type, that is, average, total, minimum, maximum, standard deviation, and so on, and whether the summary should be time weighted or event weighted.

PI Web Services uses the *PIArcManner* (page 87) class and the *PISummaryManner* (page 89) class to allow you to define retrieval manner.

PI System Data Types

There are four major data types exposed by the PI System:

- Time series
- PI event frames
- Tables
- Hierarchical structures

Currently, PI Web Services supports only the reading and writing of time series data. Upcoming releases will support retrievals from tabular, hierarchical and PI event frames data.

Error and Trace Message Logging

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* (page 30) of the Web service. For details, see *Logging and Instrumentation* (page 119).

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 DebugView* (*http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx*).

Connections to PI and AF Collectives

When PI Web Services establishes a connection to a high availability PI System to retrieve data from a PI or AF collective, 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 PI Server collective can be adjusted using the PI Connection Manager in **PI SDK Utility**, or by using the **PI System Explorer** within an AF Server collective. For details, see the Help for each application.

Time Stamps

Time Input Translation Rules

In general, use strings to define time constraints in a data retrieval Web method. Use either *relative* (page 11) or *absolute* (page 12) time.

With one *exception* (page 11), *absolute* (page 12) 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* (page 91)

To avoid ambiguity, use *ISO 8601 time string format* (page 12) and include the time zone offset.

Exception to Time Input Translation

There is an exception to the *time input translation rule* (page 10): when *InsertPIData* (page 80) is used, the time stamps of the data are used as the time inserted into the PI Server. *InsertPIData* (page 80) takes a *TimeSeries* (page 92) array as input. All time stamps in this array are defined as absolute time 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 *absolute* (page 12) 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.

Relative Time

Relative time is a range of days, hours, minutes, or seconds. A single leading operator, either + or - , is required:

```
+/- d | h | m | s
```

The default starting point for relative time is usually the current time. For example, **-8h** is eight hours before the current time. Fractional times such as **-1.5d** are supported.

Note: Multiple operators are not supported. For example, use **-23h**, not **-1d+1h**.

Absolute Time

Absolute times have one of the following formats:

Format	Description/Notes
YYYY-MM-DD[THH:MM:SS{Z|{+|-HH:MM}]	ISO 8601 format (preferred). If a time is included, either UTC (Z) or the current offset from UTC must be provided. Example: 2010-04-26T08:00:00-07:00 In this case, **-07:00** is the offset that corresponds to the UTC-7 time zone. For time zones where Daylight Saving Time (DST) changes are observed, use the appropriate offset, for example, **-05:00** when in standard time and **-4:00** when in daylight saving time.
*	current time
T	**00:00:00** on the current day (TODAY)
Y	**00:00:00** on the previous day (YESTERDAY)
Monday	**00:00:00** on the most recent Monday
Sun,Mon,Tue,Wed,Thu,Fri,Sat	**00:00:00** on the most recent Sunday, Monday, ..., Saturday

The following tables show examples of absolute time strings.

Example	Description
25	**00:00:00** on the 25th of the current month
25-Aug-09	**00:00:00** on that date
8:	**08:00:00** on the current date
25 8	**08:00:00** on the 25th of the current month
21:30:01.02	**9:30:01.0200** PM on the current date

Use caution with the default settings. Here are some examples of time stamps that might be confusing.

8:	**08:00:00** on the current date
:8	**08:00:00** on the current date
::8	**00:08:00** on the current date
:::8	**00:00:08** on the current date
0:8	**00:08:00** on the current date

The confusion comes from the ambiguity in the first two examples above. Following this theme, when minutes are added to the next examples, the time stamps are still similar.

8:01	**08:01:00** on the current date
:8:01	**08:01:00** on the current date

The difference in the two notations is evident when a date is added to the time. When a date is added to the front of the time the default notation is hh:mm:ss.ssss not :hh:mm:ss.ssss.

2 8:	**08:00:00** on the 2nd of the current month
2 :8	**00:08:00** on the 2nd of the current month
2 ::8	**00:00:08** on the 2nd of the current month

If extra colons and times are added that is greater than the given DD-MMM-YY hh:mm:ss.ssss format the last part of the time is disregarded.

2 :::8	**00:00:00** on the 2nd of the current month
2 8:01:30	**08:01:30** on the 2nd of the current month
2 :8:01:30	**00:08:01** on the 2nd of the current month

A value for the seconds must be used if sub-seconds are used. Hence use caution when considering time stamps containing sub-seconds.

8::30.01	**08:00:30.0100** on the current date
:8::30.01	**08:00:30.0100** on the current date
14 :8::30.01	**00:08:00** on the 14th of the current month

Following are examples of time stamps that do not work.

8:30.01	Ambiguous, **8** could be minutes or hours
:8:30.01	Ambiguous, **8** could be minutes or hours

Automatic WSDL Generation

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.

The Web server that hosts 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/PISearch.svc
```

Chapter 2

Install PI Web Services

PI Web Services 2010 R3 IIS Edition runs on Internet Information Services (IIS) for Windows Server, a Microsoft Web server and extension modules. When you run the PI Web Services setup kit on a Windows server, it installs:

- OSIsoft MS Runtime Redistributables 3.0.2 and 3.1.1
- PI SDK 2010 R2
- PI Buffer Subsystem 3.4.380
- PI Asset Framework 2010 R3 client
- PI OLEDB Provider 3.3.0.1
- PI Web Services 2010 R3

Before you run the PI Web Services setup kit, verify that the Web server you use meets the *system requirements* (page 16) and complete the required *pre-installation tasks* (page 15).

You may also host PI Web Services as a *Microsoft Windows service* (page 40).

Before Installation

Complete these tasks before you run the PI Web Services installation program:

- Verify the system you plan to use has the *user accounts required* (page 51) for PI Web Services.
- Verify that the server you plan to use:
  - Meets the *system requirements* (page 16).

Note: Prior to installation, the *PI Web Services setup kit* (page 15) will verify that your server fulfills the *system requirements* (page 16). The installer will exit if any components are missing.

  - Uses Microsoft Internet Information Services (IIS) for Windows Server. For details, see *Configure the Web Server* (page 18).
  - Meets the setup requirements described in *Host PI Web Services on a PI WebParts Server* (page 17) if the server on which you install PI Web Services also runs PI WebParts.

- Before you use PI Web Services, you must designate a Web site to host PI Web Services:
  - You can set up this Web site when you install PI Web Services by selecting the location of the Web site's 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 Web site has *folder and registry permissions* (page 71) that are compatible with PI Web Services.
- To *run the setup kit* (page 27):
  - If you install PI Web Services 2010 R3 on Microsoft Windows Vista, Windows 7, or Windows Server 2008, you must be logged in as an administrator.
  - .NET Framework 4 must be installed. See *Install .NET Framework 4* (page 17) for details.
  - IIS must be enabled to use the Windows features described in *Configure the Web Server* (page 18).

System Requirements

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

The server you use to install PI Web Services must be running:

- 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 Vista, Windows 7
- Internet Information Services (IIS) for Windows Server, version 6.0 or later.

Note: Although you can install either the 32-bit or 64-bit version of PI Web Services on a 64-bit computer, OSIsoft recommends that you always run IIS as a 64-bit process if you use a 64-bit Web server.

- .NET Framework 4. To download, go to `http://www.microsoft.com/net`.

The computer must also have network access to one or more of these OSIsoft servers:

- PI Server 3.3 or later
- AF Server 2.1 or later that includes *PI SQL for AF Server* (page 17)

Note: If the server you use also runs PI WebParts, see *Host PI Web Services on a PI WebParts Server* (page 17).

See *Secure Access to PI Server Data* (page 48) and *Secure Access to Data through PI Asset Framework* (page 51) for information about how to configure security settings on these servers.

Review AF Server Requirements

If you will use PI Web Services to access PI AF 2.1 or earlier, verify that PI SQL for AF Server is installed on the AF Server.

When present, PI SQL for AF Server is listed in the Windows **Add or Remove Programs** list. The PI SQL for AF Server setup kit is available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com*).

Install .NET Framework 4

You can download .NET Framework 4 from:

`http://www.microsoft.com/net` (`http://www.microsoft.com/net`)

Supported Client Platforms

PI Web Services 2010 R3 is implemented using Windows Communication Foundation (WCF).

Programmers who want to create clients to call PI Web Services should have the appropriate programming tools installed. The following software tools have been used to successfully create such clients:

- Microsoft Visual Studio 2008 with .NET Framework 2.0 or later, and Visual Studio 2010
- Microsoft Office InfoPath 2007 or 2010
- Eclipse and Apache Axis 1.4 (for testing under Java)
- Microsoft BizTalk Server 2006 R2

Note: End users who access data through PI Web Services require specific *permissions* (page 71) to Web site folders, subfolders and a registry key.

Host PI Web Services on a PI WebParts Server

If you want to host PI Web Services 2010 R3 on a Web server that has PI WebParts installed, OSIsoft recommends that you:

- Create an IIS Web site and associated application pool that are not under the control of Microsoft SharePoint. The Web site should be configured on its own port number. You must select the Web site as the host when you *run the setup kit* (page 27).
- 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.

Configure the Web Server

PI Web Services is hosted by Microsoft Internet Information Services (IIS) for Windows Server. The server on which you install PI Web Services must have, at a minimum, these required features:

- ASP.NET
- .NET Extensibility
- ISAPI Extensions
- ISAPI Filters for .NET Framework 4

If you are using *Windows 7* (page 23), you must also turn on:

- Default Document
- Static Content

Note: If the server you use does not have these IIS features activated, the setup kit will fail. If ISAPI Filters are not enabled for .NET Framework 4, you will be unable to use PI Web Services.

Refer to these topics for details about how to configure IIS, based on the Windows operating system version.

Windows Server 2003

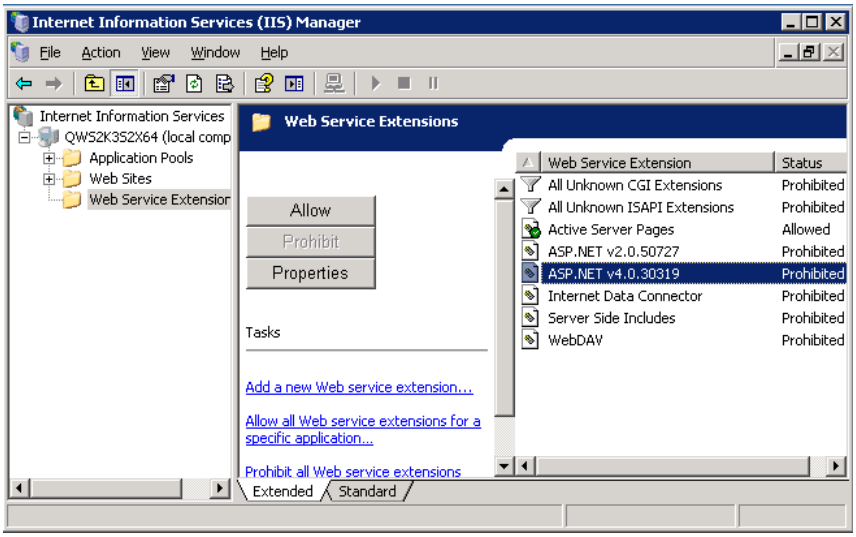
These topics explain how to install and configure Internet Information Services (IIS) on Windows Server 2003 for use with PI Web Services.

Install IIS on Windows Server 2003

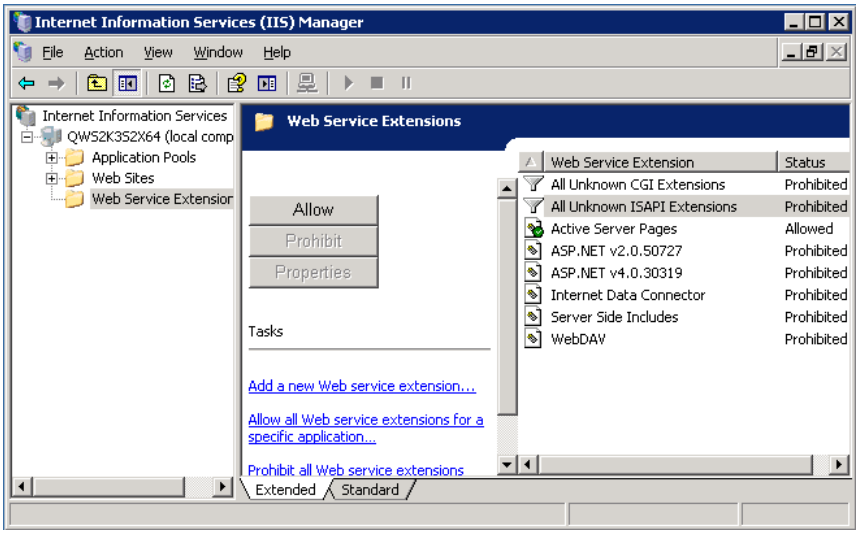
1. Click **Start**, point to **Control Panel**, and then click **Add or Remove Programs**.
2. In **Add or Remove Programs**, click **Add/Remove Windows Components**.
3. In the **Windows Components Wizard**, under Components, select Application Server.
4. Click **Next**.
5. After the wizard completes the installation, click **Finish**.
6. Enable an *ISAPI extension for .NET 4* (page 19).

Configure IIS on Windows 2003

1. Open **Internet Information Services (IIS) Manager**.
   a. Expand the node for your Web server, select the Web Service Extensions folder and then select **ASP.NET v4.xx**:

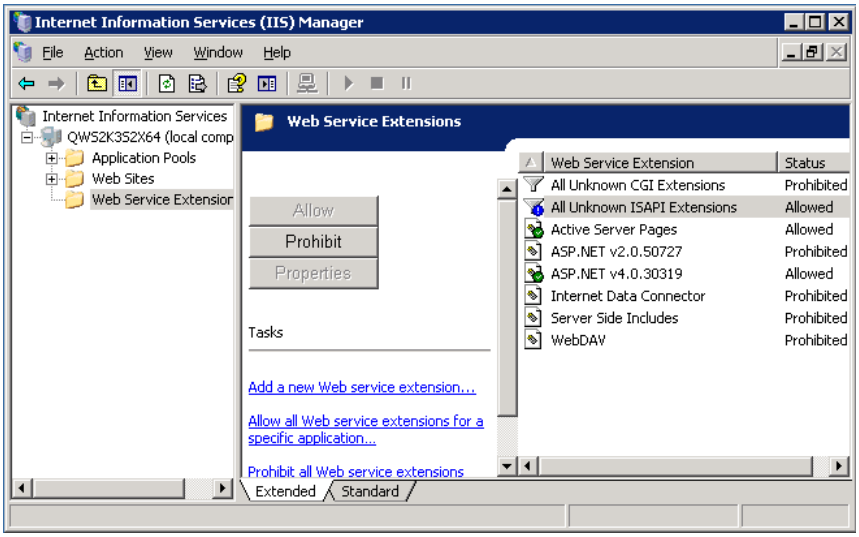


   a. Click **Allow**.
   b. Select **All Unknown ISAPI Extensions** and click **Allow**:



   c. Review the IIS Manager alert about the potential security risk.
   d. Click **Yes**.

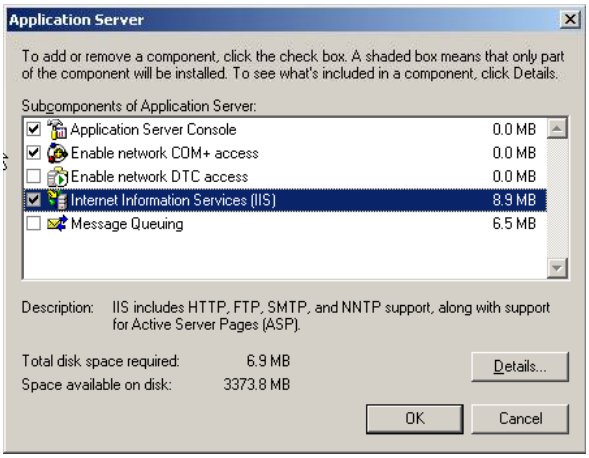
e. Verify that **ASP.NET v4.0.xx** and **All Unknown ISAPI Extensions** are Allowed:



2. Continue with the *configuration* (page 20) for PI Web Services on Windows Server 2003.

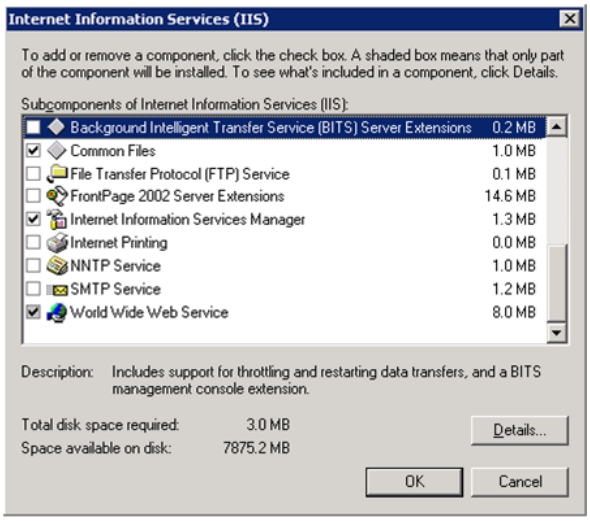
Configure IIS Application Server Settings

1. Choose **Start > Control Panel > Add or Remove Programs** and click **Add/Remove Windows Components**.
   a. In the **Windows Component Wizard**, under **Components**, select **Application Server** and click **Details**.
   b. In **Application Server** dialog:
   c. Verify that **ASP.NET** and **Internet Information Services (IIS)** are selected:



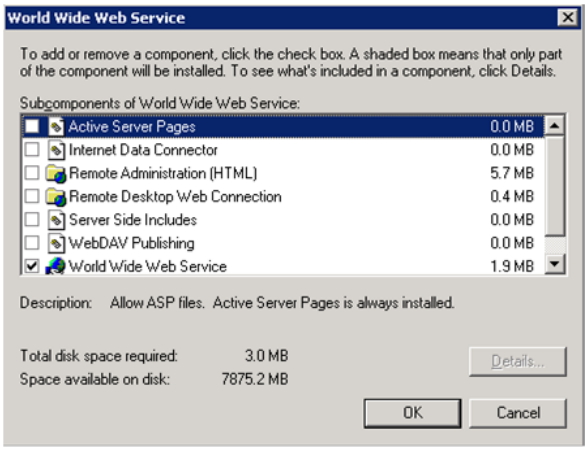
   d. Select **Internet Information Services** in **Application Server** and click **Details**.

e. Verify that **Internet Information Services Manager** and **World Wide Web Service** are selected in **Internet Information Services (IIS)**:



f. Select **World Wide Web Service** in **Internet Information Services (IIS)** and click **Details**.

g. Verify that **World Wide Web Service** is selected in **World Wide Web Service**:



h. Click **OK** in each of the three open dialogs.

2. Click Next in the **Windows Component Wizard**.
3. Click **Finish** in the **Windows Component Wizard**.
4. Run the PI Web Services *setup kit* (page 27).

Windows Server 2008 R2

These topics explain how to install and configure Internet Information Services (IIS) on Windows Server 2008 R2 for use with PI Web Services.

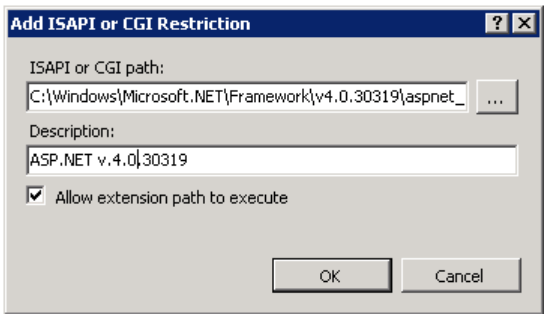
Review Role Services in IIS Server Manager

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:
   - ASP.NET
   - .NET Extensibility
   - ISAPI Extensions
   - ISAPI Filters
4. If these Role services are not installed, then add the IIS Web Server Role and Role Services and *enable an ISAPI extension for .NET 4* (page 22).

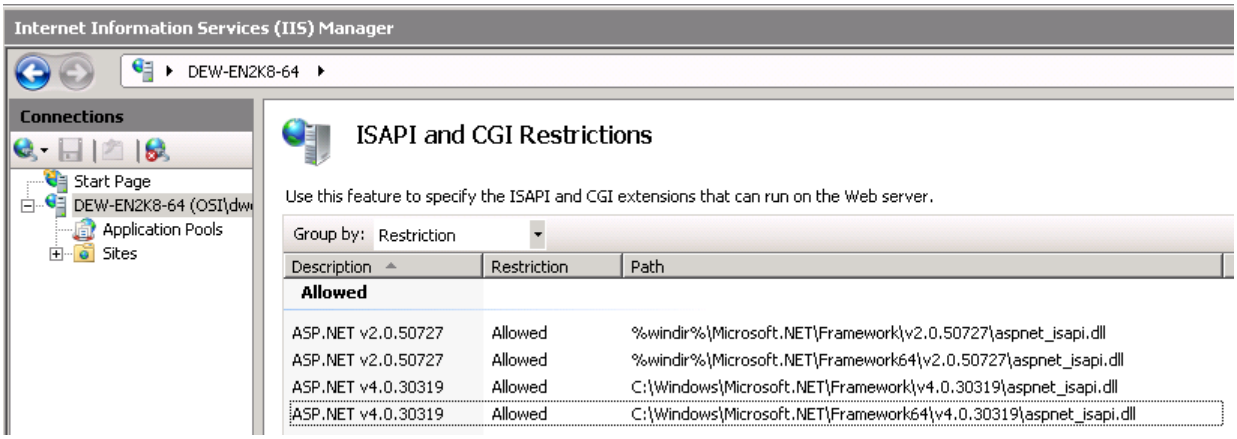
Review ASP.NET ISAPI extensions for .NET 4 Framework

ISAPI extensions must be enabled for .NET 4 Framework. To review these settings:

1. Open **Internet Information Services (IIS) Manager**.
2. In the **Connections** pane, click the server name.
3. In the **Home** pane, double-click **ISAPI and CGI Restrictions**.
4. Review the list in the ISAPI and CGI Restrictions dialog to determine whether it contains the file `aspnet_isapi.dll`. If it does not, add the file `aspnet_isapi.dll`:
5. In the **Actions** pane, click **Add ...**
   a. In the **Add ISAPI or CGI Restriction** dialog, enter the path to the binary you want to add in the ISAPI or CGI path box, the description of the binary in the Description box, and select **Allow extension path option to execute** check box to allow the binary to run on the server, and then click **OK:**



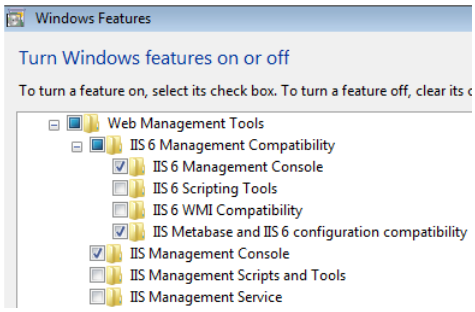
b. Right-click on the Web server in the **Connections** pane to verify that the .NET 4 Framework extension is included in the list of ISAPI extensions and that the restrictions are set to Allowed:



6. Run the PI Web Services *setup kit* (page 27).

Windows 7 and Windows Vista

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.



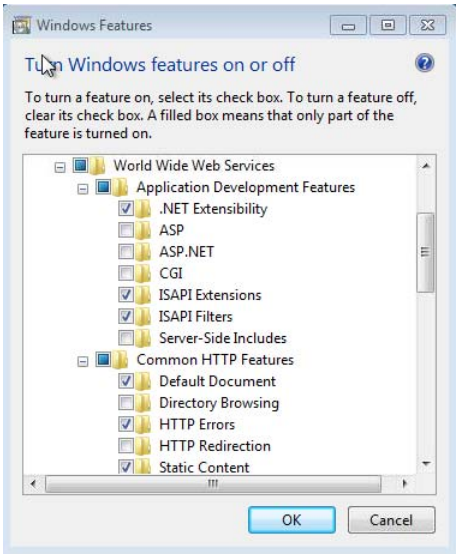
5. Click to expand **World Wide Web Services** and then:

   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**

   a. Click to expand **Common Http Features** and verify that these settings are selected:

   - **Default Document**
   - **HTTP Errors**
   - **Static Content**



   a. Click **OK**.

6. Verify that a .NET 4 Framework is enabled:

   a. In **Control Panel**:

   - For Windows 7, click **System and Security**, and then click **Administrative Tools**.
   - For Vista, click **System**, and then click **Administrative Tools**.

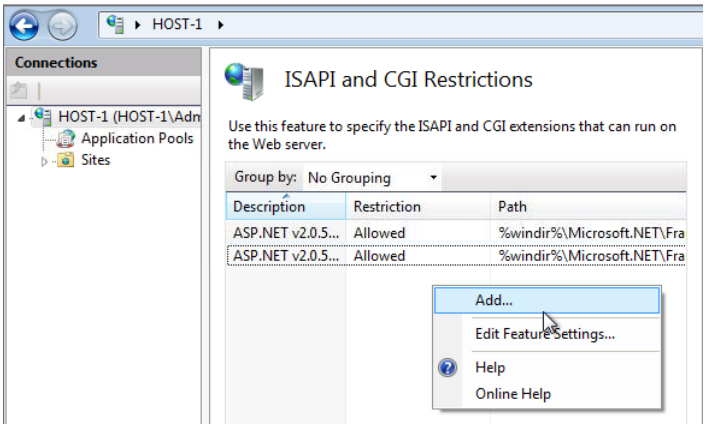
   b. Double-click **Computer Management**, and then click to expand **Services and Applications** and then click **Internet Information Services**.

   c. Double-click **ISAPI and CGI Restrictions**.

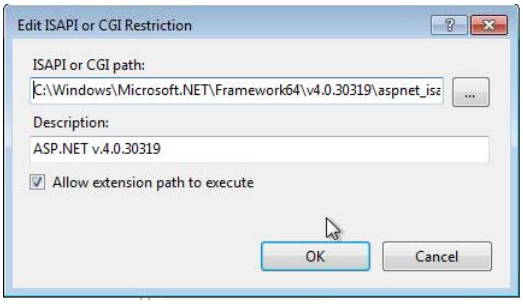
   d. Select the machine on which PI Web Services will be hosted.

   e. Review the **ISAPI and CGI Restrictions** and verify that ASP.NET v4.x is included.

f. If ASP.NET v4.x is not in the list, right-click **Add**:



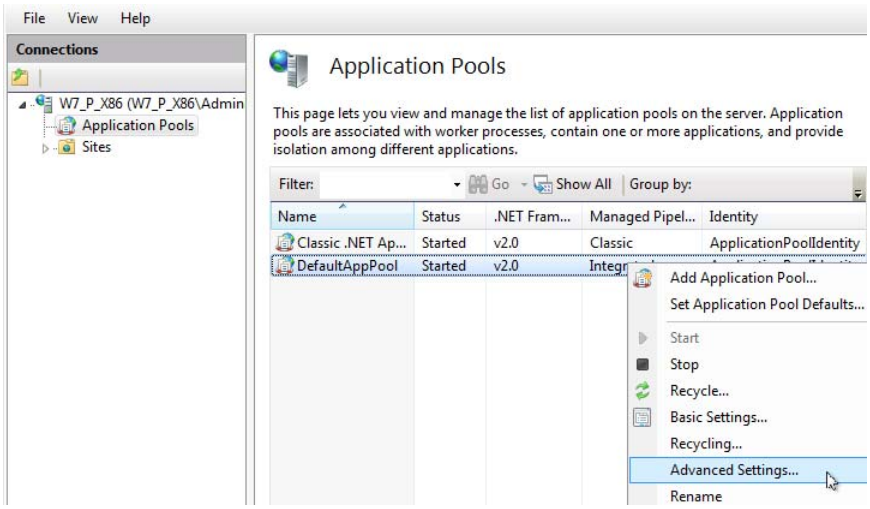
g. Enter the path to the Microsoft .NET directory containing the file `aspnet_isapi.dll`, a description and select **Allow extension path to execute**:



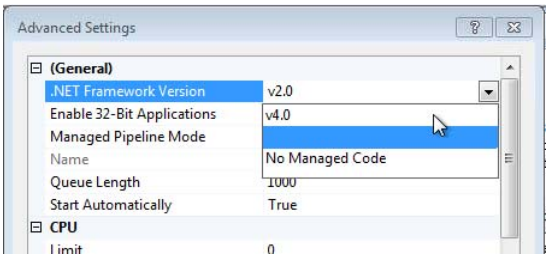
h. Click **OK**.

7. Select **Application Pools** under **Connections**, and review the **Identity** column to verify that the IIS Web server allows connections to .NET 4 Framework and that the *application pool identity* (page 51) is set to **NetworkService**.

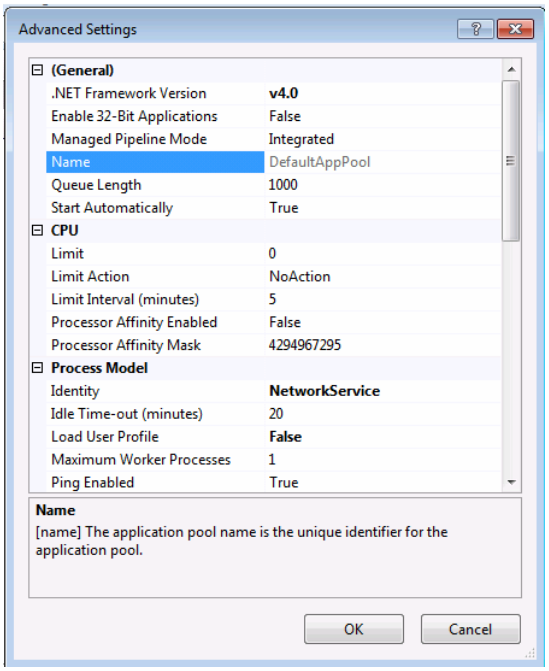
   a. To change the identity, right-click **DefaultAppPool** and select **Advanced Settings**:



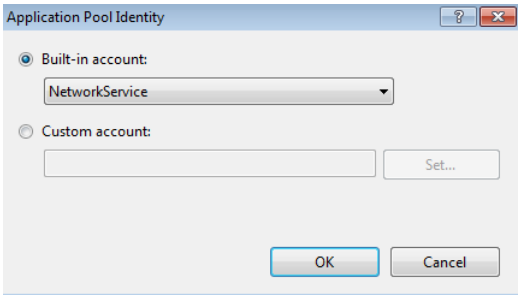
b. Use the drop-down to change **.NET Framework Version** to **v4.0**:



c. Select **Identity** and click **. . .**:

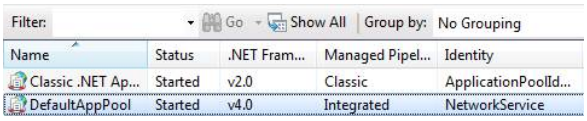


d. Select **NetworkService** from the **Built-in account** drop-down menu:



e. Click **OK** on both **Application Pool Identity** and **Advanced Settings** dialogs.

f. Verify that the **DefaultAppPool** setting is set to **NetworkService**:



8. Run the PI Web Services *setup kit* (page 27).

Run Setup Kit

If you upgrade from an earlier version of PI Web Services, the PI Web Services 2010 R3 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.

To install PI Web Services 2010 R3:

1. If you install PI Web Services on Microsoft Windows Vista, Windows Server 2008 R2, or Windows 7, verify that you are logged in as an administrator.
2. Verify that the server on which you plan to install PI Web Services includes:

- All *system requirements* (page 16), including *.NET Framework 4* (page 17)
- Microsoft Internet Information Services (IIS) on Windows Server configured as described in *Configure the Web Server* (page 18)

1. Run the PI Web Services setup kit:

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.

g. During the installation of PI Web Services, use the **Select Installation Address** dialog to select:
   - **Site:** A Web site location to host the Web site content and Web service files used by Internet Information Services (IIS) on Windows Server. By default, this file tree is stored in `[wwwroot]\PIWebServices (page 30).`
   - **Virtual directory:** A name for the IIS virtual directory. The default value is `PIWebServices`.
   - **Application Pool:** The Application Pool within the Internet Information Services Web server where PI Web Services runs.
   - **Installation directory:** If this is a new installation, 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.

Note: You can only choose a new installation directory if you install PI Web Services 2010 R3 for the first time. If you upgrade, the setup kit installs PI Web Services in the directory used for the earlier version.

After Installation

After PI Web Services is installed, you can:

- *Test the Web Server connection* (page 29).
- Verify that *PI Web Services files* (page 30) are installed.
- *Verify* (page 32) PI Web Services can retrieve PI System data.
- *Customize the setting* (page 37) that controls the size of messages sent and received between a Web client and Web server
- Review and update your *security settings* (page 38).
- Verify that your *firewall* (page 39) allows PI Web Services to communicate with your PI System.
- *Configure settings* (page 39) that determine whether PI data insertions and performance equations are allowed, and how data is displayed.

Test the Web Server Connection

Use a browser such as Windows Internet Explorer to verify your installation for data retrieval from PI Server and PI AF:

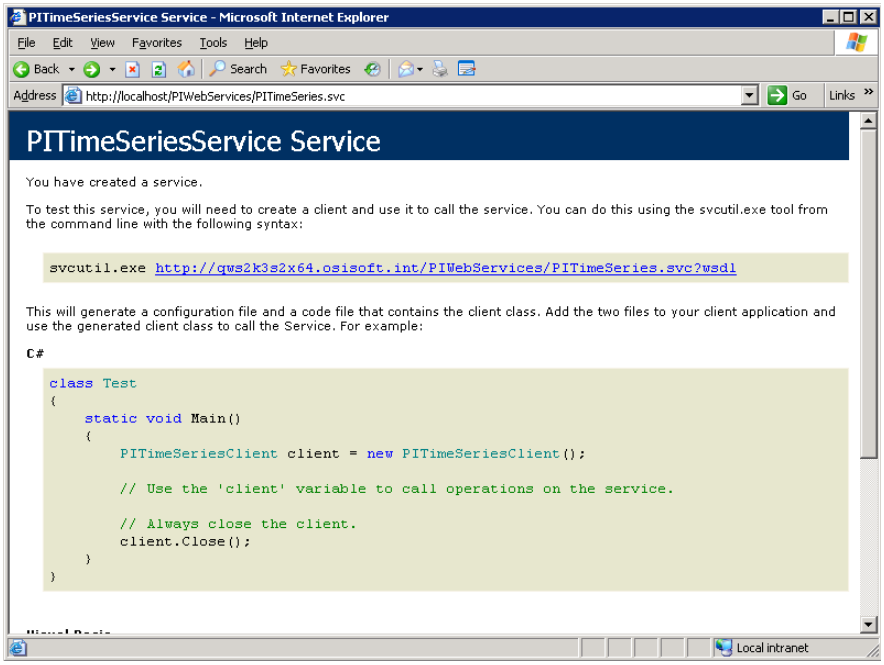
- If you used the default installation and Web site directories, enter:

  `http://localhost/PIWebServices/PITimeSeries.svc`

If you did not use the default installation and Web site directories, enter the path you used for the Web site relative URL:

- http://[servername]:[portnumber]/[PIWebServices directory name]/PITimeSeries.svc

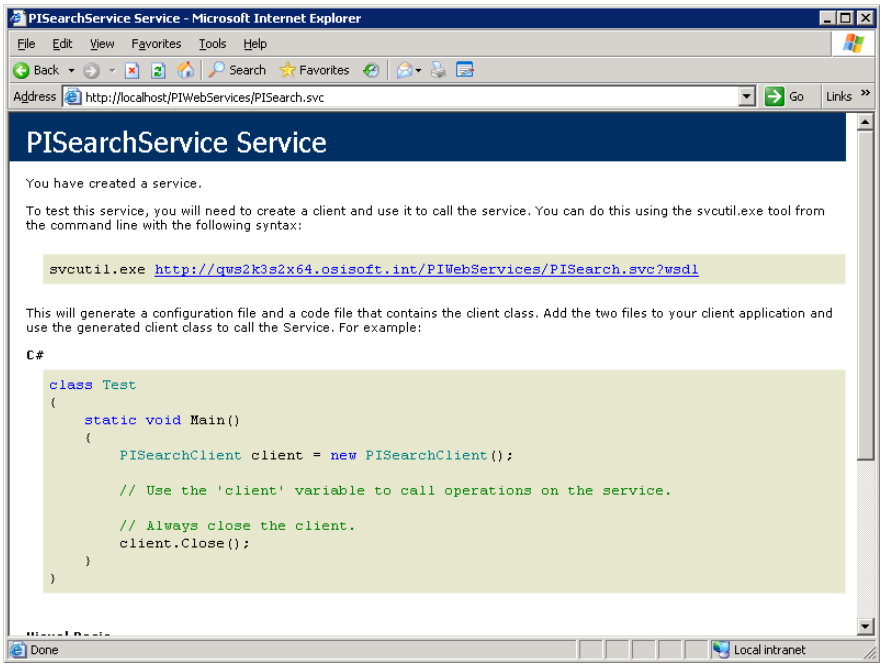
You should see this page:



To verify your installation for searches of PI Server data:

- If you used the default installation and Web site directories, enter:
  `http://localhost/PIWebServices/PISearch.svc`
- If you did not use the default installation and Web site directories, enter the path you used for the Web site relative URL:
  `http://[servername]/[PIWebServices directory name]/PISearch.svc`

You should see this page:



Find the PI Web Services File Directory

Locate the directory used for your installation and review its contents to confirm that the *PI Web Services files* (page 31) were installed correctly. This directory location will differ if:

- PI Web Services 2010 R3 is installed for the first time. Look in `[PIHOME]\PIPC\PI Web Services`.
- You upgrade to PI Web Services. Look in the Web server root directory for the Web site used to host the Web site content and service files used by Internet Information Services (IIS) on Windows Server. By default, this file tree is stored in `C:\inetpub\wwwroot\PIWebServices`. This Help refers to this directory as `[wwwroot]`.

Note: You can point to an existing Web site to host PI Web Services content and select the location of its root directory when you run the PI Web Services installer. The PI Web Services files are installed in the directory chosen during installation.

Locate PI Web Services Files

To 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* (page 30) contains:
  - `PIInstrumentation.config`
  - `PISearch.svc`
  - `PITimeSeries.svc`
  - `web.config`
- If you upgrade from an earlier version of PI Web Services, verify that:
  - The directory [`wwwroot]\PIWebServices\` contains:
    - `PITimeSeries.svc`
    - `PISearch.svc`
    - `web.config`
    - `old_web.config`
  - The directory [`wwwroot]\PIWebServices\bin` contains:
    - `PIWebServices.dll`
    - `PIWebServicesInstrumentation.dll`
    - `PIWebServicesInstrumentation.InstallState`
    - `OSIsoft.PIDataServices.Common.dll`
    - `OSIsoft.PIDataServices.Configuration.dll`
    - `OSIsoft.PIDataServices.DataAccess.dll`
    - `OSIsoft.PIDataServices.DataService.dll`
    - `OSIsoft.PIInstrumentation.dll`
    - `OSIsoft.PIInstrumentation.Listeners.dll`
- Verify that the *Process ID account* (page 51) has at least Read and Execute permissions on these directories.

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.

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 Web site http://www.wcfstorm.com/wcf/home.aspx*.

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

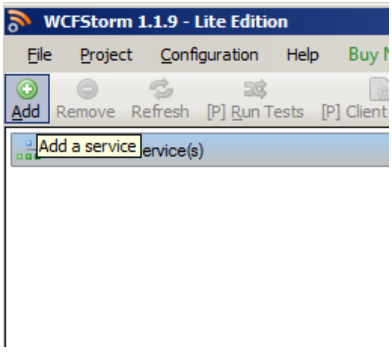
Download and Install WCFStorm

To download a trial version or shareware version of this tool, see the *WCFStorm Web site http://www.wcfstorm.com/wcf/home.aspx*.

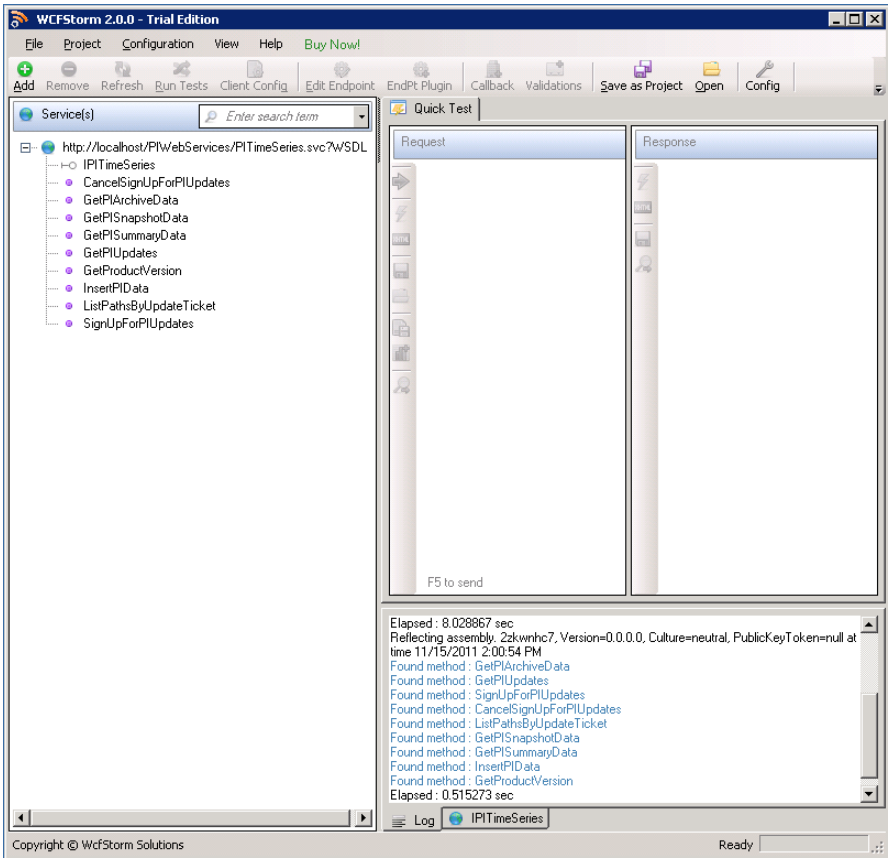
To install, extract the files to a local folder and launch `wcfstorm.exe`.

Configure WCFStorm for PI Web Services Access

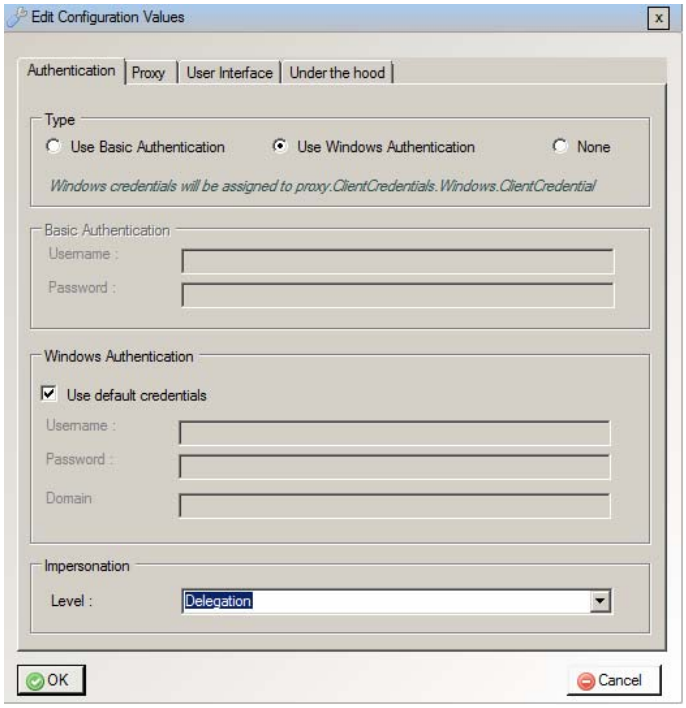
1. Open WCFStorm.
2. Enter a service endpoint:
   - If you are prompted to add a service endpoint, enter the appropriate URL. The endpoint must supply the metadata that the client will need to create a request for PI Web Services:
     - For the default installation, this endpoint is: `http://localhost/PIWebServices/PITimeSeries.svc?WSDL`.
     - 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.



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



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

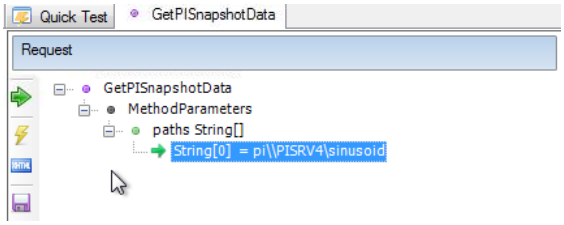


Note: PI Web Services requires the client to authorize delegation of the user's Windows identity. By default, WCFStorm authorizes impersonation but you must change the impersonation level used by WCFStorm to Delegation as described in Step 4.

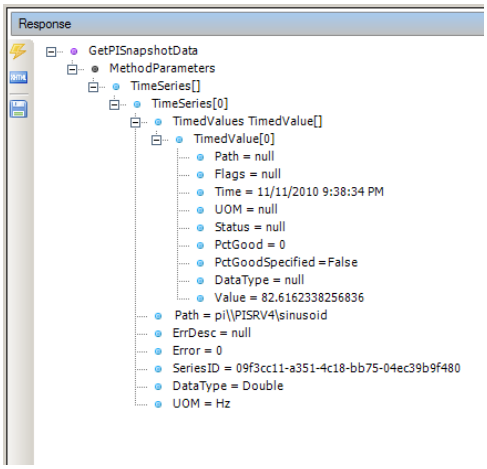
Create and Execute a Snapshot Request

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\\PISRV4\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 the **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.

Add .svc Handler Mapping to IIS

Internet Information Services (IIS) on Windows Server maintains a list of .NET assemblies, known as handlers, that are invoked when Web requests for specific file types are received. Under certain circumstances, IIS may lack a handler mapping needed by WCF.

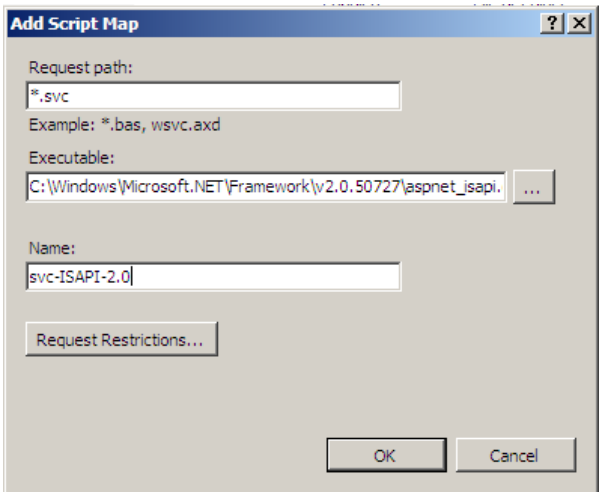
To verify that the handler exists and if necessary correct the handler in IIS 7.0:

Note: For other versions of IIS, see the MSDN article *How to: Configure an HTTP Handler Extension in IIS http://msdn.microsoft.com/en-us/library/bb515343.aspx*.

1. Open IIS Manager.
2. Expand the node for the Web server, expand Web Sites, then expand the Web site that is hosting PI Web Services.
3. Select the node for your application. For example, **PIWebServices**.
4. In the **Features View** pane, double-click **Handler Mappings**.
5. Inspect the **Handler Mappings** list and see if `*.svc` appears in the **Path** column. If `*.svc` does not appear, add a Module Mapping that associates files with the .svc extension with the ISAPI handler (`aspnet_isapi.dll`):
   a. In the **Actions** pane, click **Add Script Map**.
   b. Enter `*.svc` in the **Request path** field.
   c. In the **Executable** field, browse to the `aspnet_isapi.dll` file in `Windows\Microsoft.NET\Framework\v4.0.xx`

   Note: On 64-bit servers, the file directory is `Windows\Microsoft.NET\Framework64\v4.0.xx`.

   d. Add a name for the mapping to the **Name** field.



   e. Click **OK**.

Note: Alternatively, you can use the Microsoft command-line utility **serviceModelReg** to create the handler mapping. First, verify that the file `serviceModelReg.exe` is in `Windows\Microsoft.NET\v4.0.xx\Framework\` or `Windows\Microsoft.NET\v4.0.xx\Framework64\`.

Control Message Size

The WCF configuration parameter **maxReceivedMessageSize** allows you 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* (page 73) if PI Web Services is used to insert or retrieve large amounts of data.

This parameter is configured in the `web.config` (`http://msdn.microsoft.com/en-us/library/aa306178.aspx`) file located in the *PI Web Services installation directory* (page 30):

- If the computer was upgraded to PI Web Services 2011, see the Web application folder, `[wwwroot] (page 30)\PIWebServices`
- For computers on which PI Web Services 2001 is a new installation, see `[PIHOME]\PIPC\PI Web Services`.

To change this limit, *add the message size binding element* (page 37) to the `web.config` file and set the *appropriate message size* (page 38). If you are using .NET applications, set **maxReceivedMessageSize** in both the `app.config` and `web.config` files; in this case `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.

Each PI Web Services sample `web.config` files includes the **maxReceivedMessageSize** parameter set to the default value of **65,536 bytes (64 KB)**. If you are using one of these files, change the value of the existing parameter to set the *appropriate message size* (page 38).

Note: Do not set an arbitrary or inflated value for **maxReceivedMessageSize**. This setting protects against denial-of-service (DoS) attacks; if you set a really large value, you downgrade that protection. The impact of this setting on system security is described in this *MSDN article on security http://msdn.microsoft.com/en-us/library/ms733135.aspx*.

Add the Message Size Binding Element

This *wsHttpBinding binding* (page 55) 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>
```

Next, *set the maximum message size* (page 38).

Set Maximum Message Size

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 * number_of_paths) + (75 * number_of_values)
```

This formula is based on these characteristics:

- Size of a *GetPIArchiveData* (page 77) response message wrapper without any data: 364 bytes
- Size of a *TimeSeries* (page 92) element without any timed values and without error data: approximately 117 bytes
- Size of a *TimeSeries* (page 92) element. This can vary from the 117 byte value depending on the value of **DataType**, 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* (page 77), the size of a single *TimedValue* (page 94) element ranges from 69 to 77 bytes for point values of the Float data type. *GetPISummaryData* (page 79) 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* (page 77) and *GetPISummaryData* (page 79) are small in comparison to the response messages for these methods. By contrast, response messages are dominated by *TimeSeries* (page 92) and *TimedValue* (page 94) elements and these elements are also used in both the request and response messages for *InsertPIData* (page 80). As a result, although the recommended formula focuses on the response message for *GetPIArchiveData* (page 77), it is representative of the response message for *GetPISummaryData* (page 79), and both the request and response messages for *InsertPIData* (page 80).

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* (page 47).

Review 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* (page 68) 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* (page 69).

Control PI Web Services Features

PI Web Services includes three custom properties you can use to control these client application features:

- *Data value displays* (page 40)
- *Performance equation execution* (page 40)
- *PI System data insertions* (page 40)
- *Duration of update sign ups* (page 40)

The settings described here are found in the **PIWebServiceSettings** element of the `web.config` file:

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

The `web.config` (`http://msdn.microsoft.com/en-us/library/aa306178.aspx`) file is found in the appropriate Web application folder, depending on the location of the *PI Web Services installation files* (page 30).

Allow or Disallow Performance Equation Calls

To control whether users can execute Performance Equations, use the **AllowCalculations** parameter in the `web.config` (page 39) file:

- To prevent users from executing Performance Equations, set **AllowCalculations=False** in the web.config file on the Web server.
- By default **AllowCalculations** is set to **TRUE**, which allows PE calculations.

Allow or Disallow Insertions of Data to a PI System

To control whether users can insert data to a PI Server or AF Server, use the **AllowDataEntry** parameter in the `web.config` (page 39) file:

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

Configure Data Display Returns

To control how many decimal digits are represented in the data returned by PI Web Service calls, use the **FloatPrecision** parameter in the `web.config` (page 39) file:

- To display all digits supported by the double-precision floating point data type, set **FloatPrecision=Full**. This is the default setting.
- To display the digits as configured by the **DisplayDigits** attribute of the PI point associated with the data, set **FloatPrecision=DisplayDigits**.

Configure Duration of Update Sign Ups

To specify how long, in minutes, an update sign-up lasts before it is removed from the list of sign-ups if it is not accessed, use the **UpdatePurgeInterval** parameter in the `web.config` (page 39) file. The default value is 5 minutes.

Host PI Web Services with a Windows service

PI Web Services 2010 R3 Standalone Edition provides a Windows managed service as an alternative for hosting PI Web Services in Internet Information Services (IIS). This edition is useful for environments with security restrictions that prohibit the use of Internet Information Services. 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 2010 R3 Standalone Edition is bundled with the PI Web Services 2010 R3 setup kit and requires that the following be installed on the server that you use to install PI Web Services:

- 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 Vista, Windows 7
- .NET Framework 4. To download, go to http://www.microsoft.com/net.

The computer must also have network access to one or more of these OSIsoft servers:

- PI Server 3.3 or later
- PI AF Server 2.1 or later that includes *PI SQL for AF Server* (page 17)

See *Configure Security for PI Web Services Standalone Edition* (page 42) and *Impersonation and delegation* (page 57) for information about how to configure security settings.

Install PI Web Services Standalone Edition

To install PI Web Services 2010 R3 Standalone Edition:

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.

g. During the installation of PI Web Services, use the **Select Installation Address** dialog to select:

   - **Site:** A Web site location to host the Web site content and Web service files used by Internet Information Services (IIS) on Windows Server. By default, this file tree is stored in `[wwwroot]\PIWebServices (page 30).`
   - **Virtual directory:** A name for the IIS virtual directory. The default value is `PIWebServices`.
   - **Application Pool:** The Application Pool within the Internet Information Services Web server where PI Web Services runs.
   - **Installation directory:** You have the option to select a path to an installation folder for PI Web Services.

6. Next, *configure security* (page 42) for the standalone edition of PI Web Services.

Configure Security for PI Web Services Standalone Edition

When you host PI Web Services with a Windows service, you must reserve a namespace to instruct the operating system to route HTTP traffic for a specific URL to the service. To do so, you must use a command-line tool provided by Microsoft to invoke the HTTP Server API. For complete details, see *Configuring HTTP and HTTPS* (*http://msdn.microsoft.com/en-us/library/ms733768.aspx*).

For information about how to secure connections between a PI System and a PI Web Services client, see *PI Web Services Security* (page 47).

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:8000/PIWebServices /a
D:(A;;GA;;;NS)
```

For details on the SDDL format, see *Security Descriptor String Format* (*http://msdn.microsoft.com/en-us/library/aa379570(v=VS.85).aspx*).

Reserve Namespace on Windows 2008 R2 Server, Windows 7 or Vista

The utility for Windows Vista, Windows 7, and Windows Server 2008 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:8000/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.

If the Service Does Not Start

If sufficient rights have not been reserved through the *security configuration* (page 42), the service host will not operate properly. When you try to start the service, it will start and immediately shut down. Entries in the Windows event log do not mention a security issue. The error message is:

```
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
```

To resolve this error, adding appropriate permissions through a `urlacl` reservation as described in *Configure Security for PI Web Services Standalone Edition* (page 42).

Verify Standalone Edition Installation

To verify that PI Web Services 2010 R3 Standalone Edition was installed successfully:

- If you install PI Web Services for the first time, verify that the *PI Web Services file directory* (page 30) 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
  - PIWebServicesIInstrumentation.dll
  - PIWebServicesIInstrumentation.InstallState
  - PIWebSvcHost.exe
  - PIWebSvcHost.exe.config
  - PIWebSvcHost.InstallState

To verify data access:

1. Start the PI Web Services service.
2. Open the app.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 Web site directories, enter:

   ```
   http://server:8000/PIWebServices/TimeSeries
   ```

   If you did not use the default installation and Web site directories, enter the path you used for the Web site relative URL:

   - `http://[servername]:[portnumber]/[virtual directory name]/TimeSeries`
4. To verify your installation for searches of PI Server data:
   - If you used the default installation and Web site directories, enter:

   ```
   http://server:8000/PIWebServices/Search
   ```

   - If you did not use the default installation and Web site directories, enter the path you used for the Web site relative URL:

   ```
   http://[servername]:[portnumber]/[PIWebServices directory name]/Search
   ```

Silent Installation of PI Web Services

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

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. Edit line 5 to replace **`MyServer`** with the name of the default PI Server and **`pidemo`** with the name of the default user:

      ```
      5 = /qn PI_SERVER=MyServer PI_TYPE=3 PI_USER=pidemo ALLUSERS=1
      REBOOT=Suppress NOPISDKBUFFERING=1
      ```

   b. 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. Comment out this line:

      ```
      7 = /qn ALLUSERS=1 REBOOT=Suppress PI_SERVER=MyServer
      PI_USER=pidemo REBOOT=Suppress
      ```

   Note: The 64-bit version of PI SDK will use the information from line 5, so you do not need to include this line.

   b. 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).
   c. Uncomment this line to point to the KST_INI_FILE that you create:
      `7 = /qn IMPORT_KST=1 KST_INI_FILE="C:full path\file.ini" ALLUSERS=1 REBOOT=Suppress`
5. To change the default values of the target Web site, application pool, virtual directory and PI Web Services directory in Internet Information Services (IIS), edit this line:

   ```
   10 = /qn TARGETSITE=/LM/W3SVC/1 TARGETAPPPOOL="ASP.NET 4.0
   DefaultAppPool" TARGETVDIR=PIWebServices
   TARGETDIR="c:\Program Files\PIPC\PI Web Services"
   ```

6. Open the folder that contains the extracted files and rename the file `setup.ini` to another name such as `setup_old.ini`.
7. Rename the file `setup_silent.ini`. to `setup.ini`.
8. 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.

Chapter 3

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* (page 51)
- A Web server that uses a *secure user account* (page 51)
- *Firewall security* (page 68) that enables PI Web Services to communicate with your PI System
- *Security for the Web service bindings* (page 53)
- *Impersonation and delegation* (page 57)

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 *Windows Communication Foundation Security http://msdn.microsoft.com/en-us/library/ms732362.aspx*. For specific guidelines on configuration file syntax, see *Windows Communication Foundation Configuration Schema http://msdn.microsoft.com/en-us/library/ms731354(v=VS.100).aspx*.

Supported Security Scenarios

PI Web Services 2010 R3 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

Secure Access to PI Server Data

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

- *PI identities and mappings* (page 49) used for *Windows-integrated security* (page 48)
- *PI trusts* (page 49) (PI Server 3.3 and later)
- *Database security* (page 50) settings for the PI Server, PI modules and PI points

OSIsoft recommends that you use *Windows-integrated security* (page 48), 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 Web site* (*http://techsupport.osisoft.com*).

Windows-integrated Security

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

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* (page 49).

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* (page 51) 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 Tool (PI SMT) 3.3.1.3** or higher.
- 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* (page 6).

Note: OSIsoft recommends that you map both the *User ID and the Process ID accounts* (page 51) 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* (*http://techsupport.osisoft.com*).

PI Trusts

To provide access to pre-approved network entities on PI Servers earlier than 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* (page 49).

For further details, see *Configuring PI System Security* and the PI SMT Help, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com*).

Set Required 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* (page 6) uses it to store configuration information.

Note: Additional permissions are required to run the *setup kit* (page 27).

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.

Secure Access to Data 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* (page 51) under which Internet Information Services (IIS) on Windows Server 2003 runs 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* (page 52).
- Ensure that the *required user accounts* (page 51) have read access to AF Server.

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

Configure Required User Accounts

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

- **Process ID –** the Windows account under which the Internet Information Services (IIS) on Windows Server application pool runs
- **User ID** – the Windows account of the client calling the PI Web Services

PI Web Services uses the **Process ID** to connect to and process updated from the PI System through *PI Data Services* (page 6). 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.

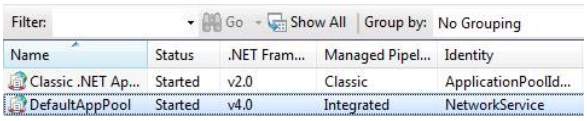
To ensure that both of these accounts allow PI Web Services to connect with a PI System:

- PI Web Services will use a secure configuration if the IIS Web server uses *Network Service for its application pool identity* (page 52).
- *Verify your identity connections* (page 52). If the IIS Web server uses a domain account other than Network Service, you must configure *Kerberos authentication* (page 52).
- *Connections that use Windows authentication* (page 52) must meet additional requirements.
- *End user folder and registry permissions* (page 71) must be set to allow access to data stored in the PI Server, PI Module Database (MDB) and PI point database.

Verify Identity Connections

To identify the **User ID** and **Process ID** used in a given session:

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



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

2. Use **PI SMT** to inspect the PI identity to which the Windows account is mapped.
3. To verify that these identities will allow PI Web Services connections, see *Connections that use Windows Authentication* (page 52), or *Connections that Kerberos Authentication* (page 52).

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 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 PI Web Services to use Kerberos Authentication* (page 111).

End User Folder and Registry Permissions

End user groups and the application pool identity require the following folder and registry permissions on the Web server:

Location	User Account	Permissions	Notes
Web site folder and subfolders	User ID	Read & Execute, Read	The Web site folder which is typically stored in `[wwwroot] (page 30)\PIWebServices`
Web site folder and subfolders	Process ID	Full Control	The Web site folder which is typically stored in `[wwwroot] (page 30)\PIWebServices`
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.

Configure Secure Web Service Access

Security settings and other details that specify how data is transported and encoded are found in the *XML element bindings* (page 54) within the `web.config`. file found in the *PI Web Services file directory* (page 30). 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**, and **TransportWithMessageControl**.
- 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* (page 61).

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* (page 56) to ensure data exchanges are successful.

These security settings are preconfigured in the *sample configuration files* (page 62) 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* (page 62) 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* (page 55)
- *netTcpBinding* (page 55)
- *netNamedPipeBinding* (page 55)

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* (page 54).

After you select the binding type, you can *configure security for the Web service bindings* (page 55).

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 http://msdn.microsoft.com/en-us/library/ms731172.aspx* and *Configuring System-Provided Bindings http://msdn.microsoft.com/en-us/library/ms731092.aspx*.

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* (page 99), 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* (page 57).

PI Web Services includes a *basicHttpBinding sample* (page 63) 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* (page 52) 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* (page 58) by default, so only minimal configuration changes are required.

If you are using *Windows authentication* (page 52), and want to use *impersonation and delegation* (page 58), this binding requires minimal edits. PI Web Services includes a *sample wsHttpBinding* (page 64) 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* (page 62) 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* (page 67) that you can use with minimal configuration. If you use this *netNamedBinding sample* (page 62) as is, the netNamedPipeBinding will perform Windows authentication but does not encrypt the data in transit.

Configure Security for Web Service Bindings

PI Web Services provides *sample configuration files* (page 62) 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* (page 30). If your IIS server is set up as described in *Configure the Web Server* (page 18), 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 http://msdn.microsoft.com/en-us/library/ms732362.aspx.*

Note: There is no administration tool provided by the operating system to manage WCF settings found in `web.config` (`http://msdn.microsoft.com/en-us/library/aa306178.aspx`). You can use a text editor or *WCF Service Configuration Editor* (page 68) to edit these files.

Change the 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 3 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* (page 56). Next, locate and edit its *corresponding configuration* (page 57).

Edit the Endpoint Binding

In PI Web Services, the `<services>` element will have a single XML child element called `<service>`. This element will in turn have at least 2 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>
```

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* (page 13) to certain newer client applications.

Edit the Active Binding Configuration

The active `<endpoint>` will have an XML attribute called binding and might have an additional XML attribute called `bindingConfiguration`. To find the actual binding configuration settings, look in the `<bindings>` element within `<system.serviceModel>`. You should find an XML child element whose name matches the binding value of the active `<endpoint>` element. In the above example, this is `<wsHttpBinding>`. 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>`.

In this example from the default *wsHttpBinding* (page 64) `web.config` file, the beginning of the XML element `<bindings>` looks like this:

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

For more information on configuring WCF bindings, see *Windows Communication Foundation Configuration Schema* (*http://msdn.microsoft.com/en-us/library/ms731354(v=VS.100).aspx*).

Impersonation and Delegation

WCF settings included in the *Web service security bindings* (page 58) 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 identify 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* (page 51) 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* (page 51).

For further information regarding the configuration of WCF security, see the MSDN article *Delegation and Impersonation with WCF http://msdn.microsoft.com/en-us/library/ms730088.aspx*.

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* (page 63).
- Clients that support profiles above the WS-I Basic Profile, i.e., clients that support WS-Security, should use the *wsHttpBinding* (page 64). 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* (page 65), or *netNamedPipeBinding* (page 67).

Note: To use the *netTcpBinding* (page 65) with PI Web Services, you must be running Internet Information Services (IIS) on Windows Server 7 or later.

Enable or Disable Impersonation

Impersonation is enabled by default in the *sample binding files* (page 62) 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 *InfoPath* (page 99). 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* (page 51) 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 http://msdn.microsoft.com/en-us/library/ff649178.aspx*.

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 .NET Clients

You can generate an `app.config` file with almost all of the necessary WCF configuration 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* (page 13) 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:

1. Generate a Web service reference as described in this MSDN article: *How to: Generate a Web Service Proxy http://msdn.microsoft.com/en-us/library/bbs97dkt(VS.90).aspx*.
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>
```

Note: Name the behavior as desired. This example uses the name `wsImpersonationBehavior`.

b. 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">
```

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* (page 54).

You can use a text editor or *WCF Service Configuration Editor* (page 68) to edit these files.

Sample Configuration Files

The directory `[WWWROOT]\PIWebServices\Bin\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.

Sample files included are:

File Name	Contents
`web_all_bindings_basic.config`	Includes all *binding types* (page 54) used by PI Web Services. The service endpoints are configured to use the basitHttpBinding with no security.
`web_config_basic_no_security.config`	A *basicHttp binding file* (page 54) that includes no active security options. Use this binding type for best interoperability with non-Windows platforms.
`web_config_netTcp.config`	A *netTcp binding file* (page 55). The sample's default setting is secure and includes additional security options.
`web_config_wsHttp.config`	A *wsHttp binding file* (page 55) suitable for cross-machine communication on an intranet between WCF clients and WCF Web servers.
`web_config_netNamedPipe.config`	A *netNamedPipe binding* (page 55) file suitable for high speed communications between processes on the same computer.

basicHttpBinding Sample

The *sample basicHttpBinding* (page 62) web.config provided has the setting **<security mode="none">**. Use this binding when it is required to communicate with SOAP 1.1 clients, such as *InfoPath* (page 99).

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 **None** security 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"
x509FindType="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 thumbprint of a certificate. For more information about the certificates used in this security setting, see this MSDN article on *service credential http://msdn.microsoft.com/en-us/library/ms731340.aspxcertificates* (*http://msdn.microsoft.com/en-us/library/ms731340.aspx*).

---

wsHttpBinding Sample

To configure the wsHttpBinding sample to provide a secure connection suitable for user impersonation and delegation over the HTTP protocol:

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:
   a. The binding attribute denotes that wsHttpBinding is the **binding** type.
   b. The bindingConfiguration attribute names the binding configuration created in Step 1.

c. 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>
```

d. 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>
```

netTcpBinding Sample

Note: 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.

To configure the sample netTcpBinding with PI Web Services hosted under IIS:

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* (page 58) for all operations.

If you are using IIS 7.0, you can use **IIS Manager** to configure the netTcpBinding:

1. Select the Web site that hosts PI Web Services.
2. Right-click and select **Edit Binding... Add netTcp**.
3. Add the port you want to use for netTcp and *, for example, **"808:*"**.
4. 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.

netNamedPipeBinding Sample

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.

To configure the Web site hosting PI Web Services to listen for netNamedPipe requests:

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/PIDataServ
   ice" 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.

If you are using IIS 7.0, you can use **IIS Manager** to configure netNamedPipes:

1. Right-click the Web site hosting PI Web Services and select **Edit Binding...**.
2. Add **net.pipe**, then ***** for the binding information.
3. Under the PI Web Services application, right-click the application, then select **Advanced Settings...**.
4. Under the Behavior section, ensure **net.pipe** 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,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) http://msdn.microsoft.com/en-us/library/ms732009.aspx*, 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* (page 69).

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

Configure Firewall Exceptions

This topic describes firewall settings for the servers hosting PI Web Services and your PI Server or AF Server.

Because of the underlying technologies that *PI Data Services* (page 6) uses to communicate with PI System servers, some ports must be open on the firewall to allow incoming TCP connections.

To use a firewall that blocks unauthorized access, but allows PI Web Services to communicate with your data servers, OSIsoft recommends these firewall exceptions:

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:
	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* (*http://techsupport.osisoft.com*).

Example: Configure Exceptions on Windows Server 2008 SP2

To configure firewall exceptions on Microsoft Windows Server 2008 R2:

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**.

For more details, see the Microsoft Web page *Firewall: frequently asked questions*:
`http://www.microsoft.com/windowsxp/using/security/internet/sp2_wfexceptions.mspx`

Chapter 4

Troubleshooting

Procedures 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 listed in *After Installation* (page 28).

If you use the default installation of PI Web Services, you will find errors in the Windows Application Log on the server hosting the Web service.

PI Web Services 2010 R3 offers further *Error and Trace Message Logging* (page 9) 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 *Logging and Instrumentation* (page 119).

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.

IIS is Not Running

This error message, if received when a user enters the URL of the Web service into the Internet Explorer browser:

```
Could not connect to
"http://<PIWebServiceHostMachineName>/PIWebServices/PITimeSeri
es.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>.
```

indicates that Internet Information Services (IIS) on Windows Server is not running. If you receive this message:

- Verify that IIS is running, or
- If you use the standalone version, verify that the Windows service is started.

Server Not Found

A SOAP fault with this message when a Web Services call is sent:

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

indicates that the Web Services call includes a *path* (page 7) that contains the name of PI Server that is not in the Known Servers Table (KST), although that server might be running.

To correct this error:

- On the Web server, use **PI SDK Utility** to verify that the target PI Server exists in the Known Servers Table (KST). If it does not, add it. See the **PI SDK Utility** Help.

Note: This message indicates that the PI SDK feature to automatically add servers to the KST is *disabled*. To avoid this error in the future, you can enable this feature, as described in the **PI SDK Utility** Help.

PI System Not Found

A SOAP fault with this message when a Web Services call is sent:

```
PI System not found.
```

indicates that the AF Server is not running. Verify that the AF Server included in the *path* (page 7) of the Web services call is running if you receive this message.

Remote Connection Failure

A SOAP fault with this message when a call or value is passed:

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

can indicate:

- A problem with the PI Server connection
- A PI Server that is not running
- A <Server> name in the *path* (page 7) that is not a valid PI Server
- Impersonation is not properly configured. See *Configure Security for .NET Clients* (page 61).

Data Entry Disallowed

This error, when received with attempts to use *InsertPIData* (page 80), where the identity of the calling user is reflected in <domain>\<user>:

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

indicates that the *InsertPIData method* (page 80) is not enabled.

To correct this error, enable the InsertPIData method if you receive in this message. Review the **AllowDataEntry** key in the PIWebServiceSettings section of `web.config` and set it to **TRUE**.

Insufficient Permissions

Users of PI Web Services must have write access to the PI points into which they attempt to insert data. If the user has no write access to the PI point included in the path of the InsertPIData call, PI data insertions will fail.

Failed events are flagged in the *TimedValue's* (page 94) Status property. To determine which insertions failed, examine the time and the path associated with *TimedValue* (page 94) Status property of:

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

Insufficient Message Size

This SOAP fault with the message is generated when returned data exceeds the size limit that the client is prepared to accept:

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

To *increase the maximum message quota* (page 38) on the Web server, set the **maxReceivedMessageSize** property in the appropriate binding element of the `web.config` file. For a WCF client, set **maxReceivedMessageSize** property `app.config`.

If the property does not appear in the `web.config` or `app.config` files, the default message size is **65,536** bytes. See *Control Message Size* (page 37) for details about how to estimate and configure message sizes.

Execution of Performance Equations Disallowed

If this error occurs when attempts to retrieve data through a call that contains a **pe** designator, where the `path =` string shows the path that the client passed to the Web service:

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

Complete these steps on the Web server to resolve this error:

1. Use the `path =` string in the error message to identify which request failed.
2. To enable this setting, set **AllowCalculations** to **TRUE** in the PIWebServiceSettings section of `web.config`.

Filters or Parameters Not Supported

Some Web method parameter values, or combinations of parameter values, are not supported at this time. If any of these situations are detected, PI Web Services will return a SOAP fault containing a message similar to:

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

If you receive a message like this, review the setting of the named parameter and use this table to identify the parameter values or combinations of parameter values that are not supported:

Source Designator	Operation	Parameter	Unsupported Value
pi	GetPIArchiveData	PIArcManner.Filter and RetrievalType	Any non-empty filter value for a RetrievalType of PlotValues
af	GetPIArchiveData	PIArcManner.Filter and RetrievalType	Any non-empty filter value for a RetrievalType of PlotValues
pe	GetPIArchiveData	PIArcManner.Filter	Any non-empty filter value
pi, pe, af	GetPISummaryData	PISummaryManner.Filter	Any non-empty filter value
pi, pe, af	GetPIArchiveData, GetPISummaryData	TimeRange	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 PE calculations
pi	InsertPIData	events	Attempting this operation with no events
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* (page 7)

Server Error in PI Web Services Application

If you receive this message in the Internet Explorer browser when you try to *test the Web Server connection* (page 29):

```
Server Error in '/PIWebServices' Application
```

Review your system account to see if its permissions on the directory specified in the error message have been restricted if you get this message. The directory must have read and write access for the *Process ID account* (page 51).

The folder **Temporary ASP.NET Files**, typically found in `C:\Windows\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.

You can also use the **aspnet_regiis.exe** utility to access these permissions. The utility can be found in this directory: `C:\Windows\Microsoft.NET\Framework\v4.0.xx`.

Note: The version of .NET 4 Framework utility can be `v4.0.30319` or later.

The command line is:

```
aspnet_regiis -ga "NT AUTHORITY\NETWORK SERVICE"
```

Invalid Tag Name

```
2147220432 - Invalid tag name.
```

An invalid tag name error appears if you do not enter correct syntax for the *path* (page 7) that designates the source of your PI System data.

Note: This error also appears when programming languages that use the backslash character (\) to represent special characters are used, as with C#. To correct this, use two backslashes to represent one backslash. For example, use `pi:\\\\piserver1\\sinusoid` to represent the path `pi:\\piserver1\sinusoid`.

Chapter 5

PI Web Services Programmer Reference

The PI Web Services Programmer Reference contains information required to develop applications that consume data from the 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* (page 77) based on the type of PI System data returned. Each interface contains the individual Web methods.

PI Web Services Interfaces

PI Web Services exposes two interfaces that provide Web methods and classes that allow you to build Web service applications that access PI System data:

- Use the *IPITimeSeries* (page 77) interface to retrieve time series data from a PI Server or PI Asset Framework, or insert time series data into a PI Server.
- Use the *IPISearch* (page 96) interface to search for PI points that are stored in a PI Server.

IPITimeSeries Interface

The IPITimeSeries interface contains methods that retrieve PI System data as collections of *TimeSeries* (page 92) objects. *TimeSeries* (page 92) data consists of time stamped data values as stored in a PI Server. *TimeSeries* (page 92) data are identified by *paths* (page 7) to PI points, performance equations or PI AF attributes.

IPITimeSeries Web Methods

GetPIArchiveData Method

GetPIArchiveData retrieves data from the PI Server as Compressed, Interpolated and Plot Values, and returns the data as an array of *TimeSeries* (page 92) 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. To sign up group of paths at one time and receive a single update ticket for the group, see *SignUpForPIUpdates* (page 83).

Syntax

```
TimeSeries (page 92)[] GetPIArchiveData(PIArcDataRequest (page
87)[] requests)
```

Arguments

Array of *PIArcDataRequest* (page 87) objects

Returns

Array of *TimeSeries* (page 92)

The returned *TimeSeries* (page 92) array has the same number of elements as the input *PIArcDataRequest* (page 87) 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.

PI Web Services reports these error codes for updates:

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 relative	4001	The end time of the request is not relative 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.

Remarks

Note: In addition to the error codes listed above, error values returned may range from **1000** to **30xx**; negative values indicate a PI SDK error.

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* (page 92) object returns an empty *TimedValue* (page 94) array. If query *paths* (page 7) 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* (page 87) object of the *PIArcDataRequest* (page 87) to **Outside**.

This is important for queries of snapshot values that specify a *TimeRange* (page 91) for the current time; these queries must specify **StartTime=*** and **EndTime=*** since the snapshot value is usually before or at current time.

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 *PIArcManner* (page 87) object is set to **TRUE**.

When the registration succeeds, the SeriesID property of the returned *TimeSeries* (page 92) object for that path is a string representation of the ticket that can be used with *GetPIUpdates* (page 84), *ListPathsByUpdateTicket* (page 86) and *CancelPIUpdates* (page 84). Each path registered will receive its own unique ticket in the SeriesID of the *TimeSeries* (page 92) object returned.

Client applications can sign up for updates from multiple paths in a group operation if you register the paths using *SignUpForPIUpdates* (page 83).

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 point paths to receive data updates if the Updates property of the *PISummaryManner* (page 89) 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* (page 83).

Syntax

```
TimeSeries (page 92)[] GetPISummaryData(PISummaryDataRequest (page
89)[] requests);
```

Arguments

Array of *PISummaryDataRequest* (page 89) objects

Returns

Array of *TimeSeries* (page 92) 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.

Data retrievals that fail for a *PISummaryDataRequest* (page 89) 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 relative`	4001	The end time of the request is not relative 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.

Remarks

Each path for which Updates is **TRUE** is signed up for updates separately and receives its own unique update ticket in the SeriesID property of the returned *TimeSeries* (page 92) object.

When the registration succeeds, the SeriesID property of the returned *TimeSeries* (page 92) object for that path is a string representation of an update ticket that may be used with *GetPIUpdates* (page 84), *ListPathsByUpdateTicket* (page 86) and *CancelPIUpdates* (page 84). 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.

GetPISnapshotData Method

GetPISnapShotData returns the current snapshot for each *path* (page 7) passed to this method.

Syntax

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

Errors

If an error occurs when data is retrieved for an individual path, PI Web Services reports the error code in the Error property of the returned *TimeSeries* (page 92) object and describes the error in the ErrDesc property.

Remarks

The returned *TimeSeries* (page 92) array has the same number of elements as the input string array.

This method is equivalent to *GetPIArchiveData* (page 77) with a *PIArcMannerRetrivalType* (page 88) of **Compressed** and *PIArcMannerBoundaries* (page 88) of **Outside** for the *TimeRange* (page 91) 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* (page 92) array contains a *TimedValue* (page 94) collection with a single *TimedValue* (page 94) object that represents the snapshot value.

InsertPIData Method

InsertPIData allows a client to insert values into one or more PI Server by passing an array of *TimeSeries* (page 92) objects, each of which contains an array of *TimedValues* (page 94).

Syntax

```
TimeSeries[] InsertPIData(TimeSeries (page 92)[] events,
PIInsertDuplicateHandling duplicateSwitch )
```

Arguments

Events: an array of *TimeSeries* (page 92) objects that contain the data to be inserted into the PI Server.

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.

TimedValue (page 94) has two properties that control what is put into the PI Server: **Status** and **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 PI Server point types including digital state points.

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

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

- If the **Status** property is set with a negative number, it is passed as is. However, the absolute 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.

- 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**.
- 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.

TimedValue (page 94) within each *TimeSeries* (page 92) must contain either the **Value** or **Status** to be inserted into the PI Server. The value's time stamp comes from the *TimedValue's* (page 94) Time member. The target path comes from the *TimeSeries* (page 92) object's path member. If the *TimedValue's* (page 94) path member is populated, its path will override the *TimeSeries* (page 92) object's path for that *TimedValue* (page 94) only.

Returns

If all data values are successfully inserted into the PI Server, this Web method returns a *TimeSeries* (page 92) object with no TimedValues. The value of the **SeriesID** property of the first *TimeSeries* (page 92) passed into the method is copied to the **SeriesID** property of the *TimeSeries* (page 92) returned to the client. Client applications can use this to keep track of asynchronous calls to InsertPIData.

If any *TimedValue* (page 94) 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* (page 94). The *TimedValue* (page 94) 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

- If the client cannot reach the Web service end point, the method returns a fault and no data are returned.
- If **duplicateSwitch** is **ReplaceOnlyDuplicate** and no value exists in the PI Server at the submitted time stamp, the **Status** property of the returned *TimedValue* (page 94) is:
  `-2147219630: No value exists at specified timestamp.`
- If **duplicateSwitch** is **ErrorDuplicate** and a value exists in the PI Server with the submitted time stamp, the **Status** property is:
  `-2147219631: Value already exists at specified timestamp.`
- If the **Status** property is set to a negative integer whose absolute value does not correspond to a system digital state string, this error is set in the **Status** of the *TimedValue* (page 94) object returned when the insertion is rejected:
  `-2147220396: Digital State not found. value`
- If a string is passed in **Status** that does not correspond to a system digital state value and the path refers to a non-string type tag, this error is returned:
  `-2147219633: Server returned write error: : [-15013] PIvalue Type or PIstring is Not tag_datatype`

Remarks

The path must be a PI path (`pi:\\`). You cannot reference a performance equation (`pe:\\`) path, or a PI AF path (`af:\\`) when inserting data into the PI Server.

The calling user must have the necessary privileges to write events to the PI Server. 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* (page 94) has a **DateTime** data type, the client must generate an absolute UTC time stamp for data to be inserted into the PI Server. PI absolute and relative 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 object* (page 94) contains an error message; this is the only instance in which the *TimedValue's* (page 94) **Status** property returned by PI Web Services does not contain a digital state.

When writing to a PI point of time stamp type, the Value property of the *TimedValue object* (page 94) 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 PI points, the **DataType** property should be set to **DateTime**. This **DataType** value may be set for either the *TimeSeries* (page 92) object or the individual *TimedValue* (page 94) objects.

Note: When used with InsertPIData, 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.

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 (page 95) SignUpForPIUpdates (string[] paths, ushort
expiration)
```

Arguments

paths: An array of strings that denotes the paths to the items for which data updates are desired.

expiration: Duration, in minutes, of the sign-up before it expires and is removed from the list of sign-ups. 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 five minutes. Any use of a given ticket in *GetPIUpdates* (page 84) or *ListPathsByUpdateTicket* (page 86) resets the expiration timer.

Returns

SignUpResult (page 95) object. 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* (page 84) 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* (page 6) may also report errors that range from 1000 to 3099, PI SDK reports errors with a negative **Status** value, such as `point not found`.

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* (page 84).

Syntax

```
void CancelPIUpdates (GUID updateticket)
```

Arguments

updateticket: A unique identifier to denote the collection of paths that is currently signed up for updates to cancel.

Remarks

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.

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* (page 77), *GetPISummaryData* (page 80), or *SignUpForPIUpdates* (page 83).

Syntax

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

Arguments

updateTicket: GUID returned when the path was signed up to receive 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 (Snapsnot), archive events (Archive), or both snapshot and archive event updates (SnapshotAndArchive).

Returns

An array of *TimeSeriesUpdates* (page 94) 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 with no TimedUpdate objects is returned and the Error property of the TimeSeriesUpdates object has the value **0**. The SeriesID property of all TimeSeriesUpdates objects is the same as the updateTicket.

Note: The TimedSeriesUpdate object has a TimedValues 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 *TimedUpdate* (page 95):

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* (page 92).

Remarks

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* (page 84) call, or used an incorrect update ticket.

ListPathsByUpdateTicket Method

ListPathsByUpdateTicket retrieves the *paths* (page 7) 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)
```

Errors

If the update ticket is not found in the list of updates, a SOAP fault is thrown with the message: `The update ticket was not found in the list of tickets registered for updates.`

Remarks

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.

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. For example: 1.2.7.0.

Errors

None

Remarks

The version is determined when the assembly is loaded and stored for subsequent use. For example, PI Web Services 2010 R3 will return a string value of 1.2.8.0.

Classes and Properties

PIArcDataRequest

Applies to (page 77)

Represents a query to a PI System for Compressed, Interpolated or Plot Values from a single *path* (page 7), subject to a single *time range* (page 91) *constraint* (page 9) that uses a single *Manner* (page 9). Since an array of PIArcDataRequest objects can be passed in a single call to *GetPIArchiveData* (page 77), any number of independent data requests can be placed at once.

Members

Path (page 7)

PIArcManner (page 87)

TimeRange (page 91)

PIARCMANNER

Member of (page 87) *Applies to* (page 77) *Members* (page 87)

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

Example

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

Members

NumValues

Member of (page 87) *Applies to* (page 77)

Use the **NumValues** property 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**.

Minimum value is **1**. Default: **400**.

PIArcMannerBoundaries

Member of (page 87) *Applies to* (page 77)

Parameter that defines how Compressed events are retrieved, relative to the time range boundaries. You can specify the boundaries as:

- **Inside (default)** – Return only events that fall within the time range or precisely at the boundaries
- **Outside** – Return events within the time range plus the first value that precedes the time range and the first value that follows the time range
- **Interpolated** – Return events within the time range plus interpolated values at the start and end of the time range

Example

```
arcmnr.Boundaries = PIArcMannerBoundaries.Outside;
```

PIArcMannerRetrievalType

Member of (page 87) *Applies to* (page 77)

An enumeration that defines the type of Archive data retrieved.

Enumeration values:

- **Compressed (default)** – Actual values 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.

Example

```
arcmnr.RetrievalType = PIArcMannerRetrievalType.Compressed
```

Updates

Member of (page 87) *Applies to* (page 89)

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 the point's tag name. Do not use the name of the PI AF data reference attribute.

PISummaryDataRequest

Applies to (page 79)

Represents a query for summary data from a single *path* (page 7) for a given time range subject to certain parameters. Since an array of PISummaryDataRequest objects can be passed in a single call to *GetPISummaryData* (page 79), any number of independent summary data requests can be placed at once.

Members

Path (page 7)

TimeRange (page 91)

PISummaryManner (page 89)

PISUMMARYMANNER

Member of (page 89) *Applies to* (page 79) *Members* (page 89)

Represents the type of summary retrieval. Use this object to specify the summary type and other properties.

Members

Intervals

Member of (page 89) *Applies to* (page 79)

Represents the number of evenly spaced intervals into which the time range of the query should be divided. The duration of the time range, divided by this value, forms the length of the interval for which the summary calculation is performed.

Set the start time or end time of the interval, with **UseStart**, a Boolean property. 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* (page 94). If **False**, the end time of the interval is reported.

Example

```
mySummaryManner.Intervals = 4;
```

PISummaryMannerSummaryValue

Member of (page 89) *Applies to* (page 79)

An enumeration specifying the type of summary calculations:

- Average (default)
- Count
- Minimum
- Maximum
- PStdDev (Population Standard Deviation)
- Range
- StdDev
- Total

Example

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

PISummaryMannerWeightType

Member of (page 89) *Applies to* (page 79)

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

Boolean value that indicates 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* (page 94). 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

Boolean value 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

Not supported.

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 *relative times* (page 11), or as PI or ISO 8601 *absolute times* (page 12).

Applies to

GetPIArchiveData (page 77)

GetPISummaryData (page 79)

Members

StartTime and EndTime (page 91)

STARTTIME AND ENDTIME

Strings to represent start and end times.

The class has the methods **IsStartAbsolute** and **IsEndAbsolute** which return a boolean **TRUE** if the respective property is an *absolute time* (page 12) or **FALSE** if it is a *relative time* (page 11).

If both **StartTime** and **EndTime** are *absolute times* (page 12), 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 *relative time* (page 11), the local time zone of the Web server hosting PI Web Services is used for both **StartTime** and **EndTime**.

TRANSITION TO OR FROM DAYLIGHT SAVINGS TIME

ISO 8601 time strings provide a Web-standard representation of *absolute time* (page 12). 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 **[+|-]HH: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

Properties (page 92)

This class represents a time series of timed values. It consists of a set of *properties* (page 92) that pertain to the time series and an array of *TimedValue* (page 94) objects that contain the actual time stamped data.

Applies to

GetPIArchiveData (page 77)

GetPISummaryData (page 79)

Data Returned as TimeSeries

Some properties in the *TimeSeries* (page 92) pertain to the entire time series. The array of *TimedValue* (page 94) objects is the most significant member of this class; it represents actual time-stamped data values.

Some members within *TimeSeries* (page 92) and *TimedValues* (page 94) have identical names and meanings. This allows for efficient data transfer, and also supports overriding *TimeSeries* (page 92) properties for individual timed values.

Properties

Properties of the TimeSeries class:

Name	Data Type	Description
Path	String	Name of a PI System data source such as a PI point or a PI AF element attribute. Every element of the *TimedValue* (page 94) array normally comes from this path. *TimedValue* (page 94) 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	String	Unit of measure of all TimedValue objects. The *TimedValue* (page 94) class also has a UOM field which can be used to override this value. **Note:** When used with *InsertPIData* (page 80), 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.

Name	Data Type	Description
Error	Integer	Non-zero value indicates an Error in retrieving the TimeSeries. Applications should use this property to test for the success or failure of an operation.
ErrDesc	String	(Optional) Error description for a failure involving the TimeSeries. If the operation is successful, ErrDesc is null.
DataType	String	XSD standard name of the data type of the values in the TimedValue array. The TimedValue class also has a DataType field which can be used to override this value.
SeriesID	String	String representation of a GUID that identifies the original query. The string includes both a Path and Manner.
TimedValues	TimedValue[]	Array of time-stamped data values.

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.

Note: When using *InsertPIData* (page 80), you do not need to include the data type.

This table summarizes the mapping between common data types and the value of the **DataType** property, as performed by the Web service:

Common Type	Value of DataType Property
Boolean	Boolean
Decimal	Decimal
single, double precision floating point	Double
int16, int32	Int
time, date time	DateTime
BLOB	Byte[]
PI Digital, string	String

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* (page 94) object whose **DataType** value is **Byte** will be **null**.

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

TIMEDVALUE

Applies to (page 92) *Properties* (page 94)

This class represents a single time-stamped data value retrieved from or sent to the PI System. The *TimeSeries* (page 92) class contains an array of TimedValues.

PROPERTIES

Properties of the *TimeValued* (page 94) class:

Name	Data Type	Description
Path	String	(Optional) Name of a PI System data source. When a TimeSeries object is returned from a Web method call, Path is reflected in the TimeSeries class for every element of the TimedValue array; This field is used primarily in InsertPIData; where it is possible to send data values to many paths in a single TimedValue 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 TimeSeries and is specified there; If this field is not blank, the UOM overrides this value only
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. **Note:** When entering data through InsertPIData, 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. 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

TimeSeriesUpdates

Applies to (page 84)

This object is derived from *TimeSeries* (page 92). In addition to the properties of that object, TimeSeriesUpdates adds the property Updates, which is an array of *TimedValueUpdate* (page 95) objects.

TimedValueUpdate

This object is derived from *TimedValue* (page 94). 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* (page 84).

When you call GetPIUpdates, the TimedValues property is null and *TimeSeriesUpdates* (page 94) is an instantiated object. *TimedValue* (page 94) is used for the regular data methods such as *GetPIArchiveData* (page 77); *TimeSeriesUpdates* (page 94) is only used for *GetPIUpdates* (page 84).

SignUpResult

Applies to (page 83)

Class used to communicate the results of explicit registration for data updates through the *SignUpForPIUpdates* (page 83) method.

Properties

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.

Remarks

Errors, and ErrDescs will always have the same number of entries. This number will match the number of paths passed into *SignUpForPIUpdates* (page 83), 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* (page 84) 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 return collections of *paths* (page 7). Returned paths can be used by methods of the *IPITimeSeries interface* (page 77) exactly as they are retrieved.

IPISearch Web Method

FindPIPathsBasic

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. Returns consist of an array of *paths* (page 7), in a syntax suitable for use with methods from the IPITimeSeries interface.

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 suitable for use with the *IPITimeSeries interface* (page 77).

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 (AF) servers. Wildcard characters are not permitted.

tagMask: String that contains the PI tag mask, including the optional wildcard characters ***** and **?**.

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. For example, a PI point named **FIC*.PV**

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* (page 7) in methods of the *IPITimeSeries* (page 77) interface. For example, the tag for a PI point named **sinusoid** on a PI Server named **piserver01**, is returned as `pi:\\piserver01\sinusoid`.

Remarks

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* (page 77). However, the retrieval of BLOB type data for PI tags is not supported.

Classes and Properties

PIPointType

Enumeration denoting a PI point type:

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

Chapter 6

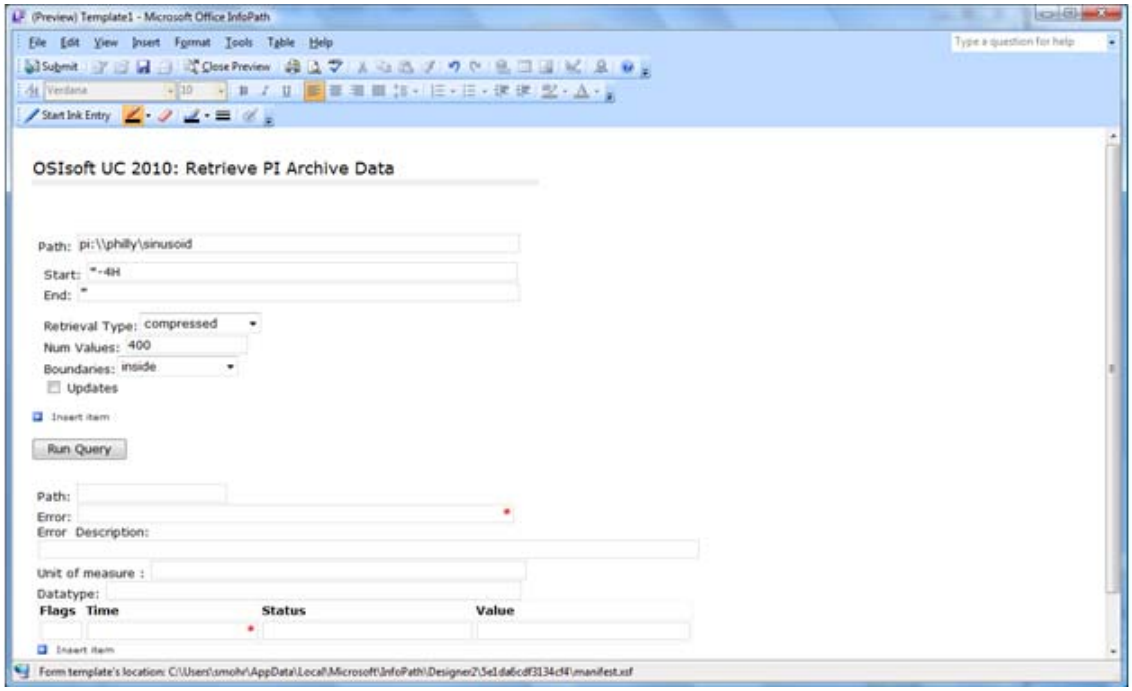
Use InfoPath with PI Web Services

To access data through PI Web Services, you must use a compatible *Web service client application* (page 3). 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* (page 54) is required when you use InfoPath.

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

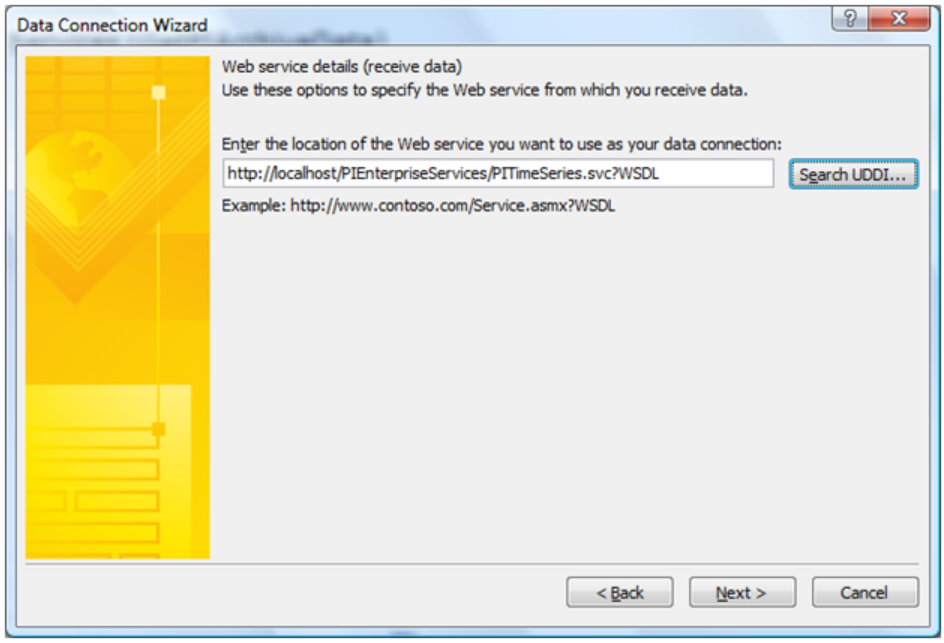


Configure the Data Connection

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



6. *Select the method* (page 101) that the form will use.

Select a Method

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:

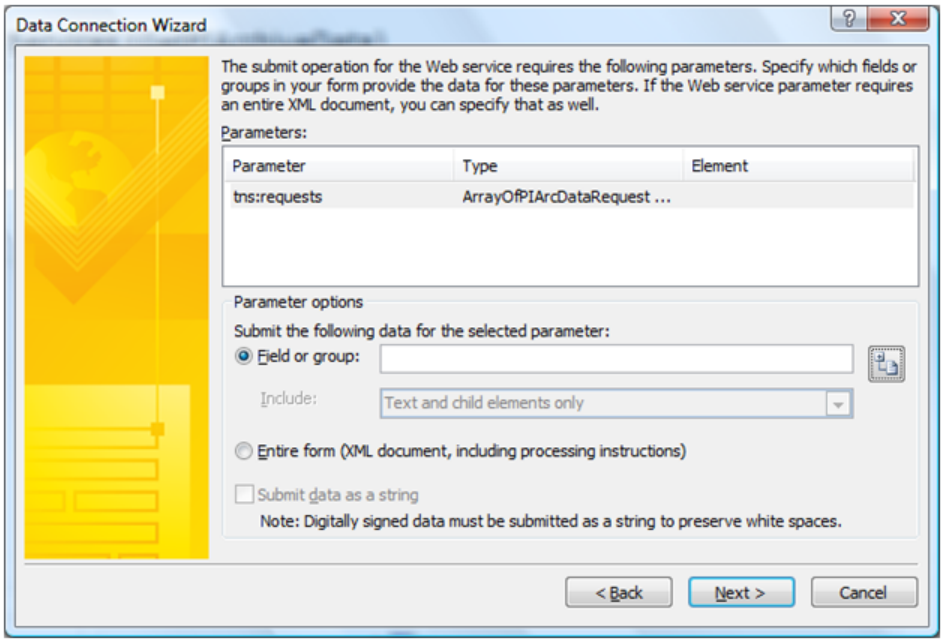
- **InsertPIData**
- **GetPISummaryData**
- **GetPIArchiveData**

1. Select **GetPIArchiveData** and click **Next**.
2. 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**.
3. *Configure the input* (page 101) that the fields will use.

Configure Field Input

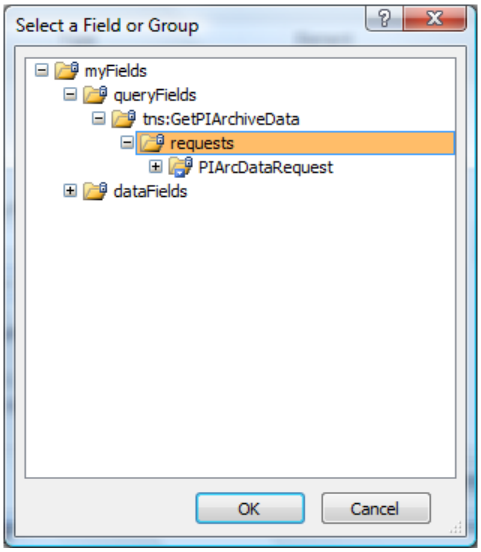
InfoPath uses the WSDL file to create empty fields in the form. To ensure the fields capture the correct data, you must associate, or bind a Web services request parameter to each field. Binding these parameters is like labeling a bucket; only data that matches the label can be added to the bucket, or in this case, the form field.

To specify how requests submitted to PI Web Services through the form are processed:



1. Select a parameter in **Parameters**.

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

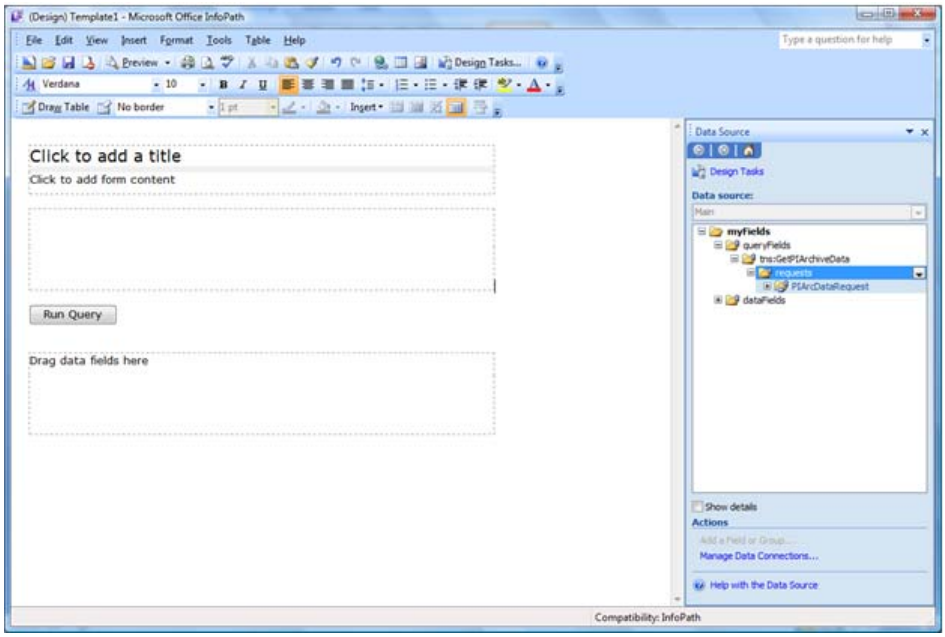


4. Click **OK**.
5. Click **Next**.
6. Click **Finish**.
7. *Design the layout* (page 102) for the form template.

Design the Form

Now you can create the template for the form that users will see when they make 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:



To lay out fields on the form that will elicit the request parameter values and display the response values:

1. Drag the Path element onto the form to create a text field within a repeating section.
2. 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* (page 107) cannot be used to submit PI relative times.

---

3. 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**.
4. Next, design the *layout of the display* (page 103) for the data returned from PI Web Services:

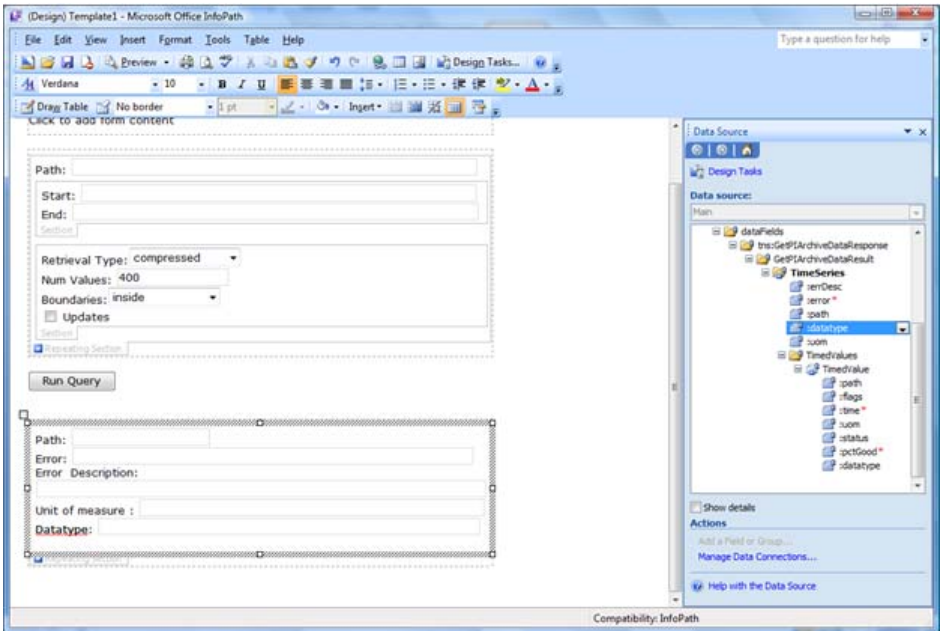
Design Returned Data Display

The data returned by the PI Web Services is an array of *TimeSeries* (page 92) objects. Each such object consists of some data pertaining to the time series as a whole, followed by an array of *TimedValue* (page 94) objects representing events returned by the PI Server. To layout the *TimeSeries* (page 92) data:

1. Drag the Path from TimeSeries onto the lower portion of the form to create a repeating section.
2. Drag **Error**, **ErrDesc**, **UOM**, and **DataType** onto the form, in succession, in the repeating section under **Path**.

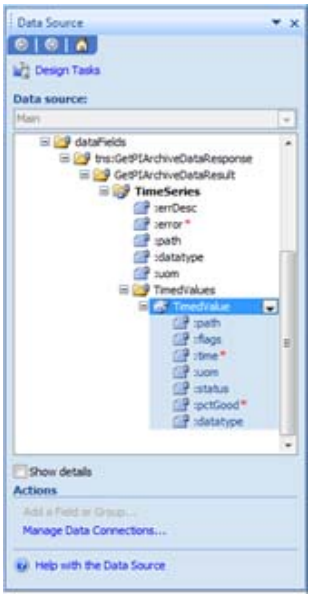
3. Edit the labels as desired:
   a. The **Error** field is a numeric error code returned if the retrieval as a whole failed.
   b. If it is non-zero, an error occurred and **ErrDesc** will contain a descriptive string.
   c. The **UOM** field contains the units of measure for the PI point or performance 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:



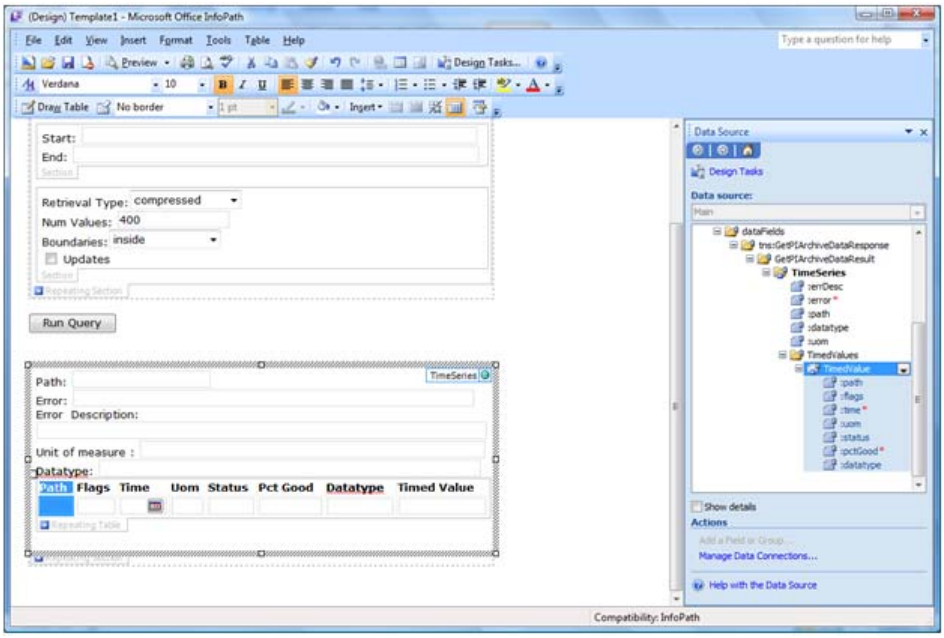
4. Next, create form fields for the individual timed values returned. If you examine the **TimedValues** field in the tree on the right of the form, you will note that there is nothing listed for value. This occurs because the value itself is returned as the textual content of an XML element in the Web service reply. In order to pick this up, drag the entire *TimedValue* (page 94) 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:

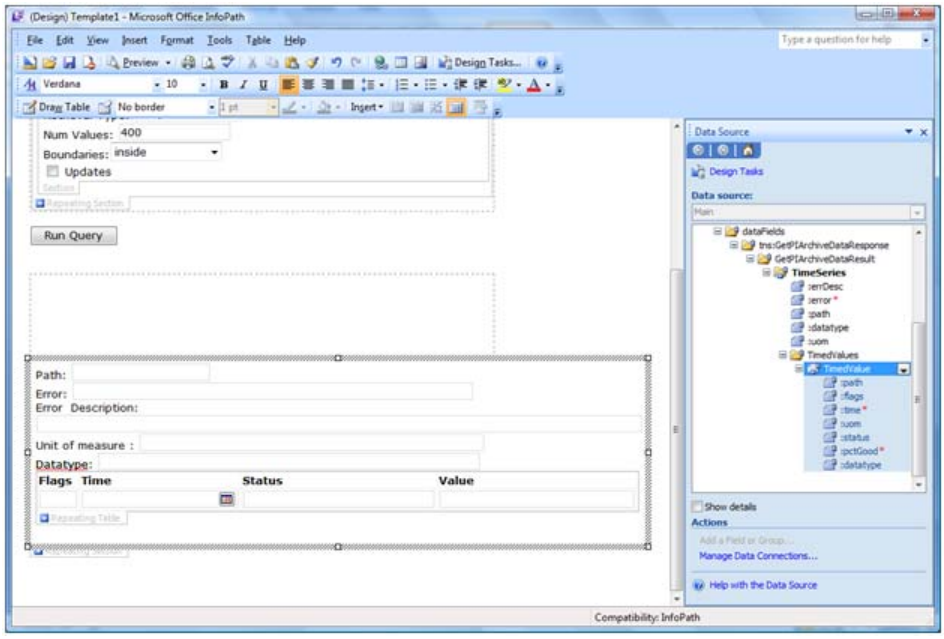


b. Select **repeating table** from the options. Note that each row is crowded. You can edit this to make it more presentable. Much of the data fields defined for the TimedValue object overlaps the TimeSeries object, or are used in different Web service methods. For example, since every TimedValue returned 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. Repeat Steps a through c for **UOM**, **PctGood**, and **DataType**. Adjust the size of the fields as desired. The results should be similar to this:



e. Decide whether to change the *Date Picker control* (page 107).

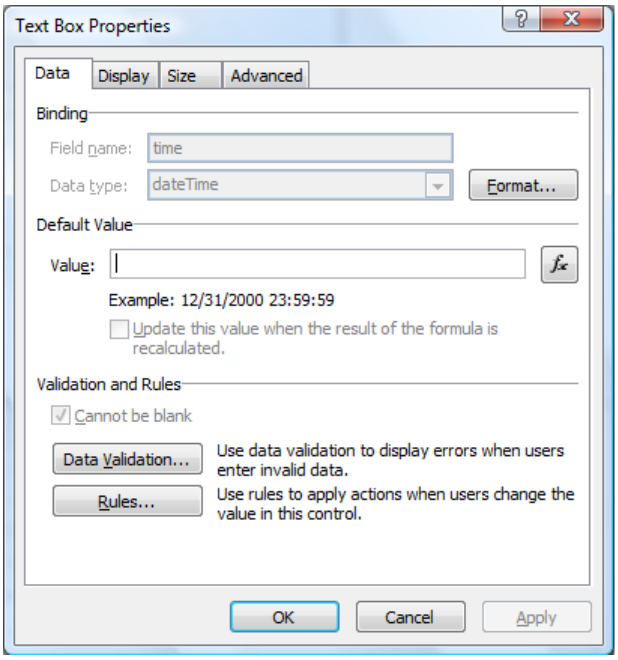
f. Next, *test the form* (page 108).

Change the Date Picker Control

By default, InfoPath turns the time field, which is indicated by dateTime in the WSDL, as a Date Picker control. If you use this default setting, the form will hide times submitted as PI relative times.

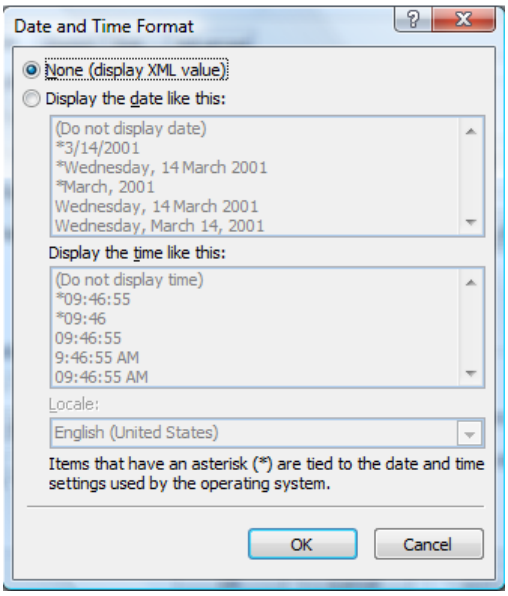
To change this setting:

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:



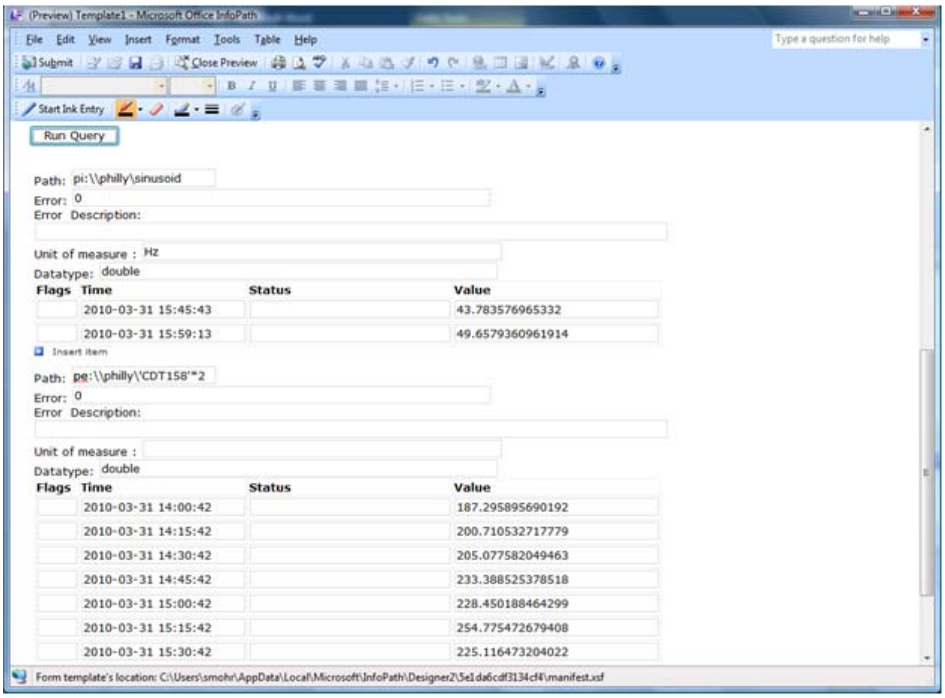
Test the Form

To use PI Web Services and test a form based on the template:

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	interpolated

Click **Run Query**, then **OK** on the security notice:



Appendix A

Configure PI Web Services to use Kerberos Authentication

This appendix describes how to set up Kerberos authentication for products that use PI Data Services, including PI Web Services.

Note: The security configuration changes described in this appendix will affect all applications running in the IIS Web application.

PI Web Services uses 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 of PI Web Services.

PI Web Services ships with a `web.config` file that is properly configured for the double-hop situation if the Web service is running on the IIS Web under the **NETWORK SERVICE** account. When an application pool uses another identity, you must configure an SPN as described here. To create an SPN, you must have domain administrator privileges.

If the 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 `web.config`

Prerequisites

To use Kerberos delegation, the Web server that hosts the product must be configured to accept the user's login credentials and relay them 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

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:

- For SharePoint Server 2010:
  `http://technet.microsoft.com/en-us/library/ee806870.aspx`
- For SharePoint Foundation 2010:
  `http://technet.microsoft.com/en-us/library/ff607695.aspx`
- For Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007:
  `http://support.microsoft.com/?id=832769`

Configure the User Account

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 **Account is sensitive and cannot be delegated is not selected**.

Associate an SPN with the Application Pool Domain Account

The Kerberos protocol relies on Service Principal Names (SPNs) that 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, for a Web service that runs under the account `piwp123\1ws` that is accessed at `http://corporatews.piwp123.com/ws.asmx`:

```
SETSPN  -A HTTP/corporatews.piwp123.com piwp123\1ws
```

Associate an SPN with the domain account

Normally, the 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\1web
SETSPN  -A HTTP/corporate1web.piwp123.com 1web
```

For a machine named `wsdomain\piwsserver1` and an account `piws1acct` on the same domain:

```
SETSPN -A HTTP/piwsserver1 wsdomain\piws1acct
SETSPN -A HTTP/piwsserver1.wsdomain.com piws1acct
```

Set the Domain Account as Trusted for Delegation

Note: To perform this step, 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:

1. From the account properties dialog, click the **Account** tab.
2. Under **Account Options**, click to select the Account is trusted for delegation.
3. Click **OK**.

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/piwsserver1`.

Use of Alternate port numbers

Web sites running under port numbers other than the default port 80 do not require any additional configuration beyond those described above. The Kerberos HTTP SPN enables the IIS W3WP process to delegate credentials on all ports, including 443 for SSL.

Use of Basic Authentication with SSL

A common configuration for authentication over the Internet is to place the Web server 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.

Using Host Headers

Within 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.

For example, a Web server name `CORPORATE1WEB` has two Web sites:

Site	Port	Header	App 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:

```
SEYSPN  -A HTTP/sample.piwp123.com piwp123\corporate1web
SETSPN  -A HTTP/example.piwp123.com piwp123\1web
```

Using Load-Balanced Web Farms

All supported SharePoint versions 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, especially if Service Principal Names must be established or if the domain policies have been restricted. As with other networking tasks, the best approach is to confirm one hop at a time.

Confirm that Client Computer is Using Kerberos

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. On the Web server, view the security event log to see which protocol is being used for authentication. There will be events for the user account that show Successful Network Logon and list the Authentication Package that was used. If the package is NTLM instead of Kerberos, double-hop cannot succeed. This might occur because the Web server is not configured to accept Kerberos or because the client account cannot use Kerberos.

Confirm that Web Server is Using Kerberos

Look in the security event log on the target machine where the data source resides, either the file server, the Web service machine or the database server. There will be events for the user account that show Successful Network Logon and list the user name and the Authentication Package that was used. If the package is NTLM and the user name is blank, the Web server is not using Kerberos to communicate with the target machine. If the client is successfully using Kerberos to connect to the Web server, this hop can fail because the target machine is not configured to use Kerberos.

Resources

KerbTray is a useful utility that helps diagnose Kerberos issues for Windows Server 2003. It displays Kerberos tickets and can purge the credentials without logging out and is available at the Microsoft Web site.

A comprehensive discussion of Kerberos troubleshooting can be found in the Microsoft article Troubleshooting Kerberos at `http://technet.microsoft.com/en-us/library/cc728430.aspx`.

Use of Alternate port numbers

Web sites running under port numbers other than the default port 80 do not require any additional configuration beyond those described above. The Kerberos HTTP SPN enables the IIS W3WP process to delegate credentials on all ports, including 443 for SSL.

Use of Basic Authentication with SSL

A common configuration for authentication over the Internet is to place the Web server 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.

Use of Host Headers

Within 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.

For example, a Web server name `CORPORATE1WEB` has two Web sites:

Site	Port	Header	App 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

All supported SharePoint versions 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, especially if Service Principal Names must be established or if the domain policies have been restricted. As with other networking tasks, the best approach is to confirm one hop at a time.

Confirm that Client Computer is Using Kerberos

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. On the Web server, view the security event log to see which protocol is being used for authentication. There will be events for the user account that show Successful Network Logon and list the Authentication Package that was used. If the package is NTLM instead of Kerberos, double-hop cannot succeed. This might occur because the Web server is not configured to accept Kerberos or because the client account cannot use Kerberos.

Confirm that Web Server is Using Kerberos

Look in the security event log on the target machine where the data source resides, either the file server, the Web service machine or the database server. There will be events for the user account that show Successful Network Logon and list the user name and the Authentication

Package that was used. If the package is NTLM and the user name is blank, the Web server is not using Kerberos to communicate with the target machine. If the client is successfully using Kerberos to connect to the Web server, this hop can fail because the target machine is not configured to use Kerberos.

Resources

KerbTray is a useful utility that helps diagnose Kerberos issues for Windows Server 2003. It displays Kerberos tickets and can purge the credentials without logging out and is available at the Microsoft Web site.

A comprehensive discussion of Kerberos troubleshooting can be found in the Microsoft article Troubleshooting Kerberos at `http://technet.microsoft.com/en-us/library/cc728430.aspx`.

Appendix B

Logging and Instrumentation

PI Web Services includes an instrumentation framework that manages performance counters and message logging. This framework allows you to run various levels of traces that monitor execution of your application without disturbing your operating environment. It generates logs of errors and events that can be easily read with any of three supported listener applications:

- Windows Event log
- PI Message log
- standard debug window

By default, the instrumentation framework is configured to log error messages to the Windows Event log on the Web server.

Note: At installation, the instrumentation framework is configured to report both warnings and errors. To help troubleshoot, you may need to increase the level of reporting to include messages that trace the details of operations. To do this, edit the `PIInstrumentation.config` file and look for the string:

<logFilter logMode="Warnings"

Change the *logMode* (page 124) from `Warnings` to `All` to enable trace messages.

Initial Settings

Parameters for log settings are preconfigured in the `PIInstrumentation.config` file, which is located in the PIPC\DAT directory. This file is automatically created when you install a product that includes the instrumentation framework.

To adjust logging and trace settings, modify `PIInstrumentation.config`. You do not need to restart your application after you modify `PIInstrumentation.config`

The `PIInstrumentation.config` file contains separate sections where you can find settings for the following:

- <*EventSources*> (page 120)
- <*Listeners*> (page 120)
- <*LogFilters*> (page 122)
- <*Formatters*> (page 122)

Most modifications are completed in the <LogFilters> section.

Note: In the event that your `PIInstrumentation.config` file become corrupted, deleted, or needs to be replaced for any reason, simply re-run the `InstallUtil.exe` application to regenerate the file. `InstallUtil.exe` is included as part of the Microsoft .NET Framework utilities.

<EventSources>

Event sources are message sources predefined by each client using the PI Instrumentation Framework.

The <EventSources> section in `PIInstrumentation.config` (page 119) includes application-specific event sources listed by type. Associated filter bindings sections act as channels to deliver messages to appropriate listeners. For example:

```
        <EventSource
name="PIWebServices.PIDataService.PIWebServicesInstrumentation.PIW
ebServicesTraceSourceTypes.PIWebServicesSearch" descriptions="PI
Web Services Search source" client="PI Web Services">
        <eventSourceParamList>
          <EventSourceParameter name="FilterBindings"
value="defaultFilterBindings" />
          <EventSourceParameter name="DebugInfo" value="false" />
        </eventSourceParamList>
      </EventSource>
      <EventSource
name="PIWebServices.PIDataService.PIWebServicesInstrumentation.PIW
ebServicesTraceSourceTypes.PIWebServicesTimeSeries"
descriptions="PI Web Services time series source" client="PI Web
Services">
        <eventSourceParamList>
          <EventSourceParameter name="FilterBindings"
value="defaultFilterBindings" />
          <EventSourceParameter name="DebugInfo" value="false" />
        </eventSourceParamList>
      </EventSource>
```

- **FilterBindings**—contains the name of the *LogFilter* (page 122) set to be used by this client source for logging messages.
- **DebugInfo**—option that enables logging of more detailed information such as stack traces and process identity information. By default it is set to **FALSE**. Change this value to **TRUE** to enable it.

<listeners>

The PI Instrumentation Framework currently supports three listeners that enable you to view logs and errors:

- **Windows Event Log**—recommended tool that comes with Microsoft Windows. By default PI Instrumentation is set for this listener.

- **PI Message Log** (in PI SMT)—for PI System admins who want to read logs while working with PI System tools.
- **DebugView for Windows**—enables more detailed logging and debugging information. This listener also enables you to save error messages in a text file that can then be sent to the OSIsoft Technical Support team for additional troubleshooting. You must download this application from the *Microsoft Web site* (*http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx*).

Note: By default only the Windows Event Log listener is configured to receive logging and error information. You can add the other listeners by modifying the `PIInstrumentation.config` file, however, keep in mind that additional listeners may result in some performance loss on your server.

```
<listeners>
<listener xsi:type="WMIListener" name="WmiTraceListener"
descriptions="Windows Event Log"
type="OSIsoft.PIInstrumentation.Listener.WmiTraceListener"
listenerDataType="OSIsoft.PIInstrumentation.Listener.WmiTraceListe
ner, OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=53b77d1d3d7a979b"
formatter="General" machineName="." />
</listeners>

<listener xsi:type="PISDKListener" name="PISDKLogEventListener"
descriptions="PISDK Message Log
"type="OSIsoft.PIInstrumentation.Listener.PISDKLogEventListener"li
stenerDataType="OSIsoft.PIInstrumentation.Listener.PISDKLogEventLi
stener OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=53b77d1d3d7a979b"
formatter="General" PIServer="trenton" />

<listener xsi:type="ConfigTraceListener" name="TraceEventListener"
descriptions="Trace message Log"
type="OSIsoft.PIInstrumentation.Listener.TraceEventListener"
listenerDataType="OSIsoft.PIInstrumentation.Listener.TraceEventLis
tener,OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=53b77d1d3d7a979b"
formatter="General" />
```

Each listener has the following parameters:

- `Name`—used to configure filter bindings
- `Descriptions`—used to describe the listener
- `Type`—used to instantiate the listener object and send messages
- `Formatter`—name used by the listener to format the message before logging

Note: Some listeners may have a few additional parameters. For example, the PI SDK message listener configuration requires a PI Server named with the PIServer parameter.

<LogFilters>

LogFilters are the main drivers of configuration for all message logging, and are frequently referenced by event source configuration entries. The default LogFilter configuration enables you to log all the warnings and errors from all client sources to the Windows Event log.

Most modifications to `PIInstallation.config` are completed in the <LogFilters> section. See *Modify Configuration* (page 123) for more details.

<formatters>

This section contains all the formatters and their associated templates. Listeners format messages according to a specified template before logging them. For example:

```
<formatters>
    <formatter name="TextFormatter" descriptions="Text Formatter"
type="OSIsoft.PIInstrumentation.Listener.Formatter.TextFormatter,
OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=53b77d1d3d7a979b"
template="Message: {message}{newline}Category:
{category}{newline}Priority: {priority}Extended Properties:
{dictionary({key} - {value}{newline})}" />
    <formatter name="XMLFormatter" descriptions="XML Formatter"
type="OSIsoft.PIInstrumentation.Listener.Formatter.XmlLogFormatter
, OSIsoft.PIInstrumentation.Listeners, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=53b77d1d3d7a979b" template="" />
</formatters>
```

Each formatter configuration has the following parameters:

- `Name`—referenced by listener configuration to select the correct formatter
- `Descriptions`—used to describe the formatter
- `Type`—used to instantiate the formatter object and format the message
- `Template`—references a key-value pair template collection used to specify a label for the message and how it should appear when logged. For example:

  ```
  Timestamp: {timestamp}{newline}
  where {timestamp} is the value substituted from the LogEntry
  object. The output format appears as
  Timestamp:  8/24/2008 2:10:54 pm
  followed by a new line.
  ```

Note: For the formatter name value XMLFormatter, the actual LogEntry object is an XML document and requires no template specification.

Modify Configuration

Most manual changes to instrumentation settings are made in the <*logFilters*> (page 122) section of the `PIInstrumentation.config` (page 119) file. Under <logFilters>, you may adjust the following:

- *Add/Remove Listeners* (page 123)— provide destinations for messages to be sent
- *Filter Messages* (page 123)—configure what messages appear in your logs

Add/Remove Listeners

Use the <*listeners*> (page 120) section of `PIInstallation.config` to configure one or more trace listeners for a single message source. For example, if you want to send critical error messages to the Windows Event Log and the PI Message Log, simply add both listeners in order to send the message to each:

```
<listeners>
    <listener>PISDKLogEventListener</listener>
    <listener>WmiTraceListener</listener>
</listeners>
```

The following table describes the listener names and details. The PI Instrumentation framework currently supports three listeners.

Name in PIInstrumentation.config	Listener Application
WmiTraceListener	Windows Event Log
PISDKLogEventListener	PI Message Log (in PI SMT)
TraceEventListener	*DebugView for Windows* (*http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx*)

Filter Messages

`PIInstrumentation.config` (page 119) contains four levels of filtering that enable you to parse the messages that get logged to your active *listeners* (page 120). When an event is traced it must satisfy the conditions for each filter in order to be logged as a message. You can modify the requirements for each filter in the <logFilters> section of `PIInstrumentation.config`. The order of filtering is as follows:

- *LogMode Filter* (page 124)—determines what kind of messages should be logged
- *Category Source Filter* (page 124)—conditions to log messages within specific categories
- *Priority Filter* (page 125)—to specify logging a message only when it falls within a certain range of priority levels
- *Keyword Filter* (page 125)—to log messages containing certain keywords

LogMode Filter

The LogMode setting determines what kind of message should be logged. There are three possible settings:

- **Errors**—logs only the error messages
- **Warnings**— logs error and warning messages
- **All**—logs all the messages, including informational messages

Category Source Filter

Categories correspond to the clients from which you receive messages. Filter these categories by setting one of the two `categoryFilterMode` settings listed below, followed by a list of `Names` to define which categories to explicitly deny or allow:

- `DenyAllExceptAllowed`–denies all categories except those explicitly allowed in the names list
- `AllowAllExceptDenied`–allows all categories except those explicitly denied in the names list

For example, if you want to log all messages from event source DataAccess and ignore the remainder, you can use the following:

```
<categorySources>
   <categorySource categoryFilterMode="DenyAllExceptAllowed"
    type="OSIsoft.PIInstrumentation.Configuration.CategorySource"
    name="default" >
      <names>DataAccess</names>
   </categorySource>
</categorySources>
```

In contrast, the following configuration logs all messages from all sources:

```
<logFilter logMode="All" name="defaultFilterBindings"
enabled="true">
<categorySources>
   <categorySource categoryFilterMode="AllowAllExceptDenied"
    type="OSIsoft.PIInstrumentation.Configuration.CategorySource"
    name="default" description="Allow all Category except those
explicitly specified as denied">
      <names/>
   </categorySource>
</categorySources>
```

If the LogFilter is not enabled, it does not load formatters or listeners. The framework loads only the necessary filters and listeners based on <LogFilter> settings.

Priority Filter

Each message object has a priority setting. If the priority meets the range between the minimum and maximum priority filter levels, then the message is logged.

In the priorityFilter section you can set the following parameters:

- `name`
- `type`
- `minimumPriority`
- `maximumPriority`

Keyword Filter

This setting logs the message if the message has any specific keyword which you would like to monitor.

In the `keywordFilter` section you can set the following parameters:

- `name`
- `keyword`

Message Throttling

The PI Instrumentation Framework includes a message throttling component that prevents 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="" />
```

Appendix C

Technical Support and Resources

You can read complete information about technical support options, and access all of the following resources at the OSIsoft Technical Support Web site:

http://techsupport.osisoft.com

For information on programming and integration with OSIsoft products see the OSIsoft vCampus Web site, or the OSIsoft vCampus section at the end of this document.

Before You Call or Write for Help

When you contact OSIsoft Technical Support, please provide:

- Product name, version, and/or build numbers
- Computer platform (CPU type, operating system, and version number)
- The time that the difficulty started
- The log files at that time

Help Desk and Telephone Support

You can contact OSIsoft Technical Support 24 hours a day. Use the numbers in the table below to find the most appropriate number for your area. Dialing any of these numbers will route your call into our global support queue to be answered by engineers stationed around the world.

Office Location	Access Number	Local Language Options
San Leandro, CA, USA	1 510 297 5828	English
Philadelphia, PA, USA	1 215 606 0705	English
Johnson City, TN, USA	1 423 610 3800	English
Montreal, QC, Canada	1 514 493 0663	English, French
Sao Paulo, Brazil	55 11 3053 5040	English, Portuguese
Frankfurt, Germany	49 69 951 555 333	English, German
Manama, Bahrain	973 1758 4429	English, Arabic
Singapore	65 6391 1811 86 021 2327 8686	English, Mandarin Mandarin
Perth, WA, Australia	61 8 9282 9220	English

Support may be provided in languages other than English in certain centers (listed above) based on availability of attendants. If you select a local language option, we will make best efforts to connect you with an available Technical Support Engineer (TSE) with that language skill. If no local language TSE is available to assist you, you will be routed to the first available attendant.

If all available TSEs are busy assisting other customers when you call, you will be prompted to remain on the line to wait for the next available TSE or else leave a voicemail message. If you choose to leave a message, you will not lose your place in the queue. Your voicemail will be treated as a regular phone call and will be directed to the first TSE who becomes available.

If you are calling about an ongoing case, be sure to reference your case number when you call so we can connect you to the engineer currently assigned to your case. If that engineer is not available, another engineer will attempt to assist you.

Search Support

From the OSIsoft Technical Support Web site, click **Search Support**.

Quickly and easily search the OSIsoft Technical Support Web site's support solutions, documentation, and support bulletins using the advanced MS SharePoint search engine.

E-Mail–Based Technical Support

techsupport@osisoft.com

When contacting OSIsoft Technical Support by e-mail, it is helpful to send the following information:

- Description of issue: Short description of issue, symptoms, informational or error messages, history of issue.
- Log files: See the product documentation for information on obtaining logs pertinent to the situation.

Online Technical Support

From the OSIsoft Technical Support Web site, click **My Support** > **My Calls**.

Using OSIsoft's Online Technical Support, you can:

- Enter a new call directly into OSIsoft's database (monitored 24 hours a day)
- View or edit existing OSIsoft calls that you entered
- View any of the calls entered by your organization or site, if enabled
- See your licensed software and dates of your Service Reliance Program agreements

Remote Access

From the OSIsoft Technical Support Web site, click **Contact Us** > **Remote Support Options**.

OSIsoft Support Engineers may remotely access your server in order to provide hands-on troubleshooting and assistance. See the Remote Support Options page for details on the various methods you can use.

On-Site Service

From the OSIsoft Technical Support Web site, click **Contact Us** > **On-site Field Service Visit**.

OSIsoft provides on-site service for a fee. Visit our On-site Field Service Visit page for more information.

Knowledge Center

From the OSIsoft Technical Support Web site, click **Knowledge Center**.

The Knowledge Center provides a searchable library of documentation and technical data, as well as a special collection of resources for system managers. For these options, click **Knowledge Center** on the Technical Support Web site.

- The Search Support feature allows you to search Support Solutions, Bulletins, Support Pages, Known Issues, Enhancements, and Documentation (including user manuals, release notes, and white papers).
- System Manager Resources include tools and instructions that help you manage archive sizing, backup scripts, daily health checks, daylight saving time configuration, PI Server security, PI System sizing and configuration, PI trusts for interface nodes, and more.

Upgrades

From the OSIsoft Technical Support Web site, click **Contact Us** > **Obtaining Upgrades**.

You are eligible to download or order any available version of a product for which you have an active Service Reliance Program (SRP), formerly known as Tech Support Agreement (TSA). To verify or change your SRP status, contact your Sales Representative or *Technical Support* (*http://techsupport.osisoft.com/*) for assistance.

OSIsoft Virtual Campus (vCampus)

The OSIsoft Virtual Campus (vCampus) Web site offers a community-oriented program that focuses on PI System development and integration. The Web site's annual online subscriptions provide customers with software downloads, resources that include a personal development PI System, online library, technical webinars, online training, and community-oriented features such as blogs and discussion forums.

OSIsoft vCampus is intended to facilitate and encourage communication around PI programming and integration between OSIsoft partners, customers and employees. See the OSIsoft vCampus Web site, *http://vCampus.osisoft.com* (*http://vCampus.osisoft.com*) or contact the OSIsoft vCampus team at vCampus@osisoft.com for more information.

Index

<

<EventSources> • 120
<formatters> • 122
<listeners> • 120
<LogFilters> • 122
<Path to Item> • 8
<Server> • 7
<Source Designator> • 7

A

About the OSIsoft PI Data Access Suite • 2
Absolute Time • 12
Add .svc Handler Mapping to IIS • 35
Add the Message Size Binding Element • 37
Add/Remove Listeners • 123
After Installation • 28
Allow or Disallow Insertions of Data to a PI System • 40
Allow or Disallow Performance Equation Calls • 40
Architecture • 3
Associate an SPN with the Application Pool Domain Account • 112
Associate an SPN with the domain account • 113
Automatic WSDL Generation • 13

B

basicHttpBinding • 54
basicHttpBinding Sample • 63
Before Installation • 15
Binding Types used by PI Web Services • 54

C

CancelPIUpdates Method • 84
Category Source Filter • 124
Change the Date Picker Control • 107
Change the Endpoint and Active Configuration Bindings • 56
Classes and Properties • 87, 97
Configure Data Display Returns • 40
Configure Duration of Update Sign Ups • 40
Configure Field Input • 101
Configure Firewall Exceptions • 69
Configure IIS Application Server Settings • 20
Configure IIS on Windows 2003 • 19
Configure PI Web Services to use Kerberos Authentication • 111
Configure Required User Accounts • 51
Configure Secure Web Service Access • 53
Configure Security for .NET Clients • 61
Configure Security for PI Web Services Standalone Edition • 42
Configure Security for Web Service Bindings • 55
Configure the Data Connection • 100
Configure the User Account • 112
Configure the Web Server • 18
Configure WCFStorm for PI Web Services Access • 32
Confirm that Client Computer is Using Kerberos • 116
Confirm that Web Server is Using Kerberos • 116
Connections that Kerberos Authentication • 52
Connections that use Windows Authentication • 52
Connections to PI and AF Collectives • 10
Constraint • 9
Control Message Size • 37
Control PI Web Services Features • 39
Create an SPN • 112
Create and Execute a Snapshot Request • 34

D

Data Entry Disallowed • 73
Data Returned as TimeSeries • 92
Design Returned Data Display • 103
Design the Form • 102
Download and Install WCFStorm • 32

E

Edit the Active Binding Configuration • 57
Edit the Endpoint Binding • 56
Enable or Disable Impersonation • 58
End User Folder and Registry Permissions • 53
Error and Trace Message Logging • 9
Example
Configure Exceptions on Windows Server 2008 SP2 • 69
Examples • 8
Exception to Time Input Translation • 11
Execution of Performance Equations Disallowed • 74
Explicit Sign-ups • 6

F

Filter • 88, 91
Filter Messages • 123
Filters or Parameters Not Supported • 74
Find the PI Web Services File Directory • 30
FindPIPathsBasic • 96
Firewall Security • 68

G

GetPIArchiveData Method • 77
GetPISnapshotData Method • 80
GetPISummaryData Method • 79
GetPIUpdates Method • 84
GetProductVersion Method • 86

H

Host PI Web Services on a PI WebParts Server • 17
Host PI Web Services with a Windows service • 40

I

If the Service Does Not Start • 43
IIS is Not Running • 71
Impersonation and Delegation • 57
Implicit Sign-ups • 6
In this Guide • 1
Initial Settings • 119
InsertPIData Method • 80
Install .NET Framework 4 • 17
Install IIS on Windows Server 2003 • 18
Install PI Web Services • 15
Install PI Web Services Standalone Edition • 41
Insufficient Message Size • 73
Insufficient Permissions • 73
Intervals • 89
Introduction • 1
Invalid Tag Name • 76
IPISearch Interface • 96
IPISearch Web Method • 96
IPITimeSeries Interface • 77
IPITimeSeries Web Methods • 77

K

Keyword Filter • 125

L

ListPathsByUpdateTicket Method • 86
Locate PI Web Services Files • 31
Logging and Instrumentation • 119
LogMode Filter • 124

M

Manner • 9
Members • 87, 89
Message Throttling • 125
Modify Configuration • 123

N

netNamedPipeBinding • 55
netNamedPipeBinding Sample • 67
netTcpBinding • 55
netTcpBinding Sample • 65
NumValues • 87

P

Path • 7
PI Identities and Mappings • 49
PI System Data Types • 9
PI System Not Found • 72
PI Trusts • 49
PI Web Services Interfaces • 77
PI Web Services Programmer Reference • 77
PI Web Services Security • 47
PIArcDataRequest • 87
PIArcManner • 87
PIArcMannerBoundaries • 88
PIArcMannerRetrievalType • 88
PIPointType • 97
PISummaryDataRequest • 89
PISummaryManner • 89
PISummaryMannerSummaryValue • 90
PISummaryMannerWeightType • 90
Prerequisites • 111
Priority Filter • 125
Properties • 92, 94

R

Relative Time • 11
Remote Connection Failure • 72
Reserve Namespace on Windows 2003 Server • 42
Reserve Namespace on Windows 2008 R2 Server, Windows 7 or Vista • 43
Resources • 117
Retrieval of Data Updates • 5
Returned Time Stamps • 11
Review AF Server Requirements • 17
Review ASP.NET ISAPI extensions for .NET 4 Framework • 22

Review Firewall Setup • 39
Review Role Services in IIS Server Manager • 22
Review Security Configuration • 38
Role of PI Data Services • 6
Run Setup Kit • 27

S

Sample Configuration Files • 62
Secure Access to Data through PI Asset Framework • 51
Secure Access to PI Server Data • 48
Security Binding Samples • 62
Select a Method • 101
Server Error in PI Web Services Application • 75
Server Not Found • 72
Set Maximum Message Size • 38
Set Required User Permissions • 50
Set the Domain Account as Trusted for Delegation • 113
Sign up for data updates • 5
SignUpForPIUpdates Method • 83
SignUpResult • 95
Silent Installation of PI Web Services • 45
StartTime and EndTime • 91
Summary of Features • 4
Supported Client Platforms • 17
Supported Security Scenarios • 48
System Requirements • 16

T

Technical Support and Resources • 127
Test the Form • 108
Test the Web Server Connection • 29
Time Input Translation Rules • 10
Time Stamps • 10
TimedValue • 94
TimedValueUpdate • 95
TimeRange • 91
TimeSeries • 92
TimeSeriesUpdates • 94
Transition To or From Daylight Savings Time • 91
Troubleshooting • 71, 116

U

Updates • 88, 90
Use InfoPath with PI Web Services • 99
Use of Alternate port numbers • 115
Use of Basic Authentication with SSL • 115
Use of Host Headers • 115
Use of Load-Balanced Web Farms • 116
UseStart • 90

V

Verify Data Access • 32
Verify Identity Connections • 52
Verify Standalone Edition Installation • 44

W

WCF Service Configuration Editor • 68
Web Service Bindings that use Impersonation and Delegation • 58
Web Service Inputs • 7
Windows 7 and Windows Vista • 23
Windows Server 2003 • 18
Windows Server 2008 R2 • 22
Windows-integrated Security • 48
wsHttpBinding • 55
wsHttpBinding Sample • 64