PI OLEDB Enterprise 2012

User Guide

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

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 OLEDB Enterprise 2012 User Guide

Version: 1.3.1.5

Published: 02/06/2013

Table of Contents

Chapter 1

Introduction

PI OLEDB Enterprise is an OLE DB data provider you can use to access asset metadata stored in PI Asset Framework (AF). PI OLEDB Enterprise supports read-only access to PI AF server data. Only the AF asset namespaces are exposed.

PI OLEDB Enterprise also provides read-only access to data from PI Servers, since AF attributes can be configured to reference PI points. PI OLEDB Enterprise exposes this information through a relational view of AF databases, which is accessible through SQL queries.

The PI OLEDB Enterprise data provider meets the OLE DB 2.7 specification. The SQL Engine is embedded and supports SQL (Structured Query Language) compliant with ANSI SQL 92 Entry Level.

Microsoft's OLE DB specification is a powerful middleware interface that is supported by many commercially available applications, including Microsoft Office and SQL Server. In addition to using these commercial applications for OLE DB, you can also write custom applications that access data through OLE DB.

PI OLEDB Enterprise is a member of the *PI Data Access product suite* (page 2).

Which PI OLEDB Provider Should I Use?

Both PI OLEDB Enterprise and PI OLEDB Provider 3.3.*x* provide a relational view of the real-time and configuration data stored in PI Systems.

Use PI OLEDB Enterprise if you want to navigate through the asset hierarchy and access the metadata stored in the PI Asset Framework (PI AF), or if you want read-only access to AF data references, including *PI point data* (see "*Introduction*" on page 1).

Note: PI OLEDB Enterprise supports read-only access to PI AF server data. Only the AF asset namespaces are exposed.

Use the PI OLEDB Provider 3.3.*x* if you do not use PI AF, or want to use direct PI archive queries to achieve higher performance, or access the PI point database, the PI Module Database or the PI Batch Database.

Note: A combination of both data providers may be required, depending on your site's needs and environment.

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* (*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 PI AF servers. PSA is a runtime license to access PI System data using any of the programmatic access methods licensed through the PSA, including PI OLEDB Enterprise. For more information, see the *OSIsoft Web site* (*http://www.osisoft.com*) or contact *OSIsoft Technical Support* (*http://techsupport.osisoft.com/*).

Supported Scenarios

While PI OLEDB Enterprise can be used programmatically, such as through the use of Microsoft's ADO.NET components, its main use consists of integrating with products already known as OLE DB consumers.

Applications that PI OLEDB Enterprise is known to integrate well with include:

- Microsoft SQL Server (as a Linked Server, including Integration Services, Analysis Services and Reporting Services capabilities). See *Recommended Configuration for Linked Server in Microsoft SQL Server* (page 93).
- Microsoft SharePoint (including Excel Services)
- Microsoft Excel (including PowerPivot)
- PI WebParts

PI OLEDB Enterprise includes the **PI SQL Commander**, a tool that helps you learn the product, develop queries, and test functionality. **PI SQL Commander** offers two kinds of sample queries:

- One sample SQL query for each table in the PI AF server catalog. For details, see *Run Sample SQL Queries* (page 21).

  Note: The amount of data queried is typically small enough to avoid timeouts but the predefined queries can also be customized to adapt to customer-specific database needs.

- A compendium of SQL queries that can be opened as a Solution in **PI SQL Commander**. See *More Queries - PI SQL Query Compendium Solution* (page 92) for details.

Webinars, tutorials, and programming examples are also available as part of the *OSIsoft vCampus* (*http://vCampus.osisoft.com*) program.

Deployment Options

PI OLEDB Enterprise is deployed as a PI AF client that can run on either the same machine as the PI AF server, or on a different machine. OSIsoft recommends that you install it on the machine that includes the OLE DB consumer application. For example, install PI OLEDB Enterprise on the Microsoft SQL Server machine if it is called as a Linked Server.

Note: Your PI AF server must include *PI SQL for AF server* (page 7). If you use PI AF server 2.1, you must install this component separately on the server. PI SQL for AF server is included with PI AF server 2010 and later.

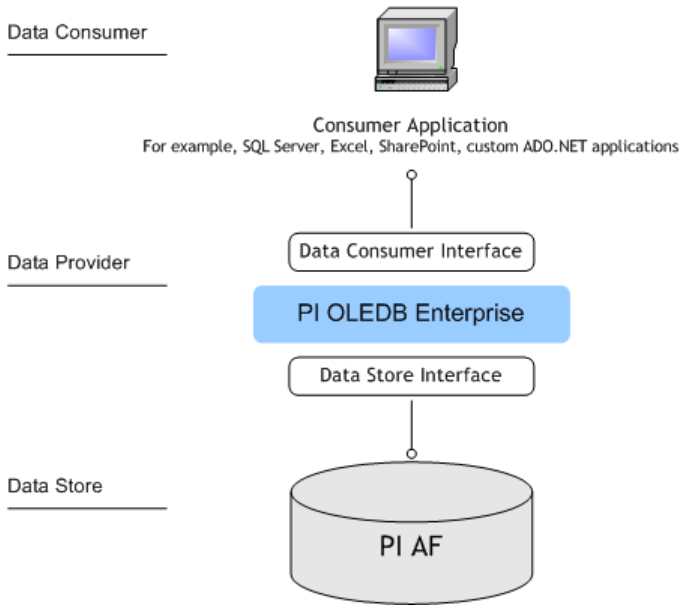
Architecture

When OLE DB is the interface used for data access, the application is called the OLE DB data consumer, and the middleware program is called the OLE DB data provider.

In the case of PI OLEDB Enterprise, the data consumer relies on the PI OLEDB Enterprise data provider to recognize the format of PI System data stores and provide access to it.

The PI OLEDB Enterprise data provider:

- Accepts requests for data from the application
- Accesses the data from the PI System
- Returns the data to the application



PI OLEDB Enterprise Agent

The PI OLEDB Enterprise Agent is a Windows service installed by PI OLEDB Enterprise, which acts as a bridge between PI OLEDB Enterprise and PI AF server. This service segregates .NET (managed) code from the provider, and thus removes incompatibilities found in some environments such as Microsoft SQL Server.

The PI OLEDB Enterprise setup kit configures the agent service to run under the **Local Service** account. For PI OLEDB Enterprise to work properly, the service must be running.

The PI OLEDB Enterprise Agent configuration file `PIOLEDBENTAgent.exe.config` is installed in the `[Program Files]\PIPC\OLEDB` folder. This file contains settings that control some behavioral aspects of the agent. As an example, the agent process contains one thread pool per user; each thread has its own AF SDK cache. The thread pool size is predefined by the **threadPoolSize** property.

Certain use cases warrant changes to these settings, however the effects of such changes are complex and depend on several factors, such as the quantity and type of queries, number of users, and frequency of data retrieval. For this reason, OSIsoft strongly recommends that you do not modify this file unless you are instructed to do so by *OSIsoft Technical Support* (page 99).

Chapter 2

Installation

PI OLEDB Enterprise consists of a server and a client component:

- PI SQL for AF server (`afsqlservice_2.x.x.x.exe`)
  - An add-in to be installed on the PI AF server
  - PI Asset Framework 2010 and later include PI SQL for AF server, however patches might be distributed separately

Note: High Availability support in PI AF server requires AF 2010 or later.

- PI OLEDB Enterprise (`PI OLEDB Enterprise_2012_.exe`)
  - A client component that installs the OLE DB provider and PI System Explorer software. It also includes **PI SQL Commander**, a query development and testing tool.

Note: The installation must be run from an account that has administrative privileges.

Before you install (see "*Before Installation*" on page 6) PI OLEDB Enterprise:

1. Consider your *deployment options* (page 3)
2. Review and verify *system requirements* (page 6)
3. Configure Microsoft SQL Server

Then:

1. Install *PI SQL for AF server* (page 7) version 2.1.2.19 if you have PI AF server 2.1.
2. Optionally, install *PI SQL for AF server* (page 7) 2.3.1.0 if you have PI AF server 2.3.
3. Run the *PI OLEDB Enterprise Setup Kit* (page 10).
4. *Configure PI OLEDB Enterprise* (page 12) data access.
5. *Validate* (page 19) the installation.

Before Installation

Review System Requirements

The server on which you install *PI OLEDB Enterprise* (page 11) requires:

- Windows XP or later

PI OLEDB Enterprise works through *PI SQL for AF Server* that is part of the PI AF Server installation.

- PI AF Server 2010 or later is required

For details on how to install PI AF server, see the *PI AF Installation and Maintenance Guide*, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

If you use PI AF Server 2.3 (2010 R2), then PI SQL for AF Server can optionally be updated to version PI SQL for AF Server 2.3.1.0. This version provides higher performance for Attribute searches, compared to the version included in PI AF Server 2.3.

Note: PI OLEDB Enterprise Event Frames support requires PI AF server 2012 or later.

Configure SQL Server

The following applies to PI AF Server versions 2010 and 2010 R2 only:

PI SQL query authorization and data filtering is performed directly at the SQL Server back end. The PI SQL for AF server setup kit installs a CLR assembly named `OSIsoft.Authorization` that requires SQL Server to be:

- CLR enabled
- granted access to **Win32 API security validation** functions

The login used is also named `OSIsoft.Authorization`.

If you host your SQL Server on a 32-bit OS or have a 64-bit SQL Server, the setup kit for *PI SQL for AF server* (page 7) configures these requirements.

Note: The second requirement can only be met if the assembly is strong name signed. This strong name is created inside SQL Server as an asymmetric key, and has a corresponding login with **UNSAFE ASSEMBLY** permission.

If, however, you want to use a 32-bit version of SQL Server on a 64-bit Windows server, you must enable CLR *before* you install *PI SQL for AF server* (page 7) since this configuration requires a restart. Restart is not required for other server types. For more details, see this

MSDN article http://msdn.microsoft.com/en-us/library/ms175193.aspx on the option to enable CLR.

The success or failure of configuration settings is logged to a file in the PI AF server directory:

```
PIHOME\AF\afsqlserviceAutoConfig.log
```

Note: `PIHOME` represents the directory to which you install PI System applications and interfaces. For example, `C:\Program Files\PIPC`.

Install PI OLEDB Enterprise

Note: The installation must be run from an account that has administrative privileges.

Install PI SQL for AF Server 2.3 (2010 R2) optional update

PI SQL for AF server is an add-in to the PI AF server that contains scripts for the Microsoft SQL Server database.

If you want to install these PI SQL scripts for a 32-bit SQL Server that runs on a 64-bit version of Windows, verify that the server uses the *required settings* (page 6) before you run the setup kit.

Note: You must use an account with administrative privileges to run this setup kit.

See *Deployment Options* (page 3) for complete details and recommendations about the machines you use to install PI SQL for AF server.

Run the PI SQL for AF Setup Kit

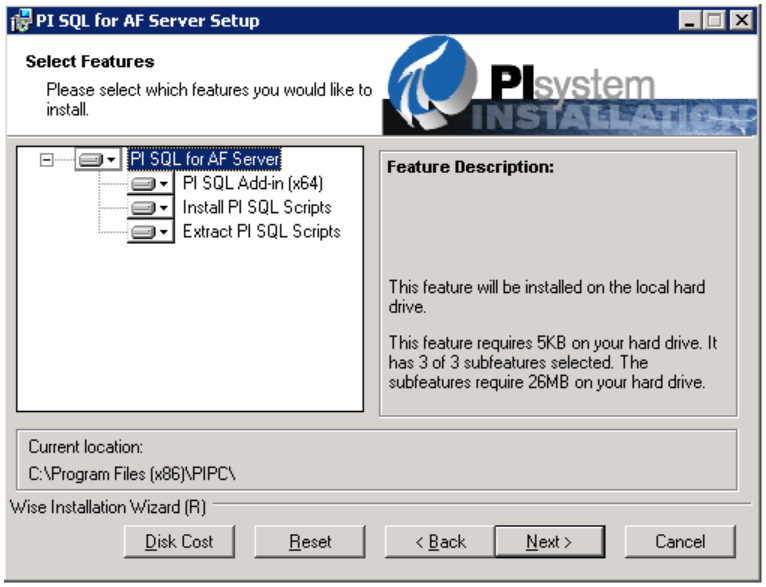
1. Run the *PI SQL for AF server* (page 7) setup kit `afsqlservice_x.x.x.x`.exe.

   Note: If the PI AF server on which you run the PI SQL for AF server setup kit already has a newer PI SQL for AF server version, the setup kit will exit. In this case, proceed to the *PI OLEDB installation* (page 11).

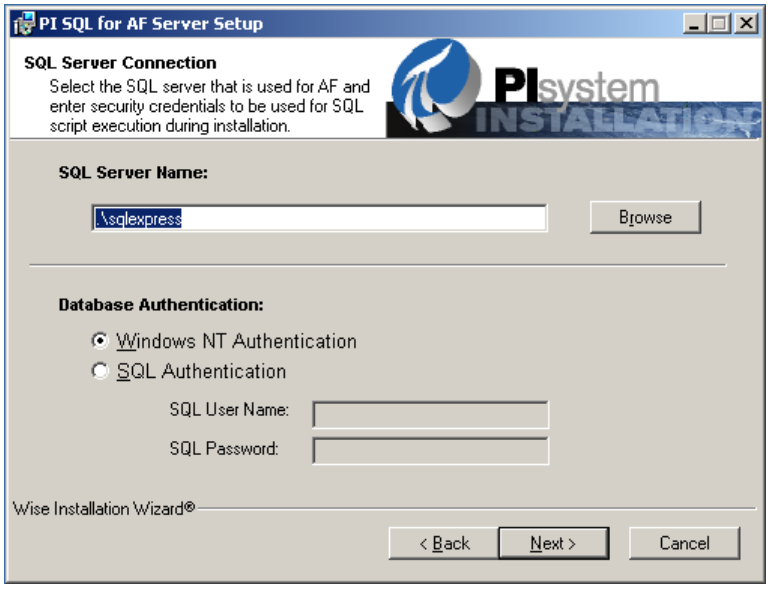
2. Select from the available PI SQL features:
   - **PI SQL Add-in**. The setup kit detects whether the machine you use has a 64-bit or 32-bit operating system and installs the appropriate software. If the machine does not run PI AF server, this option is not available.
   - **Install PI SQL Scripts**. The setup kit contains scripts for the SQL Server back end database. If multiple PI AF server machines are installed for the same back-end database, **Install PI SQL Scripts** needs to be selected at least once during the

installation. You can also exclusively select this feature if you are running the PI for SQL Setup Kit on the local SQL Server computer that may not have the PI AF server service installed.

- **Extract PI SQL Scripts**. The setup kit stores PI SQL scripts on disk for manual execution on the back end MS SQL Server that hosts the PIFD database: `PIHOME\SQLDAS\SQL2310`.



3. Select the SQL server to be used with PI AF server and select security credentials to be used for SQL script execution:



- You can successfully run the PI SQL scripts on a machine that is not the SQL Server, however, the account that runs the installation must have **sysadmin** server role on the SQL Server.

- Alternatively, a SQL Server administrator account can be specified if the SQL Server has mixed mode login enabled.

4. The setup kit can grant **Everyone** access to the local Windows group created on the PI AF server. You might select this option to accommodate prior releases of clients, such as PI WebParts 3.0, which can only execute PI SQL queries if the process that hosts the provider is member of a local Windows group created on the PI AF server:

Caution: OSIsoft recommends that you maintain this configuration only temporarily to test your installation. Upon validating your installation, replace **Everyone** with identities for specific Process IDs or User IDs.

Installation Details

Files Installed

To review the list of files installed, see the PI OLEDB Enterprise release notes for the version you use. Release notes are available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

Location of Files

The Installation procedure runs SQL scripts against the back end MS SQL Server that hosts the PIFD database. These scripts install additional procedures, views, and so on, all registered under schema name **AfSql**. Files are installed in the `PIHOME\SQLDAS` directory and the Windows Global Assembly Cache (GAC). The `PIHOME\AF\AFService.exe.config` file gets modified and the PI AF server Windows service gets restarted.

Note: `PIHOME` is the directory to which you install PI client applications and interfaces. For example, `C:\Program Files\PIPC`.

Review SQL Server Log Files

The following message in the SQL Server log is a result of the SQL Server configuration and is not an error:

```
Unsafe assembly 'osisoft.authorization<c/> version=0.9.0.0<c/>
culture=neutral<c/> publickeytoken=b026e21f53854bab<c/>
processorarchitecture=msil' loaded into appdomain …
```

PI OLEDB Enterprise Setup Kit

The PI OLEDB Enterprise setup kit installs:

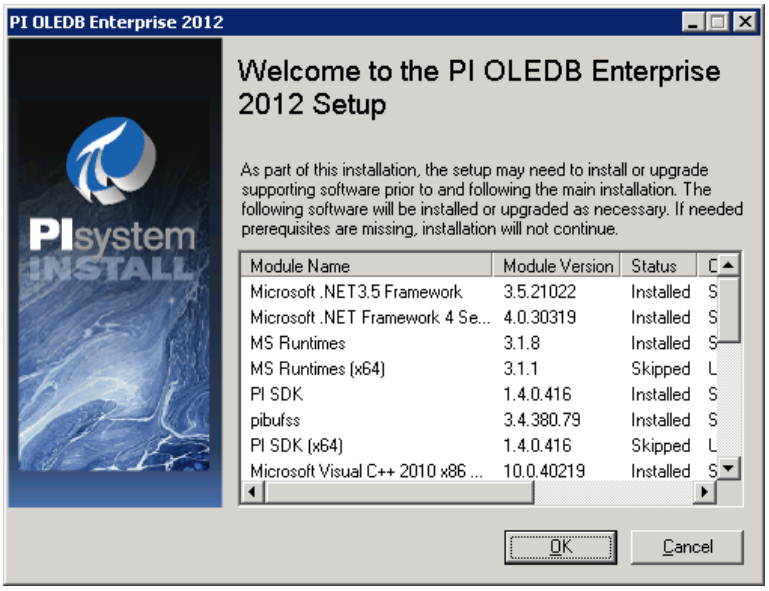
- files in the `PIHOME\OLEDB` directory
- a Windows service called PI OLEDB Enterprise Agent
- a shortcut to launch the **PI SQL Commander** in **Start > Programs > PI System**
- Visual Studio Shell redistributables
- PI OLEDB Provider 3.3

Note: `PIHOME` is the directory to which you install PI client applications and interfaces. For example, `C:\Program Files\PIPC`.

Run the PI OLEDB Enterprise Setup Kit

Note: The installation must be run from an account that has administrative privileges.

1. Run `PI OLEDB Enterprise_2012_.exe`



Note: The installation of the Visual Studio Shell redistribution package may take several minutes without updating the progress bar.

2. Follow the prompts to install the *components required* (page 10) for PI OLEDB Enterprise.
3. Configure *PI OLEDB Enterprise* (page 12) and *verify your installation* (page 19).

Installation Details

To review the list of files installed, see the PI OLEDB Enterprise release notes for the version you use. Release notes are available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

To review the list of files installed by Visual Studio Shell redistributables, see the *Microsoft Visual Studio Shell download pages* (*http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=19670*, *http://www.microsoft.com/download/en/details.aspx?id=1366*).

Silent Installation

The PI OLEDB Enterprise setup kit extracts several installation modules. The components of the installation process, their order, and the arguments used to launch the components are provided in a configuration file named `setup.ini`. If you modify this file, you can provide different command line arguments for different stages of the setup. This may be useful within a well-controlled environment with options that are known in advance, such as in the case of an embedded installation. The setup kit also contains a file named `silent.ini` that contains modifications to `setup.ini` that are typically needed to run a silent installation. You can augment these arguments by adding any of the options described below. For PI SDK and PI AF Client installations and arguments, see the PI SDK Help and release notes, and the *PI AF Installation and Maintenance* guide, respectively.

Individual arguments must contain no spaces unless they are surrounded by quotes.

Argument	Description
`/i`	Specifies an installation
`/qn`	Specifies the "quiet mode" and suppresses dialog boxes and prompts
`ALLUSERS`	Specifies the per-machine or per-user installation context. Use a value of **1** for silent installations.
`REBOOT`	Restarts the computer. Use a value of **Suppress** for silent installations.

Use this syntax for a silent installation:

```
msiexec.exe /i PIOLEDBEnterprise.msi REBOOT=Suppress
ALLUSERS=1 /qn
msiexec.exe /i PIOLEDBEnterprise64.msi REBOOT=Suppress
ALLUSERS=1 /qn
```

Note: On a 64-bit Windows both versions, 64-bit and 32-bit, need to be installed, even if you only plan on using one.

Configure Data Access

Set Initialization Properties

Before you can establish an OLE DB connection, first set the properties that initialize PI OLEDB Enterprise connections. Enter the initialization properties as strings of keywords and values, each separated by an equal sign, that is **=**. These strings persist as a series of keyword/value pairs separated by semicolons. For example:

```
Keyword1 = Value1; Keyword2 = Value2;
```

For details about the initialization properties to use in PI OLEDB Enterprise connection strings, see:

- *General OLE DB Initialization Properties* (page 13)
- *PI OLEDB Enterprise-Specific Initialization Properties* (page 14)

A Universal Data Link is a text file with a `.udl` extension containing the connection string. UDL files serve for persisting and sharing of the connection information. See *Use a UDL File* (page 16) for more information.

General OLE DB Initialization Properties

These general OLE DB initialization properties are supported for PI OLEDB Enterprise:

OLE DB Property	Description	Connection String Keyword	Examples
	OLE DB provider ProgID	Provider	`Provider = PIOLEDBENT;`
DBPROP_AUTH_ INTEGRATED	Name of the authentication service Setting the property to Security Service Provider Interface (SSPI) invokes a trusted connection	Integrated Security	`Integrated Security = ; (default)` `Integrated Security = SSPI;`
DBPROP_AUTH_ USERID	User name	User ID	`User ID = piadmin;`
DBPROP_AUTH_ PASSWORD	Password	Password	`Password = PI;`
DBPROP_INIT_ CATALOG	Initial catalog (database)	Initial Catalog = Configuration;	`Initial Catalog = Configuration;`
DBPROP_INIT_ DATASOURCE	Name of the PI AF server that is the data source	Data Source	`Data Source = localhost;`
DBPROP_INIT_ HWND	Window handle from the calling application - used as a parent window of the dialog that prompts for the missing information	This property is not persisted in the connection string	N/A
DBPROP_INIT_ PROMPT	Prompt mode - designates if the provider should ask for the missing information	This property is not persisted in the connection string	N/A
DBPROP_INIT_ PROVIDERSTRING	Provider-specific properties	Extended Properties	`Extended Properties = "Always Return Rowset = True;"`
DBPROP_INIT_ TIMEOUT	Connection timeout in seconds This property is currently not used You must use PI System Explorer to change the default connection timeout	Connect Timeout	`Connect Timeout = 0;`

PI OLEDB Enterprise-Specific Initialization Properties

Connection String Keyword	Description	Examples
Allow Expensive	If set to **TRUE**, expensive SQL commands can be executed.	`Allow Expensive = False; (default)` `Allow Expensive = True;`
Always Return Rowset	If set, all SQL commands return rowsets that contain the number of rows affected by the execution.	`Defer Execution = False; (default)` `Defer Execution = True;`
Cancel On Low Resources	Cancels query execution if available memory pages reach a critical level (greater than five percent). Only available when run on Windows XP or later.	`Cancel On Low Resources = True; (default)`
Command Timeout	Command timeout in seconds. Overrides the **DBPROP_COMMANDTIMEOUT** property.	`Command Timeout = -1; (default; not set)` `Command Timeout = 60;`
Defer Execution	If set, the execution of requests that would ask for all table rows is deferred to the first access to the resulting rowset data.	`Defer Execution = False; (default)` `Defer Execution = True;`
Disable Server Selection	If set, the server combo box in the login dialog is disabled. This prevents users from changing the server during the connection procedure.	`Disable Server Selection = False; (default)` `Disable Server Selection = True;`
Embed Errors	If set, embed error messages in the value column.	`Embed Errors = False; (default)` `Embed Errors = True;`
Function Errors As NULL	If set to **TRUE**, SQL functions that reach an error state will return N**ULL**.	`Function Errors As NULL = False; (default)`
Ignore Errors	If set, an error retrieving a value will not cause the whole query to fail. Instead the row in error will be omitted.	`Ignore Errors = False; (default)` `Ignore Errors = True;`
Integers as Value	If set, integer point values are returned in the value column of the picomp and piinterp tables.	`Integers as Value = False; (default)` `Integers as Value = True;`

Connection String Keyword	Description	Examples
Keep Default Ordering	If set, the returned table data is ordered according to the criteria defined by the table metadata. If not set, the returned data might not keep this ordering. Setting this property to **FALSE** can speed up the query execution.	`Keep Default Ordering = True; (default)` `Keep Default Ordering = False;`
Log File	Full log file path/name.	`Log File = C:\Temp\Log\PIOLEDB.log;`
Log Level	Log level. Determines what information is to be logged. (For details, see *Activate Logging* (page 18)).	`Log Level = 0; (default)` `Log Level = 1;`
Optimization Log Limit	Logging of optimization criteria is limited to the first **n** parameters. This reduces log file size and improves readability. It should only be increased for optimization troubleshooting.	`Optimization Log Limit = 100; (default)`
Shorten Primary Keys	If set, the maximum length of string primary key columns is shortened to **255** characters. Microsoft Access does not support string primary keys longer than **255** characters. This property allows you to overcome this restriction.	`Shorten Primary Keys = False; (default)` `Shorten Primary Keys = True;`
Time as Double	If set, time columns use double precision floating-point number as the underlying data type (values are in seconds). This property increases the time precision to subseconds.	`Time as Double = False; (default)` `Time as Double = True;`

Security

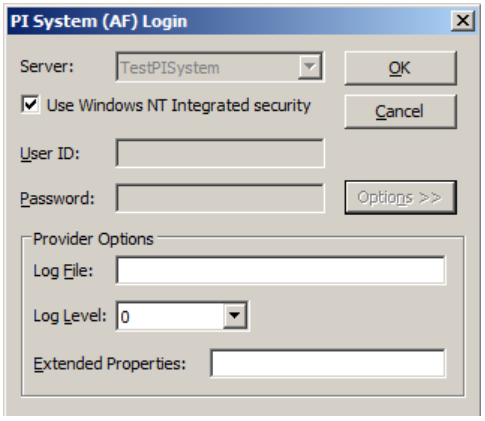
PI OLEDB Enterprise supports the Security Service Provider Interface (SSPI), which is the Microsoft version of the Generic Security Service API (GSSAPI) standard. SSPI is an interface that allows calls of various security functions that use authentication protocols supported by the operating system.

To configure SSPI, use either the connection string **Integrated Security=SSPI** or select the **Use Windows NT Integrated security** option in the *Login dialog* (page 16).

Note: If use of PI OLEDB Enterprise involves more than two machines, as in the case of middleware scenarios, you may need to configure Kerberos delegation. Examples are if PI OLEDB Enterprise connections are hosted from SharePoint or SQL Server.

Login Dialog

The login dialog is displayed during the initialization of the provider if enabled by the **DBPROP_INIT_PROMPT** property. Use the login dialog to set or change the initialization properties at runtime.



The PI AF server name must be configured for the machine that runs PI OLEDB Enterprise. Use **PI System Explorer** to configure the server name. For details, see *PI AF Overview Guide* and the **PI System Explorer** Help, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

Universal Data Link (UDL)

A Universal Data Link (UDL) file is a text file with a .udl extension that contains the connection string. Use UDL files for persisting and sharing of the connection information.

Use a UDL File

Use these steps to create a UDL file:

1. Open **Windows Explorer**.
2. Browse to the folder, in which the UDL file is to be created.
3. Right-click the folder and select **New-Text Document**.
4. In the **Tools** menu, click **Folder Options**. On the **View** tab, clear the **Hide file extensions for known file types** check box and then click **OK**.
5. Rename the file to have the `.udl` extension.

To add AF server connections:

1. Double-click the UDL file to open the Data Link Properties dialog.
2. Open the **Provider** tab and verify that PI OLEDB Enterprise is selected.
3. Click the **Connection** tab and enter the name of the default server.
4. Select the appropriate *security* (page 15) option.
5. Click **Test Connection**.

High Availability (HA)

PI OLEDB Enterprise is built with the AF SDK and therefore supports connection failover to servers in a PI collective when used with PI AF server versions 2.2 and later.

If a server connection becomes unavailable, SQL statements that are in progress may fail. Whether they fail depends on many factors, such as which tables are involved and whether the failover is fast enough so that the AF SDK method calls do not time out, and whether the statement execution is in the midst of an AF SDK call or between AF SDK calls. It is important to note that there is no guarantee for a successful execution, which means that the OLE DB consumer application needs to implement proper error handling and possible retry attempts.

If no SQL statements are in progress when a server connection failure and the automatic transition to another collective member occurs, the user or application may notice a longer execution time, or timeout, for the next statement executed. To avoid this timeout, command timeout values may be increased for HA connections.

Note: High Availability (HA) support is based on AF SDK functionality. For further details, see the topic AF High Availability Using an AF Collective in the *PI AF Overview Guide*, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

Date and Time Considerations

PI OLEDB Enterprise supports local time zone timestamps only, so that:

- Time references used in query strings must use local time
- Timestamps returned are in local time

Activate Logging

You can turn on logging with the Log File and Log Level initialization properties. To successfully create the log file, its parent directory must exist.

PI OLEDB Enterprise supports the following log levels:

Log level	Messages logged
0	- Initialization information, - SQL Engine errors
1	- OLE DB COM interface calls (excluding IRowset interface), - OLE DB COM errors and warnings, - SQL Engine information (optimization, execution time, and so on), - SQL Engine warnings
2	- OLE DB COM interface call input and output arguments (excluding IRowset interface),
3	- OLE DB IRowset COM interface calls, their input and output arguments

Disable OLE DB Services

There are two ways to disable OLE DB services:

- Edit the registry entry **[HKEY_CLASSES_ROOT\CLSID\{0D7893B6-1D54-4a22-8D40-AF2FF6B1955F}]** and use one of the following values that disables the service(s) as needed:

Services enabled	Value in connection string
All services (the default)	0xffffffff
All services except pooling	0xfffffffe
All services except pooling and auto-enlistment	0xfffffffc
All services except client cursor	0xfffffffb
All services except client cursor and pooling	0xfffffffa
No services	0x00000000

- For ADO-based applications, add one of the following strings to the connection string:

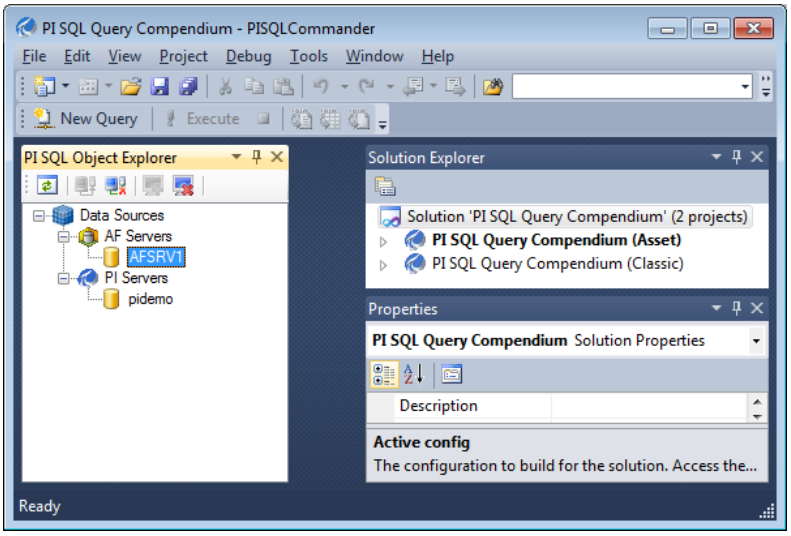
Services enabled	Value in connection string
All services (the default)	"OLE DB Services = -1;"
All services except pooling	"OLE DB Services = -2;"
All services except pooling and auto-enlistment	"OLE DB Services = -4;"
All services except client cursor	"OLE DB Services = -5;"
All services except client cursor and pooling	"OLE DB Services = -6;"
No services	"OLE DB Services = 0;"

Validate PI OLEDB Enterprise Installation

PI OLEDB Enterprise includes **PI SQL Commander**, an application you can use to validate your installation, learn the product, and develop and test queries. For more information, see **PI SQL Commander** Help, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

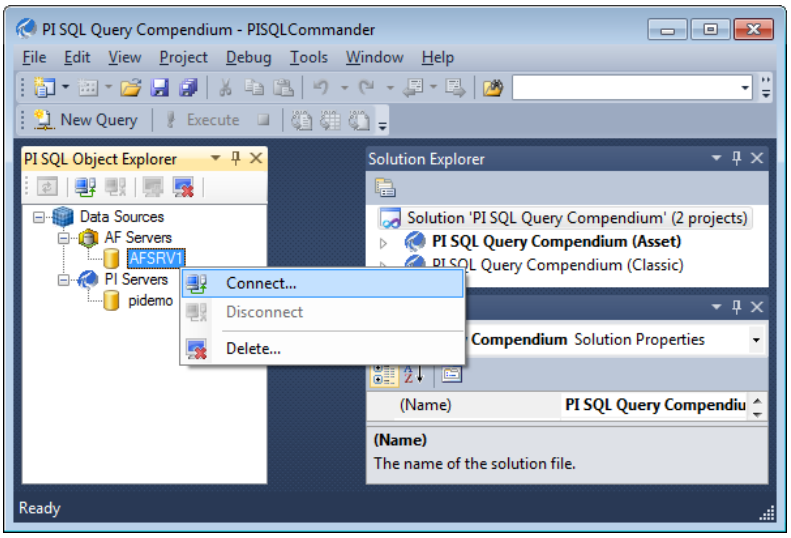
Open PI SQL Commander

1. Click **Start > All Programs > PI System > PI SQL Commander**.
2. Connect to a data source, as explained in *Connect to PI AF server* (page 20).

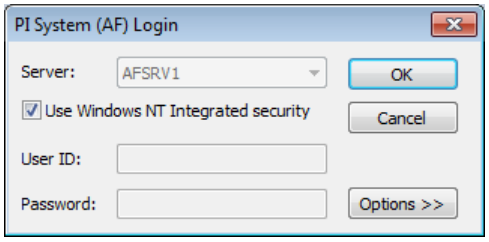


Connect to PI AF Server

1. Verify that the **PI OLEDB Enterprise Agent** service is running:
   a. Open Windows **Control Panel > Administrative Tools > Services** and find **PI OLEDB Enterprise Agent** in the list of services.
   a. Verify that the status of **PI OLEDB Enterprise Agent** is started.
2. In **PI SQL Commander**, verify that your **PI SQL Object Explorer** is visible. If it is not, click **View > PI SQL Object Explorer**.
3. Right-click a PI AF server icon in **PI SQL Object Explorer** and select **Connect**:

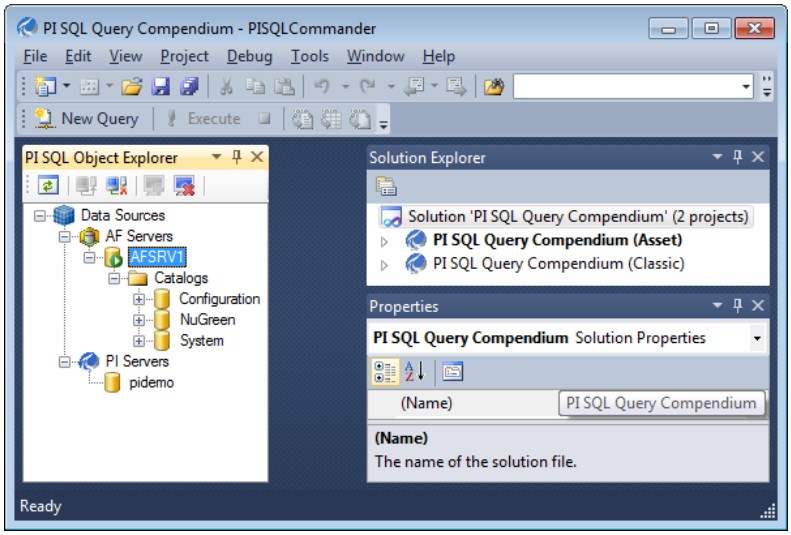


4. Provide credentials for a Windows user account, or select **Use Windows NT Authentication** to use your Windows credentials to connect to the PI AF server:



Note: To access the various PI AF data sources, PI data references, and so on, you must enter login information for an account with valid permissions for the machine that runs the PI AF server. For details, see the topic **AF Security through a Firewall** in the *PI AF Installation and Maintenance User Guide*, available at the *OSIsoft Technical Support Web site* (*http://techsupport.osisoft.com/*).

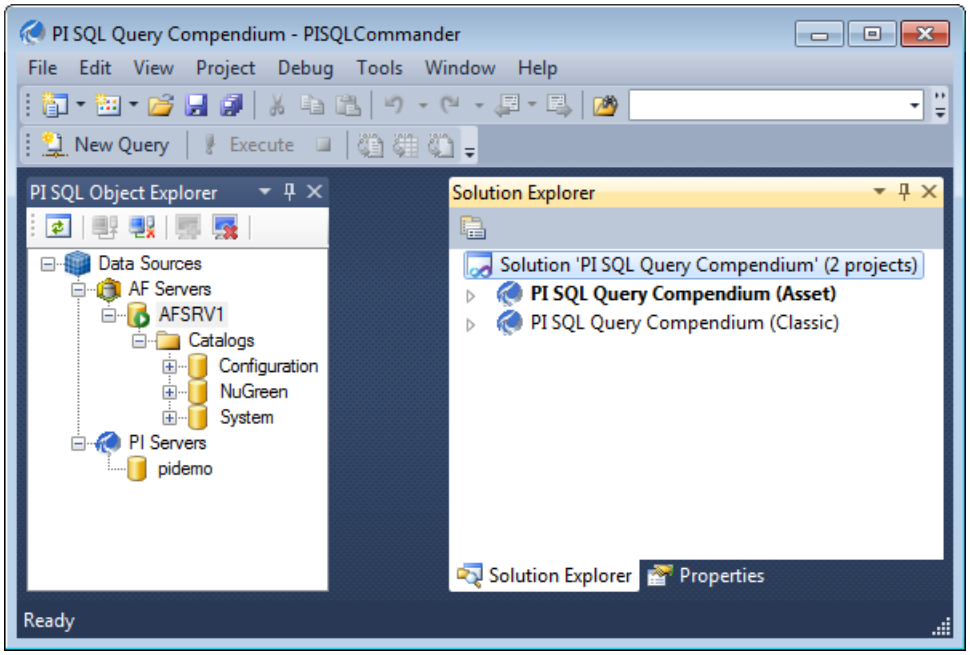
An arrow next to the server icon indicates that the connection is successful:



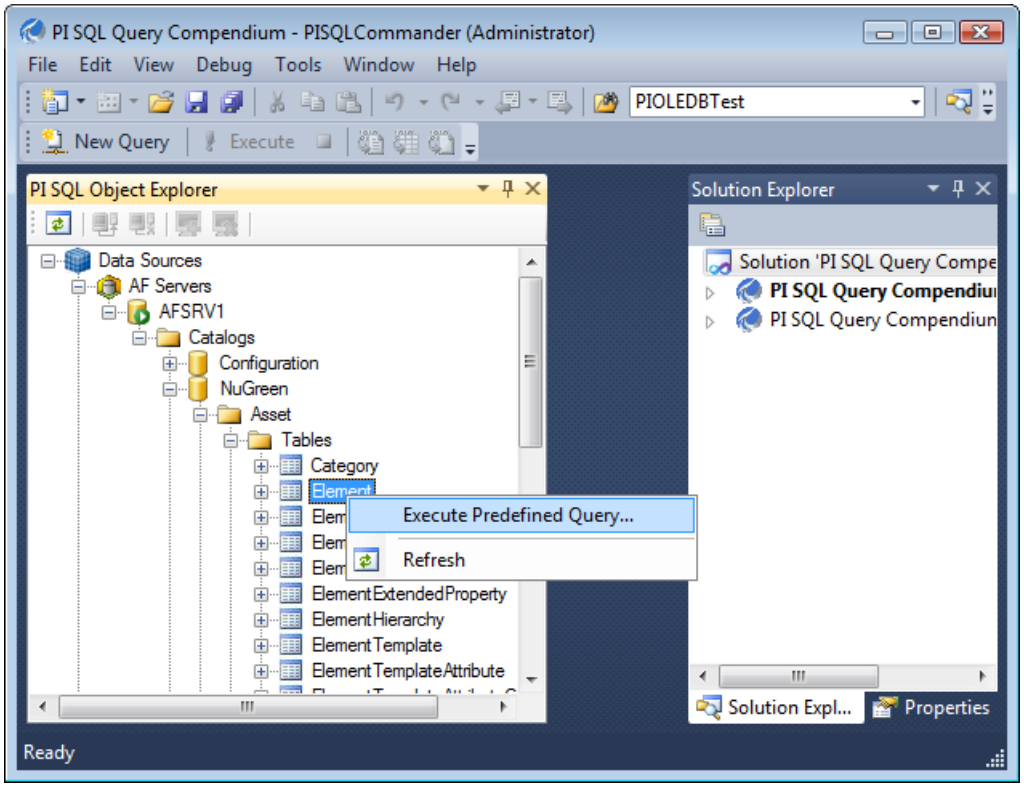
Run Sample SQL Queries

PI OLEDB Enterprise includes one sample SQL query for each table in the catalogs for PI AF server. Use these queries to explore PI SQL Commander functionality, or edit and save these queries to use as a custom query.

1. *Connect to a PI AF server* (page 20) and expand the schema under PI AF servers in **PI SQL Object Explorer**:

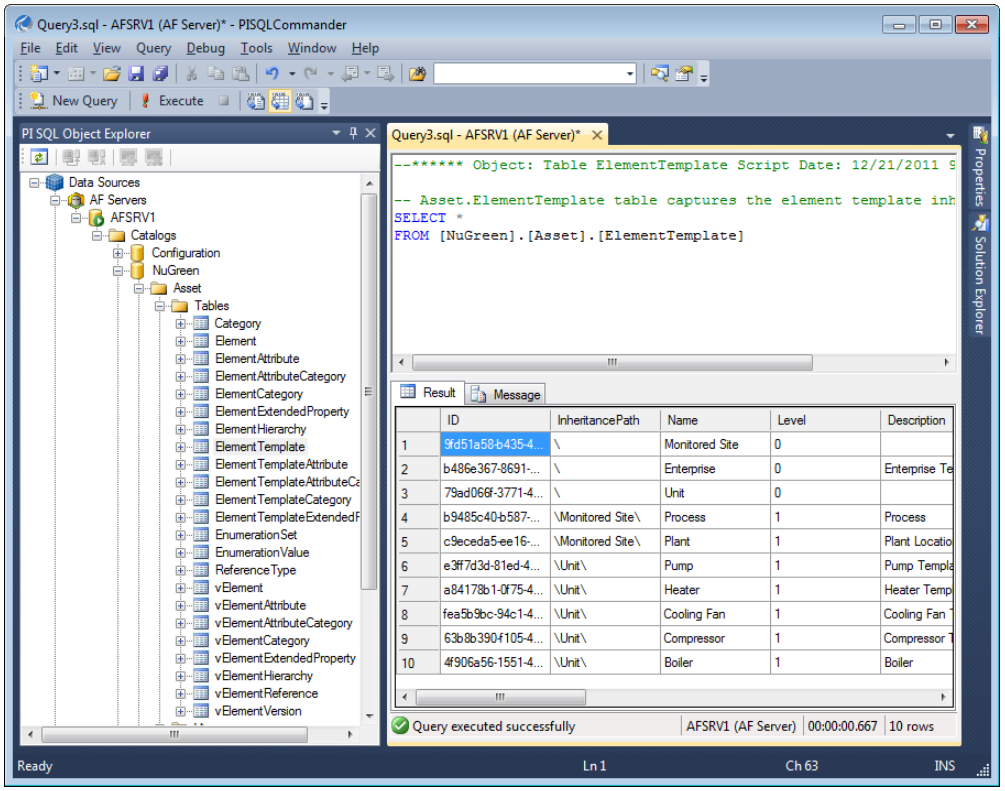


2. Right-click an object in the catalog that represents a table, view or function, and then select **Execute Predefined Query**:



Note: You can edit a sample query and save it as a customized query. Execute the query, save the updated query and add it to a project.

3. View the query in the **PI SQL Query Editor** viewer and its results in the **Result** pane:



Note: For information about two additional collections of sample queries, or projects, that you can load onto your system using **PI System Explorer**, see **PI System Explorer** Help.

To Remove PI OLEDB Enterprise

To uninstall PI OLEDB Enterprise, it is important to note that:

- Supporting components that the PI OLEDB Enterprise setup kit installs, such as PI SDK and AF SDK, must be removed separately.
- If you want to uninstall PI AF server, you must first remove *PI SQL for AF server* (applies to all PI AF Server versions between 2.1 and 2012).

Remove PI OLEDB Enterprise

Run the setup kit again and choose the Remove option, or use Windows Control Panel to remove PI OLEDB Enterprise.

Chapter 3

SQL Command Language

PI OLEDB Enterprise supports Structured Query Language (SQL) that is compliant with ANSI SQL 92 Entry Level.

Data Types

PI OLEDB Enterprise supports the following data types:

Name	OLE DB Data Type Indicator
Int8	DBTYPE_I1
Int16	DBTYPE_I2
Int32	DBTYPE_I4
Int64	DBTYPE_I8
UInt8	DBTYPE_UI1
UInt16	DBTYPE_UI2
UInt32	DBTYPE_UI4
UInt64	DBTYPE_UI8
Single	DBTYPE_R4
Double	DBTYPE_R8
Boolean	DBTYPE_BOOL
AnsiString	DBTYPE_STR
String	DBTYPE_WSTR
DateTime	DBTYPE_DATETIME
Variant	DBTYPE_VARIANT
Time	DBTYPE_TIME

SQL Statements

The basic structure in SQL is the statement. Each statement begins with a unique keyword. SQL statements consist of clauses which begin with a keyword. SQL syntax is not case sensitive.

A statement has no explicit limit in terms of string length, however implicit limits may apply to configuration details such as process space, hardware, OS configuration. In practice, only auto-generated statements exceed a size of **100,000** characters and OSIsoft recommends that you implement special scalability testing for such applications.

SELECT Statement

The SQL SELECT statement queries data from tables in the database.

Syntax

Note: Brackets ([]) denote optional parts of the statement, braces ({}) with vertical bars (|) denote mutually exclusive parts.

```
<select_statement> ::=
<query>
[UNION [ALL] <query> [UNION … ]]
[ORDER BY <expression> [ASC | DESC] [, …]]
[OPTION (option [, …])]

<query> ::=
SELECT [ALL | DISTINCT] [TOP integer_value] <select_list>
[FROM table_source [, …]
[WHERE <condition>]
[GROUP BY [ALL] <expression> [, …]]
[HAVING <condition>]
```

```
<select_list> ::=
{* |
{table_name | view_name | table_alias} .* |
<expression> [[AS] column_alias] |
column_alias = <expression>}
[, …]

<table_source> ::=
table_name [[AS] table_alias] |
view_name [[AS] table_alias] |
<select_statement> [[AS] table_alias] |
<joined_table> [[AS] table_alias]

<joined_table> ::=
<table_source> [INNER | LEFT OUTER] JOIN <table_source> ON
<condition>
<table_source> { CROSS | OUTER} APPLY <table_valued_function>

<table_valued_function> ::=
table_valued_function_name ([<expression> [, …]])|
table_values_function_template_name < [<expression> [, …]]>
([<expression> [, …]])



<option> ::=
FORCE ORDER | IGNORE ERRORS | EMBED ERRORS | ALLOW EXPENSIVE

<condition> ::=
{[NOT] <predicate> | (<condition>)}
[{AND | OR} <condition>]
[, …]
```

```
<predicate> ::=
<expression> {= | > | < | >= | <= | <> | !=} [ALL | ANY |
SOME] <expression> |
<expression> [NOT] IN ({<expression> [, …] |
<select_statement>}) |
<expression> [NOT] BETWEEN <expression> AND <expression> |
<expression> [NOT] LIKE <expression> [ESCAPE <expression>] |
<expression> IS [NOT] NULL |
EXISTS (<select_statement>)

<expression> ::=
<expression_factor> |
<expression > {+ | - | * | / |  %} <expression>

<expression_factor> ::=
integer_value | float_value | string_value |
True | False |
NULL | ? | column_name |
CAST (<expression> AS data_type_name) |
COUNT (*) |
aggr_function_name ([ALL | DISTINCT] <expression>) |
nonaggr_function_name ([<expression> [, …]]) |
[+ | -] <expression_factor> |
(<expression>) |
(IF <condition> THEN <expression> ELSE <expression>) |
NULLIF (<expression>, <expression>) |
COALESCE (<expression>, …) |
CASE <expression> WHEN <expression> THEN <expression> [WHEN …]
[ELSE <expression>] END |
CASE WHEN <condition> THEN <expression> [WHEN …] [ELSE
<expression>] END |
 (<select_statement>)
```

OPTION Query Hint

The **OPTION** clause specifies that the indicated query hint should be used throughout the entire query. Each query hint can be specified only one time, although multiple query hints are permitted. Only one **OPTION** clause can be specified with the statement.

A hint that is specified for a query only applies to tables in the query that support that hint.

OPTION (FORCE ORDER)

This query hint tells the query execution engine to retrieve data in the same order as the tables are listed in the query; for example, after the **FROM** clause. Intermediate results for one table can be used to parameterize data retrieval of the next table. For example it is advised to retrieve elements first to parameterize the attribute table or get attributes first in order to know what values should be retrieved.

OPTION (IGNORE ERRORS)

Normally a SQL query returns an error as soon a database call throws an exception. This query hint allows continuing retrieving data even if previous rows are in error. The result set will just omit rows in error but return all other rows.

This query hint is not supported by all tables. Tables that support **OPTION (IGNORE ERRORS)** are listed in *Table Structures* (page 55).

OPTION (EMBED ERRORS)

Tables in the Data catalog support this query hint to embed error messages in the value column.
If **OPTION (IGNORE ERRORS)** is also specified and a table supports both query hints, then **OPTION (EMBED ERRORS)** takes precedence.

Query hints can be combined with a comma separated list.

For example:

```
OPTION (FORCE ORDER, IGNORE ERRORS)
```

OPTION (ALLOW EXPENSIVE)

Unrestricted queries for a number of tables are considered expensive. To protect the system from unintended load such queries will return an error.
For example:

```
SELECT ea.*
FROM [NuGreen].[Asset].[ElementAttribute] ea
OPTION (FORCE ORDER)
```

returns the following error:

```
[NuGreen.Asset.ElementAttribute ea] 'ElementID' column is not
restricted. Such a query is considered expensive.
```

Possible solutions:

- Add a join with 'ElementHierarchy' or 'Element' table or, if the join is already present, reorder tables in the FROM clause and use the 'OPTION (FORCE ORDER)' query hint.
- Add 'OPTION (ALLOW EXPENSIVE)' query hint.

Setting the Option overwrites the standard behavior and allows the query to run.

CREATE VIEW Statement

The CREATE VIEW statement creates a new database view. A view is effectively a SQL query stored in the catalog.

Syntax

```
CREATE VIEW view_name [(column_name, [, …])] AS <select_statement>
```

Remarks

View name must be fully qualified, that is, `catalog..table`.

View metadata is stored in the PI Module Database, therefore PI Servers without a PI Module Database do not support creation of views.

ALTER VIEW Statement

The ALTER VIEW Statement modifies a previously created view.

Syntax

```
ALTER VIEW view_name [(column_name, [, …])] AS <select_statement>
```

DROP VIEW Statement

The DROP VIEW Statement removes a previously created view and its description from the catalog.

Syntax

```
DROP VIEW view_name
```

USE Statement

The USE Statement dynamically sets the default Catalog (PI AF database). It overwrites the Initial Catalog initialization property.

Syntax

```
USE database_name
```

Remarks

Because Catalog names are custom, the USE Statement allows generalizing queries and making them portable to other systems or allows queries being used with different Catalogs (AF databases).

For example, the statement:

```
SELECT et.Name ElementTemplate
FROM NuGreen.Asset.Category c
INNER JOIN NuGreen.Asset.ElementTemplateCategory etc ON
etc.CategoryID = c.ID
INNER JOIN NuGreen.Asset.ElementTemplate et ON et.ID =
etc.ElementTemplateID
WHERE c.Name = N'Equipment Assets'
```

can be shortened to:

```
SELECT et.Name ElementTemplate
FROM Asset.Category c
INNER JOIN Asset.ElementTemplateCategory etc ON etc.CategoryID =
c.ID
INNER JOIN Asset.ElementTemplate et ON et.ID =
etc.ElementTemplateID
WHERE c.Name = N'Equipment Assets'
```

and because Asset is the default schema it can be even shorter:

```
SELECT et.Name ElementTemplate
FROM Category c
INNER JOIN ElementTemplateCategory etc ON etc.CategoryID = c.ID
INNER JOIN ElementTemplate et ON et.ID = etc.ElementTemplateID
WHERE c.Name = N'Equipment Assets'
```

Operators and Functions

Note: In case of an evaluation error, operators and functions return **NULL**.

Arithmetic Operators

PI OLEDB Enterprise supports arithmetic operators for all numeric data types:

`+, -, *, /, %`

It also supports these operator overloads:

```
String Operator+(s1 String, s2 String)
WString Operator+(s1 WString, s2 WString)
DateTime Operator+(x DateTime, y Time)
DateTime Operator+(x Time y DateTime)
Time Operator+(x Time, y Time)

Time Operator-(x DateTime, y DateTime)
DateTime Operator-(x DateTime, y Time)
Time Operator-(x Time, y Time)

Time Operator*(x Int32, y Time)
Time Operator*(x Time, y Int32)
Time Operator/(x Time, y Int32)
```

Mathematical Functions

ABS

```
Int32 ABS(x Int32)
Float32 ABS(x Float32)
Float64 ABS(x Float64)
```

ACOS

```
Float32 ACOS(x Float32)
Float64 ACOS(x Float64)
```

ASIN

```
Float32 ASIN(x Float32)
Float64 ASIN(x Float64)
```

ATAN

```
Float32 ATAN(x Float32)
Float64 ATAN(x Float64)
```

ATAN2

```
Float32 ATAN2(x Float32, y Float32)
Float64 ATAN2(x Float64, y Float32)
```

CEILING

```
Float32 CEILING(x Float32)
Float64 CEILING(x Float64)
```

COS

```
Float32 COS(x Float32)
Float64 COS(x Float64)
```

COSH

```
Float32 COSH(x Float32)
Float64 COSH(x Float64)
```

EXP

```
Float32 EXP(x Float32)
Float64 EXP(x Float64)
```

FLOOR

```
Float32 FLOOR(x Float32)
Float64 FLOOR(x Float64)
```

LOG

```
Float32 LOG(x Float32)
Float64 LOG(x Float64)
```

LOG10

```
Float32 LOG10(x Float32)
Float64 LOG10(x Float64)
```

PI

```
Float64 PI()
```

POWER

```
Float32 POWER(x Float32, y Float32)
Float64 POWER(x Float64, y Float64)
```

ROUND

```
Float32 ROUND(x Float32)
Float32 ROUND(x Float32, y Int32)
Float64 ROUND(x Float64)
Float64 ROUND(x Float64, y Int32)
```

SIN

```
Float32 SIN(x Float32)
Float64 SIN(x Float64)
```

SINH

```
Float32 SINH(x Float32)
Float64 SINH(x Float64)
```

SQRT

```
Float32 SQRT(x Float32)
Float64 SQRT(x Float64)
```

TANH

```
Float32 TAN(x Float32)
Float64 TAN(x Float64)
```

String Functions

CONCAT

```
String CONCAT(s String, …)
WString CONCAT(s WString, …)
```

FORMAT

Specify a format for number data types:

```
String FORMAT(x Float64, s String)
String FORMAT (x Float32, s String)
String FORMAT (x Int32, s String)
WString FORMAT (x Float64, s WString)
WString FORMAT (x Float32, s WString)
WString FORMAT (x Int32, s WString)
```

The format string (2nd parameter) can contain one of the following characters:

Character	Description
#	Displays a digit. Each instance of the character represents a position for one number. If no value exists in a position, PI OLEDB Enterprise displays a blank space. For example, if you apply this format: `#,###`, but have a value of **45**, you see nothing but the number **45**. If you have a value of **12145**, PI OLEDB Enterprise displays **12,145** – even though you defined only one placeholder to the left of the thousands separator.
0	Displays a digit. Each instance of the character represents a position for one number. If no value exists in a position, PI OLEDB Enterprise displays a zero (**0**).
Thousands and decimal separators	Indicates where you want to place the thousands and decimal separators. Use the separators defined in your Windows regional settings.
E+, E- –or– e+, e-	Displays values in scientific (exponential) notation. Use this option when the predefined scientific format does not provide enough room for your values. Use E+ or e+ to display values as positive exponents, and E- or e- to display negative exponents. You must use these placeholders with other characters. For example, suppose you apply the format `0.000E+00` to a numeric field and then enter **612345**. PI OLEDB Enterprise displays **6.123E05**. PI OLEDB Enterprise first rounds the number of decimal places down to three (the number of zeroes to the right or left of the decimal separator). Next, PI OLEDB Enterprise calculates the exponent value from the number of digits that fall to the right (or left, depending on your language settings) of the decimal separator in the original value. In this case, the original value would have placed the five-digit value **02223** to the right of the decimal point. For that reason, PI OLEDB Enterprise displays **6.123E+05**, and the resulting value is the equivalent of `6.123 x 105`.

EXAMPLES

```
SELECT Format(5459.4, '##,##0.00')  -- Returns "5,459.40".
SELECT Format(334.9, '###0.00')     -- Returns "334.90".
SELECT Format(5, '0.00')            -- Returns "5.00".
SELECT Format(123.456, '##.##')     -- Returns "123.46".
```

Specify a format for DateTime data type:

```
String FORMAT(x DataTime, s String)
WString FORMAT (x DataTime, s WString)
```

The format string (2nd parameter) can contain one of the following characters:

Character	Description
d or dd	Used to display the day of the month as one or two digits. For one digit, use a single placeholder. For two digits, use two placeholders.
M or MM	Used to display the month as either a one-digit or two-digit number.
MMM	Used to abbreviate the name of the month to three letters. For example, October appears as Oct.
MMMM	Used to spell out all month names.
yyyy	Used to display the 4 digits of the year.
yy	Used to display the last two digits of the year.
H or HH	Used to display the hour as one or two digits. 24-hour format.
h or hh	Used to display the hour as one or two digits. 12-hour format.
tt	Used to display 12-hour clock values with a trailing AM or PM.
m or mm	Used to display minutes as one or two digits.
s or ss	Used to display seconds as one or two digits.
f or ff or fff or ffff or fffff	Used to display subsecond fractions of up to 5 digits.
/\:-.	Characters, including spaces, allowed as delimiter.

EXAMPLES

```
SELECT Format(MyDateTime, 'h:m:s')       -- Returns "5:4:23".
SELECT Format(MyDateTime, 'hh:mm:ss tt')  -- Returns --
"05:04:23 PM".
SELECT Format(MyDateTime, 'MMM d yyyy')   -- Returns -- "Jan
27 2011".
SELECT Format(MyDateTime, 'HH:mm:ss')    -- Returns "17:04:23"
SELECT Format(MyDateTime, 'hh:mm:ss.fff') -- Returns --
"05:04:23.000".
```

INSTR

```
Int32 INSTR(s1 String, s2 String, n Int32, m Int32)
Int32 INSTR(s1 String, s2 String, n Int32)
Int32 INSTR(s1 String, s2 String)
Int32 INSTR(s1 WString, s2 WString, n Int32, m Int32)
Int32 INSTR(s1 WString, s2 WString, n Int32)
Int32 INSTR(s1 WString, s2 WString)
```

Search **s2** in **s1** from the n-'th position the m-'th occurrence (default for n and m is **1**), result is the position found in **s1** or **0**.
n and m must have values >= **1**, otherwise the result is **NULL**.

The search is case insensitive.

LEFT

```
String LEFT(s String, n Int32)
WString LEFT(s WString, n Int32)
```

LEN

```
Int32 LEN(s String)
Int32 LEN(s WString)
```

LENGTH

Same as LEN

LOWER

```
String LOWER(s String)
WString LOWER(s WString)
```

LTRIM

```
String LTRIM(s String)
WString LTRIM(s WString)
```

REPLACE

```
String REPLACE(s String, findWhat String, replaceWith String)
WString REPLACE(s WString, findWhat WString, replaceWith WString)
```

RIGHT

```
String RIGHT(s String, n Int32)
WString RIGHT(s WString, n Int32)
```

RTRIM

```
String RTRIM(s String)
WString RTRIM(s WString)
```

SPACE

```
String SPACE(n Int32)
```

SUBSTR

```
String SUBSTR(s String, start Int32, len Int32)
WString SUBSTR(s WString, start Int32, len Int32)
```

TRIM

```
String TRIM(s String)
WString TRIM(s WString)
```

UPPER

```
String UPPER(s String)
WString UPPER(s WString)
```

Date/Time Functions

DAY

```
Int32 DAY(x DateTime)
```

FRACTION

```
Float64 FRACTION(x DateTime)
Float64 FRACTION(x Time)
```

HOUR

```
Int32 HOUR(x DateTime)
Int32 HOUR(x Time)
```

MINUTE

```
Int32 MINUTE(x DateTime)
Int32 MINUTE(x Time)
```

MONTH

```
Int32 MONTH(x DateTime)
```

ROUND

```
DateTime ROUND(x DateTime, fractionPrecision Int32 = 0)
```

SECOND

```
Int32 SECOND(x DateTime)
Int32 SECOND(x Time)
```

YEAR

```
Int32 YEAR(x DateTime)
```

Aggregate Functions

Aggregate functions **AVG, COUNT, MAX, MIN, SUM** are implemented in compliance with ANSI SQL.

AVG, COUNT, MAX, MIN, SUM

Aggregate functions are implemented in compliance with ANSI SQL.

List

See under PI SQL Functions

Table-Valued Functions

Table-Valued Functions are functions that return a table.

GetPIPoint

See under <AF Database>.Asset.GetPIPoint (ElementAttributeID GUID)

Transpose Table-Valued Functions

Transpose Table-Valued Functions (TVFs) are created using PI SQL Commander. See Transposition of Data Tables.

Transpose Table-Valued Functions can also dynamically be instantiated using one of the following templates:

```
[v]TransposeSnapshot<template_name String,
                     attribute_path String,
                     include_subtree Boolean,
                     values_as_variant Boolean>


[v]TransposeArchive<template_name String,
                     attribute_path String,
                     include_subtree Boolean,
                     values_as_variant Boolean>

[v]TransposeInterpolateRange<template_name String,
                     attribute_path String,
                     include_subtree Boolean,
                     values_as_variant Boolean>

[v]TransposeInterpolateDiscrete<template_name String,
                     attribute_path String,
```

```
                        include_subtree Boolean,
                        values_as_variant Boolean>
```

The prefix `[v]` denotes Transpose TVFs for versioned Data tables.

EXAMPLE

```
SELECT
    e.ID ElementID,
    e.Name PIAF_ElementName,
    t.Name PIAF_TemplateName,
    date(FORMAT(tir.Time, 'yyyy-MM-dd')) DateKey,
    time(FORMAT(tir.Time, 'HH:mm:ss')) TimeKey,
    tir.[\Production Metrics\Hourly Market Value] [Hourly Market Value Value]
FROM
(
 SELECT ID, Name
 FROM [Ontario_Wind]..[ElementTemplate] t
 WHERE t.InheritancePath like N'\Wind turbine\%' /*Inheritance: templateName*/
 UNION
 SELECT ID, Name
 FROM [Ontario_Wind]..[ElementTemplate] t
 WHERE t.Name = N'Wind turbine'                 /*Implementation: templateName*/
) t
INNER JOIN [Ontario_Wind]..[Element] e
ON e.ElementTemplateID = t.ID
--
-- dynamically use Transpose TVF
--
CROSS APPLY [Ontario_Wind]..[TransposeInterpolateRange]
<
    N'Wind turbine' /*templateName*/,
    '\'             /*attributePath*/,
    TRUE            /*includeSubtree*/,
    FALSE           /*valuesAsVariant*/
>
(
    e.ID,
    N'9/1/2011 12:00:00 AM' /*startTime*/,
    N'9/15/2011 12:00:00 AM' /*endTime*/,
    N'15m'          /*timeStep*/
) tir
WHERE tir.[\Production Metrics\Hourly Market Value] IS NOT NULL
OPTION (IGNORE ERRORS, FORCE ORDER)
```

PI SQL Functions

ParentName

```
String ParentName(Path String, Level = 0 Int32)
```

Returns parent name. Level can vary from zero (direct parent) to the path length - 1.

GetPIPoint

Also available as function table **ft_GetPIPoint.**

See under <AF Database>.Asset.GetPIPoint (ElementAttributeID GUID)

List

```
String List(s String)
String List(s String, Separator String)
```

Returns the concatenated strings of a given field (column or expression). Substrings can be divided by an optional separator.

VarType

```
String VarType(s String)
```

Returns the embedded data type of a given VARIANT field. This function should only be used for fields of type VARIANT.

UOMID

```
GUID UOMID(uomNameOrAbbreviation String)
```

Returns the ID for a given Unit of Measure (UOM). The function searches through UOM names and UOM abbreviations.

UOMName

```
String UOMName(uomID GUID)
```

Returns the name for a given UOM ID.

UOMAbbreviation

```
String UOMAbbreviation(uomID GUID)
```

Returns the abbreviation for a given UOM ID.

UOMClassName

```
String UOMClassName(uomID GUID)
```

Returns the class name for a given UOM ID.

UOMCanonicalID

```
GUID UOMID(uomID GUID)
```

Returns the canonical UOM ID for a given Unit of Measure.

UOMConvert

```
Double UOMConvert(x Double, fromUomID GUID, toUomID GUID)
```

Chapter 4

Metadata

PI OLEDB Enterprise Catalogs

PI OLEDB Enterprise exposes one *fixed* catalog – System. Other catalogs expose data from the various PI AF databases within a PI AF server. Their names are patterned after PI AF database names.

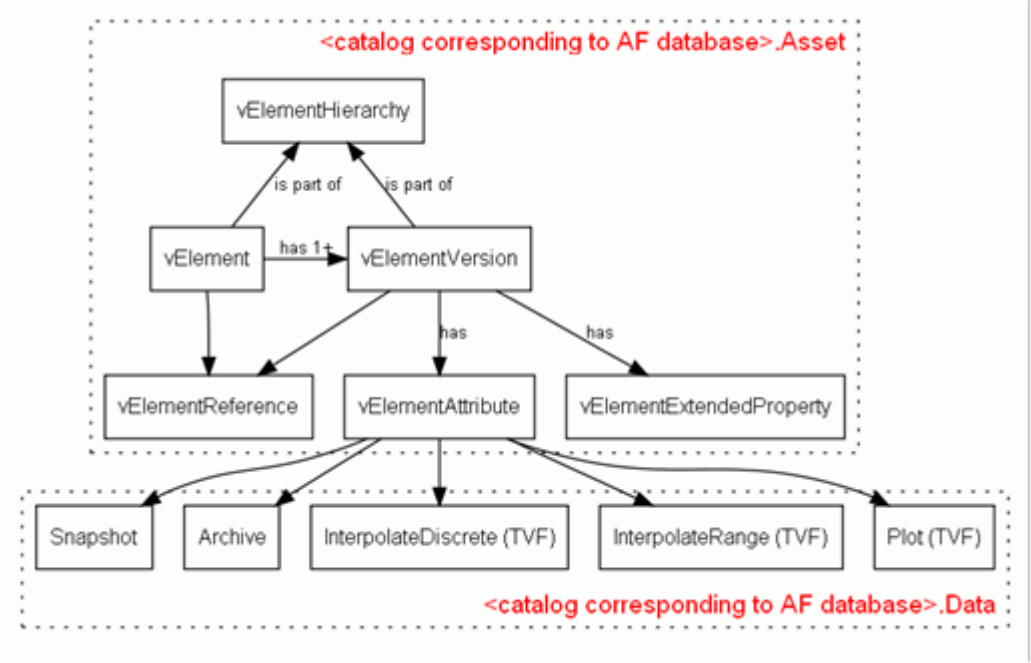
Catalog	Schema	Database Object	Type
<AF database>	Asset	Category	Table
		Element	Table
		ElementAttribute	Table
		ElementExtendedProperty	Table
		ElementHierarchy	Table
		ElementAttributeCategory	
		ElementCategory	Table
		ElementTemplate	Table
		ElementTemplateAttribute	
		ElementTemplateAttributeCategory	Table
		ElementTemplateCategory	Table
		ElementTemplateExtendedProperty	Table
		EnumerationSet	Table
		EnumerationValue	Table
		vElement	
		vElementAttribute	Table
		vElementExtendedProperty	Table
		vElementHierarchy	
		vElementReference	Table
		vElementVersion	Table
		ReferenceType	Table
		ft_GetPIPoint	Table

	Data	Snapshot	Table
		ft_InterpolatedDiscrete	Function Table
		InterpolateDiscrete	Table Valued Function
		ft_InterpolateRange	Function Table
		InterpolateRange	Table Valued Function
		ft_Plot	Function Table
		Plot	Table Valued Function
		Archive	Table
	DataT		custom Table-Valued Functions and Function Tables
	EventFrame	EventFrame	Table
		EventFrameAttribute	Table
		EventFrameAttributeCategory	Table
		EventFrameAttributeHierarchy	Table
		EventFrameAttributeReference	Table
		EventFrameAttributeReferencedElement	Table
		EventFrameTemplate	Table
		EventFrameTemplateAttribute	Table
		EventFrameTemplateAttributeCategory	Table
		EventFrameTemplateCategory	Table
		ft_GetPIPoint	Table
System	AF	PlugIn	Table
	Connection	ConnectionProperty	Table
		ConnectionStatus	Table
		ProductVersion	Table
	UnitOfMeasure	BaseUOMClass	Table
		UOM	Table
		UOMClass	Table
		UOMDatabase	Table

PI OLEDB Enterprise E-R Model

This entity-relationship model reflects how various PI System data is represented with tables.

Asset and Data Schema



Four tables in the upper part of the diagram show the Asset schema that forms the core of the asset database representation: ElementHierarchy, Element, ElementVersion and ElementAttribute.

Note: Tables that support element versions have a **v** prefix.

Tables and Table-Valued Functions (TVFs) in the lower part of the diagram show the Data schema that represents the gateway to time-series data.

Asset.vElementHierarchy table is usually the starting point for most queries. This table represents the element hierarchy and supports retrieval of elements based on various criteria, such as query date, path pattern or hierarchy level restriction, and so on.

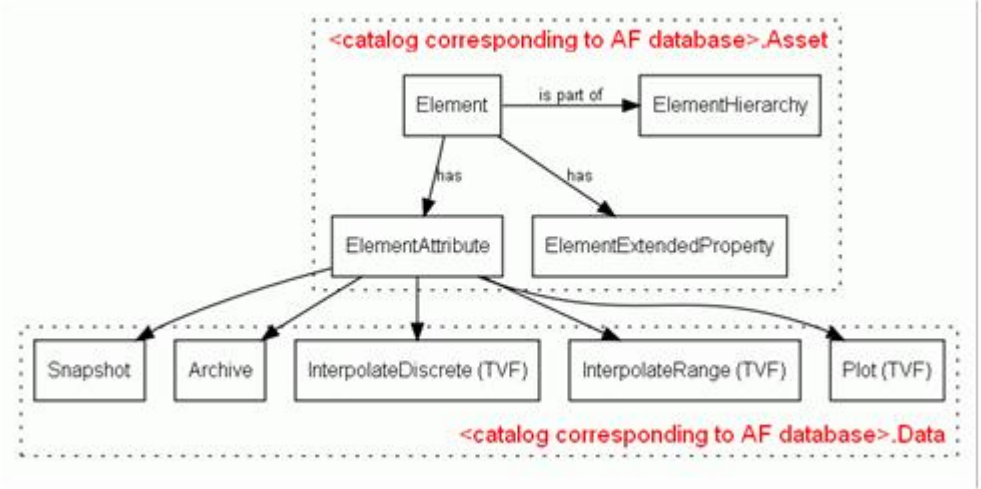
Typically, queries into **Asset.vElementHierarchy** are extended as follows:

- Element attribute query = inner join of **Asset.vElementHierarchy**, **Asset.vElementVersion**, and **Asset.vElementAttribute**.
- Element extended property query = inner join of **Asset.vElementHierarchy**, **Asset.vElementVersion**, and **Asset.vElementExtendedProperty**.

Time-series data query = inner join **Asset.vElementHierarchy**, **Asset.vElementVersion**, **Asset.vElementAttribute**, and any one table from the Data schema.

Typical queries are as follows:

- Element attribute query = inner join of **Asset.ElementHierarchy** and **Asset.ElementAttribute**.
- Element extended property query = inner join of **Asset.ElementHierarchy** and **Asset.ElementExtendedProperty**.
- Time-series data query = inner join **Asset.ElementHierarchy**, **Asset.ElementAttribute**, and any one table or TVF dedicated to element attributes from the Data schema.

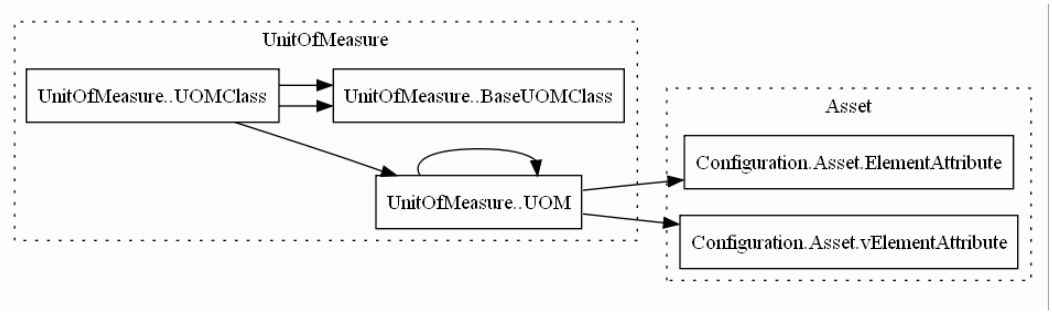


Current Hierarchy

Alternatively, queries that refer to the current element hierarchy can also be formulated with simplified element tables. As there is just one version valid at the current time, the simplified E-R model does not reference a table of element versions. Typical queries are as follows:

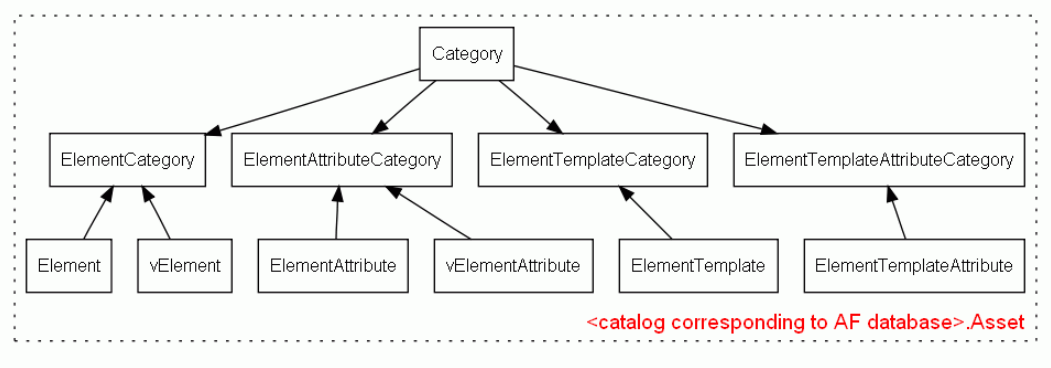
- Element attribute query = inner join of **Asset.ElementHierarchy** and **Asset.ElementAttribute**.
- Element extended property query = inner join of **Asset.ElementHierarchy** and **Asset.ElementExtendedProperty**.
- Time-series data query = inner join **Asset.ElementHierarchy**, **Asset.ElementAttribute**, and any one table or TVF dedicated to element attributes from the Data schema.

Units of Measure



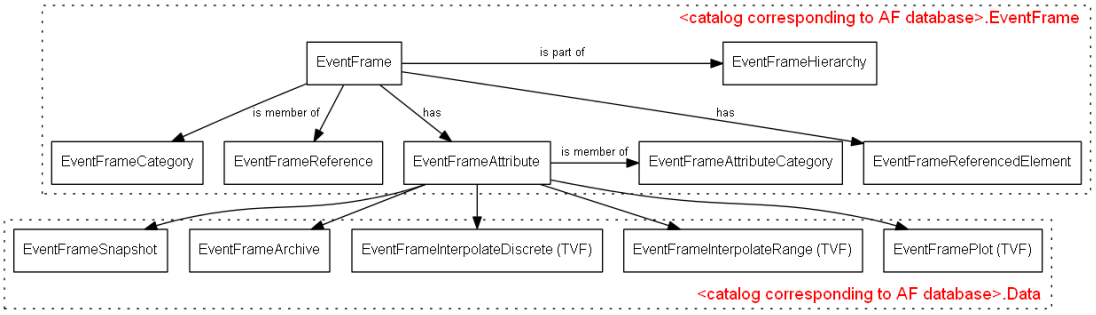
Units of measure are represented by three tables: **UOMClass**, **UOMBaseClass**, and **UOM**. Default element attribute **UOM** is represented as a link from **ElementAttribute / vElementAttribute** table to **UOM** table.

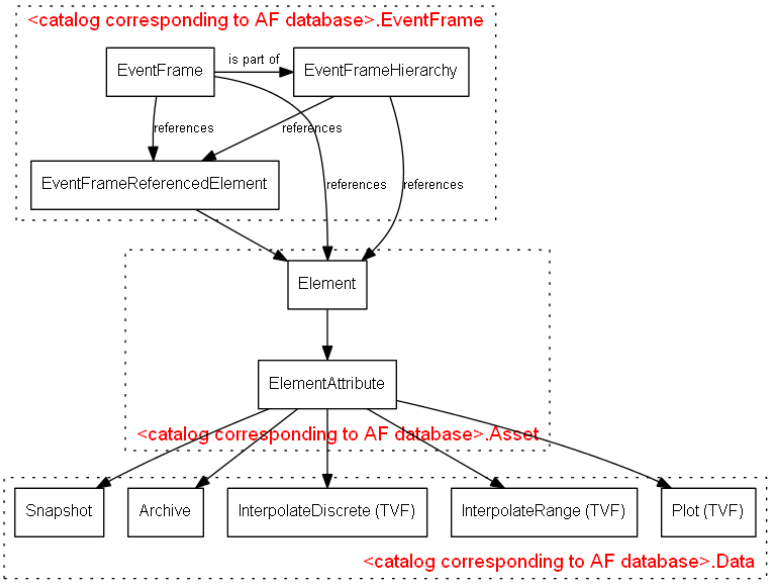
Categories



Elements, element attributes, element templates, and element template attributes can be placed into one or more categories, which helps with organization and searching. Categorization is captured by **ElementCategory**, **ElementAttributeCategory**, **ElementTemplateCategory**, and **ElementTemplateAttributeCategory** tables which refer to the global category table as well as to the element, element attribute, element template, and element template attribute tables (M-N relationship decompositions).

EventFrame Schema*





Tables representing Event Frames, such as **EventFrame**, **EventFrameHierarchy** , **EventFrameAttribute**, etc., are grouped into the EventFrame schema.

Event Frames can reference elements (column PrimaryReferencedElementID) and refer to data via element attributes.

Event Frames can also have attributes (table **EventFrameAttributes**) and refer to data via these attributes. Event Frame attributes reference data tables in the Data schema, prefixed EventFrame (e.g. **EventFrameSnapshot**, **EventFrameArchive**, ft_**EventFramePlot**).

Typical queries are as follows:

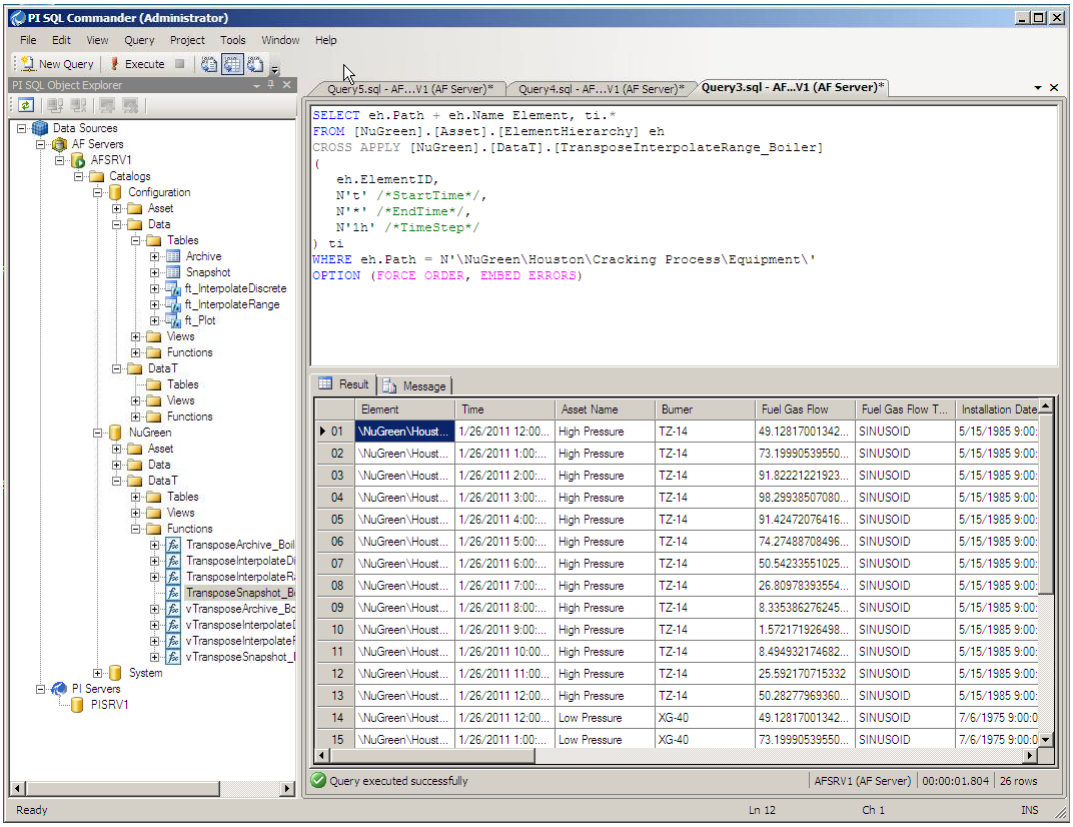
- Event Frame Element attribute query = inner join of **EventFrame.EventFrame** and **Asset.ElementAttribute**.
- Event Frame attribute query = inner join of **EventFrame.EventFrame** and **EventFrame.EventFrameAttribute**.

- Time-series data query = inner join **EventFrame.EventFrame**,**Asset.ElementAttribute**, and any one data table or TVF dedicated to element attributes from the Data schema.
  or
  inner join **EventFrame.EventFrame**,**EventFrame.EventFrameAttribute**, and any one data table or TVF dedicated to Event Frames from the Data schema

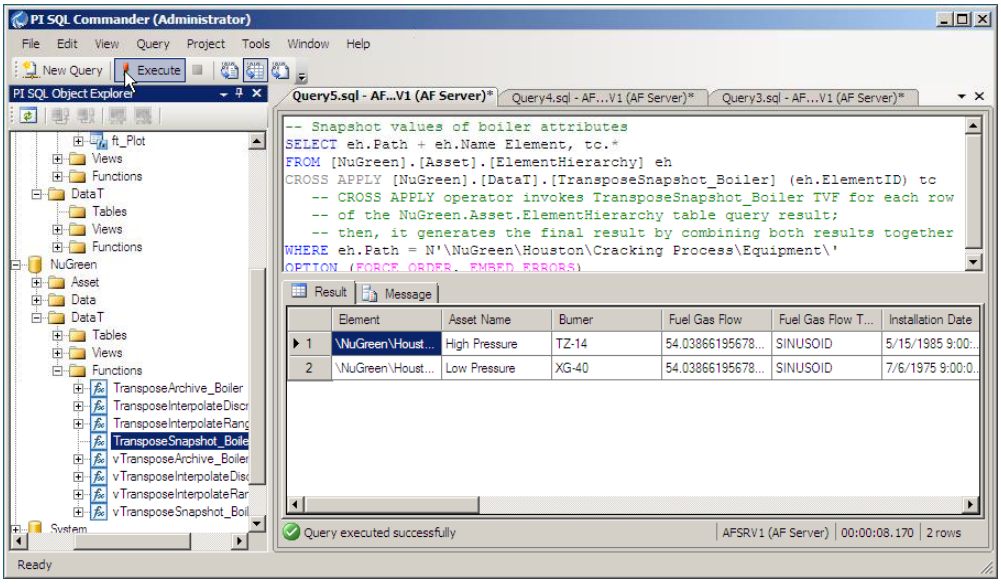
*__Note:__ PI OLEDB Enterprise Event Frames support requires PI AF server 2012 or later.

Transposition of Data Tables

Built-in data tables such as Snapshot, Archive, and so on, return attributes and their values in rows.



For various use cases such as reports or OLAP cubes, attribute values need to be returned in a way so that each column represents an attribute. This is contrary to a typical relational representation, where each value of each attribute is normally represented in consecutive rows. To represent multiple attributes in this "one column per attribute" format, one could join data tables with itself multiple times, but the resulting query string would be rather large and complex. To help with this, we provide a way to create custom Table-Valued Functions (TVFs) and derived function tables, to get "transposed" result sets of the related data tables.



PI OLEDB Enterprise supports custom Table-Valued Functions (TVFs) based on AF element templates. In this case, the element template and the attribute templates it contains, defines the metadata for the TVF. Typically, TVFs are incorporated into queries that use the **CROSS APPLY** keyword. For details, see *Compendium of PI SQL Statements (AF)* (page 83).

For certain use cases where the environment does not allow the use of TVFs, the provider also supports the creation of function tables derived from these TVFs. These function tables have the same metadata and simply require that parameter columns be constrained (like a function). Use this as an alternative to the **CROSS APPLY** keyword and syntax.

Note: This user guide refers to parameter columns as columns that have the same name as the parameters in corresponding TVFs.

You can create transposed TVFs and corresponding function tables through a wizard in **PI SQL Commander**. For details, see the file `TransposedData.sql` in **PI SQL Commander**:

1. Open **PI SQL Commander**.
2. Go to **View > PI SQL Object Explorer** > **Object Explorer**.
3. Select **PI SQL Query Compendium (Asset)**, then open the **Queries > Transposed Data** folder.
4. Double-click the file `TransposedData.sql`.

Metadata Updates

Metadata (for example, table columns and their type) can change due to the execution of Data Manipulation Language (DML) statements or the external system configuration. The PI OLEDB Enterprise provider has an internal refresh rate of three minutes and automatically updates metadata. For example, if one user creates a Transpose TVF function, that function will be visible and callable by another user within three minutes.

Custom Database Objects

Custom database objects such as Views, Transpose Functions and Transpose Function Tables are stored in the PI AF Configuration database. When exporting a database via PI AF System Explorer, these objects are not included. If a database needs to be re-imported or moved to another system, the custom database objects need to be copied separately.

The PI SQL Commander offers an export/import wizard for custom database objects. For details, see chapters “Import custom database objects” and “Export custom database objects” in the PI SQL Commander Help.

Note: Direct export/import of the Configuration database elements that represent PI SQL custom database objects does not work.

Table and Table-Valued Function Structures

In the following section tables and Table-Valued Functions are listed by their names. In queries they can be referenced using fully qualified names

<AF Database>.<Schema>.<table or TVF>

or

.<Schema>.<table or TVF> (using the default database)

Note: Function Tables are marked using the icon  and Table-Valued Functions are marked with 

Asset Schema

.Asset.Category

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
Type	String	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.Element

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
Comment	String	Yes	No
Revision	Int32	No	No
HasChildren	Boolean	No	No
HasMultipleVersions	Boolean	No	No

Name	Data type	Is nullable?	Primary Key
ElementTemplateID	GUID	Yes	No
DBReferenceTypeID	GUID	Yes	No
DefaultElementAttributeID	GUID	Yes	No
SecurityDescriptor	String	No	No
Created*	DateTime	Yes	No
CreatedBy*	String	Yes	No
Modified*	DateTime	Yes	No
ModifiedBy*	String	Yes	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

* Column available when connected to PI AF Server 2010 R3 Patch or later version

.Asset.vElement

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
HasMultipleVersions	Boolean	No	No
DBReferenceTypeID	GUID	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.ElementAttribute

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
ValueType	String	No	No
EnumerationSetID	GUID	Yes	No

Name	Data type	Is nullable?	Primary Key
EnumerationSetName	String	Yes	No
IsConfigurationItem	Boolean	No	No
DataReferencePlugInID	GUID	Yes	No
ConfigString	String	Yes	No
DefaultUOMID	GUID	Yes	No
ElementTemplateAttributeID	GUID	Yes	No
ParentID	GUID	Yes	No
ElementID	GUID	No	No

.Asset.vElementAttribute

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
ValueType	String	No	No
EnumerationSetID	GUID	Yes	No
EnumerationSetName	String	Yes	No
IsConfigurationItem	Boolean	No	No
DataReferencePlugInID	GUID	Yes	No
ConfigString	String	Yes	No
DefaultUOMID	GUID	Yes	No
ElementTemplateAttributeID	GUID	Yes	No
ParentID	GUID	Yes	No
ElementVersionID	GUID	No	No

.Asset.ElementExtendedProperty

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
ValueInt	Int64	Yes	No
ValueDbl	Float64	Yes	No
ValueStr	String	Yes	No
ValueGuid	GUID	Yes	No
ElementID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

.Asset.vElementExtendedProperty

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
ValueInt	Int64	Yes	No
ValueDbl	Float64	Yes	No
ValueStr	String	Yes	No
ValueGuid	GUID	Yes	No
ElementVersionID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

.Asset.ElementHierarchy

Name	Data type	Is nullable?	Primary Key
Path	String	No	Yes
Name	String	No	Yes
Level	Int16	No	No
ElementID	GUID	No	No
ParentElementID	GUID	Yes	No
ReferenceTypeID	GUID	No	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.vElementHierarchy

Name	Data type	Is nullable?	Primary Key
Path	String	No	Yes
Name	String	No	Yes
Level	Int16	No	No
Effective	DateTime	No	Yes
Ineffective	DateTime	No	No
ElementID	GUID	No	No
ElementVersionID	GUID	No	No
ElementVersionEffective	DateTime	No	No
ElementVersionIneffective	DateTime	No	No
ParentElementID	GUID	Yes	No
ParentElementVersionID	GUID	Yes	No
ReferenceTypeID	GUID	No	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.vElementVersion

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Effective	DateTime	No	No
Ineffective	DateTime	No	No
Obsolete	DateTime	Yes	No
Comment	String	Yes	No
ElementTemplateID	GUID	Yes	No
DefaultElementAttributeID	GUID	Yes	No
Revision	Int32	No	No
HasChildren	Boolean	No	No
ElementID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

.Asset.ElementCategory

Name	Data type	Is nullable?	Primary Key
ElementID	GUID	No	Yes
CategoryID	GUID	No	Yes

.Asset.vElementCategory

Name	Data type	Is nullable?	Primary Key
ElementVersionID	GUID	No	Yes
CategoryID	GUID	No	Yes

.Asset.ElementAttributeCategory

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
CategoryID	GUID	No	Yes

.Asset.vElementReference

Name	Data type	Is nullable?	Primary Key
ParentElementVersionID	GUID	No	Yes
ChildElementID	GUID	No	Yes
ReferenceTypeID	GUID	No	No

.Asset.ElementReference

Name	Data type	Is nullable?	Primary Key
ParentElementVersionID	GUID	No	Yes
ChildElementID	GUID	No	Yes
ReferenceTypeID	GUID	No	No

.Asset.ReferenceType

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
ParentName	String	No	No
ChildName	String	No	No
Strength	UInt8	No	No
AllowedParentElementTemplateID	GUID	Yes	No
AllowedChildElementTemplateID	GUID	Yes	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.ElementTemplate

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
InheritancePath	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
Type	String	No	No
InstanceType	String	Yes	No
AllowElementToExtend	Boolean	No	No
DefaultElementTemplateAttributeID	GUID	Yes	No
BaseElementTemplateID	GUID	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.ElementTemplateCategory

Name	Data type	Is nullable?	Primary Key
ElementTemplateID	GUID	No	Yes
CategoryID	GUID	No	Yes

.Asset.ElementTemplateExtendedProperty

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
ValueInt	Int64	Yes	No
ValueDbl	Float64	Yes	No
ValueStr	String	Yes	No
ValueGuid	GUID	Yes	No
ElementTemplateID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

.Asset.ElementTemplateAttribute

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
IsConfigurationItem	Boolean	No	No
ValueType	String	No	No
EnumerationSetID	GUID	Yes	No
EnumerationSetName	String	Yes	No
DataReferencePlugInID	GUID	Yes	No
ConfigString	String	Yes	No
DefaultValueInt	Int64	Yes	No
DefaultValueDbl	Float64	Yes	No
DefaultValueStr	String	Yes	No
DefaultValueDateTime	DateTime	Yes	No
DefaultValueGuid	GUID	Yes	No
DefaultUOMID	GUID	Yes	No
IsIndexed	Boolean	No	No
ParentID	GUID	Yes	No
ElementTemplateID	GUID	No	No

.Asset.ElementTemplateAttributeCategory

Name	Data type	Is nullable?	Primary Key
ElementTemplateAttributeID	GUID	No	Yes
CategoryID	GUID	No	Yes

.Asset.EnumerationSet

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.Asset.EnumerationValue

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
Value	Int32	No	No
EnumerationSetID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

.Asset.GetPIPoint (ElementAttributeID GUID)

Name	Data type	Is nullable?	Primary Key
Path	String	Yes	No
Server	String	Yes	No
Tag	String	Yes	No

.Asset.ft_GetPIPoint

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	No
Path	String	Yes	No
Server	String	Yes	No

Name	Data type	Is nullable?	Primary Key
Tag	String	Yes	No

EventFrame Schema

.EventFrame.EventFrame

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
StartTime	DateTime	No	No
EndTime	DateTime	Yes	No
EventFrameTemplateID	GUID	Yes	No
PrimaryParentID	GUID	Yes	No
PrimaryParentReferenceTypeID	GUID	Yes	No
PrimaryReferencedElementID	GUID	Yes	No
Revision	Int32	No	No
IsRoot	Boolean	No	No
HasChildren	Boolean	No	No
HasReferencedElements	Boolean	No	No
SecurityDescriptor	String	No	No
Modified	DateTime	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.EventFrame.EventFrameAttribute

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
IsConfigurationItem	Boolean	No	No

Name	Data type	Is nullable?	Primary Key
ValueType	String	No	No
EnumerationSetID	GUID	Yes	No
EnumerationSetName	String	Yes	No
DataReferencePlugInID	GUID	Yes	No
ConfigString	String	Yes	No
DefaultUOMID	GUID	Yes	No
EventFrameTemplateAttributeID	GUID	Yes	No
EventFrameID	GUID	No	No

.EventFrame.EventFrameAttributeCategory

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
CategoryID	GUID	No	Yes

.EventFrame.EventFrameCategory

Name	Data type	Is nullable?	Primary Key
EventFrameID	GUID	No	Yes
CategoryID	GUID	No	Yes

.EventFrame.EventFrameHierarchy

Name	Data type	Is nullable?	Primary Key
IDPath*	String	No	Yes
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
StartTime	DateTime	No	No
EndTime	DateTime	Yes	No
EventFrameTemplateID	GUID	Yes	No
ParentID	GUID	Yes	No
ParentReferenceTypeID	GUID	Yes	No
DefaultEventFrameAttributeID	GUID	Yes	No
PrimaryReferencedElementID	GUID	Yes	No
Revision	Int32	No	No

Name	Data type	Is nullable?	Primary Key
IsRoot	Boolean	No	No
HasChildren	Boolean	No	No
HasReferencedElements	Boolean	No	No
SecurityDescriptor	String	No	No
Modified	DateTime	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

* Column available when connected to PI AF Server versions later than PI AF Server 2012

.EventFrame.EventFrameReference

Name	Data type	Is nullable?	Primary Key
ParentEventFrameID	GUID	No	Yes
ChildEventFrameID	GUID	No	Yes

.EventFrame.EventFrameReferencedElement

Name	Data type	Is nullable?	Primary Key
EventFrameID	GUID	No	Yes
ElementID	GUID	No	Yes

.EventFrame.EventFrameTemplate

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	No
InheritancePath	String	No	Yes
Name	String	No	Yes
Level	Int16	No	No
Description	String	Yes	No
Type	String	No	No
AllowEventFrameToExtend	Boolean	No	No
DefaultEventFrameTemplateAttributeID	GUID	Yes	No
BaseEventFrameTemplateID	GUID	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No

Name	Data type	Is nullable?	Primary Key
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

.EventFrame.EventFrameTemplateAttribute

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Path	String	No	No
Name	String	No	No
Level	Int16	No	No
Description	String	Yes	No
IsConfigurationItem	Boolean	No	No
ValueType	String	No	No
EnumerationSetID	GUID	Yes	No
EnumerationSetName	String	Yes	No
DataReferencePlugInID	GUID	Yes	No
ConfigString	String	Yes	No
DefaultValue	Variant	Yes	No
DefaultValueInt	Int64	Yes	No
DefaultValueDbl	Double	Yes	No
DefaultValueStr	String	Yes	No
DefaultValueGuid	GUID	Yes	No
DefaultValueDateTime	DateTime	Yes	No
DefaultUOMID	GUID	Yes	No
IsIndexed	Boolean	No	No
ParentID	GUID	Yes	No
EventFrameTemplateID	GUID	No	No

.EventFrame.EventFrameTemplateAttributeCategory

Name	Data type	Is nullable?	Primary Key
EventFrameTemplateAttributeID	GUID	No	Yes
CategoryID	GUID	No	Yes

.EventFrame.EventFrameTemplateCategory

Name	Data type	Is nullable?	Primary Key
EventFrameTemplateID	GUID	No	Yes
CategoryID	GUID	No	Yes

. EventFrame.GetPIPoint (EventFrameAttributeID GUID)

Name	Data type	Is nullable?	Primary Key
Path	String	Yes	No
Server	String	Yes	No
Tag	String	Yes	No

. EventFrame.ft_GetPIPoint

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	No
Path	String	Yes	No
Server	String	Yes	No
Tag	String	Yes	No

Data Schema

All tables and Table-Valued Functions in the Data schema share the following columns:

.Data.<table>

Name	Data type	Is nullable?	Primary Key
Time	DateTime	No	Yes
Value	Variant	Yes	No
ValueInt	Int64	Yes	No
ValueDbl	Float64	Yes	No
ValueStr	String	Yes	No
ValueGuid	GUID	Yes	No
ValueDateTime	DateTime	Yes	No
Status	Int32	No	No
Annotated	Boolean	No	No
IsGood	Boolean	No	No
Questionable	Boolean	No	No
Substituted	Boolean	No	No

Below are the data tables and Table-Valued Functions, showing their specific columns only:

.Data.Snapshot

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
ElementTemplateAttributeID	GUID	Yes	No
…			

.Data.EventFrameSnapshot

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
EventFrameTemplateAttributeID	GUID	Yes	No
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The Snapshot and EventFrameSnapshot tables use the **AFAttribute.GetValue** method.

.Data.Archive

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
…			

.Data.EventFrameArchive

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The Archive and EventFrameArchive tables use the **AFData.RecordedValues** method if the method is supported by the Data Reference. For custom Data References that do not support **AFData.RecordedValues** the Provider falls back using the **AFAttribute.GetValues** method.

.Data.InterpolateDiscrete (ElementAttributeID Guid,Time DateTime)

Name	Data type	Is nullable?	Primary Key
…			

.Data.EventFrameInterpolateDiscrete (EventFrameAttributeID Guid,Time DateTime)

Name	Data type	Is nullable?	Primary Key
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

This InterpolateDescrete and EventFrameInterpolateDescrete tables use the **AFAttribute.GetValue** method.

.Data.ft_InterpolateDiscrete

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
…			

.Data.ft_EventFrameInterpolateDiscrete

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

This table uses the **AFAttribute.GetValue** method.

.Data.InterpolateRange (ElementAttributeID Guid, StartTime DateTime, EndTime DateTime, TimeStep Time)

Name	Data type	Is nullable?	Primary Key
…			

.Data.EventFrameInterpolateRange (EventFrameAttributeID Guid, StartTime DateTime, EndTime DateTime, TimeStep Time)

Name	Data type	Is nullable?	Primary Key
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The InterpolateRange and EventFrameInterpolateRange Table-Valued Functions use the **AFAttribute.GetValues** method.

.Data.ft_InterpolateRange

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
TimeStep	Time	No	No
…			

.Data.ft_EventFrameInterpolateRange

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
TimeStep	Time	No	No
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The InterpolateRange and EventFrameInterpolateRange function tables use the **AFAttribute.GetValues** method.

.Data.Plot (ElementAttributeID Guid, StartTime DateTime, EndTime DateTime, IntervalCount UInt32)

Name	Data type	Is nullable?	Primary Key
…			

.Data.EventFramePlot (EventFrameAttributeID Guid, StartTime DateTime, EndTime DateTime, IntervalCount UInt32)

Name	Data type	Is nullable?	Primary Key
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The Plot and EventFramePlot Table-Valued Functions use the **AFAttribute.GetValues** method.

.Data.ft_Plot

Name	Data type	Is nullable?	Primary Key
ElementAttributeID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
IntervalCount	UInt32	No	No
…			

.Data.ft_EventFramePlot

Name	Data type	Is nullable?	Primary Key
EventFrameAttributeID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
IntervalCount	UInt32	No	No
…			

QUERY HINT SUPPORT

```
OPTION (IGNORE ERRORS), OPTION (EMBED ERRORS)
```

IMPLEMENTATION DETAILS

The Plot and EventFramePlot function tables use the **AFAttribute.GetValues** method.

DataT Schema

.DataT.<Transpose Snapshot>,
.DataT.v<Transpose Snapshot>

Name	Data type	Is nullable?	Primary Key
ElementID or ElementVersionID	GUID	No	Yes
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
...			

.DataT. <Transpose Snapshot> (ElementID Guid),
.DataT. <vTranspose Snapshot> (ElementVersionID Guid)

Name	Data type	Is nullable?	Primary Key
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
…			

QUERY HINT SUPPORT

`OPTION (IGNORE ERRORS)`,
`OPTION (EMBED ERRORS)` only if the **Values as VARIANT** option has been selected when creating the TVF.

IMPLEMENTATION DETAILS

These tables use the **AFAttribute.GetValue** method.

.DataT.<Transpose Archive>,
.DataT.<vTranspose Archive>

Name	Data type	Is nullable?	Primary Key
ElementID or ElementVersionID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
Time	DateTime	No	Yes

Name	Data type	Is nullable?	Primary Key
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
...			

.DataT. <Transpose Archive> (ElementID Guid, StartTime DateTime, EndTime DateTime),
.DataT. <vTranspose Archive> (ElementVersionID Guid, StartTime DateTime, EndTime DateTime)

Name	Data type	Is nullable?	Primary Key
Time	DateTime	No	Yes
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
…			

.DataT. <Transpose Archive> (EventFrameAttributeID Guid, StartTime DateTime, EndTime DateTime, IntervalCount UInt32)

Name	Data type	Is nullable?	Primary Key
…			

QUERY HINT SUPPORT

`OPTION (IGNORE ERRORS)`,
`OPTION (EMBED ERRORS)` only if the **Values as VARIANT** option has been selected when creating the TVF.

IMPLEMENTATION DETAILS

These tables use the **AFData.RecordedValues** method.

.DataT.<Transpose InterpolateDiscrete>,
.DataT.v<Transpose InterpolateDiscrete>

Name	Data type	Is nullable?	Primary Key
ElementID or ElementVersionID	GUID	No	Yes
Time	DateTime	No	Yes

Name	Data type	Is nullable?	Primary Key
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
...			

.DataT. <Transpose InterpolateDiscrete > (ElementID Guid, Time DateTime), .DataT. <vTranspose InterpolateDiscrete > (ElementVersionID Guid, Time DateTime)

Name	Data type	Is nullable?	Primary Key
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
…			

QUERY HINT SUPPORT

`OPTION (IGNORE ERRORS),`
`OPTION (EMBED ERRORS)` only if the **Values as VARIANT** option has been selected when creating the TVF.

IMPLEMENTATION DETAILS

These tables use the **AFAttribute.GetValue** method.

.DataT.<Transpose InterpolateRange>, .DataT.v<Transpose InterpolateRange>

Name	Data type	Is nullable?	Primary Key
ElementID or ElementVersionID	GUID	No	Yes
StartTime	DateTime	No	No
EndTime	DateTime	No	No
TimeStep	Time	No	No
Time	DateTime	No	Yes
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
...			

.DataT.<Transpose InterpolateRange> (ElementID Guid, StartTime DateTime, EndTime DateTime, TimeStep Time),
.DataT.<Transpose InterpolateRange> (ElementID Guid, StartTime DateTime, EndTime DateTime, TimeStep Time)

Name	Data type	Is nullable?	Primary Key
<Attribute> or <Path>\<Attribute>	Variant or Attribute specific	Yes	No
…			

QUERY HINT SUPPORT

`OPTION (IGNORE ERRORS)`,
`OPTION (EMBED ERRORS)` only if the **Values as VARIANT** option has been selected when creating the TVF.

IMPLEMENTATION DETAILS

These tables use the **AFAttribute.GetValues** method.

System Catalog Schemas

System.AF.PlugIn

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
AssemblyFileName	String	No	No
AssemblyTime	DateTime	No	No
CompatibilityVersion	Int32	No	No
Version	String	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

System.UnitOfMeasure.BaseUOMClass

Name	Data type	Is nullable?	Primary Key
UOMClassID	GUID	No	Yes
BaseUOMClassID	GUID	No	Yes
BasePower	Int16	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

System.UnitOfMeasure.UOM

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Abbreviation	String	No	No
Description	String	Yes	No
CanonicalFactor	Float64	No	No
CanonicalOffset	Float64	No	No
RefUOMID	GUID	Yes	No
RefFactor	Float64	Yes	No
RefOffset	Float64	Yes	No
RefFormulaFrom	String	Yes	No
RefFormulaTo	String	Yes	No
UOMClassID	GUID	No	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

System.UnitOfMeasure.UOMClass

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Name	String	No	No
Description	String	Yes	No
CanonicalUOMID	GUID	Yes	No
Modified	DateTime	No	No
ModifiedBy	String	Yes	No

System.UnitOfMeasure.UOMDatabase

Name	Data type	Is nullable?	Primary Key
ID	GUID	No	Yes
Modified	DateTime	No	No
ModifiedBy	String	Yes	No
SecurityDescriptor	String	No	No
CheckOutTime	DateTime	Yes	No
CheckOutUserName	String	Yes	No
CheckOutMachineName	String	Yes	No

System.Connection.ConnectionProperty

Name	Data type	Is nullable?	Primary Key
Name	String	No	Yes
Value	Variant	No	No

System.Connection.ConnectionStatus

Name	Data type	Is nullable?	Primary Key
Name	String	No	Yes
Value	Variant	Yes	No

System.Connection.ProductVersion

Name	Data type	Is nullable?	Primary Key
Item	String	No	Yes
Description	String	Yes	No
Version	String	No	No
Major	Int32	No	No
Minor	Int32	No	No
Build	Int32	No	No
Revision	Int32	No	No
Architecture	String	Yes	No
OSName	String	Yes	No
OSVersion	String	Yes	No

Chapter 5

Compendium of PI SQL Statements for PI AF

This chapter gives a tour of SQL statements that target certain use cases.

Asset Schema Queries

This section provides SQL statements that demonstrate how to retrieve data for PI OLEDB Enterprise from:

- The Element hierarchy
- Element attribute recorded values
- Element extended properties

Retrieve Elements of Element Hierarchy as it Exists at a Certain Date

```
SELECT Name FROM NuGreen.Asset.vElementHierarchy
WHERE Path = N'\NuGreen\Houston\Cracking Process\Equipment\'
 /* "Path = …" condition restricts hierarchy traversal to single
    element*/
AND Effective <= N'01-Aug-2011'     /*query date*/
AND Ineffective > N'01-Aug-2011'    /*query date*/
```

For the query date specification, you can use one of the following timestamp literal formats:

- ***** (refers to the current time)
- yyyy-mm-dd hh:mm:ss.fr
- dd-mmm-yyyy hh:mm:ss.fr

In most cases, it is desirable to specify query date as shown above. Without query date restriction, query into vElementHierarchy traverses entire element hierarchy history.

Retrieve Children Elements of Current Element Hierarchy

```
SELECT Name FROM NuGreen.Asset.ElementHierarchy
WHERE Path = N'\NuGreen\'
```

ElementHierarchy table query is equivalent to vElementHierarchy query with query date restriction set to current time. You can imagine the current element hierarchy table as a predefined *view* on top of an element hierarchy table that contains all element versions.

Retrieve Subtree of Current Element Hierarchy

```
SELECT Name FROM NuGreen.Asset.ElementHierarchy
WHERE Path LIKE N'\NuGreen\Houston\Cracking Process\Equipment\%'
/*"Path LIKE ..." condition restricts hierarchy traversal to
subtree*/
```

Retrieve Subtree Part of Current Element Hierarchy

```
SELECT Name FROM NuGreen.Asset.ElementHierarchy
WHERE Path LIKE N'\NuGreen\Houston\%'
AND Level <= 2 /*Level column can be used to restrict depth of
hierarchy traversal*/
```

Retrieve Element Attributes

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name Path
FROM NuGreen.Asset.vElementHierarchy eh
INNER JOIN NuGreen.Asset.vElementVersion ev ON ev.ID =
eh.ElementVersionID
INNER JOIN NuGreen.Asset.vElementAttribute ea ON
ea.ElementVersionID = ev.ID
WHERE eh.Path LIKE N'\NuGreen\Houston\Cracking Process\Equipment\'
AND eh.Effective <= N'01-Aug-2011' /*query date*/
AND eh.Ineffective > N'01-Aug-2011' /*query date*/
AND eh.Name LIKE N'B-%'
```

PI OLEDB Enterprise may fail to prepare optimal execution plan for more complex join queries. Thus, it is better to use an **OPTION (FORCE ORDER)** clause and order tables in the **FROM** clause by the intended execution order. For example, query above is expected to first retrieve elements from the hierarchy, then it should look for element version information and finally it should execute the element attribute query.

Retrieve Element Attributes and Optional UOM

```
SELECT eh.Path + eh.Name Element, ea.Path + ea.Name
ElementAttribute, uom.Name
FROM NuGreen.Asset.ElementHierarchy eh
INNER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID =
eh.ElementID
LEFT OUTER JOIN System.UnitOfMeasure.UOM uom ON uom.ID =
ea.DefaultUOMID
WHERE eh.Path = N'\NuGreen\Houston\Cracking Process\Equipment\'
OPTION (FORCE ORDER)
```

Retrieve Elements and Specific Attribute Values When Present

```
SELECT eh.Path + eh.Name Element, ea.Path + ea.Name
ElementAttribute, i.Value
FROM NuGreen.Asset.ElementHierarchy eh
LEFT OUTER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID
= eh.ElementID AND ea.Name = N'Blades'
OUTER APPLY NuGreen.Data.InterpolateDiscrete(ea.ID, 't') i
WHERE eh.Path = N'\NuGreen\Houston\Cracking Process\Equipment\'
OPTION (FORCE ORDER)
```

Retrieve Element Attribute Recorded Values

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name Path, r.*
FROM NuGreen.Asset.vElementHierarchy eh
INNER JOIN NuGreen.Asset.vElementVersion ev ON ev.ID =
eh.ElementVersionID
INNER JOIN NuGreen.Asset.vElementAttribute ea ON
ea.ElementVersionID = ev.ID
INNER JOIN NuGreen.Data.Archive r ON r.ElementAttributeID = ea.ID
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Effective <= N'*' /*query date*/
AND eh.Ineffective > N'*' /*query date*/
AND eh.Name LIKE N'B-%'
AND ea.Name LIKE N'%Flow'
```

Retrieve Element Extended Properties

```
SELECT eh.Path + eh.Name + ep.Name Path, ep.*
FROM NuGreen.Asset.vElementHierarchy eh
INNER JOIN NuGreen.Asset.vElementVersion ev ON ev.ID =
eh.ElementVersionID
INNER JOIN NuGreen.Asset.vElementExtendedProperty ep ON
ep.ElementVersionID = ev.ID
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Effective <= N'01-Aug-2011' /*query date*/
AND eh.Ineffective > N'01-Aug-2011' /*query date*/
OPTION (FORCE ORDER)
```

Retrieve Element Attribute Recorded Values Using Simplified Asset Database E-R Model

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name Path, r.*
FROM NuGreen.Asset.ElementHierarchy eh
INNER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID =
eh.ElementID
INNER JOIN NuGreen.Data.Archive r ON r.ElementAttributeID = ea.ID
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Name LIKE N'B-%'
AND ea.Name LIKE N'%Flow'
AND r.Time BETWEEN N'01-Aug-2011' AND N'02-Aug-2011'
OPTION (FORCE ORDER)
```

Search for Elements By Attribute Values Using Simplified Asset Database E-R Model

This query searches for all `Boiler` elements where the value of the `Manufacturer` attribute is equal to `KPC, Limited`.

```
SELECT eh.Path + eh.Name Element
FROM NuGreen.Asset.ElementTemplate et
INNER JOIN NuGreen.Asset.ElementTemplateAttribute eta ON
eta.ElementTemplateID = et.ID
INNER JOIN NuGreen.Data.Snapshot s ON s.ElementTemplateAttributeID
= eta.ID
INNER JOIN NuGreen.Asset.ElementAttribute ea ON
s.ElementAttributeID = ea.ID
INNER JOIN NuGreen.Asset.ElementHierarchy eh ON ea.ElementID =
eh.ElementID
WHERE et.Name = 'Boiler'
AND eta.Name = 'Manufacturer'
AND s.ValueStr = 'KPC, Limited'
OPTION (FORCE ORDER)
```

When performing queries on the Data.Snapshot table, you need to follow some best practices for internal optimization mechanisms to take place. Specifically, attributes must have the following configuration in the Element Templates:

- The **Value Type** is set to one of the **Basic Types** or an **Enumeration Set** (that is, not an array or object).
- The **Indexed** check box is selected.
- No data reference is used (the **Data Reference** field is set to **<None>**).

Please see the PI SQL Query Compendium included in the PI SQL Commander tools for more information on these types of queries.

The same optimization mechanisms take place when using Transpose Function Tables:

```
SELECT eh.Path + eh.Name Element, tc.*
FROM [NuGreen].[DataT].[ft_TransposeSnapshot_Boiler] tc
INNER JOIN [NuGreen].[Asset].[ElementHierarchy] eh ON eh.ElementID
= tc.ElementID
WHERE tc.[Manufacturer] = 'KPC, Limited'
OPTION (FORCE ORDER, IGNORE ERRORS, EMBED ERRORS)
```

Retrieve Element Attribute Current Values Using Simplified Asset Database E-R Model

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name ElementAttribute, c.*
FROM NuGreen.Asset.ElementHierarchy eh
INNER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID =
eh.ElementID
INNER JOIN NuGreen.Data.Snapshot c ON c.ElementAttributeID = ea.ID
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Name LIKE N'B-%'
AND ea.Name LIKE N'%Flow'
OPTION (FORCE ORDER)
```

Retrieve Interpolated Values Using Simplified Asset Database E-R Model

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name ElementAttribute, i.*
FROM NuGreen.Asset.ElementHierarchy eh
INNER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID =
eh.ElementID
CROSS APPLY NuGreen.Data.InterpolateRange
(
      ea.ID                    /*ElementAttributeID*/,
      DATE('*') - TIME('1h')   /*StartTime*/,
      DATE('*')                /*EndTime*/,
      TIME('10m')              /*TimeStep*/
) i
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Name LIKE N'B-%'
AND ea.Name LIKE N'%Flow'
OPTION (FORCE ORDER)
```

Above query retrieves interpolated values through InterpolateRange Table-Valued Function (TVF), which performs calculation over the specified time interval with the specified time step.

Table-Valued Functions are typically incorporated into the query using the **CROSS APPLY** operator which invokes the Table-Valued Function for each row returned by join of tables which precede the TVF in the **FROM** clause.

The following query is equivalent to the previous one. Each built-in TVF (like **InterpolateRange**) has a function table counterpart. Function tables offer alternative syntax for calling TVFs. They are intended to be used from third party clients that do not allow executing queries containing TVFs directly (for example, SQL Server). Function arguments are represented by columns which must be restricted in the **WHERE** clause.

Retrieve Interpolated Values Using Simplified Asset Database E-R Model (2)

```
SELECT eh.Path + eh.Name + ea.Path + ea.Name ElementAttribute, i.*
FROM NuGreen.Asset.ElementHierarchy eh
INNER JOIN NuGreen.Asset.ElementAttribute ea ON ea.ElementID =
eh.ElementID
INNER JOIN NuGreen.Data.ft_InterpolateRange i ON
i.ElementAttributeID = ea.ID
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
AND eh.Name LIKE N'B-%'
AND ea.Name LIKE N'%Flow'
AND i.StartTime = DATE('*') - TIME('1h')
AND i.EndTime = DATE('*')
AND i.TimeStep = TIME('10m')
OPTION (FORCE ORDER)
```

Retrieve Current Values Using TransposeSnapshot Table-Valued Function

```
SELECT eh.Name Element, tc.*
FROM [NuGreen].[Asset].[ElementHierarchy] eh
CROSS APPLY [NuGreen].[DataT].[TransposeSnapshot_Cooling
Fan](eh.ElementID) tc
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
OPTION (FORCE ORDER)
```

Retrieve Interpolated Values Using TransposeInterpolatedDiscrete Table-Valued Function

```
SELECT eh.Name Element, ti.*
FROM NuGreen.[Asset].[ElementHierarchy] eh
CROSS APPLY NuGreen.[DataT].[TransposeInterpolateDiscrete_Cooling
Fan]
(
eh.ElementID,
DATE('*') - TIME('1h')
) ti
WHERE eh.Path LIKE N'\NuGreen\Little Rock\Extruding Process\%'
OPTION (FORCE ORDER)
```

Retrieves interpolated values of attributes inherited from Cooling Fan element template. The query is restricted to elements contained in the `\NuGreen\Little Rock\Extruding Process\` subtree.

Retrieve Interpolated Values Using TransposeInterpolatedDiscrete Function Table

```
SELECT eh.Path + eh.Name Element, tc.*
FROM [NuGreen].[Asset].[ElementHierarchy] eh
INNER JOIN
[NuGreen].[DataT].[ft_TransposeInterpolateDiscrete_Boiler] tc
      ON tc.ElementID = eh.ElementID
WHERE eh.Path LIKE  N'\%' AND tc.Time = '*'
OPTION (FORCE ORDER)
```

Retrieves interpolated values of all attributes of elements that are inherited from the Boiler template.

Use LIST Function if an Element is in multiple Categories

```
SELECT Path+Element "Plant", LIST(Category,', ') "Categories"
FROM (
   SELECT eh.name "Element", c.Name "Category", Path
   FROM [NuGreen].[Asset].[ElementCategory] ec
      , [NuGreen].[Asset].[ElementHierarchy] eh
      , [NuGreen].[Asset].[Category] c
   WHERE ec.ElementID = eh.ElementID
    AND ec.CategoryID = c.ID
    AND eh.Name LIKE '%'
    AND eh.Level = 2
   ORDER BY Element, Category
) AllRelations
GROUP BY Path+Element
```

Elements in hierarchy level 2 are listed by category. An element that is member of multiple categories has the category names listed in the Categories column.

EventFrame Schema Queries

Here are example statements that demonstrate how to retrieve

- Event Frames
- Event Frame attribute values

Search Event Frame by Category

```
SELECT ef.name, ef.Starttime, ef.endTime
FROM eventframe.eventframe ef
INNER JOIN eventframe.eventframecategory efc
ON ef.id=efc.EventframeID
INNER JOIN asset.category ac
ON efc.CategoryID=ac.ID
WHERE ac.name='OSIBatch'and ef.starttime
BETWEEN '5-jun-2012' AND '8-jun-2012'
ORDER BY ef.ID;
```

Returns all event frames of a specific category.

Search Child Event Frames

```
SELECT c.Name as ChildEventFrameName,c.Description,
c.StartTime,c.EndTime
FROM (Eventframe.EventFrame c
INNER JOIN EventFrame.EventFrameReference efr
ON efr.ChildEventFrameID=c.ID)
INNER JOIN Eventframe.EventFrame p
ON efr.ChildEventFrameID=p.ID
WHERE p.name='2700'
AND c.StartTime BETWEEN '25-may-2012' AND '30-may-2012';
```

Returns all child event frames having a parent with a specific name.

Search Event Frames Referencing a Specific Element

```
SELECT ef.Name as EF_Name, ef.starttime, ef.endtime, e.name
FROM EventFrame.eventframe ef
INNER JOIN Eventframe.EventFrameReferencedElement efr
ON efr.EventframeID=ef.ID
INNER JOIN Asset.Element e
ON e.ID=efr.ElementID
WHERE e.name='Reactor R12' AND ef.endtime < '6-jun-2012' AND
ef.HasChildren=FALSE
ORDER BY 1;
```

Returns all event frames that reference a specific element, have no children and end before a specific date.

Search Event Frames by Template

```
SELECT ef.name, ef.Starttime, ef.endTime
FROM eventframe.eventframe ef
INNER JOIN eventframe.eventframeTemplate eft
ON ef.EventFrameTemplateID=eft.ID
WHERE eft.name='OSIProcedure'
AND ef.StartTime BETWEEN '04-jun-2012' AND '06-jun-2012'
ORDER BY 1;
```

Returns all event frames derived from a specific template and starting in a certain time range.

Search Event Frames by Element Attribute

```
SELECT ef.Name, ef.StartTime, ef.EndTime
FROM EventFrame.EventFrame ef
INNER JOIN Asset.ElementAttribute ea
ON ea.ElementID = ef.PrimaryReferencedElementID
WHERE ef.PrimaryReferencedElementID IS NOT NULL
AND ea.Name = 'Productivity'
AND ef.starttime BETWEEN '03-jun-2012' AND '06-jun-2012'
ORDER BY 1;
```

Find all event frames referencing any elements with a specific attribute and starting in a certain time range.

Retrieve Event Frames and Attribute Values

```
SELECT ef.Name AS EventFrame, ef.StartTime, ef.EndTime,
       efa.Name AS EventFrameAttribute,
       efa.Path AS EventFrameAttributePath,
       efa.Level AS EventFrameAttributeLevel,
       efa.IsConfigurationItem,
       cast(efs.Value as String) as AttrValue
FROM EventFrame.EventFrame ef
INNER JOIN EventFrame.EventFrameAttribute efa
ON efa.EventFrameID = ef.ID
INNER JOIN Data.EventFrameSnapshot efs
ON efs.EventFrameAttributeID = efa.ID
WHERE ef.StartTime > N'01-Jul-2012'
AND ef.EndTime < N'02-Jul-2012'
AND efa.EventFrameTemplateAttributeID IS NULL
ORDER BY ef.Name, ef.StartTime, ef.EndTime, efa.Path, efa.Name;
```

Find Event Frames in a certain time range and show individual attributes and their snapshot values.

System Database Queries

Another Use of the LIST Function

```
SELECT LIST((Name + ' = ' + cast(Value as String)), ', ')
"Connection Status"
FROM System..ConnectionStatus;
```

Reports the connection status as comma separated list.

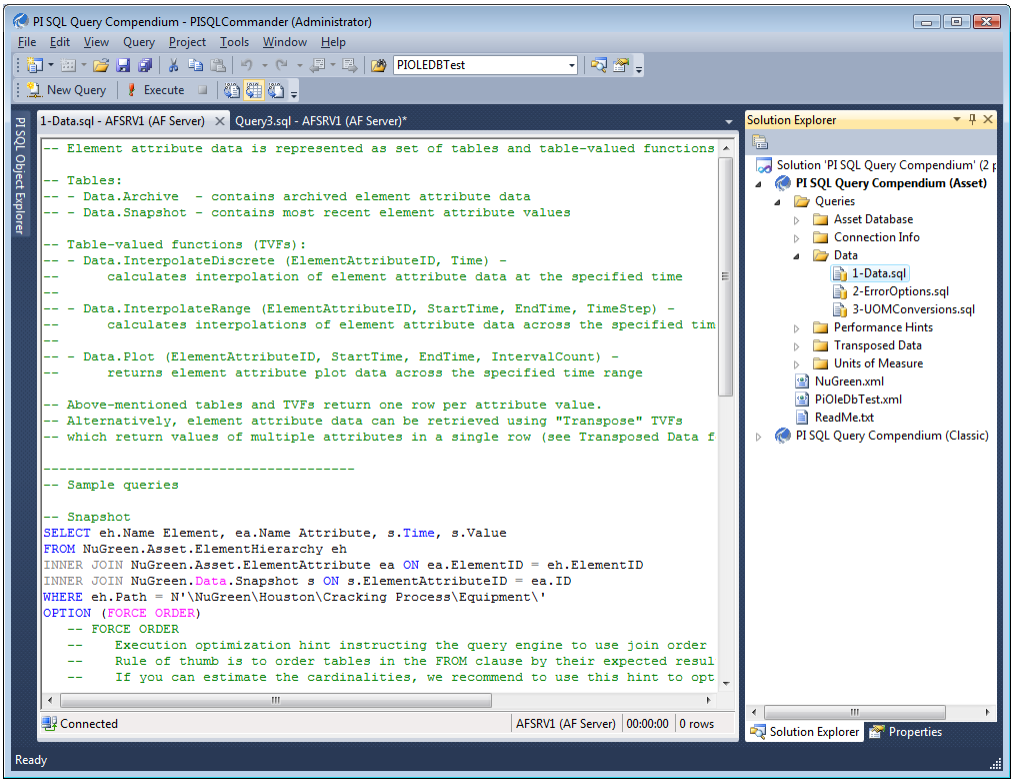
More Queries - PI SQL Query Compendium Solution

PI SQL Commander includes an extensive collection of sample queries in a ready-to-use solution. It consists of two sample PI Asset Framework (PI AF) databases NuGreen and PiOleDbTest that you can import into your system with **PI System Explorer**. The files are located in the following directory:

`[PIHOME]\OLEDB\Tools\PI SQL Commander\MyProjects\PI SQL Query Compendium`

Note: `[PIHOME]` is the directory to which you install PI client applications and interfaces. For example, `C:\Program Files\PIPC`.

Open the solution **PI SQL Query Compendium** in the **PI SQL Commander Solution Explorer** and follow the guidelines given in the project file `ReadMe.txt`. There are also comments in the query project files found in the **Queries** folder:



Appendix A

Recommended Configuration for Linked Server in Microsoft SQL Server

A very common use case for PI OLEDB Enterprise is to configure a Linked Server in Microsoft SQL Server, to expose the PI OLEDB Enterprise catalogs and tables as if they were native to SQL Server. This configuration allows applications, reports, and other components that leverage SQL Server tables, to access PI System data seamlessly.

To create a Linked Server, the PI OLEDB Enterprise provider must be installed on the SQL Server machine. The creation of a Linked Server is generally done with SQL Server Management Studio (SSMS) and requires the configuration of a series of parameters in three major categories: General, Security and Server options. The tables in this section summarize the recommended settings for these parameters.

Note: If you run SSMS under Windows 2008, Vista or later, you may need to launch SSMS as adminstrator.

General Settings

Parameter	Description
Linked server	A name for the linked server being created. For example, LINKEDPI.
Provider	PI OLEDB Enterprise (selected from the drop-down list)
Product name	PIOLEDBENT
Data source	Name of the PI Asset Framework (AF) Server
Provider string	Additional initialization properties for this Linked Server (for example, Windows Integrated Security, Time Zone, Logging). See *General OLE DB Initialization Properties* (page 13) and *PI OLEDB Enterprise-Specific Initialization Properties* (page 14) for details. **Note:** If you are using PI Server with Windows Integrated Security (WIS), OSIsoft recommends that you set up the Linked Server to ensure that users who access this Linked Server will connect with their Windows credentials. Specify **Integrated Security = SSPI;** in this field to enable this setting.
Location	N/A
Catalog	Use default setting of **SYSTEM**

Security Settings

OSIsoft recommends that you use Windows Integrated Security (WIS) and therefore select the **Be made using the login's current security context** option.

It is also possible to connect using a specific PI User or create associations between Windows Users and PI Users, but this is outside the scope of this document.

Server Options

Use the Server Options to determine how the linked server handles connections, passes queries and triggers timeouts. OSIsoft recommends that you use these options:

Parameter	Description
Collation Compatible	It is strongly recommended that this parameter be set. When set, Microsoft SQL Server assumes that all characters in the Linked Server are compatible with the local SQL Server, and is allowed to send comparisons on text columns to the OLE DB provider directly. If the option is not set, SQL Server will evaluate text comparisons itself, which has a significant performance impact.
Connection Timeout and Query Timeout	Use these settings optionally. The Connection Timeout option represents the connection timeout, while the Query Timeout represents the execution timeout. Both options are in seconds.

OLE DB Provider Options

A fourth category of parameters is available outside of the environment where the Linked Server is being created; that is because these parameters are shared by all Linked Servers configured with the same OLE DB provider (for example, PI OLEDB Enterprise). OSIsoft recommends that you use these OLE DB provider options:

Option name	Recommended Setting	Description
Dynamic parameters	**Yes**	Allows SQL placeholders ? in parameterized queries.
Nested queries	**Yes**	Allows nested SELECT statements in the FROM clause, that is, sub-queries.
Level Zero only	No	Only base level OLEDB interfaces are invoked against the provider. PI OLEDB Enterprise supports more than base level interfaces, so do not enable this option.
Allow InProcess	**Yes**	If set, Microsoft SQL Server allows the provider to be instantiated as an in-process server. For performance reasons, OSIsoft strongly recommends that this option be enabled so that PI OLEDB Enterprise is run as an in-process server. Should you decide to not set this option, additional security settings must be configured in order to initialize the provider out-of-process;for details contact *OSIsoft Technical Support* (page 99).
Non transacted updates	No	If non-zero, SQL Server allows updates, even if the ITransactionLocal interface is not available. PI OLEDB Enterprise supports the ITransactionLocal interface, so OSIsoft recommends that you do not enable this option.
Index as access path	False	If non-zero, SQL Server attempts to use indexes of the provider to fetch data, which PI OLEDB Enterprise does not support.
Disallow adhoc access	False	If set, SQL Server does not allow executing pass-through queries against the OLEDB provider. While this option can be checked, it is sometimes appropriate to execute pass-through queries.
Support LIKE operator	**Yes**	Indicates that the provider supports queries using the LIKE keyword.

If you are running SQL Management Studio Express, the Provider Options button is not available. You must run the scripts manually from SQL Server Management Studio to enter these settings. Use this script to enter a new query:

```
EXEC sys.sp_MSset_oledb_prop 'PIOLEDB', 'AllowInProcess', 1
EXEC sys.sp_MSset_oledb_prop 'PIOLEDB', 'DynamicParameters', 1
EXEC sys.sp_MSset_oledb_prop 'PIOLEDB', 'NestedQueries', 1
EXEC sys.sp_MSset_oledb_prop N'PIOLEDB', N'SqlServerLIKE', 1
```

Appendix B

Technical Support and Resources

For technical assistance, contact OSIsoft Technical Support at +1 510-297-5828 or techsupport@osisoft.com. The OSIsoft Technical Support website offers additional contact options for customers outside of the United States.

When you contact OSIsoft Technical Support, be prepared to provide this information:

- Product name, version, and build numbers
- Computer platform (CPU type, operating system, and version number)
- Time that the difficulty started
- Log files at that time
- Details of any environment changes prior to the start of the issue
- Summary of the issue, including any relevant log files during the time the issue occurred

The OSIsoft Virtual Campus *(vCampus)* website has subscription-based resources to help you with the programming and integration of OSIsoft products.

Index