
Plconnect Documentation

Release 0.9.1

Hugo Lapré; Stijn de Jong

Aug 11, 2021

CONTENTS

1	Contents	3
1.1	Installation	3
1.2	Tutorials	3
1.3	API Reference	13
1.4	Contributing	30
1.5	Credits	32
1.6	History	32
1.7	Copyrights	33
1.8	Things left to do	34
2	Indices and tables	35
	Python Module Index	37
	Index	39

A python connector to the OSISoft PI and PI-AF databases

This connector allows access to the OSISoft PI System through their proprietary SDK. It provides a number of classes, mostly mirroring the AF SDK structure, but at the same time implementing the cool stuff we use Python for.

The documentation for *PIconnect* is roughly structured according to Daniele Procida's proposed structure:

- The basic introductions to using *PIconnect* are covered in the *Tutorials*.
- The technical reference to all modules within *PIconnect* are covered in *API Reference*. These also include several links to the original SDK documentation for important implementation details.
- Background explanations and how-to guides still need to be written.

CONTENTS

1.1 Installation

1.1.1 Stable release

To install *PIconnect*, run this command in your terminal:

```
$ pip install PIconnect
```

This is the preferred method to install *PIconnect*, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

1.1.2 From sources

The sources for *PIconnect* can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/Hugovdberg/PIconnect
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/Hugovdberg/PIconnect/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

1.2 Tutorials

To start using the *PIconnect* package a number of short tutorials are available. It is recommended to start with the *Basics* tutorials, which cover the connections to the different PI Systems.

1.2.1 Basics

Connecting to a PI Server

To connect to a PI Server you need to use the *PIServer* class. The following code connects to the default database and prints its name.:

```
import PIconnect as PI

with PI.PIServer() as server:
    print(server.server_name)
```

The next step is to get a list of *PIPoint* objects from the server, and print the first ten of those:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')
    for point in points[:10]:
        print(point)
```

To get the data as stored in the archive for a given point, we can use the *PIPoint.recorded_values* method. The following snippet gets the data recorded in the last 48 hours:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')
    data = points[0].recorded_values('*-48h', '*')
    print(data)
```

The resulting *data* object is essentially a decorated version of a *pandas.Series*, and can be used for any further processing.

Connecting to other servers

By default *PIServer* connects to the default server, as reported by the SDK. To find out which other servers are available you can use the *servers* dictionary. The keys to the dictionary are the server names. You can get the list of server names like this:

```
import PIconnect as PI
print(list(PI.PIServer.servers.keys()))
```

To explicitly connect to any of the available servers, you pass the name of the server to the *PIServer* constructor.

```
import PIconnect as PI

with PI.PIServer(server='ServerName') as server:
    print(server.server_name)
```

Note: When the server name is not found in the dictionary, a warning is raised and a connection to the default server is returned instead.

Connecting to a PI AF Database

To retrieve data from the PI Asset Framework, the *PIAFDatabase* object should be used. The following code connects to the default database on the default server, and prints its server name:

```
import PIconnect as PI

with PI.PIAFDatabase() as database:
    print(database.server_name)
```

The Asset Framework represents a hierarchy of elements, with attributes on the elements. The database has a dictionary of *children*, which you can loop over as follows:

```
import PIconnect as PI

with PI.PIAFDatabase() as database:
    for root in database.children.values():
        print("Root element: {r}".format(r=root))
```

The keys of the dictionary are the names of the elements inside. The following snippet first gets the first key in the dictionary, and then uses that to get the corresponding *PIAFElement* from the dictionary. Then from this element its *PIAFAttribute* are extracted:

```
import PIconnect as PI

with PI.PIAFDatabase() as database:
    key = next(iter(database.children))
    element = database.children[key]
    for attr in element.attributes:
        print(element.attributes[attr])
```

To get the data for the last 48 hours from a given attribute you need the *PIAFAttribute.recorded_values* method:

```
import PIconnect as PI

with PI.PIAFDatabase() as database:
    key = next(iter(database.children))
    element = database.children[key]
    attribute = next(iter(element.attributes.values()))
    data = attribute.recorded_values("*-48h", "")
    print(data)
```

Note: Attributes on root elements within the database might not have meaningful summaries. To get a better result take a look at *Finding descendants in the hierarchy* below.

Finding descendants in the hierarchy

Whilst it is possible to traverse the hierarchy one at a time, by using the *PIAFElement.children* dictionaries, it is also possible to get a further descendant using the *PIAFElement.descendant* method. Assuming the database has a root element called *Plant1* with a child element *Outlet*, the latter element could be accessed directly as follows:

```
import PIconnect as PI

with PI.PIAFDatabase() as database:
    element = database.descendant(r"Plant1\Outlet")
```

Note: Elements in the hierarchy are separated by a single backslash \, use either raw strings (using the *r* prefix, as in the example above) or escape each backslash as `\\`.

Connecting to other servers or databases

When no arguments are passed to the *PIAFDatabase* constructor, a connection is returned to the default database on the default server. It is possible to connect to other servers or databases, by passing the name of the server and database as arguments to the *PIAFDatabase* constructor.

```
import PIconnect as PI

with PI.PIAFDatabase(server="ServerName", database="DatabaseName") as database:
    print(database.server_name)
```

Note: It is also possible to specify only server or database. When only server is specified, a connection to the default database on that server is returned. Similarly, when only a database is specified, the connection is made to that database on the default server.

A list of the available servers can be found in the *PIAFDatabase.servers* attribute. This is a dictionary, where the keys are the server names. To get the list of server names you can use the following code.

```
import PIconnect as PI
print(list(PI.PIAFDatabase.servers.keys()))
```

A list of the databases on a given server can be retrieved from the same *PIAFDatabase.servers* attribute. Each item in the dictionary of servers is a dictionary with two items, *server* and *databases*. The first contains the raw server object from the SDK, while the *databases* item is a dictionary of {name: object} pairs. So to get the databases for a given server you can use the following code:

```
import PIconnect as PI
print(list(PI.PIAFDatabase.servers["ServerName"]["databases"].keys()))
```

1.2.2 Data extraction

Extracting recorded values

The data in the PI archives are typically compressed¹. To get the exact values as they are stored in the archive, the *recorded_values* method should be used. It is also possible to extract a single historic value using *recorded_value*. This is available on both *PIPoint*, and *PIAFAAttribute* objects.

For simplicity this tutorial only uses *PIPoint* objects, see the tutorial on *PIAF* to find how to access *PIAFAAttribute* objects.

Single vs Multiple values

We start of by extracting a the value from the first *PIPoint* that is returned by the server as it was 5 minutes ago.

```
import PIconnect as PI

with PI.PIServer() as server:
    point = server.search('*')[0]
    data = point.recorded_value('-5m')
    print(data)
```

You will see *PISeries* is printed containing a single row, with the *PIPoint* name as the Series name, the point value as the value, and the corresponding timestamp as the index.

By default the PI Server automatically detects which value to select and returns it with the requested timestamp as the index. To control which value is returned, we pass the *retrieval_mode* argument to *recorded_value*:

```
import PIconnect as PI
from PIconnect.PIConsts import RetrievalMode

with PI.PIServer() as server:
    point = server.search('*')[0]
    data = point.recorded_value('-5m', retrieval_mode=RetrievalMode.AT_OR_BEFORE)
    print(data)
```

Since it is unlikely there is a value at exactly 5 minutes ago, the PI Server now returns the latest value that's before the requested time. Also the index is now no longer at the requested time, but at the time the value was recorded.

Now to get a time series of the values from the *PIPoint* we use the *recorded_values* method, and pass a *start_time* and *end_time*:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.recorded_values('*-48h', '*')
    print(data)
```

¹ More information on the compression algorithm can be found in this youtube video: [OSIsoft: Exception and Compression Full Details](#).

Boundary types

By default only the data between the *start_time* and *end_time* is returned. It is also possible to instead return the data from the last value before *start_time* up to and including the first value after *end_time*, by setting the *boundary_type* to *outside*:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.recorded_values('*-48h', '*', boundary_type='outside')
    print(data)
```

Warning: The *boundary_type* argument currently takes a string as the key to the internal `__boundary_types` dictionary. This will change in a future version to an enumeration in *PIConsts*.

Finally, it is also possible to interpolate the values surrounding both boundaries such that a value is returned exactly at the requested timestamp:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.recorded_values('*-48h', '*', boundary_type='interpolate')
    print(data)
```

Filtering values

Sometimes it is desirable to exclude certain values from the returned data. This is possible using the *filter_expression* argument of the *PIPoint.recorded_values* method. Only values matching the expression are returned.

The simplest test is to only return values below a given value. To test if the values of a tag called *Plant1_Flow_out* are below the value 100, you need the *filter_expression*="*Plant1_Flow_out* < 100". *PIPoint.recorded_values* provides a shortcut to include the tag name, by replacing *%tag%* with the current tag name:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    print(points.recorded_values(
        '*-48h',
        '*',
        filter_expression="%tag%' < 115"
    ))
```

Multiple tests can be combined with the keywords *and* and *or*:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
```

(continues on next page)

(continued from previous page)

```
print(points.recorded_values(
    '*-48h',
    '*',
    filter_expression="%tag%" > 100 and '%tag%' < 115"
))
```

Extracting interpolated values

Since the data in the PI archive is compressed by default, the time interval between consecutive values is typically irregular. To get values at regular intervals the *interpolated_values* method is used. This is available on both *PIPoint*, and *PIAFAttribute* objects.

For simplicity this tutorial only uses *PIPoint* objects, see the tutorial on *PIAF* to find how to access *PIAFAttribute* objects.

Basic usage

The basic example takes the first *PIPoint* that is returned by the server and gets the data for the last hour at 5 minute intervals, by specifying the *start_time*, *end_time*, and *interval* arguments to *PIPoint.interpolated_values*:

```
import Plconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.interpolated_values('*-1h', '*', '5m')
    print(data)
```

Filtering values

To filter the interpolated values the same *filter_expression* syntax as for *Filtering values* can be used.

Extracting summaries

Extracting a summary for a single time interval

The PI system allows multiple types of summaries to be calculated from data. To get the maximum value of a *PIPoint* in the last 14 days, you would use the *PIPoint.summary* method. This takes at least three arguments, *start_time*, *end_time* and *summary_types*, as shown in the following code:

```
import Plconnect as PI
from Plconnect.PIConsts import SummaryType

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summary('*-14d', '*', SummaryType.MAXIMUM)
    print(data)
```

The returned *data* is a `pandas.DataFrame` with the timestamps as index and a column for each requested summary. The timestamp in this case is the datetime at which the maximum occurred. This is more obvious when requesting multiple summaries over the same time span:

```
import PIconnect as PI
from PIconnect.PIConsts import SummaryType

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summary('*-14d', '*', SummaryType.MAXIMUM | SummaryType.MINIMUM)
    print(data)
```

Similarly, a `PIAFAttribute` also has a `PIAFAttribute.summary` method, that works in the same way:

```
import PIconnect as PI
from PIconnect.PIConsts import SummaryType

with PI.PIAFDatabase() as database:
    key = next(iter(database.children))
    element = database.children[key]
    attribute = next(iter(element.attributes.values()))
    data = attribute.summary('*-14d', '*', SummaryType.MAXIMUM | SummaryType.MINIMUM)
    print(data)
```

Note: Attributes on root elements within the database might not have meaningful summaries. To get a better result take a look at *Finding descendants in the hierarchy*.

Summary timestamps

Since the minimum and maximum of a point never occur at the same timestamp, the `DataFrame` in the previous example will typically occur two row. It is possible to reduce that to a single timestamp, when the time at which the summary value occurs is of no value.

There are two possibilities for the timestamp, the beginning of the requested time interval, or the end of the interval. Which to return is specified using the *time_type* argument. To always return the beginning of the interval, you should use the `TimestampCalculation.EARLIEST_TIME` constant from `PIConsts`:

```
import PIconnect as PI
from PIconnect.PIConsts import SummaryType, TimestampCalculation

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summary(
        '*-14d',
        '*',
        SummaryType.MAXIMUM | SummaryType.MINIMUM,
        time_type=TimestampCalculation.EARLIEST_TIME
    )
    print(data)
```

Similarly, the `TimestampCalculation.MOST_RECENT_TIME` constant always returns the time at the end of the interval:

```
import PIconnect as PI
from PIconnect.PIConsts import SummaryType, TimestampCalculation

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summary(
        '*-14d',
        '*',
        SummaryType.MAXIMUM | SummaryType.MINIMUM,
        time_type=TimestampCalculation.MOST_RECENT_TIME
    )
    print(data)
```

Event weighting

Summaries of multiple data points, or events, in time can be calculated in several ways. By default each event is weighted according to the period of time for which it is valid. This period depends on the type of data, whether it is stepped or continuous data.

To get an unweighted summary, in which every event has equal weight, the *CalculationBasis.EVENT_WEIGHTED* constant from the *PIConsts* module should be used:

```
import PIconnect as PI
from PIconnect.PIConsts import CalculationBasis, SummaryType

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summary(
        '*-14d',
        '*',
        SummaryType.MAXIMUM | SummaryType.MINIMUM,
        calculation_basis=CalculationBasis.EVENT_WEIGHTED
    )
    print(data)
```

Extracting summaries at regular time intervals

Besides extracting a single summary over an entire period of time, it is also possible to extract summaries at fixed intervals within a period of time. This is done using the *PIPoint.summaries* or *PIAFAttribute.summaries* methods. In addition to the singular *summary()* method, this takes an *interval* as an argument. The following code extracts the maximum value for each hour within the last 14 days:

```
import PIconnect as PI
from PIconnect.PIConsts import SummaryType

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.summaries('*-14d', '*', '1h', SummaryType.MAXIMUM)
    print(data)
```

Just as the `summary()` methods, the `summaries()` methods support both changing the *Event weighting* and *Summary timestamps*.

Configuring timezones

Warning: Default timezone changed from Europe/Amsterdam to UTC in 0.8.0

By default the data is extracted in the UTC timezone. This is done since no fool proof way is available to detect the local timezone. It is however possible to configure the timezone used by *PIconnect*. This is done using the *PIConfig.DEFAULT_TIMEZONE* option. It takes any valid *pytz* timezone name, such as *Europe/Amsterdam* or *America/Sao_Paulo*.

```
import PIconnect as PI

print(PI.PIConfig.DEFAULT_TIMEZONE)
with PI.PIServer() as server:
    points = server.search('*')
    data = points[0].recorded_values('-1h', '*')

print(data.index.tz)

PI.PIConfig.DEFAULT_TIMEZONE = 'Etc/GMT-1'

with PI.PIServer() as server:
    points = server.search('*')
    data = points[0].recorded_values('-1h', '*')

print(data.index.tz)
```

The output is always a `pandas.Series` object with a timezone aware `pandas.DatetimeIndex`, so it is also possible to convert the timezone afterwards like:

```
data.index = data.index.tz_convert('Europe/Amsterdam')
```

Extracting event frames

Since the data in the PI archive is compressed by default, the time interval between consecutive values is typically irregular. To get values at regular intervals the *interpolated_values* method is used. This is available on both *PIPoint*, and *PIAFAttribute* objects.

For simplicity this tutorial only uses *PIPoint* objects, see the tutorial on *PIAF* to find how to access *PIAFAttribute* objects.

Basic usage

The basic example takes the first *PIPoint* that is returned by the server and gets the data for the last hour at 5 minute intervals, by specifying the *start_time*, *end_time*, and *interval* arguments to *PIPoint.interpolated_values*:

```
import PIconnect as PI

with PI.PIServer() as server:
    points = server.search('*')[0]
    data = points.interpolated_values('*-1h', '*', '5m')
    print(data)
```

Filtering values

To filter the interpolated values the same *filter_expression* syntax as for *Filtering values* can be used.

1.2.3 Data manipulation

Update value in PI

Writing a value back to PI using Python is a interesting feature. Having this capability we can use PIconnect for implementing collecting data process it somehow (e.g. a prediction model) and write back the results somehow it can be used by final users

After discussion with @Hugovdberg & with contribution of @ldariva we finally implemented an interface for the AFSDK UpdateValue method with 4 parameters value as AFValue time as python datetime.datetime with specified timezone replace_option as AFUpdateOption buffer_option as AFBufferOption.

```
from datetime import datetime
import pytz
import PIconnect as PI
from PIconnect.PIConsts import UpdateMode, BufferMode
tag = 'foo'
with PI.PIServer(server='foo') as server:
    point = server.search(tag)[0]
    point.update_value(1.0, datetime.now(pytz.utc), UpdateMode.NO_REPLACE, BufferMode.
    ↪ BUFFER_IF_POSSIBLE)
```

1.3 API Reference

PIconnect Connector to the OSIsoft PI and PI-AF databases.

1.3.1 PI Data Archive related modules

PIconnect.PI module

```
class PIconnect.PI.PIServer(server=None, username=None, password=None, domain=None,  
                           authentication_mode=AuthenticationMode.PI_USER_AUTHENTICATION,  
                           timeout=None)
```

Bases: `object`

PIServer is a connection to an OSIsoft PI Server

Parameters

- **server** (*str, optional*) – Name of the server to connect to, defaults to None
- **username** (*str, optional*) – can be used only with password as well
- **password** (*str, optional*) – -//-
- **todo** – domain, auth
- **timeout** (*int, optional*) – the maximum seconds an operation can take

Note: If the specified *server* is unknown a warning is thrown and the connection is redirected to the default server, as if no server was passed. The list of known servers is available in the *PIServer.servers* dictionary.

servers Dictionary of known servers, as reported by the SDK

default_server Default server, as reported by the SDK

search(*query, source=None*)

Search PIPoints on the PIServer

Parameters

- **query** (*str or [str]*) – String or list of strings with queries
- **source** (*str, optional*) – Defaults to None. Point source to limit the results

Returns A list of *PIPoint* objects as a result of the query

Return type `list`

Todo: Reject searches while not connected

property server_name

Name of the connected server

version = '0.2.2'

```
class PIconnect.PI.PIPoint(pi_point)
```

Bases: *PIconnect.PIData.PISeriesContainer*

Reference to a PI Point to get data and corresponding metadata from the server.

Parameters **pi_point** (*AF.PI.PIPoint*) – Reference to a PIPoint as returned by the SDK

property current_value

Return the current value of the attribute.

property description

Return the description of the PI Point.

Todo: Add setter to alter displayed description

filtered_summaries(*start_time, end_time, interval, filter_expression, summary_types, calculation_basis=None, filter_evaluation=None, filter_interval=None, time_type=None*)

Return one or more summary values for each interval within a time range

Parameters

- **start_time** (*str* or *datetime*) – String containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using [AF.Time.AFTimeRange](#).
- **end_time** (*str* or *datetime*) – String containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using [AF.Time.AFTimeRange](#).
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using [AF.Time.AFTimeSpan.Parse](#).
- **filter_expression** (*str*, *optional*) – Defaults to “”. Query on which data to include in the results. See [Filtering values](#) for more information on filter queries.
- **summary_types** (*int* or [PIConsts.SummaryType](#)) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or [PIConsts.CalculationBasis](#), *optional*) – Event weighting within an interval. See [Event weighting](#) and [CalculationBasis](#) for more information. Defaults to [CalculationBasis.TIME_WEIGHTED](#).
- **filter_evaluation** (*int* or [PIConsts.ExpressionSampleType](#), *optional*) – Determines whether the filter is applied to the raw events in the database, or if it is applied to an interpolated series with a regular interval. Defaults to [ExpressionSampleType.EXPRESSION_RECORDED_VALUES](#).
- **filter_interval** (*str*, *optional*) – String containing the interval at which to extract apply the filter. This is parsed using [AF.Time.AFTimeSpan.Parse](#).
- **time_type** (*int* or [PIConsts.TimestampCalculation](#), *optional*) – Timestamp to return for each of the requested summaries. See [Summary timestamps](#) and [TimestampCalculation](#) for more information. Defaults to [TimestampCalculation.AUTO](#).

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type [pandas.DataFrame](#)

interpolated_value(*time*)

Return a [PISeries](#) with an interpolated value at the given time

Parameters **time** (*str*) – String containing the date, and possibly time, for which to retrieve the value. This is parsed, using [AF.Time.AFTime](#).

Returns

A **PISeries** with a single row, with the corresponding time as the index

Return type **PISeries**

interpolated_values(*start_time*, *end_time*, *interval*, *filter_expression=""*)

Return a **PISeries** of interpolated data.

Data is returned between *start_time* and *end_time* at a fixed *interval*. All three values are parsed by `AF.Time` and the first two allow for time specification relative to “now” by use of the asterisk.

filter_expression is an optional string to filter the returned values, see OSIsoft PI documentation for more information.

The AF SDK allows for inclusion of filtered data, with filtered values marked as such. At this point PIconnect does not support this and filtered values are always left out entirely.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using `AF.Time.AFTimeSpan.Parse`.
- **filter_expression** (*str*, *optional*) – Defaults to ‘’. Query on which data to include in the results. See [Filtering values](#) for more information on filter queries.

Returns Timeseries of the values returned by the SDK

Return type **PISeries**

property last_update

Return the time at which the last value for this PI Point was recorded.

property name

property raw_attributes

Return a dictionary of the raw attributes of the PI Point.

recorded_value(*time*, *retrieval_mode=RetrievalMode.AUTO*)

Return a **PISeries** with the recorded value at or close to the given time

Parameters

- **time** (*str*) – String containing the date, and possibly time, for which to retrieve the value. This is parsed, using `AF.Time.AFTime`.
- **retrieval_mode** (int or `PIConsts.RetrievalMode`) – Flag determining which value to return if no value available at the exact requested time.

Returns

A **PISeries** with a single row, with the corresponding time as the index

Return type **PISeries**

recorded_values(*start_time*, *end_time*, *boundary_type='inside'*, *filter_expression=""*)

Return a **PISeries** of recorded data.

Data is returned between the given *start_time* and *end_time*, inclusion of the boundaries is determined by the *boundary_type* attribute. Both *start_time* and *end_time* are parsed by `AF.Time` and allow for time specification relative to “now” by use of the asterisk.

By default the *boundary_type* is set to ‘inside’, which returns from the first value after *start_time* to the last value before *end_time*. The other options are ‘outside’, which returns from the last value before *start_time* to the first value before *end_time*, and ‘interpolate’, which interpolates the first value to the given *start_time* and the last value to the given *end_time*.

filter_expression is an optional string to filter the returned values, see OSIsoft PI documentation for more information.

The AF SDK allows for inclusion of filtered data, with filtered values marked as such. At this point PIconnect does not support this and filtered values are always left out entirely.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **boundary_type** (*str*, optional) – Defaults to ‘inside’. Key from the `__boundary_types` dictionary to describe how to handle the boundaries of the time range.
- **filter_expression** (*str*, optional) – Defaults to ‘’. Query on which data to include in the results. See *Filtering values* for more information on filter queries.

Returns Timeseries of the values returned by the SDK

Return type `PISeries`

Raises `ValueError` – If the provided *boundary_type* is not a valid key a `ValueError` is raised.

summaries(*start_time*, *end_time*, *interval*, *summary_types*,
calculation_basis=`CalculationBasis.TIME_WEIGHTED`,
time_type=`TimestampCalculation.AUTO`)

Return one or more summary values for each interval within a time range

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using `AF.Time.AFTimespan.Parse`.
- **summary_types** (*int* or `PISconsts.SummaryType`) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or `PISconsts.CalculationBasis`, optional) – Event weighting within an interval. See *Event weighting* and *CalculationBasis* for more information. Defaults to `CalculationBasis.TIME_WEIGHTED`.
- **time_type** (*int* or `PISconsts.TimestampCalculation`, optional) – Timestamp to return for each of the requested summaries. See *Summary timestamps* and *TimestampCalculation* for more information. Defaults to `TimestampCalculation.AUTO`.

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type `pandas.DataFrame`

summary(*start_time*, *end_time*, *summary_types*, *calculation_basis*=*CalculationBasis.TIME_WEIGHTED*, *time_type*=*TimestampCalculation.AUTO*)

Return one or more summary values over a single time range.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **summary_types** (*int* or `PIConsts.SummaryType`) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or `PIConsts.CalculationBasis`, *optional*) – Event weighting within an interval. See *Event weighting* and *CalculationBasis* for more information. Defaults to `CalculationBasis.TIME_WEIGHTED`.
- **time_type** (*int* or `PIConsts.TimestampCalculation`, *optional*) – Timestamp to return for each of the requested summaries. See *Summary timestamps* and *TimestampCalculation* for more information. Defaults to `TimestampCalculation.AUTO`.

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type `pandas.DataFrame`

property `units_of_measurement`

Return the units of measument in which values for this PI Point are reported.

update_value(*value*, *time*=*None*, *update_mode*=*UpdateMode.NO_REPLACE*, *buffer_mode*=*BufferMode.BUFFER_IF_POSSIBLE*)

Update value for existing PI object.

Parameters

- **value** – value type should be in cohesion with PI object or it will raise `PIException`: [-10702] STATE Not Found
- **time** (*datetime*, *optional*) – it is not possible to set future value, it raises `PIException`: [-11046] Target Date in Future.

You can combine `update_mode` and `time` to change already stored value.

version = '0.3.0'

1.3.2 PI Asset Framework related modules

PIconnect.PIAF module

```
class PIconnect.PIAF.PIAFDatabase(server=None, database=None)
    Bases: object
    Context manager for connections to the PI Asset Framework database.
    servers Dictionary of known servers, as reported by the SDK
    default_server Default server, as reported by the SDK
    property children
        Return a dictionary of the direct child elements of the database.
    property database_name
        Return the name of the connected PI AF database.
    descendant(path)
        Return a descendant of the database from an exact path.
    event_frames(start_time=None, start_index=0, max_count=1000,
                 search_mode=EventFrameSearchMode.FORWARD_FROM_START_TIME,
                 search_full_hierarchy=False)
    property server_name
        Return the name of the connected PI AF server.
    version = '0.2.0'

class PIconnect.PIAF.PIAFElement(element)
    Bases: PIconnect.PIAFBase.PIAFBaseElement
    Container for PI AF elements in the database.
    property attributes
        Return a dictionary of the attributes of the current element.
    property categories
    property children
        Return a dictionary of the direct child elements of the current element.
    descendant(path)
        Return a descendant of the current element from an exact path.
    property description
    property name
        Return the name of the current element.
    property parent
        Return the parent element of the current element, or None if it has none.
    version = '0.1.0'
```

1.3.3 Generic utility modules

Plconnect.AFSDK module

AFSDK Loads the .NET libraries from the OSIsoft AF SDK

class `Plconnect.AFSDK.AF`

Bases: `object`

Mock class of the AF namespace

class `AFDatabase(name)`

Bases: `object`

Mock class of the AF.AFDatabase class

class `AFFelement(name)`

Bases: `object`

Mock class of the AF.AFFelement class

class `Data`

Bases: `object`

Mock class of the AF.Data namespace

class `AFBoundaryType(value)`

Bases: `enum.IntEnum`

Mock class of the AF.Data.AFBoundaryType enumeration

Inside = 0

Interpolated = 2

Outside = 1

class `PI`

Bases: `object`

Mock class of the AF.PI namespace

class `PIPoint`

Bases: `object`

Mock class of the AF.PI.PIPoint class

static `FindPIPoints(connection, query, source, attribute_names)`

Stub to mock querying PIPoints

class `PIServer(name)`

Bases: `object`

Mock class of the AF.PI.PIServer class

Connect(*retry*)

Stub for connecting to test server

Disconnect()

Stub for disconnecting from test server

class `PIServers`

Bases: `object`

Mock class of the AF.PI.PIServers class

```

    DefaultPIServer = None

class PISystem(name)
    Bases: object

    Mock class of the AF.PISystem class

    Connect()
        Stub to connect to the testing system

    Disconnect()
        Stub to disconnect from the testing system

class InternalDatabases
    Bases: object

    Mock class for the AF.PISystem.Databases property

class PISystems
    Bases: object

    Mock class of the AF.PISystems class

    DefaultPISystem = None

    Version = '0.0.0.0'

class Time
    Bases: object

    Mock class of the AF.Time namespace

class AFTimeRange(start_time, end_time)
    Bases: object

    Mock class of the AF.Time.AFTimeRange class

class AFTimeSpan
    Bases: object

    Mock class of the AF.Time.AFTimeSpan class

    static Parse(interval)
        Stub for parsing strings that should return a AFTimeSpan

```

PIconnect.PIConsts module

```

class PIconnect.PIConsts.AuthenticationMode(value)
    Bases: enum.IntEnum

    AuthenticationMode indicates how a user authenticates to a PI Server

    Detailed information is available at AF.PI.PIAuthenticationMode.

    PI_USER_AUTHENTICATION = 1
        Use the PI User authentication mode when making a connection

    WINDOWS_AUTHENTICATION = 0
        Use Windows authentication when making a connection

class PIconnect.PIConsts.BufferMode(value)
    Bases: enum.IntEnum

    Indicates buffering option in updating values, when supported by the Data Reference.

```

Detailed information is available at [AF.Data.AFBufferOption](#)

BUFFER = 2

BUFFER_IF_POSSIBLE = 1

Try updating data reference values with buffer. If fails (e.g. data reference AFDataMethods does not support Buffering, or its Buffering system is not available), then try updating directly without buffer.

DO_NOT_BUFFER = 0

Updating data reference values without buffer.

class PIconnect.PIConsts.CalculationBasis(*value*)

Bases: [enum.IntEnum](#)

CalculationBasis indicates how values should be weighted over a time range

Detailed information is available at [AF.Data.AFCalculationBasis](#).

EVENT_WEIGHTED = 1

Each event is weighted equally.

EVENT_WEIGHTED_EXCLUDE_EARLIEST = 5

Each event is weighted equally, except data at the beginning of the interval is excluded.

EVENT_WEIGHTED_EXCLUDE_MOST_RECENT = 4

Each event is weighted equally, except data at the end of the interval is excluded.

EVENT_WEIGHTED_INCLUDE_BOTH_ENDS = 6

Each event is weighted equally, data at both boundaries of the interval are explicitly included.

TIME_WEIGHTED = 0

Each event is weighted according to the time over which it applies.

TIME_WEIGHTED_CONTINUOUS = 2

Each event is time weighted, but interpolation is always done as if it is continuous data.

TIME_WEIGHTED_DISCRETE = 3

Each event is time weighted, but interpolation is always done as if it is discrete, stepped, data.

class PIconnect.PIConsts.EventFrameSearchMode(*value*)

Bases: [enum.IntEnum](#)

EventFrameSearchMode defines the interpretation and direction from the start time when searching for event frames.

Detailed information is available at https://techsupport.osisoft.com/Documentation/PI-AF-SDK/html/T_OSIsoft_AF_EventFrame_AFEventFrameSearchMode.htm, including a graphical display of event frames that are returned for a given search mode.

BACKWARD_FROM_END_TIME = 3

Backward from end time, also known as ending before

BACKWARD_FROM_START_TIME = 1

Backward from start time, also known as starting before

BACKWARD_IN_PROGRESS = 5

Backward in progress, also known as starting before and in progress

ENDING_AFTER = 4

ENDING_BEFORE = 3

FORWARD_FROM_END_TIME = 4

Forward from end time, also known as ending after

FORWARD_FROM_START_TIME = 2

Forward from start time, also known as starting after

FORWARD_IN_PROGRESS = 6

Forward in progress, also known as starting after and in progress

NONE = 0

Uninitialized

STARTING_AFTER = 2

STARTING_AFTER_IN_PROGRESS = 6

STARTING_BEFORE = 1

STARTING_BEFORE_IN_PROGRESS = 5

class `PIconnect.PIConsts.ExpressionSampleType(value)`

Bases: `enum.IntEnum`

`ExpressionSampleType` indicates how expressions are evaluated over a time range.

Detailed information is available at [AF.Data.AFSampleType](#).

EXPRESSION_RECORDED_VALUES = 0

The expression is evaluated at each archive event.

INTERVAL = 1

The expression is evaluated at a sampling interval, passed as a separate argument.

class `PIconnect.PIConsts.RetrievalMode(value)`

Bases: `enum.IntEnum`

`RetrievalMode` indicates which recorded value should be returned

Detailed information is available at [AF.Data.AFRetrievalMode](#).

AFTER = 7

The first after the requested time

AT_OR_AFTER = 2

At the exact time if available, else the first after the requested time

AT_OR_BEFORE = 1

At the exact time if available, else the first before the requested time

AUTO = 0

Automatic detection

BEFORE = 6

The first before the requested time

EXACT = 4

At the exact time if available, else return an error

class `PIconnect.PIConsts.SummaryType(value)`

Bases: `enum.IntFlag`

`SummaryType` indicates which types of summary should be calculated.

Based on `enum.IntEnum` in Python 3.5 or earlier. `SummaryType`'s can be or'ed together. Python 3.6 or higher returns a new `IntFlag`, while in previous versions it will be casted down to `int`.

```
>>> SummaryType.MINIMUM | SummaryType.MAXIMUM # Returns minimum and maximum
<SummaryType.MAXIMUM|MINIMUM: 12> # On Python 3.6+
12 # On previous versions
```

Detailed information is available at [AF.Data.AFSummaryTypes](#).

ALL = 24831

A convenience to retrieve all summary types

ALL_FOR_NON_NUMERIC = 8320

A convenience to retrieve all summary types for non-numeric data

AVERAGE = 2

Average value over the time span

COUNT = 128

The sum of the event count (when the calculation is event weighted). The sum of the event time duration (when the calculation is time weighted.)

MAXIMUM = 8

The maximum value in the time span

MINIMUM = 4

The minimum value in the time span

NONE = 0

No summary data

PERCENT_GOOD = 8192

The percentage of the data with a good value over the time range. Based on time for time weighted calculations, based on event count for event weighed calculations.

POP_STD_DEV = 64

The population standard deviation of the values over the time span

RANGE = 16

The range of the values (max-min) in the time span

STD_DEV = 32

The sample standard deviation of the values over the time span

TOTAL = 1

A total over the time span

TOTAL_WITH_UOM = 16384

The total over the time span, with the unit of measurement that's associated with the input (or no units if not defined for the input).

class `Plconnect.PIConsts.TimestampCalculation(value)`

Bases: `enum.IntEnum`

TimestampCalculation defines the timestamp returned for a given summary calculation

Detailed information is available at [AF.Data.AFTimestampCalculation](#).

AUTO = 0

The timestamp is the event time of the minimum or maximum for those summaries or the beginning of the interval otherwise.

EARLIEST_TIME = 1

The timestamp is always the beginning of the interval.

MOST_RECENT_TIME = 2

The timestamp is always the end of the interval.

class `PIconnect.PIConsts.UpdateMode(value)`

Bases: `enum.IntEnum`

Indicates how to treat duplicate values in the archive, when supported by the Data Reference.

Detailed information is available at [AF.Data.AFUpdateOption](#)

INSERT = 1

Add the value to the archive. Any existing values at the same time are not overwritten.

INSERT_NO_COMPRESSION = 5

Add the value to the archive without compression. If this value is written to the snapshot, the previous snapshot value will be written to the archive, without regard to compression settings. Note that if a subsequent snapshot value is written without the InsertNoCompression option, the value added with the InsertNoCompression option is still subject to compression.

NO_REPLACE = 2

Add the value to the archive only if no value exists at the same time. If a value already exists for that time, the passed value is ignored.

REMOVE = 6

Remove the value from the archive if a value exists at the passed time.

REPLACE = 0

Add the value to the archive. If any values exist at the same time, will overwrite one of them and set its Substituted flag.

REPLACE_ONLY = 3

Replace an existing value in the archive at the specified time. If no existing value is found, the passed value is ignored.

`PIconnect.PIConsts.get_enumerated_value(enumeration, value, default)`

PIconnect.PIData module

PIData contains a number of auxiliary classes that define common functionality among `PIPoint` and `PIAFAttribute` objects.

class `PIconnect.PIData.PISeriesContainer`

Bases: `abc.ABC`

With the ABC class we represent a general behaviour with PI Point object (General class for objects that return PISeries objects).

Todo: Move `__boundary_types` to `PIConsts` as a new enumeration

property `current_value`

Return the current value of the attribute.

filtered_summaries(*start_time, end_time, interval, filter_expression, summary_types, calculation_basis=None, filter_evaluation=None, filter_interval=None, time_type=None*)

Return one or more summary values for each interval within a time range

Parameters

- **start_time** (*str* or *datetime*) – String containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – String containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using `AF.Time.AFTimeSpan.Parse`.
- **filter_expression** (*str*, *optional*) – Defaults to ‘’. Query on which data to include in the results. See *Filtering values* for more information on filter queries.
- **summary_types** (*int* or `PIConsts.SummaryType`) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or `PIConsts.CalculationBasis`, *optional*) – Event weighting within an interval. See *Event weighting* and *CalculationBasis* for more information. Defaults to `CalculationBasis.TIME_WEIGHTED`.
- **filter_evaluation** (*int* or `PIConsts.ExpressionSampleType`, *optional*) – Determines whether the filter is applied to the raw events in the database, or if it is applied to an interpolated series with a regular interval. Defaults to `ExpressionSampleType.EXPRESSION_RECORDED_VALUES`.
- **filter_interval** (*str*, *optional*) – String containing the interval at which to extract apply the filter. This is parsed using `AF.Time.AFTimeSpan.Parse`.
- **time_type** (*int* or `PIConsts.TimestampCalculation`, *optional*) – Timestamp to return for each of the requested summaries. See *Summary timestamps* and *TimestampCalculation* for more information. Defaults to `TimestampCalculation.AUTO`.

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type `pandas.DataFrame`

interpolated_value(*time*)

Return a `PISeries` with an interpolated value at the given time

Parameters **time** (*str*) – String containing the date, and possibly time, for which to retrieve the value. This is parsed, using `AF.Time.AFTime`.

Returns

A `PISeries` with a single row, with the corresponding time as the index

Return type `PISeries`

interpolated_values(*start_time*, *end_time*, *interval*, *filter_expression*=’')

Return a `PISeries` of interpolated data.

Data is returned between *start_time* and *end_time* at a fixed *interval*. All three values are parsed by `AF.Time` and the first two allow for time specification relative to “now” by use of the asterisk.

filter_expression is an optional string to filter the returned values, see OSIsoft PI documentation for more information.

The AF SDK allows for inclusion of filtered data, with filtered values marked as such. At this point PIconnect does not support this and filtered values are always left out entirely.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using `AF.Time.AFTimeSpan.Parse`.
- **filter_expression** (*str*, *optional*) – Defaults to `''`. Query on which data to include in the results. See [Filtering values](#) for more information on filter queries.

Returns Timeseries of the values returned by the SDK

Return type PISeries

abstract name()

recorded_value(*time*, *retrieval_mode=RetrievalMode.AUTO*)

Return a PISeries with the recorded value at or close to the given time

Parameters

- **time** (*str*) – String containing the date, and possibly time, for which to retrieve the value. This is parsed, using `AF.Time.AFTime`.
- **retrieval_mode** (int or `PIConsts.RetrievalMode`) – Flag determining which value to return if no value available at the exact requested time.

Returns

A PISeries with a single row, with the corresponding time as the index

Return type PISeries

recorded_values(*start_time*, *end_time*, *boundary_type='inside'*, *filter_expression=""*)

Return a PISeries of recorded data.

Data is returned between the given *start_time* and *end_time*, inclusion of the boundaries is determined by the *boundary_type* attribute. Both *start_time* and *end_time* are parsed by `AF.Time` and allow for time specification relative to “now” by use of the asterisk.

By default the *boundary_type* is set to ‘inside’, which returns from the first value after *start_time* to the last value before *end_time*. The other options are ‘outside’, which returns from the last value before *start_time* to the first value before *end_time*, and ‘interpolate’, which interpolates the first value to the given *start_time* and the last value to the given *end_time*.

filter_expression is an optional string to filter the returned values, see OSIsoft PI documentation for more information.

The AF SDK allows for inclusion of filtered data, with filtered values marked as such. At this point PIconnect does not support this and filtered values are always left out entirely.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **boundary_type** (*str*, *optional*) – Defaults to ‘inside’. Key from the `__boundary_types` dictionary to describe how to handle the boundaries of the time range.

- **filter_expression** (*str*, *optional*) – Defaults to “”. Query on which data to include in the results. See [Filtering values](#) for more information on filter queries.

Returns Timeseries of the values returned by the SDK

Return type `PISeries`

Raises **ValueError** – If the provided *boundary_type* is not a valid key a *ValueError* is raised.

summaries(*start_time*, *end_time*, *interval*, *summary_types*,
calculation_basis=*CalculationBasis.TIME_WEIGHTED*,
time_type=*TimestampCalculation.AUTO*)

Return one or more summary values for each interval within a time range

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **interval** (*str*) – String containing the interval at which to extract data. This is parsed using `AF.Time.AFTimeSpan.Parse`.
- **summary_types** (*int* or `PIConsts.SummaryType`) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or `PIConsts.CalculationBasis`, *optional*) – Event weighting within an interval. See [Event weighting](#) and [CalculationBasis](#) for more information. Defaults to `CalculationBasis.TIME_WEIGHTED`.
- **time_type** (*int* or `PIConsts.TimestampCalculation`, *optional*) – Timestamp to return for each of the requested summaries. See [Summary timestamps](#) and [TimestampCalculation](#) for more information. Defaults to `TimestampCalculation.AUTO`.

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type `pandas.DataFrame`

summary(*start_time*, *end_time*, *summary_types*, *calculation_basis*=*CalculationBasis.TIME_WEIGHTED*,
time_type=*TimestampCalculation.AUTO*)

Return one or more summary values over a single time range.

Parameters

- **start_time** (*str* or *datetime*) – Containing the date, and possibly time, from which to retrieve the values. This is parsed, together with *end_time*, using `AF.Time.AFTimeRange`.
- **end_time** (*str* or *datetime*) – Containing the date, and possibly time, until which to retrieve values. This is parsed, together with *start_time*, using `AF.Time.AFTimeRange`.
- **summary_types** (*int* or `PIConsts.SummaryType`) – Type(s) of summaries of the data within the requested time range.
- **calculation_basis** (*int* or `PIConsts.CalculationBasis`, *optional*) – Event weighting within an interval. See [Event weighting](#) and [CalculationBasis](#) for more information. Defaults to `CalculationBasis.TIME_WEIGHTED`.

- **time_type** (*int* or `PIConsts.TimestampCalculation`, *optional*) – Timestamp to return for each of the requested summaries. See [Summary timestamps](#) and [TimestampCalculation](#) for more information. Defaults to `TimestampCalculation.AUTO`.

Returns

Dataframe with the unique timestamps as row index and the summary name as column name.

Return type `pandas.DataFrame`

abstract units_of_measurement()

update_value(*value*, *time=None*, *update_mode=UpdateMode.NO_REPLACE*, *buffer_mode=BufferMode.BUFFER_IF_POSSIBLE*)

Update value for existing PI object.

Parameters

- **value** – value type should be in cohesion with PI object or it will raise `PIException`: [-10702] STATE Not Found
- **time** (*datetime*, *optional*) – it is not possible to set future value, it raises `PIException`: [-11046] Target Date in Future.

You can combine *update_mode* and *time* to change already stored value.

version = '0.1.0'

PIconnect.config module

class `PIconnect.config.PIConfigContainer`

Bases: `object`

property `DEFAULT_TIMEZONE`

PIconnect.time module

`PIconnect.time.timestamp_to_index(timestamp)`

Convert `AFTIME` object to `datetime` in local timezone.

Parameters **timestamp** (`System.DateTime`) – Timestamp in .NET format to convert to *datetime*.

Returns `datetime` with the timezone info from `PIConfig.DEFAULT_TIMEZONE`.

Return type *datetime*

`PIconnect.time.to_af_time(time)`

Convert a time to a `AFTIME` value.

Parameters **time** (*str* | *datetime*) – Time to convert to `AFTIME`.

Returns Time range covered by the start and end time.

Return type `AF.Time.AFTIME`

`PIconnect.time.to_af_time_range(start_time, end_time)`

Convert a combination of start and end time to a time range.

Both *start_time* and *end_time* can be either a `datetime.datetime` object or a string. *datetime* objects are first converted to a string, before being passed to `AF.Time.AFTIMERange`. It is also possible to specify either end as a *datetime* object, and then specify the other boundary as a relative string.

Parameters

- **start_time** (*str* / *datetime*) – Start time of the time range.
- **end_time** (*str* / *datetime*) – End time of the time range.

Returns Time range covered by the start and end time.

Return type `AF.Time.AFTimeRange`

1.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

1.4.1 Types of Contributions

Report Bugs

Report bugs at <https://github.com/Hugovdberg/PIconnect/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

Write Documentation

PIconnect could always use more documentation, whether as part of the official PIconnect docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/Hugovdberg/PIconnect/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

1.4.2 Get Started!

Ready to contribute? Here's how to set up *PIconnect* for local development.

1. Fork the *PIconnect* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/PIconnect.git
```

3. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your fork for local development:

```
$ cd PIconnect/
$ pipenv sync -d
$ pipenv install -e .
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, format the code with black, check that your changes pass pylint and the tests, including testing other Python versions with tox:

```
$ black PIconnect
$ pylint PIconnect tests
$ python setup.py test or py.test
$ tox
```

Pylint and tox will be installed automatically by pipenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

1.4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7, 3.8 and 3.9. Testing is automated through GitHub Actions, so you get feedback on your pull request where things are not up to standards.

1.4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_piconnect
```

1.5 Credits

1.5.1 Development Lead

- Hugo Lapré <hugo.lapre@brabantwater.nl> <<https://github.com/Hugovdberg>>

1.5.2 Contributors

- Stijn de Jong
- AlcibiadesCleinias <<https://github.com/AlcibiadesCleinias>>
- Leandro Dariva Pinto <<https://github.com/ldariva>>

1.6 History

1.6.1 0.9.1 (2021-08-11)

- Fixes the Exception type to swallow (related to #580)
- Fixes missing dependency in wheel (#595)

1.6.2 0.9.0 (2021-08-10)

Warning: This is the final version to support python 2.7

- Added support to write values to the databases (#573)
- Added support for extracting Event frames from PI-AF (#587)
- Added methods to extract a single historic value from both *PIPoint* and *PIAFAttribute* objects. (#523)

- Added options to login to the PI Server using provided credentials (#522)
- Added option to set the connection timeout for data extraction (#572)
- Better loading of the configured servers (#580)
- All data extracting functions now support both extraction using strings and *datetime* objects. (#574)

1.6.3 0.8.0 (2020-03-03)

- Added option to configure the timezone for the returned index. Changed default from Europe/Amsterdam to UTC! Adds *pytz* as new dependency(#499)
- More robust detection of the default PI AF server (#496, #501)
- Removed *pytest-runner* dependency unless explicitly requested (#503)
- Exiting the context manager for a *PIAFDatabase* no longer explicitly disconnects from the server, but leaves it up to SDK. (#487)
- Various updates of the package dependencies

1.6.4 0.7.1 (2019-08-16)

- Improved documentation
- Changed *PIData.PISeriesContainer* to an Abstract Base Class

1.6.5 0.7.0 (2018-11-14)

- Add *summary*, *summaries*, and *filtered_summaries* methods to *PIPoint* and *PIAFAttribute*

1.6.6 0.6.0 (2018-07-05)

1.6.7 0.5.1 (2017-11-25)

1.6.8 0.4.0 (2017-11-25)

- First release on PyPI.

1.7 Copyrights

Even though the PIconnect package itself is licensed under the MIT license, it requires the OSIsoft PI AF SDK to actually make the connections. All copyrights to the SDK remain explicitly to OSIsoft.

OSIsoft, the OSIsoft logo and logotype, Managed PI, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developer's Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC driver, PI Manual Logger, PI Notifications, PI ODBC, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, RLINK and RtReports are all trademarks of OSIsoft, LLC.

1.8 Things left to do

Todo: Reject searches while not connected

(The [original entry](#) is located in /home/docs/checkouts/readthedocs.org/user_builds/piconnect/checkouts/stable/PIconnect/PI.py:docstring of PIconnect.PI.PIServer.search, line 11.)

Todo: Add setter to alter displayed description

(The [original entry](#) is located in /home/docs/checkouts/readthedocs.org/user_builds/piconnect/checkouts/stable/PIconnect/PI.py:docstring of PIconnect.PI.PIPoint.description, line 3.)

Todo: Move `__boundary_types` to PIconsts as a new enumeration

(The [original entry](#) is located in /home/docs/checkouts/readthedocs.org/user_builds/piconnect/checkouts/stable/PIconnect/PIData.py:docstring of PIconnect.PIData.PISeriesContainer, line 4.)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`PIconnect`, [13](#)

`PIconnect.AFSDK`, [20](#)

`PIconnect.config`, [29](#)

`PIconnect.PIConsts`, [21](#)

`PIconnect.PIData`, [25](#)

`PIconnect.time`, [29](#)

A

AF (class in *PIconnect.AFSDK*), 20
 AF.AFDatabase (class in *PIconnect.AFSDK*), 20
 AF.AFElement (class in *PIconnect.AFSDK*), 20
 AF.Data (class in *PIconnect.AFSDK*), 20
 AF.Data.AFBoundaryType (class in *PIconnect.AFSDK*), 20
 AF.PI (class in *PIconnect.AFSDK*), 20
 AF.PI.PIPoint (class in *PIconnect.AFSDK*), 20
 AF.PI.PIServer (class in *PIconnect.AFSDK*), 20
 AF.PI.PIServers (class in *PIconnect.AFSDK*), 20
 AF.PISystem (class in *PIconnect.AFSDK*), 21
 AF.PISystem.InternalDatabases (class in *PIconnect.AFSDK*), 21
 AF.PISystems (class in *PIconnect.AFSDK*), 21
 AF.Time (class in *PIconnect.AFSDK*), 21
 AF.Time.AFTimeRange (class in *PIconnect.AFSDK*), 21
 AF.Time.AFTimeSpan (class in *PIconnect.AFSDK*), 21
 AFTER (*PIconnect.PIConsts.RetrievalMode* attribute), 23
 ALL (*PIconnect.PIConsts.SummaryType* attribute), 24
 ALL_FOR_NON_NUMERIC (*PIconnect.PIConsts.SummaryType* attribute), 24
 AT_OR_AFTER (*PIconnect.PIConsts.RetrievalMode* attribute), 23
 AT_OR_BEFORE (*PIconnect.PIConsts.RetrievalMode* attribute), 23
 attributes (*PIconnect.PIAF.PIAFElement* property), 19
 AuthenticationMode (class in *PIconnect.PIConsts*), 21
 AUTO (*PIconnect.PIConsts.RetrievalMode* attribute), 23
 AUTO (*PIconnect.PIConsts.TimestampCalculation* attribute), 24
 AVERAGE (*PIconnect.PIConsts.SummaryType* attribute), 24

B

BACKWARD_FROM_END_TIME (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22
 BACKWARD_FROM_START_TIME (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22

BACKWARD_IN_PROGRESS (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22
 BEFORE (*PIconnect.PIConsts.RetrievalMode* attribute), 23
 BUFFER (*PIconnect.PIConsts.BufferMode* attribute), 22
 BUFFER_IF_POSSIBLE (*PIconnect.PIConsts.BufferMode* attribute), 22
 BufferMode (class in *PIconnect.PIConsts*), 21

C

CalculationBasis (class in *PIconnect.PIConsts*), 22
 categories (*PIconnect.PIAF.PIAFElement* property), 19
 children (*PIconnect.PIAF.PIAFDatabase* property), 19
 children (*PIconnect.PIAF.PIAFElement* property), 19
 Connect() (*PIconnect.AFSDK.AF.PI.PIServer* method), 20
 Connect() (*PIconnect.AFSDK.AF.PISystem* method), 21
 COUNT (*PIconnect.PIConsts.SummaryType* attribute), 24
 current_value (*PIconnect.PI.PIPoint* property), 14
 current_value (*PIconnect.PIData.PISeriesContainer* property), 25

D

database_name (*PIconnect.PIAF.PIAFDatabase* property), 19
 default_server (*PIconnect.PI.PIServer* attribute), 14
 default_server (*PIconnect.PIAF.PIAFDatabase* attribute), 19
 DEFAULT_TIMEZONE (*PIconnect.config.PIConfigContainer* property), 29
 DefaultPIServer (*PIconnect.AFSDK.AF.PI.PIServers* attribute), 20
 DefaultPISystem (*PIconnect.AFSDK.AF.PISystems* attribute), 21
 descendant() (*PIconnect.PIAF.PIAFDatabase* method), 19
 descendant() (*PIconnect.PIAF.PIAFElement* method), 19
 description (*PIconnect.PI.PIPoint* property), 14

description (*PIconnect.PIAF.PIAFElement* property), 19

Disconnect() (*PIconnect.AFSDK.AF.PI.PIServer* method), 20

Disconnect() (*PIconnect.AFSDK.AF.PISystem* method), 21

DO_NOT_BUFFER (*PIconnect.PIConsts.BufferMode* attribute), 22

E

EARLIEST_TIME (*PIconnect.PIConsts.TimestampCalculation* attribute), 24

ENDING_AFTER (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22

ENDING_BEFORE (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22

event_frames() (*PIconnect.PIAF.PIAFDatabase* method), 19

EVENT_WEIGHTED (*PIconnect.PIConsts.CalculationBasis* attribute), 22

EVENT_WEIGHTED_EXCLUDE_EARLIEST (*PIconnect.PIConsts.CalculationBasis* attribute), 22

EVENT_WEIGHTED_EXCLUDE_MOST_RECENT (*PIconnect.PIConsts.CalculationBasis* attribute), 22

EVENT_WEIGHTED_INCLUDE_BOTH_ENDS (*PIconnect.PIConsts.CalculationBasis* attribute), 22

EventFrameSearchMode (class in *PIconnect.PIConsts*), 22

EXACT (*PIconnect.PIConsts.RetrievalMode* attribute), 23

EXPRESSION_RECORDED_VALUES (*PIconnect.PIConsts.ExpressionSampleType* attribute), 23

ExpressionSampleType (class in *PIconnect.PIConsts*), 23

F

filtered_summaries() (*PIconnect.PI.PIPoint* method), 15

filtered_summaries() (*PIconnect.PIData.PISeriesContainer* method), 25

FindPIPoints() (*PIconnect.AFSDK.AF.PI.PIPoint* static method), 20

FORWARD_FROM_END_TIME (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22

FORWARD_FROM_START_TIME (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 22

tribute), 22

FORWARD_IN_PROGRESS (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23

G

get_enumerated_value() (in module *PIconnect.PIConsts*), 25

I

INSERT (*PIconnect.PIConsts.UpdateMode* attribute), 25

INSERT_NO_COMPRESSION (*PIconnect.PIConsts.UpdateMode* attribute), 25

Inside (*PIconnect.AFSDK.AF.Data.AFBoundaryType* attribute), 20

Interpolated (*PIconnect.AFSDK.AF.Data.AFBoundaryType* attribute), 20

interpolated_value() (*PIconnect.PI.PIPoint* method), 15

interpolated_value() (*PIconnect.PIData.PISeriesContainer* method), 26

interpolated_values() (*PIconnect.PI.PIPoint* method), 16

interpolated_values() (*PIconnect.PIData.PISeriesContainer* method), 26

INTERVAL (*PIconnect.PIConsts.ExpressionSampleType* attribute), 23

L

last_update (*PIconnect.PI.PIPoint* property), 16

M

MAXIMUM (*PIconnect.PIConsts.SummaryType* attribute), 24

MINIMUM (*PIconnect.PIConsts.SummaryType* attribute), 24

module

PIconnect, 13

PIconnect.AFSDK, 20

PIconnect.config, 29

PIconnect.PIConsts, 21

PIconnect.PIData, 25

PIconnect.time, 29

MOST_RECENT_TIME (*PIconnect.PIConsts.TimestampCalculation* attribute), 24

N

name (*PIconnect.PI.PIPoint* property), 16

name (*PIconnect.PIAF.PIAFElement* property), 19

name() (*PIconnect.PIData.PISeriesContainer* method), 27
 NO_REPLACE (*PIconnect.PIConsts.UpdateMode* attribute), 25
 NONE (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23
 NONE (*PIconnect.PIConsts.SummaryType* attribute), 24

O

Outside (*PIconnect.AFSDK.AF.Data.AFBoundaryType* attribute), 20

P

parent (*PIconnect.PIAF.PIAFElement* property), 19
 Parse() (*PIconnect.AFSDK.AF.Time.AFTimeSpan* static method), 21
 PERCENT_GOOD (*PIconnect.PIConsts.SummaryType* attribute), 24
 PI_USER_AUTHENTICATION (*PIconnect.PIConsts.AuthenticationMode* attribute), 21
 PIAFDatabase (class in *PIconnect.PIAF*), 19
 PIAFElement (class in *PIconnect.PIAF*), 19
 PIConfigContainer (class in *PIconnect.config*), 29
 PIconnect
 module, 13
 PIconnect.AFSDK
 module, 20
 PIconnect.config
 module, 29
 PIconnect.PIConsts
 module, 21
 PIconnect.PIData
 module, 25
 PIconnect.time
 module, 29
 PIPoint (class in *PIconnect.PI*), 14
 PISeriesContainer (class in *PIconnect.PIData*), 25
 PIServer (class in *PIconnect.PI*), 14
 POP_STD_DEV (*PIconnect.PIConsts.SummaryType* attribute), 24

R

RANGE (*PIconnect.PIConsts.SummaryType* attribute), 24
 raw_attributes (*PIconnect.PI.PIPoint* property), 16
 recorded_value() (*PIconnect.PI.PIPoint* method), 16
 recorded_value() (*PIconnect.PIData.PISeriesContainer* method), 27
 recorded_values() (*PIconnect.PI.PIPoint* method), 16
 recorded_values() (*PIconnect.PIData.PISeriesContainer* method), 27
 REMOVE (*PIconnect.PIConsts.UpdateMode* attribute), 25

REPLACE (*PIconnect.PIConsts.UpdateMode* attribute), 25
 REPLACE_ONLY (*PIconnect.PIConsts.UpdateMode* attribute), 25
 RetrievalMode (class in *PIconnect.PIConsts*), 23

S

search() (*PIconnect.PI.PIServer* method), 14
 server_name (*PIconnect.PI.PIServer* property), 14
 server_name (*PIconnect.PIAF.PIAFDatabase* property), 19
 servers (*PIconnect.PI.PIServer* attribute), 14
 servers (*PIconnect.PIAF.PIAFDatabase* attribute), 19
 STARTING_AFTER (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23
 STARTING_AFTER_IN_PROGRESS (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23
 STARTING_BEFORE (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23
 STARTING_BEFORE_IN_PROGRESS (*PIconnect.PIConsts.EventFrameSearchMode* attribute), 23
 STD_DEV (*PIconnect.PIConsts.SummaryType* attribute), 24
 summaries() (*PIconnect.PI.PIPoint* method), 17
 summaries() (*PIconnect.PIData.PISeriesContainer* method), 28
 summary() (*PIconnect.PI.PIPoint* method), 18
 summary() (*PIconnect.PIData.PISeriesContainer* method), 28
 SummaryType (class in *PIconnect.PIConsts*), 23

T

TIME_WEIGHTED (*PIconnect.PIConsts.CalculationBasis* attribute), 22
 TIME_WEIGHTED_CONTINUOUS (*PIconnect.PIConsts.CalculationBasis* attribute), 22
 TIME_WEIGHTED_DISCRETE (*PIconnect.PIConsts.CalculationBasis* attribute), 22
 timestamp_to_index() (in module *PIconnect.time*), 29
 TimestampCalculation (class in *PIconnect.PIConsts*), 24
 to_af_time() (in module *PIconnect.time*), 29
 to_af_time_range() (in module *PIconnect.time*), 29
 TOTAL (*PIconnect.PIConsts.SummaryType* attribute), 24
 TOTAL_WITH_UOM (*PIconnect.PIConsts.SummaryType* attribute), 24

U

units_of_measurement (*PIconnect.PI.PIPoint* prop-

erty), [18](#)
`units_of_measurement()` (*PIconnect.PIData.PISeriesContainer* *method*),
[29](#)
`update_value()` (*PIconnect.PI.PIPoint* *method*), [18](#)
`update_value()` (*PIconnect.PIData.PISeriesContainer* *method*), [29](#)
`UpdateMode` (*class in PIconnect.PIConsts*), [25](#)

V

`Version` (*PIconnect.AFSDK.AF.PISystems* *attribute*), [21](#)
`version` (*PIconnect.PI.PIPoint* *attribute*), [18](#)
`version` (*PIconnect.PI.PIServer* *attribute*), [14](#)
`version` (*PIconnect.PIAF.PIAFDatabase* *attribute*), [19](#)
`version` (*PIconnect.PIAF.PIAFElement* *attribute*), [19](#)
`version` (*PIconnect.PIData.PISeriesContainer* *attribute*), [29](#)

W

`WINDOWS_AUTHENTICATION` (*PIconnect.PIConsts.AuthenticationMode* *attribute*),
[21](#)