2015 OSIsoft TechCon

Learn the Basics of PI Programming
# Table of Contents

Table of Contents .......................................................................................................................................... 1

## 1 Introducing the PI System..................................................................................................................... 3

1.1 Simple as PI? ........................................................................................................................................ 3

1.2 The Interface ....................................................................................................................................... 4

1.3 The PI Server ....................................................................................................................................... 4

1.3.1 What is a PI AF Element? ............................................................................................................. 4

1.3.2 What is a PI AF Attribute? ........................................................................................................... 5

1.3.3 What is a PI Point or PI Tag? ...................................................................................................... 5

1.4 Directed Exercise – Seeing the Sights with PI System Explorer .................................................... 6

1.5 Time and the PI System..................................................................................................................... 8

1.5.1 PI System Time Abbreviations............................................................................................... 8

1.5.2 PI System Time Expressions .................................................................................................. 9

1.5.3 Timestamp Specification......................................................................................................... 9

1.5.4 How Does the PI System Adjust for Time Zones and DST? ................................................. 10

1.6 Introducing PI Developer Technologies .......................................................................................... 11

1.7 Directed Exercise – Get in Shape with PI Square........................................................................... 12

1.8 Choose Your Own Adventure ....................................................................................................... 13

## 2 PI AF SDK ............................................................................................................................................. 14

2.1 Introduction ....................................................................................................................................... 14

2.2 Exercise – Exploring PI AF SDK Reference ................................................................................... 15

2.3 Exercise – Building PI AF Elements with PI AF SDK ................................................................. 17

2.4 Exercise – Designing a PI AF SDK Client Application ............................................................... 20

2.5 Exercise – Accessing the PI Data Archive through Rich Data Access ......................................... 23

2.6 Where to Go Next ............................................................................................................................ 26

## 3 PI Web API ........................................................................................................................................... 27

3.1 Introduction ....................................................................................................................................... 27

3.2 Exercise – Exploring PI Web API Reference ................................................................................. 28

3.3 Exercise – Simple PI Web API Calls ............................................................................................. 29

3.4 Exercise – Making calls to the Index Search Crawler ................................................................... 34
3.5 Exercise – Developing a Web Application With jQuery and PI Web API ........................................... 36
3.6 Where to Go Next .................................................................................................................................. 41

4 PI SQL Framework ....................................................................................................................................... 42
4.1 Introduction ........................................................................................................................................... 42
4.2 Review of SQL Select Statements ........................................................................................................... 43
4.3 Exercise – A Salute to PI SQL Commander ............................................................................................ 43
4.4 Exercise – Serving Data with PI SQL Data Access Technologies ............................................................ 46
  4.4.1 A Table for One, Please .................................................................................................................. 46
  4.4.2 A Table for Two, Please ................................................................................................................ 46
  4.4.3 A Table for Three, Please .............................................................................................................. 47
  4.4.4 A Table for Four, Please .............................................................................................................. 47
  4.4.5 Exercise – A Table with a View, Please ......................................................................................... 48
4.5 Exercise – Data Transpositions .............................................................................................................. 49
4.6 Where to Go Next .................................................................................................................................. 51

OSIsoft Virtual Learning Environment ........................................................................................................ 51
1 Introducing the PI System

Whether you’re developing an application that requires that you retrieve data from somewhere or integrating different systems together, it can be difficult or frustrating to get access to the data you need to design your solution. This is especially maddening when you’re working with multiple standards, geographical locations, or departments within the company. At OSIsoft, we think that, as long as you have permission, it should be easy to access data within your organization. When you want to find information about an asset or process, there should be a single location where you can find it, and you should be able to extract it using any standard you want. That’s why we made the PI System, and that is exactly what it does for you.

The PI System is an infrastructure that sits between data sources in a facility, and the users and applications that require access to that data. A “data source” can be pretty much anything, and the data can be accessed by using a huge number of methods, standards or client tools. The PI System is essentially a “universal translator” for plant data. Rather than developing with many different standards, requiring specific languages or operating systems to access process data from different places in the plant, you’ll only need to use one. This class will focus on what kind of data can be accessed using the PI System and what options you have when accessing this data programmatically.

For example, you are tasked to design an application to report on pump maintenance in an organization. To do this, you have to get data from several disparate sources. You’d probably require time-series data from the Pump SCADA system, times, dates and names from a maintenance database, and maybe location based weather data that you have to scrape from a government website. If you made the use of a PI System, you’d have a PI Server in the center referencing and gathering all of this data, ready to be accessed easily with your choice of standard methods of data retrieval.

1.1 Simple as PI?

At its simplest, PI is a data infrastructure. A simple PI System consists of:

- the data source
- the data collector for that data source
- the PI Data Archive combined with its Asset Framework server (The “PI Server”)
- and a client - an appropriate visualization tool on a PC, or an application using one of the PI Development Technologies, which we’ll study in this class.
1.2 The Interface
PI Interfaces, usually installed within the control system network, constantly gather time-series data exposed by a source, and continuously send that data through to the PI Data Archive. For example, your organization has a SCADA system that exposes data using the OPC standard.

1.3 The PI Server
The central component of the PI System is the PI Server, which stores, categorizes and references data. The PI Server is made up of two parts, the PI Data Archive, which efficiently stores time-series data, and PI Asset Framework (PI AF), which categorizes and references not only time-series data, but also relational data from all over the organization. Together, these server applications act as a high performance view into any plant data you require, whether it be in real time, historical, static, relational, or even occurring in the future.

In the first part of this lab, we're going to take a look at what can be stored or referenced in the PI Server, by viewing PI Asset Framework with an administration tool called PI System Explorer. But first, we should define some terms we’re going to use throughout this lab.

1.3.1 What is a PI AF Element?
The PI AF Server is a part of the PI System. It contains asset or “metadata” usually organized according to the assets containing the attributes being monitored. An Element is an object in PI AF that can represent an asset, geographical location or process. To generalize a little, an Element is a “thing” that can be compared to another like “thing”. Two boilers can be compared, two plants can be compared, two different processes in a plant can be compared, and two employees can be compared. There’s one exception here, in that you should not define an event – such as a batch or downtime – as an element. There’s a different object type for that, the Event Frame. Elements can be thought of as “permanent” objects.

PI AF can be helpful to users of the PI Data Archive who know the assets, but are not familiar with attribute nomenclature. With Elements, data can be located without understanding the technical details of each piece of equipment. Organized assets help find all of the attributes associated with a specific piece of equipment.
1.3.2 What is a PI AF Attribute?
Attributes represent a unique property associated with an object in PI AF. If we have a “Tank” element, as above, then it might have attributes such as Density, Level, Manufacturer, Serial Number, Mass and Volume.

- An attribute may be a constant, like Serial Number.
- It could be a value from an internal PI AF table or a value from an external database, for example this organization has a table containing every tank’s Serial Number and their corresponding Manufacturers.
- It could be a pointer to a data stream in the PI Data Archive, like the Level of liquid in the tank.

A PI AF attribute is simply a single point of measurement, in the context of its owning AF object.

1.3.3 What is a PI Point or PI Tag?
For all intents and purposes, PI Point, PI Tag and PI Data Stream have all been used historically and are all interchangeable terms. A PI Point It is a unique storage point for a stream of data in the PI Data Archive. It is simply a single point of measurement. An example of a PI Point is the current level of liquid in a tank, read from a named stream of data from a SCADA system. The time-series data coming from the tank level is stored in the PI Data Archive, then referenced as above in PI AF to better represent it to a user or application.
1.4 Directed Exercise – Seeing the Sights with PI System Explorer

Let’s have a look at the data that the PI System can store or reference. To do this, we’ll use PI System Explorer (PSE). PI System Explorer is more of an administration tool than a client tool, but lets us easily view the Asset Framework structure in the PI Server. Think of PSE as something akin to SQL Server Management Studio, but for PI.

Exercise Objectives

- List the kinds of data that can be shown in PI System Explorer (PSE)
- Navigate the PI Asset Framework (AF) hierarchy using PI System Explorer

Approach

1. Open PI System Explorer (PSE) using the shortcut on the desktop.

2. Ensure you are connected to the “OSIsoft Enterprises” database. You can tell this by looking at the title bar of the PSE window, which should say “\PISRV1\OSIsoft Enterprises...”. If you are not connected to this database, select the “Databases” button in the top left of the window, then select the “OSIsoft Enterprises” database and press “OK”.

3. This database represents the processes and assets found in oil refineries and other associated processes owned by this organization. We’re going to use this database for most of the lab. Expand out the hierarchy Elements > Houston > Cracking Process > Boiler-210. Select Boiler-210.

4. Click on the “Attributes” tab in the right hand pane.

5. Take a look at the kind of data that is stored in attributes under the boiler. Navigate around the elements in the left hand pane and see what kind of data is under other assets.

6. You will discuss the following questions as a class:
   - What kinds of data can be stored in Attributes?
   - Is there any non-time series data in PSE?
• Would it make sense to put ambient temperatures of a city in a PI AF Database? How would this look?

• Would it make sense to put company employees and their details in AF? How would this look?

• Can you think of something interesting you've seen at the convention so far that developers have done with the PI System and PI Asset Framework?
1.5 Time and the PI System

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

1.5.1 PI System Time Abbreviations

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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Time-unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>second</td>
</tr>
<tr>
<td>m</td>
<td>minute</td>
</tr>
<tr>
<td>h</td>
<td>hour</td>
</tr>
<tr>
<td>d</td>
<td>day</td>
</tr>
<tr>
<td>w</td>
<td>week</td>
</tr>
<tr>
<td>mo</td>
<td>month</td>
</tr>
<tr>
<td>y</td>
<td>year</td>
</tr>
</tbody>
</table>

Table 1: Time Unit Abbreviation

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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full</th>
<th>Reference-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td></td>
<td>Current time.</td>
</tr>
<tr>
<td>t</td>
<td>today</td>
<td>00:00:00 (midnight) of the current day</td>
</tr>
<tr>
<td>y</td>
<td>yesterday</td>
<td>00:00:00 (midnight) of the previous day</td>
</tr>
<tr>
<td>sun</td>
<td>sunday</td>
<td>00:00:00 (midnight) on the most recent Sunday</td>
</tr>
<tr>
<td>mon</td>
<td>monday</td>
<td>00:00:00 (midnight) on the most recent Monday</td>
</tr>
<tr>
<td>tue</td>
<td>tuesday</td>
<td>00:00:00 (midnight) on the most recent Tuesday</td>
</tr>
<tr>
<td>wed</td>
<td>wednesday</td>
<td>00:00:00 (midnight) on the most recent Wednesday</td>
</tr>
<tr>
<td>thu</td>
<td>thursday</td>
<td>00:00:00 (midnight) on the most recent Thursday</td>
</tr>
<tr>
<td>fri</td>
<td>friday</td>
<td>00:00:00 (midnight) on the most recent Friday</td>
</tr>
<tr>
<td>sat</td>
<td>saturday</td>
<td>00:00:00 (midnight) on the most recent Saturday</td>
</tr>
<tr>
<td>YYYY</td>
<td></td>
<td>00:00:00 (midnight) on the current day and month in year YYYY</td>
</tr>
<tr>
<td>M-D or M/D</td>
<td></td>
<td>00:00:00 (midnight) on the Dth day of month M in the current year</td>
</tr>
<tr>
<td>DD</td>
<td></td>
<td>00:00:00 (midnight) on the DDth day of the current month</td>
</tr>
</tbody>
</table>

Table 2: Reference-time Abbreviations
1.5.2 PI System Time Expressions

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

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

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

You can only include one time offset in an expression. Including multiple offsets can lead to unpredictable results. For example, the following time expressions are not valid:

\[ *+1d+4h \quad t-1d+12h \]

1.5.3 Timestamp Specification

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

Fixed times

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>23-aug-12</td>
<td>3:00 p.m. on August 23, 2012</td>
</tr>
<tr>
<td>25-sep-12</td>
<td>00:00:00 (midnight) on September 25, 2012</td>
</tr>
</tbody>
</table>

Reference-time Abbreviations

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

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

Reference-time Abbreviations or Fixed Times with a Time Offset

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>*-1h</td>
<td>One hour ago</td>
</tr>
<tr>
<td>t+8h</td>
<td>08:00:00 (8:00 a.m.) today</td>
</tr>
<tr>
<td>Time Offset</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>y-8h</td>
<td>16:00:00 (4:00 p.m.) the day before yesterday</td>
</tr>
<tr>
<td>mon+14.5h</td>
<td>14:50:00 (2:30 p.m.) last Monday</td>
</tr>
<tr>
<td>sat-1m</td>
<td>23:59:00 (11:59 p.m.) last Friday</td>
</tr>
<tr>
<td>1-jan-11-1d</td>
<td>00:00:00 (12:00 a.m.) 31 December 2010</td>
</tr>
</tbody>
</table>

**Time Offsets**

Entered alone in a time field, time offsets specify a time relative to an implied reference-time. The implied reference-time depends on the field where you enter the expression:

- For a start time, the reference-time is the current clock time.
- For an end time, the reference-time is the start time.
- For a single time stamp, the reference-time is the current clock time.

<table>
<thead>
<tr>
<th>Time field</th>
<th>Input</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time</td>
<td>-1d</td>
<td>One day before the current clock time (24 hours before the current clock time)</td>
</tr>
<tr>
<td>End time</td>
<td>+6h</td>
<td>Six hours after the start time</td>
</tr>
<tr>
<td>End time</td>
<td>-30m</td>
<td>30 minutes before the start time</td>
</tr>
<tr>
<td>Time stamp</td>
<td>-15s</td>
<td>15 seconds before the current clock time</td>
</tr>
</tbody>
</table>

**1.5.4 How Does the PI System Adjust for Time Zones and DST?**

The short answer is: we do not!

When we collect data, we convert it to UTC (Universal Coordinated Time), or what used to be called Greenwich Mean Time (GMT). This means that each day has exactly 24 hours. The local machine clock of the user looking at the data makes any adjustments for time, such as time zone or DST.

If your region observes DST, once a year the day will look like it has 23 hours and another 25, but the PI Server never knows anything other than 24-hour days.

In addition, because the clients and PI Server know what time zone they are in, the data can be viewed with respect to the server time or the client time. This is determined by a setting in the client tool.
1.6 Introducing PI Developer Technologies

PI Developer Technologies (also called PI System Access, or PI Data Access) are a family of products designed to support the implementation of custom applications on top of the PI System, as well as the integration of PI System data with other applications and business systems such as Microsoft Office or SQL Server, Enterprise Resource Planning systems (ERPs), reporting and analytics platforms, web portals, geospatial and maintenance systems, just to name a few. The PI System Access suite covers a wide range of use cases in various environments, programming languages, operating systems and infrastructures. This chapter will cover the different technologies in the PI System Access family. The technologies that may be covered in the rest of this class are detailed below:

- **PI AF SDK** provides comprehensive, high performance, Windows based .NET programmatic access to the PI System.
- **PI Web API** exposes a REST API, enabling operating system and device independent programmatic access to the PI System.
- **PI SQL Framework** exposes the data on a PI Server as if it were on a relational database. This lab will use PI OLEDB Enterprise, one of the three recommended products in this framework.

There are more ways to access the PI System programmatically, but the above three cover a vast majority of use cases. After this introduction, you’ll get some experience with one of the above three technologies. You will have to make a choice on which one you’ll study for the rest of the workshop.

The below table should enable you to make your decision more easily:

- **PI AF SDK**
  - Has the best performance of all PI Development Technologies.
  - More methods and options available than any other Technology.
  - Limited to Windows operating systems.
- **PI Web API**
  - Operating system and programming language agnostic.
  - Lower performance than PI AF SDK.
  - Limited methods and options available compared with PI AF SDK.
- **PI SQL Framework**
  - Compatible with other systems that use Structured Query Language (SQL).
  - Lower performance than PI AF SDK.
  - Limited methods and options available compared with PI AF SDK.
  - Limited to Read-only access to the PI System

Documentation for these products can be found at the following location:

https://livelibrary.osisoft.com/LiveLibrary/content/en/DeveloperTechnologies-v1

You can find a shortcut to this location on your desktop titled “Documentation”.

11 | Page
1.7 Directed Exercise – Get in Shape with PI Square

By far, the best resource for developing with the PI System is PI Square. PI Square is an online community that enables OSIsoft’s customers, partners and employees to converse and share ideas. More importantly to you, the PI Developers Club space that enables OSIsoft’s customers, partners and employees to converse and share ideas about development.

PI Square will be used throughout the rest of this lab for references and exercises.

Exercise Objectives

- Log in to PI Square
- Find exercise solutions using PI Square

Approach

1. Go to the PI Developers Club section of PI Square by following this link (you can also find this on your desktop):

   https://pisquare.osisoft.com/community/developers-club

2. Click the “Log in” link on the top right of the page

3. If you already have an SSO account, use it to log in. If you don’t, create one with “Register”. If you’re having trouble logging in, call the instructor over for assistance.

4. Take a look at the “PI .NET FrameWork (PI AF SDK)”, “PI SQL Data Access Technologies” and “RESTful PI System Access” spaces. If you ever have questions on the use of these technologies, these are the spaces to post in.

5. Type the following into the search box in the top right of the page and hit enter:

   learn_basics_pi_programming

Throughout the rest of this lab, you’ll be doing hands-on exercises. If you need to copy or paste code, or would like to see exercise solutions, you’ll find what you need when performing this search and navigating to the corresponding article. All content published relating to this lab is published to PI Square with the above tag.
1.8 Choose Your Own Adventure

The rest of this class will be self-paced. Each self-paced module will take around one to two hours to complete, so you may be able to do more than one. You’ll now have to choose which module you’d like to complete in full. We suggest to go with a technology that you’ll be using after this workshop, when you get back to work.

Think back to a nifty application or presentation that you saw in the conference so far that used one of the PI Development Technologies (more specifically, PI AF SDK, PI Web API, or any of the products in the PI SQL Framework family). Write down what it did, and which technology it used:

Which one of the three technologies would you like to work with for the rest of this workshop, PI AF SDK, PI OLEDB Enterprise, or PI Web API?

Why?

Once you’ve written down your chosen technology, skip ahead to the corresponding page number. It’s like a choose-your-own-adventure book, but even more fun.

PI AF SDK ..................................................................................................................................................... 14
PI Web API ................................................................................................................................................... 27
PI SQL Framework ....................................................................................................................................... 42
2  PI AF SDK

2.1  Introduction

If you need to develop your own application to interact with the PI System, you have many options. All of the family of products that make up the PI SQL Framework, PI Web API, or PI AF SDK are at your disposal. However, if you want to get the best performance out of your application, the PI AF SDK has the greatest potential of all. This is all assuming that you’re developing, and running, the application on a Microsoft Windows environment.

The PI AF SDK provides a comprehensive, Windows based programmatic interface to the PI System. It provides an object-oriented approach to interact with AF structures and data, as well as the PI Data Archive directly. The PI AF SDK is designed for easy access from Microsoft .NET languages like Visual Basic .NET, C#, and Managed C++. The development machine and the machine that will eventually run any applications developed with PI AF SDK must have at least Microsoft .NET Framework 3.5 installed. However, for full use of all features we recommend using the PI AF SDK .NET 4.5 assemblies, which would require .NET 4.5 to be installed on both the development machine, and any other machines that will run your application.

There are many reasons why you might choose to develop with the PI AF SDK:

- You can quickly build out complex PI AF hierarchies with the PI AF SDK. Many of our customers and partners prefer to build AF hierarchies this way, rather than using the PI AF Builder plugin to Microsoft Excel.

- You have the power to create rich and customized Windows client applications with PI AF SDK. The client applications that make up the PI Visualization Suite were developed using the PI AF SDK and its predecessors.

- The PI AF SDK is well suited to developing middleware applications sitting between the PI System and other systems. PI AF SDK gives you flexibility on how you want to integrate other systems with the PI System, and allows you to do so in the most resource efficient way possible. It is also possible to automate element generation and/or synchronization between PI AF and other systems.

- ... and many more.

During this chapter we will examine PI AF SDK documentation and develop a few simple applications using the .NET 4.5 assemblies.
2.2 Exercise – Exploring PI AF SDK Reference

Exercise Objectives

- Locate the PI AF SDK Reference Manual
- Find methods to accomplish tasks

Approach

An important resource when developing with the PI AF SDK is the AFSDK.chm reference file. If you need this file in the future, it can be downloaded from the OSIsoft download center. Follow these steps to become more familiar with this file. You should use this file throughout this lab to look up methods that are used.

1. Open PI System Explorer (PSE) from the desktop of your machine and ensure you are connected to the “OSIsoft Enterprises” database. We will reference this database when explaining some of the namespaces and classes in the PI AF SDK.

2. Open AFSDK.chm from the desktop of your machine.

3. The PI AF SDK contains many namespaces, organized into hierarchies. Ensure that the “Contents” tab is showing, and expand out the “Hierarchies” heading and click on “PISystem Hierarchy”.

4. The PISystem Hierarchy shows us the namespaces in PI AF SDK. If you look further down the page, the OSIsoft.AF.Asset namespace contains classes relating to PI AF Elements, and a few other related classes. Note that inside the OSIsoft.AF.Asset namespace each class appears to be doubled – it exists as a plural and a singular. Each plural class is a collection of objects of its corresponding singular class. For example, the Elements class represents a collection of Element objects. Click on the “AFElement” class under the OSIsoft.AF.Asset namespace. Ensure you click on the non-pluralized form “AFElement”.

5. We are now at the description of the AFElement class. We see a hierarchy describing all of the classes with relationships to this class. The arrows and lines indicate relationships between objects. Generally, you can get an AFElement object from an AFElements object, and so on and so forth. We’d like to see everything we can do with an AFElement object, so click on “Members” right at the top of the screen underneath the title “AFElement Class”.

6. Here you’ll see the contents of the AFElement class. Scroll down and pay special attention to the Find* methods. Most of your work with Elements, Attributes and Points will be using Find* methods like these.

Note: the PI AF SDK, for backwards compatibility reasons, refers to a PI AF Server as a PISystem, and a PI Data Archive as a PIServer. This is confusing, but OSIsoft takes great care in backwards compatibility of its products and this is a necessary complication.
Use the PI AF SDK reference to answer the following questions. Try to get used to navigating around with the hierarchies at the top of the left pane, and the different links right below the title of the current selection.

1. Most applications you write with the PI AF SDK will require that you first connect to a PI AF Server. What PI AF SDK method would you use to connect to a PI AF Server?

2. If you’re not connecting to a PI AF Server in your code, then you’ll be connecting directly to a PI Data Archive. This is most often done when developing custom interfaces. What PI AF SDK method would you use to connect to a PI Data Archive?

3. During the next exercise we’ll be using the AFEElements.Add method. What does the AFEElements.Add method do?

4. After making changes to PI AF, your changes are kept in a sandbox until checked in. Only checked in changes will be visible to other users. What PI AF SDK method would you use to “check in” all changes made to a single PI AF Database?
2.3 Exercise – Building PI AF Elements with PI AF SDK

One of the common uses of PI AF SDK is to automatically build out large, complex databases. In this exercise we take a look at how we can access a PI AF asset hierarchy with PI AF SDK, and make some changes through code.

Disclaimer: This sure looks like one of those tried and true “copy and paste the code snippets” exercises. But... it’s not. When you encounter problems, put your debugging hat on and try to figure out what’s happening. The steps will tell you if further action needs to be taken.

Exercise Objectives

- Start a project and reference PI AF SDK
- Connect to a PI AF Server
- Read data from a PI AF Server
- Make changes to the PI AF Database
- Check in/Check out your changes to the PI AF Database

Approach

1. Open Visual Studio on your development machine.
2. Create a new project, and select Visual C# > Console Application.
3. Choose a name for the project, and select OK.
4. Add a reference to the PI AF SDK assembly to the project:
   i. Click on Project > Add Reference...
   ii. Select Assemblies > Extensions.
   iii. Check the checkbox next to OSIsoft.AFSDK. Ensure you check the 4.0 version of the extension.
   iv. Press OK.
5. Add the following “using” statements to the top of your code:

```csharp
using OSIsoft.AF;
using OSIsoft.AF.Asset;
```

6. Next, add the following code to your main method to explicitly connect to your PI AF Server and a set AF Database. We’ll be using the “OSIsoft Enterprises” database:

```csharp
// Connect to the default AF Server and OSIsoft Enterprises Database
PISystems pisystems = new PISystems();
PISystem pisystem = pisystems.DefaultPISystem;
pisystem.Connect();
AFDatabase OSIsoftEnterprises = pisystem.Databases["OSIsoft Enterprises"]; Console.WriteLine("Database: {0}", OSIsoftEnterprises.Name);
```
7. Make a breakpoint by clicking in the grey margin next to the “static void...” line, then click the Start button with the green triangle (or from the DEBUG menu, click Start Debugging) to start a debugging session and check your work so far. Step through your code with F10 until you pass the last line. Select the command window to check on progress. A proper solution at this point will have the following output:

```
Database: OSIsoft Enterprises
```

8. Stop debugging with DEBUG > Stop Debugging.

9. The OSIsoft Enterprises element hierarchy begins with locations, which contains processes, which contains equipment. Once you’re successfully retrieving a reference to the OSIsoft Enterprises database, add code to retrieve references to the Tucson location, the Distilling Process in Tucson, and the Equipment used in that process:

```csharp
// Retrieve specific elements involved in the distilling process in Tucson
AFNamedCollectionList<AFElement> tucsons = AFElement.FindElements(OSIsoftEnterprises, null, "Tucson", AFSearchField.Name, true, AFSortField.Name, AFSortOrder.Ascending, 10);
if (tucsons.Count < 1)
{
    return;
}
AFElement tucson = tucsons[0];
AFElement distilling = tucson.Elements["Distilling Process"];  
AFElement equipment = distilling.Elements["Equipment"];  
```

10. The distilling equipment in Tucson includes a boiler named B-117. Use the reference to the Tucson distilling process to change the description for this boiler. In order to modify an element, you must first check it out, then check it back in when your modifications are complete:

```csharp
// Edit the existing boiler used in the distilling process in Tucson
AFElement boiler = equipment.Elements["B-117"];  
Console.WriteLine("Boiler Description: {0}", boiler.Description);
boiler.CheckOut();
boiler.Description = "Original boiler used in Tucson.";
boiler.CheckIn();
Console.WriteLine("Boiler Description: {0}", boiler.Description);
```

11. Click the Start button with the green triangle (or from the DEBUG menu, click Start Debugging) to start a debugging session and check your work so far. A proper solution at this point will have the following output:

```
Database: OSIsoft Enterprises
Boiler Description: Distilling B-117
Boiler Description: Original boiler used in Tucson.
```

```csharp
```
12. Now add a second boiler named B-118 in Tucson. Note that both the new boiler and the Tucson distilling Equipment element need to be checked in. Also note that when creating the boiler, we’re using an already defined Template that is stored in the PI AF Server. You can use PI System Explorer to see what this template looks like by using the Library tab in the bottom left corner of the application > Templates > Element Templates > Boiler.

```csharp
// Add another boiler to the Tucson distilling process
newBoiler.Description = "New boiler to be used in Tucson."
newBoiler.CheckIn();
equipment.CheckIn();
Console.WriteLine("Boiler Description: {0}", newBoiler.Description);
```

13. Click the Start button with the green triangle (or from the DEBUG menu, click Start Debugging) to start a debugging session and check your work so far. A proper solution at this point will have the following output:

![Output](image)

Note: This will also be unsuccessful without further editing of the above code and you will have to make changes to make it work. Only move on after you’ve got the above output for the exercise.

14. Note that there is no difference this time between the description of B-117 before and after it is checked back in, because the new description was already checked-in during a prior debugging session. Press Enter to stop the debugging session and return to Visual Studio.

15. Without making any modifications to your source code, immediately start another debugging session, and note that we have an unhandled exception:

![Error](image)

16. Add code to your program to prevent this error from happening. Debug to confirm that your code can handle this scenario properly. Save your work and move on to the next exercise.
2.4 Exercise – Designing a PI AF SDK Client Application

Exercise Objectives

- Design a Windows Forms application using PI AF SDK
- Explore the built-in PI AF SDK forms objects.

Problem Description

PI AF SDK comes with user interface objects for common client side tasks. These are incredibly useful when starting out developing client tools. Follow the steps below to build a simple client application to connect to an AF Server and display an element hierarchy tree.

Approach

1. Open Visual Studio on your development machine
2. Create a new project, and select Visual C# > Windows Forms Application. Name the project “PIAFApplet”.
3. Add a reference to the PI AF SDK assembly to the project. The procedure may be slightly different depending on your version of Microsoft Visual Studio:
   i. Click on Project > Add Reference...
   ii. Select Assemblies > Extensions.
   iii. Check the checkboxes next to OSIsoft.AFSDK and OSIsoft.AF.UI. Ensure you check the 4.0 version of each extension.
   iv. Press OK.
4. Add the PI AF SDK objects to your project:
   i. On the “Form1.cs” design pane, select “Toolbox” in the top left, underneath the upper toolbar.
   ii. Right click on “All Windows Forms” below “Toolbox”, and select “Choose Items”. Wait until the items are fully loaded.
   iii. Check the box next to the following items. Using the “Filter” box will help you find them.
      - PISystemPicker
      - AFDatabasePicker
      - AFTreeView
   iv. After importing, you can find them by searching for AF and PI from the search under the toolbox heading.
5. Click and drag each of the new items out to your form. Your form should look like this:
6. Configure the objects for ease of use:
   
i. Select the “PISystemPicker” object inside your Form1 and locate the “Properties” pane in the lower right of the window. Scroll down to the properties starting with “ShowBegin” and change all corresponding options to mirror the picture below:

   ![Properties settings](image)

   ii. Scroll up to the “Connect Automatically” property, and change that property to “True”. This will make the applet automatically connect to the PI AF Server.

   iii. Select the “AFDatabasePicker” and change its properties so the only properties in the AF section that are “True” are ShowComboBox, ShowNoEntries and ShowImages.
7. We now have objects on our form, but they don’t interact with each other. We need to write code to connect these objects. Double click on the form and add the following code underneath `public Form1():`

```csharp
InitializeComponent();
afDatabasePicker1.SystemPicker = piSystemPicker1;
```

While you’re at it, add the following “using” statements to the top of your code:

```csharp
using OSIsoft.AF;
using OSIsoft.AF.Asset;
using OSIsoft.AF.Time;
using OSIsoft.AF.UnitsOfMeasure;
```

**Tip** These “using” statements are always useful to have. We recommend getting into the habit of adding them whenever you’re using the PI AF SDK.

8. Back on your “Form1.cs [Design]” tab, select your “AFDatabasePicker” and look for the lightning bolt “Events” button in the Properties pane in the bottom right of the window. Select it.

9. Scroll to “SelectionChange” and double-click on the white space next to it. This will add a default event handler that we can use to input our code.

10. In the new SelectionChange event handler method, write the following code:

```csharp
AFDatabase db = afDatabasePicker1.AFDatabase;

//Clear the treeview
afTreeView1.AFRoot = null;

if (db != null)
{
    //Set the root of the Treeview
    afTreeView1.AFRoot = db.Elements;
}
```

11. Start debugging your code. Use your interface to connect to your PI AF server, select your database and navigate through your element tree.

12. Stop debugging and save your project.
2.5 Exercise – Accessing the PI Data Archive through Rich Data Access

Exercise Objectives

- Develop a custom console application in order to get data from the PI Data Archive.
- Explore the objects from OSIsoft.AF.PI namespace.

Problem Description

PI AF SDK provides the ability to access the PI Data Archive directly without the need to use attribute references in the PI AF Server. This is useful when you need highly efficient writes to the PI Data Archive. Follow the steps below to build a console application that will:

1. Connect to the default PI Data Archive.
2. Get the snapshot, recorded and interpolated values from the PI Point Sinusoid.
3. Find all the PI Points whose name contains the letter A.
4. Get the following PI Point attributes from the sinusoid: location1, descriptor, excdev.
5. Write a value to the PI Point Sinusoid.

Approach

1. Open Visual Studio on your development machine
2. Create a new project, and select Visual C# > Console Application. Name the project “AccessingThePIDataArchive”.
3. Add a reference to the PI AF SDK assembly to the project. The procedure may be slightly different depending on your version of Microsoft Visual Studio:
   i. Click on Project > Add Reference...
   ii. Select Assemblies > Extensions.
   iii. Check only OSIsoft.AFSDK. Ensure you check the 4.0 version of the extension.
   iv. Press OK.
4. Add PI AF SDK namespace to the using Directive.

    ```csharp
    using OSIsoft.AF.PI;
    using OSIsoft.AF;
    using OSIsoft.AF.Time;
    using OSIsoft.AF.Asset;
    ```
5. Connect to the default PI Data Archive.

    PIServers myPIServers = new PIServers();
    PIServer myPIServer = myPIServers.DefaultPIServer;
    myPIServer.Connect();

6. Find the Sinusoid PI Point.

    PIPoint myPIPoint = PIPoint.FindPIPoint(myPIServer, "sinusoid");

7. Get the snapshot value.

    AFValue SinusoidSnapshot = myPIPoint.Snapshot();
    Console.WriteLine("Snapshot from Sinusoid: ", SinusoidSnapshot.Value + "\t" + SinusoidSnapshot.Timestamp.LocalTime);

8. Get the recorded values from the last day.

    AFValues SinusoidRecorded = myPIPoint.RecordedValues(new AFTimeRange("*-1d", "*"), OSIsoft.AF.Data.AFBoundaryType.Inside, ",", false);
    Console.WriteLine("Showing Recorded data from Sinusoid...");
    foreach (AFValue myValue in SinusoidRecorded)
    {
        Console.WriteLine(myValue.Value + "\t" + myValue.Timestamp.LocalTime);
    }

9. Get the interpolated values from the last day with one hour of interval.

    AFValues SinusoidInterpolated = myPIPoint.InterpolatedValues(new AFTimeRange("*-1d", "*"), new AFTimeSpan(0, 0, 1, 0, 0, 0), ",", false);
    Console.WriteLine("Showing Interpolated data from Sinusoid...");
    foreach (AFValue myValue in SinusoidInterpolated)
    {
        Console.WriteLine(myValue.Value + "\t" + myValue.Timestamp.LocalTime);
    }

10. Find all the PI Points that starts with the letter s.

    PIPointQuery query1 = new PIPointQuery(PICommonPointAttributes.Tag, OSIsoft.AF.Search.AFSearchOperator.Equal, "s*");
    IEnumerable<PIPoint> foundPoints = PIPoint.FindPIPoints(myPIServer, new PIPointQuery[] { query1 });
    foreach (PIPoint currentPIPoint in foundPoints)
    {
        Console.WriteLine(currentPIPoint.Name);
    }
11. Get the descriptor, location1 and excdev from the Sinusoid PI Point.

```csharp
IDictionary<string, object> PIPointAttributes = myPIPoint.GetAttributes(new string[] { "descriptor","location1","excdev" });
Console.WriteLine("\n\nSinusoid PI Point attributes: \n");
Console.WriteLine("descriptor = " + PIPointAttributes["descriptor"]);
Console.WriteLine("location1 = " + PIPointAttributes["location1"]);
Console.WriteLine("excdev = " + PIPointAttributes["excdev"]);
```

12. Start debugging your code checking the content of the variables on each line in order to have a glimpse about the logic of the program.

13. Write a few lines of code to change the current value of the point Sinusoid to 200 with a timestamp of the current time. You’ll need to use the PIPoint.UpdateValue method. Search the documentation for more information on this method. After changing its value, print the new value and timestamp to the console.
2.6 Where to Go Next

This section has been a light introduction to PI AF SDK. If you have any questions or would like more information about the product, please check the associated section in PI Square:

https://pisquare.osisoft.com/community/developers-club/pi-net-framework-pi-af-sdk

If you still have time during this workshop, feel free to start on the other sections of this booklet. See below for a list of page numbers:

PI Web API................................................................................................................................................... 27
PI SQL Framework....................................................................................................................................... 42
In a world that is increasingly full of “Internets of Things”, it’s getting more and more important to integrate systems of different functions and standards with each other. The PI Web API makes it easy to do just this; integrate the PI System with non-Windows systems such as Smartphones, UNIX servers or other network-connected devices. The PI Web API gives rich, cross-platform access to the PI System from any application that can send and receive HTTP or HTTPS. In other words, adding a PI Web API Server to a PI System will enable client access to the PI Server via a RESTful API.

Under normal operation, a client makes HTTP calls to the PI Web API Service. The PI Web API Service gets the data needed from the PI Data Archive and PI Asset Framework servers, and returns the data in the form of JSON to the requesting client. The PI Web API also features an Index Search Crawler, a highly efficient way to search for objects within the PI System. Every now and again, this service will crawl the entire system, keeping an index of available objects for quick searching by clients.

We will start exploring simple queries with a manual client called Postman, a plugin to Google Chrome, and also experiment with the Index Search Crawler. Then, we will develop a web application that will update values to the PI System through PI Web API.
3.2 Exercise – Exploring PI Web API Reference

Exercise Objectives

- Explore the help page
- Explore the documentation

Approach

Open Google Chrome to the following URL to visit the PI Web API help page. Once you’re there, use it to answer the following questions:

https://PISRV1/piwebapi/help

1. Which PI Web API method would you use for seeing the names and properties of child elements of a given element in your PI AF database?

2. What do the “Stream” category of PI Web API methods do?

3. You are designing an application that requires the results of time-weighted averages of a few different data streams. Can you get the results of a summary calculation like this with PI Web API? Which method would you use?

4. Your application needs to be able to write back to a PI AF Attribute. Which Method would you use to accomplish this?
3.3 Exercise – Simple PI Web API Calls

Activity Objectives

- Make simple calls to the PI Web API
- Determine the structure of the data returned

Approach

For the next few exercises, we’re going to use a PI AF Database that contains weather data for a few cities in the USA. In preparation, let’s take a look at this database:

1. Open up PI System Explorer using the shortcut on your desktop.
2. Click on the “Database” button in the top left of the window.
3. Select the USAWeather database and click OK.

You should now see the following structure:

As you can see, the USAWeather database element hierarchy has a “Cities” element with five child elements named “Chicago”, “Los Angeles”, “New York”, “San Francisco” and “Washington”. Each city element is derived from the “City” element template which has five attribute templates: “Cloud cover”, “Humidity”, “Pressure”, “Temperature”, “Visibility” and “Wind Speed”.
We could take a look at this database using PI Web API using a simple browser, but we are going to both read and write to it. Reading and writing requires us to make both GET and POST HTTP requests. So, we’ll need to use an HTTP client that supports both of these. Postman is an extension of Google Chrome that makes it easy to test RESTful services by manually crafting HTTP requests, and does the task we’re looking for. While you are learning to use the PI Web API you will most likely make many, many queries using a tool like this.

You will find Postman on your training machine. If you’d like to revisit this exercise at a later date, you can download and install Google Chrome from google.com/chrome/browser and, once it is installed, get the Postman client from getpostman.com.

Complete the following tasks below to become more familiar with the query structure when making calls to the PI Web API:

1. Open Postman by double clicking on the shortcut on your desktop.

2. Inside Postman, type the following into the address field, making sure that the GET method is selected.

   ![Image](https://PISRV1/piwebapi/)

   After typing, hit enter or press the “Send” button. This returns some top level properties of the PI Web API. Note that none of these properties are actual production or configuration data, they are links to other queries and the help file. We really don’t get much from this query.

3. Let’s go one level deeper. Write the following in the address field and press enter or click “Send”:

   ![Image](https://PISRV1/piwebapi/assetservers/)

   Note that this returns a JSON that contains the names, IDs and details of any PI AF Servers that the PI Web API can access.
Displayed on the previous page is the output of the query we just ran. We are already familiar with most of the properties that are returned. However, there appear to be two different IDs for the AF Server object. We see one “Id” and the “WebId”. The “Id” is the same unique AF Server ID that we see when connecting with PI System Explorer. The WebId, however, is something new altogether.

When retrieving information, PI Web API encodes both the ID and path to the requested object to form this “WebId”. This forms a new unique identifier for use in further queries. Most PI Web API queries require one or more WebIds be included in the call.

When you make a query with an input WebId, the PI Web API decodes this WebId and finds the object’s unique ID along with the object’s path. It then attempts to perform the query using the object’s unique ID. If this fails, it will perform the query using the path instead. This gives the PI Web API a redundant, efficient way to reference objects in the PI System.

4. In the previous call, the PI Web API gave us “Links”, pointing us to queries that can be performed on the returned object. Examine the syntax of the URL next to the “Databases” link. It is in the form:

https://PISRV1/piwebapi/assetservers/PI-Asset-Server-WebID/databases

This query references the unique ID of the PI AF Server, and asks for the databases that belong to that server. Click on the link and click on the “Send” button.

5. Now we have our database names, IDs and associated information. Try to find the “USAWeather” database, then display its “Elements”. You will see a “Cities” element. Drill down one level more by viewing its “Elements” and you’ll see a list of cities. Get the attributes of “Chicago”, then get Interpolated data for the “Wind Speed” attribute.

6. Up until now, we have been dealing exclusively with GET requests. The PI Web API can also write to the PI Server with the use of POST requests. We’re going to use a POST to add a new attribute to an existing element in our database. A PI Web API POST request consists of two parts. A URL, indicating the object you wish to POST to, and an associated JSON packet, containing the information you wish to send. We are going to create a new attribute under a parent element, so will have to prepare a URL referring to the attributes under a given element, and a JSON packet to send through with the attribute’s parameters.

First off, we need the URL indicating the object we’d like to post to. Take a look at the help file for this kind of request. It’s at:

https://PISRV1/piwebapi/help/controllers/attribute

We need to make a query in the form POST elements/{webId}/attributes. We need a webId of the element that we’re going to post to. Revisit the previous steps, navigate through your AF server using the PI Web API links, and grab the webId of the “Cities” element. Then, type the following into the URL section of Postman, replacing element-webId with the WebId of your “Cities” element:

https://PISRV1/piwebapi/elements/element-webId/attributes
• Beside the URL, Select “POST” as the type of request.
• Further to the right of POST, select “Headers” and enter a header of “Content-type” and a value of “application/json”
• Below the URL select “raw” as the text type
• Beside the “raw” selector, ensure that you are making the request in the form of “JSON”.

Write the following into the text entry pane:

```json
{
    "Name": "ExampleAttribute",
    "Description": "ExampleString",
    "DataReferencePlugIn": "PI Point",
    "ConfigString": "\\PISRV1\CDT158;UOM=m;ReadOnly=False",
    "IsConfigurationItem": true,
}
```

Your configuration should look similar to the following screenshot:

![Screenshot of request configuration](image)

Finally, click on the “Send” button. Ensure you receive the “201 Created” Status code.

7. Change the request back to GET and click the “Send” button. You should see the properties of your created attribute. Ensure your attribute has a DataReferencePlugIn value of “PI Point” and note that it has a WebId value when queried.
8. Write a new query to get equally spaced interpolated data every 30 minutes for the last 24 hours for your new attribute. Try to build the URL yourself.

Hint: The resources in the “GET streams/…” resource type return raw data, interpolated data or calculated data based on a single data stream. A “data stream” can be a PI AF Attribute or a PI Point. You need to use the “GET streams/{webId}/interpolated” resource to get evenly spaced interpolated data for a point or attribute. See the help page for more information. You’ll also need to specify parameters at the end of the URL to suit the requirements in the question, try using Postman’s URL params feature, which will automatically append your parameters to your request. Check the form of your URL after making the request and see the syntax that is used.

Write your query in the blank space below (you can skip any webIDs, otherwise we’ll be here all day):

9. Write a single value of “35” to the attribute you created in step 6, “ExampleAttribute”. Which method do you use? What are the contents of your JSON packet? Write down the format your query including the JSON below:
3.4 Exercise – Making calls to the Index Search Crawler

Activity Objectives

- Make simple calls to the PI Web API search
- Determine the REST structure of the data returned

Approach

When you work with the PI System, you will commonly need to search for objects within your code. The PI Web API offers a high performance indexed search service built in. We’re going to look at how we can take advantage of this feature in our code. We will be performing searches for elements contained in the PI AF database called “USAWeather” in preparation for the next exercise.

1. Before a PI AF database can be searched with the Index Search Crawler, it must be indexed. We need to confirm that the AF database we are going to search through is indexed. Open a browser and navigate to the below URL.

   https://PISRV1/piwebapi/admin/search/database.html

   An entry for the AF:\PISRV1\USAWeather database should already be on the page. If it’s not, click “Add Database” and type the following text into the “Add Database” textbox, replacing AF-Server-Name with the name of your AF Server:

   AF:\\AF-Server-Name\USAWeather

   Once the database is added, it will be shown on the page and scheduled for indexing. By default, the PI Indexed Search Crawler Service picks up changes every three minutes. Wait for it to have a recent time for “Last Crawled”, then progress to the next step.

2. Open Postman and type the following into the address field and hit enter:

   https://PISRV1/piwebapi/search/query?q=Chicago

   If we look in the help file under “GET search/query”, it states that if no field is specified, the following fields will all be used: name, description, afcategories, afelementtemplate, attributename, attributedescription. This means that the query will show all PI AF Elements and PI Points which have the search query within:

   - Their name.
   - Their description.
   - One of their AF categories name.
   - Their AF element template.
   - Their AF attributes name.
   - Their AF attributes description.
3. Let’s search again but we want to get the elements that have attributes with the name “Temperature”.
   
   https://PISRV1/piwebapi/search/query?q=attributename:Temperature

4. Let’s change the query in order to return only the second and third object found.
   
   https://PISRV1/piwebapi/search/query?q=attributename:Temperature&start=1&count=2

5. Finally, we are going to query only Chicago again with no fields specified. Nevertheless, on the results, we want to see only the name, webid and matchedfields.
   
   https://PISRV1/piwebapi/search/query?q=Chicago&fields=matchedfields;paths;name;

6. You should see all items returned with their name, path and the matched fields.
3.5  Exercise – Developing a Web Application With jQuery and PI Web API

Activity Objectives:

- Make a simple web application that interacts with the PI System.
- Send data to the PI System through a web application.

Approach:

This exercise is heavy with copy/pasting code, and you can find a copy of it to copy/paste from on PI Square. It can be found by searching for the tag learn_basics_pi_programming.

We are going to develop a web application that will enable a user to manually write to the PI Server, to update weather data for a few cities around the USA. We will use the USAWeather PI AF Database for this, just as we have been for the last few exercises.

1. Open Visual Studio and create a new ASP.NET Empty Web Application project. This can be found under Installed > Templates > Visual C# > Web.
2. In the “Solution Explorer” pane on the right side of the screen:

- Right click on SampleApp_PIWebAPI_PostingData, Add a folder and rename it to “js”.

- Do the same and create another folder and rename it to “css”.

- Do the same, but this time Add a new HTML page. Rename the page to sending_data_to_pi.html.

- Go to your computer’s desktop and open the shortcut to “Exercise Files”. Drag and drop the default.css file onto the css folder on your project in Visual Studio.

- Right click on “js” and Add a JavaScript file. Rename it to sending_data_to_pi.js.

3. Check your configuration against the screenshot shown on the right of the page above.

4. Open the file sending_data_to_pi.html and paste the following content:

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <!--Adding references to jQuery, Cascade Style Sheet and JavaScript libraries -->
  <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
  <script src="js/sending_data_to_pi.js"></script>
  <link href="css/default.css" rel="stylesheet" />
  <title>Sending Value to the PI System through PI Web API</title>
</head>
<body>
  <h1>Sending Value to the PI System through PI Web API</h1>
  <p>Select the element and attribute that you want to send your value:</p>
  <!--Adding one drop down list for cities, one for attributes, a text value for the value to be sent and a button to submit -->
  <!--When the button is clicked it will trigger the SendValue() JavaScript function -->
  <label for="city_name">Select city</label>
  <select name="city_name" id="city_name" size="1"></select>
  <label for="attribute_name">Select attribute</label>
  <select name="attribute_name" id="attribute_name" size="1"></select>
  <input type="text" name="value" id="value" value="" />
  <input type="button" id="UpdateBtn" value="Send Value" onclick="SendValue()" />
</body>
</html>
```
5. Take a look at the HTML code snippet. The drop down lists for the cities and attributes are currently empty. We will fill the lists using jQuery by getting the values from PI Web API. Therefore, when the page is loaded, jQuery will make two REST calls against PI Web API to fill the drop down list options. After clicking on the “Send Value” button, the value typed on the corresponding field that will be sent to the PI System.

6. It is time to edit the sending_data_to_pi.js file, which is currently empty. Open the file by double clicking it. We’ll define the MakeAjaxRequest() function responsible for making calls against the PI Web API. This method has four inputs:

- Type: HTTP Method: GET, POST, PATCH, PUT or DELETE
- Url: Address of the resource
- successCallBack: Variable function to be executed in case the request is successful.
- Data: Data used on POST and PUT request. The GET and DELETE methods do not use it.

Paste the following code into the file:

```javascript
function MakeAjaxRequest(type, url, successCallBack, data) {
  $.ajax({
    type: type,
    url: url,
    cache: false,
    async: true,
    data: data,
    contentType: "application/json",
    success: successCallBack,
    xhrFields: {
      withCredentials: true
    },
    error: function (error, variable) {
      alert('There was an error with the request');
    }
  });
}
```

7. We now need functions to populate the drop down lists on our page. The following functions GetCitiesName() and GetAttributesName() add the proper items to the drop down lists by calling the MakeAjaxRequest function with the appropriate URL. You will need to replace both URLs in the below functions with URLs appropriate to your system. Use Postman to get a URL that finds:

- All elements underneath the “Cities” element in the USAWeather Database and use it in the GetCitiesName() function. You will need to enter your own WebID for the “City” element in the code.
- The attributes of the “City” element template and use it in the GetAttributesName() function. You will need to enter your own WebID for the “Cities Template” element template in the code.
```javascript
function GetCitiesName() {
    var url = "https://PISRV1/piwebapi/elements/Your-WebId/elements";
    MakeAjaxRequest('GET', url, function (data) {
        for (var i = 0; i < data.Items.length; i++) {
            $('#city_name').append($('<option>', {
                value: data.Items[i].Name,
                text: data.Items[i].Name
            }));
        }
    });
}

function GetAttributesName() {
    var url = "https://PISRV1/piwebapi/elementtemplates/Your-WebId/attributetemplates";
    MakeAjaxRequest('GET', url, function (data) {
        for (var i = 0; i < data.Items.length; i++) {
            $('#attribute_name').append($('<option>', {
                value: data.Items[i].Name,
                text: data.Items[i].Name
            })), null);
        }
    });
}

8. There’s one thing missing concerning loading the drop down list options. We need to call both
GetCitiesName() and GetAttributesName() functions once the HTML page is loaded. This is
achieved by using the $(document).ready function. Add the following to your code:

$(document).ready(function () {
    GetCitiesName();
    GetAttributesName();
});

9. The SendValue() function will send the value to the PI System. It stores the value to be sent, the
name of the city and the name of the attribute to write to all in JavaScript variables. Make a GET
request in order to receive the response with the URL related to the value of the selected
attribute from the selected city. Make a POST request to update the value to the PI System. As
we are making asynchronous calls, the variable sendValueFunction contains the function for
updating the value for a given URL and data. This variable is the successCallback variable for the
MakeAjaxRequest to get the value URL before making the POST request. The function is as
follows:
```javascript
function SendValue() {
    var value = $('"#value"').val();
    var cityName = $('"#city_name"').val();
    var attributeName = $('"#attribute_name"').val();
    var url = 'https://PISRV1/piwebapi/attributes?path=\\PISRV1\USAWeather\Cities\" + cityName + "\" + attributeName;
    var sendValueFunction = function(data) {
        var valueUrl = data['Links']['Value'];
        MakeAjaxRequest('POST', valueUrl, function() {
            alert("Value has being sent successfully");
        }, "{'Value': " + value + " }");
    };
    MakeAjaxRequest('GET', url, sendValueFunction, null);
}
```

10. Select your `sending_data_to_pi.html` file, then press F5 to debug your application, and a browser should pop up. Select one of the cities, and one of the attributes, and use the application to write a value back to the PI Data Archive. Use PI System Explorer or Postman to check if you were successful.
3.6 Where to Go Next

This section has been a light introduction to PI Web API. If you have any questions or would like more information about the product, please check the associated section in PI Square:

https://pisquare.osisoft.com/community/developers-club/restful-pi-system-access

If you still have time during this workshop, feel free to start on the other sections of this booklet. See below for a list of page numbers:

PI AF SDK ..................................................................................................................................................... 14
PI SQL Framework ....................................................................................................................................... 42
4 PI SQL Framework

4.1 Introduction

The PI SQL Framework is a category of PI System Access products comprising if PI OLEDB Enterprise, PI JDBC Driver, and PI ODBC Driver, among others. The products in this category expose PI asset structures and time series data as if the PI System were a relational database. That is, each exposes PI data (both real time and contextual) as tables available for access via SQL queries. These technologies are typically used when integrating the PI System with other systems that can import data in this format. This lab will focus on PI OLEDB Enterprise (architecture above). However, any of the queries we make here will also work with both the PI JDBC and PI ODBC Drivers. PI JDBC and ODBC Drivers have a slightly different architecture to PI OLEDB Enterprise (below).

These Developer Technologies are particularly useful when the PI System has a role to play in Reporting or Business Intelligence (BI) scenarios, where it must interact with other products or systems that communicate using Structured Query Language (SQL). In general terms, if you want to access the PI System as if it were a relational database, you’ll most likely use one of these three products.
4.2 Review of SQL Select Statements

Before we go on to PI OLEDB Enterprise, we need to know what a SQL Select statement is, and be familiar with the syntax that is used. If you’re already familiar with the syntax and use of these statements including aliases and table joining, feel free to skip right to the exercises in this chapter.

If you need a review of SQL Select statements, you’ll find a document on PI Square with the name “Review of SQL Select Statements”. You’ll be able to find it with the learn_basics_pi_programming tag.

4.3 Exercise – A Salute to PI SQL Commander

PI OLEDB Enterprise exposes objects in the PI System through a relational view of PI AF databases, which is accessible through SQL queries. The PI OLEDB Enterprise installation includes a test environment which handles the OLE connection process, allowing developers to easily test queries and results. This test environment is PI SQL Commander. PI SQL Commander not only tests queries, but also enables creation of transpose functions and views against PI AF using PI OLEDB Enterprise.

Exercise Objectives

- Get familiar with PI OLEDB Commander
- Discover the interactions between PI OLEDB Enterprise tables

Approach

1. Launch PI System Explorer (PSE) by double clicking on the shortcut on your desktop.
2. Launch PI SQL Commander by double clicking on the shortcut on your desktop.
3. Open the Documentation Live Library by double clicking on the link on your desktop. Navigate through to the PI OLEDB Enterprise 2012 section.

The rest of this section will regularly reference this Documentation, and will use PI SQL Commander to test queries. It’s always good to have PI System Explorer open while writing these queries so you have another view into the system. We’ll use these two applications, along with the documentation, to take a look at what PI OLEDB Enterprise can do for us.

4. In PI SQL Commander, right click on “PISRV1” under Data Sources > AF Servers > PISRV1. Click “Connect”. Ensure that Windows Integrated Security is selected, then click “OK”. Your connection was successful if you see a green arrow next to your AF server name.
5. Expand the “Catalogs” list under PISRV1, and you will see the different AF databases available. Today we are using “OSIsoft Enterprises” Database. Expand this database by selecting the “+” symbol to the left.
6. Spend some time expanding out some of the tables under “OSIsoft Enterprises”, and their corresponding “Columns” objects. Look at the different tables and think about what they might represent.
7. Right click on the Asset.Element table and select “Execute Predefined Query...”. Look at the columns, and try to match some of the information you see to information in PI System Explorer (PSE).
You should be able to see all human-readable properties of each element object in PSE, that is, you should be able to see everything that isn’t a unique identifier GUID – these aren’t exposed to users in PSE. Every object in AF has a GUID, and these GUIDs are what we’ll be using to join the different tables exposed in PI OLEDB Enterprise. More on this later.

8. Do the same predefined query for the Asset.ElementHierarchy table. What information does this query show you?

9. Edit the Asset.ElementHierarchy predefined query to remove the whole WHERE clause. Your query should now read:

```sql
SELECT *
FROM [OSIsoft Enterprises].[Asset].[ElementHierarchy]
```

10. Execute the query. What do the columns in this query show you?

11. Read through the table descriptions below and execute the corresponding predefined queries. You might need to edit each predefined query after running to make it more general, and make it catch more levels of the element or attribute hierarchies. Try to match the described columns to the results of your queries and see what kind of data they hold. In later exercises, we’ll be joining theses tables to each other on their various ID columns.

**The Asset.ElementHierarchy Table**

This table contains the data that is in the treeview. That is, it is the hierarchical relationships between the elements (parent / child etc). There is no ID column, the elements are linked on the ElementID field which is their GUID.

**The Asset.Element Table**

This table holds information about each Element. It is the information you would see on the General tab if you were viewing this element in PI System Explorer. The ID field is the GUID for the element named in the row.
The Asset.ElementAttribute Table

This table contains each of the Element attributes. Note that root elements may contain no attributes so will likely return an empty set. The ID column contains the attribute GUID. The table also has an ElementID column, which is the ID for the element the attribute belongs to.

The Asset.ElementTemplate Table

This table contains the names and properties of element templates in the database. The ID column in this table represents the GUID of the element template object in the database.

The Asset.ElementTemplateAttribute Table

This table represents the template attributes in the database, showing their names and configuration, including their reference configuration strings. The ID column in this table represents the GUID of the attribute template object in the database.

The Data.Snapshot Table

This table contains most recent data for all element attributes in the database including their values and timestamps, but does not contain the names of the element attributes. The table does not have an ID column, it instead has an ElementAttributeID column, corresponding to the GUID of the attribute represented in each row.

The Data.Archive Table

This table contains all historical time series data in the PI Data Archive corresponding to element attributes, including values and timestamps, but does not contain the names of the element attributes. The table does not have an ID column, it instead has an ElementAttributeID column, corresponding to the GUID of the attribute represented in each row.

Note: Tables prefixed with a “v” – we will not use these tables in this lab. These tables are used when referencing elements with versioning.
4.4 Exercise – Serving Data with PI SQL Data Access Technologies

Exercise Objectives:

- Become more familiar with the different tables available in PI OLEDB Enterprise
- Learn how the tables interact with each other
- Join PI OLEDB Enterprise tables together
- Create a View of your solution

Approach:

We are tasked with writing a query to grab real time fuel gas flows for all assets in the enterprise for use on a reporting system. Each reading will have to have contextual information to go along with it, including the location of the asset, name of asset, and description of asset. We also need to know the current value of the reading, as well as when it was last updated.

We’ll go through gradually, adding more and more complexity to our query as we go. This exercise is very barebones in this book to be more of a challenge. You’re welcome to look up the solution on PI Square by searching for the learn_basics_pi_programming tag. Please attempt the exercises before racing for the solution, you’ll have way more fun.

4.4.1 A Table for One, Please
Write a query (or edit one of the above selected pre-defined queries) to return all the names and descriptions of all elements in the “OSIsoft Enterprises” database – not just those at the region level. You’ll have to look through the table columns of each table, but you should be able to do this without any table joins. Execute the query to ensure you have the correct solution.

Expected result:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wichita</td>
<td>The Wichita Plant</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

4.4.2 A Table for Two, Please
Add to your previous query, making the results show the names, descriptions, and paths to the parents of all elements in the database. You will need to join two of the above mentioned tables on corresponding GUIDs, using some of the ID related columns. Generally speaking, if you want to do anything using PI OLEDB Enterprise – you’re going to have to join at least two tables together.

Expected result:

<table>
<thead>
<tr>
<th>Path</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>Houston</td>
<td>The Houston Plant</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
4.4.3 A Table for Three, Please
Write a query to return the “Fuel Gas Flow” attributes of all elements in the hierarchy, including the Attribute names, owning element, parent of owning element, and the element’s description. Note that you will need to use aliases here and the columns from the last exercises will also need to be renamed. This needs to be done otherwise you’ll have two “Name” columns, one for the Element.Name column, and one for the Attribute.Name column. Your query needs to rename the columns like so:

- Path => Element Parent
- Name => Element Name
- Description => Element Description
- And a new column from the extra table join: Attribute Name

Expected Result:

<table>
<thead>
<tr>
<th>Element Parent</th>
<th>Element Name</th>
<th>Element Description</th>
<th>Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Tucson\Cracking Process</td>
<td>Boiler-309</td>
<td>Cracking Boiler B-309</td>
<td>Fuel Gas Flow</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

4.4.4 A Table for Four, Please
Extend the previous query to join all four tables. Show the same columns as above, in addition to the current Timestamp and Value of each attribute listed. Edit the above query and execute, ensuring you see the following columns:

<table>
<thead>
<tr>
<th>Element Parent</th>
<th>Element Name</th>
<th>Element Description</th>
<th>Attribute Name</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Tucson\Cracking Process</td>
<td>Boiler-309</td>
<td>Cracking Boiler B-309</td>
<td>Fuel Gas Flow</td>
<td>2015-04...</td>
<td>66.4...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Keep your query open, you’ll use it for the next exercise. Write your query in the space below:
4.4.5 Exercise – A Table with a View, Please

In relational databases, a ‘view’ is a saved result set that becomes like a ‘virtual table’ which can be queried just like any other table.

1. Copy the contents of the query that should still be open in your query window – the solution to “A Table for Four, Please”.

2. In PI SQL Commander, navigate through to Data Sources > AF Servers > PISRV1 > Catalogs > OSIsoft Enterprises > Asset > Views. Right click on “Views” and select “Create View…”

3. Replace the [ <view name> ] text with [Fuel Gas Flows]

4. Paste your query, replacing the <query> section of the text in the window that pops up.

5. Execute the query.

6. Right click on “Views” and select “Refresh”

7. Expand out “Views” and right click on your new view and select “Execute Predefined Query”

What does the predefined query for a view show you?

[Blank]

Why are views useful?

[Blank]
4.5 Exercise – Data Transpositions

We have already dealt with the Asset and Data branches of the Catalog. Next we’ll turn our attention to the DataT section. Result sets we have seen so far look like this:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute</th>
<th>Current Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pump 1</td>
<td>Flow</td>
<td>43 m³/h</td>
</tr>
<tr>
<td>Pump 1</td>
<td>Power draw</td>
<td>17.1 A</td>
</tr>
<tr>
<td>Pump 1</td>
<td>Spindle speed</td>
<td>1960 RPM</td>
</tr>
<tr>
<td>Pump 2</td>
<td>Flow</td>
<td>29 m³/h</td>
</tr>
<tr>
<td>Pump 2</td>
<td>Power draw</td>
<td>13.8 A</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

However, it is much more legible and makes more sense if we rotate or transpose the values as:

<table>
<thead>
<tr>
<th>Element</th>
<th>Flow</th>
<th>Power draw</th>
<th>Spindle speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pump 1</td>
<td>43 m³/h</td>
<td>17.1 A</td>
<td>1960 RPM</td>
</tr>
<tr>
<td>Pump 2</td>
<td>29 m³/h</td>
<td>13.8 A</td>
<td>1754 RPM</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

We can use PI OLEDB Enterprise to generate transpositions similar to what is above. Transpositions are not only more useful when visually perusing a table but are suited to BI applications like Power Pivot when analyzing result sets. Like views, transpose functions do not exist until they are created.

**Activity objectives:**

- Create a Transpose Function
- Execute a query on a Transpose Function

**Approach:**

1. Using PI SQL Commander, navigate to Data Sources > AF Servers > PISRV1 > Catalogs > OSIsoft Enterprises > DataT > Functions. Right click on functions and left click on New Transpose Functions > (asset):

2. Check the Boiler template and click next.

3. Click next on the Attribute Path screen without making changes.

4. Check the “TransposeSnapshot” function and click next.

5. Keep the defaults on the Function Names screen and click next.
6. Click through next / execute to complete the creation.

Creating functions like this creates both a function, and a table underneath DataT. We’re going to use the table for the rest of this exercise.

7. Navigate to ...DataT > Tables > ft_TransposeSnapshot_Boiler. Right click on the new table and select “Execute Predefined Query”. Edit the query so it shows us some meaningful data and execute it again.

8. Lastly, we can save this as a view by adding the following at the top of the modified query and executing again:

```
CREATE VIEW [OSIsoft Enterprises].[DataT].[BoilerSnapshots] AS
```
4.6 Where to Go Next
This section has been a light introduction to PI OLEDB Enterprise. If you have any questions or would like more information about the product, please check the associated section in PI Square:
https://pisquare.osisoft.com/community/developers-club/pi-sql-data-access-technologies

If you still have time during this workshop, feel free to start on the other sections of this booklet. See below for a list of page numbers:

PI AF SDK ..................................................................................................................................................... 14
PI Web API................................................................................................................................................... 27

OSIsoft Virtual Learning Environment

The OSIsoft Virtual Environment provides you with virtual machines where you can complete the exercises contained in this workbook. After you launch the Virtual Learning Environment, connect to PISRV1 with the credentials: pischool\student01, student.

The environment contains the following machines:

PISRV1: a windows server that runs the PI System and that contains all the software and configuration necessary to perform the exercises on this workbook. This is the machine you need to connect to. This machine cannot be accessed from the outside except by rdp, however, from inside the machine, you can access Coresight and other applications with the url: http://pisrv1/, (i.e. http://pisrv1/coresight).

PIDC: a domain controller that provides network and authentication functions.

The system will create these machines for you upon request and this process may take between 5 to 10 minutes. During that time you can start reading the workbook to understand what you will be doing in the machine.

After you launch the virtual learning environment your session will run for up to 8 hours, after which your session will be deleted. You can save your work by using a cloud storage solution like onedrive or box. From the virtual learning environment you can access any of these cloud solutions and upload the files you are interested in saving.

System requirements: the Virtual Learning Environment is composed of virtual machines hosted on Microsoft Azure that you can access remotely. In order to access these virtual machines you need a Remote Desktop Protocol (RDP) Client and you will also need to be able to access the domain cloudapp.net where the machines are hosted. A typical connection string has the form cloudservicename.cloudapp.net:xxxxx, where the cloud service name is specific to a group of virtual machines and xxxxx is a port in the range 41952-65535. Therefore users connecting to Azure virtual machines must be allowed to connect to the domain *.cloudapp.net throughout the port range 41952-65535. If you cannot connect, check your company firewall policies and ensure that you can connect to this domain on the required ports.