R - Basic Example

PI Web API Connection

Installing Packages

The PI Web API requires Basic Authentication headers in order to connect. In R there aren't built-in functions that can handle connecting to a server with authentication headers, so we need to install a package that will handle this for us. A package that can do this is `RCurl`. To install `RCurl`, in the console type the following command and press `Enter`.

```
> install.packages("RCurl")
```

Now we can load the library in our script by adding the following line at the beginning the of our program.

```
library(RCurl)
```

This will make all the functions that `RCurl` provides available to us in our script. The main function we are interested in is the `getURL` function which will, given the required username and password, handle the authentication headers required to connect to the PI Web API.

```
data.request <- getURL(encoded.url,
                       ssl.verifypeer=FALSE,
                       ssl.verifyhost=FALSE,
                       userpwd = "USERNAME:PASSWORD",
                       httpauth=1L)
```

Obtaining URL

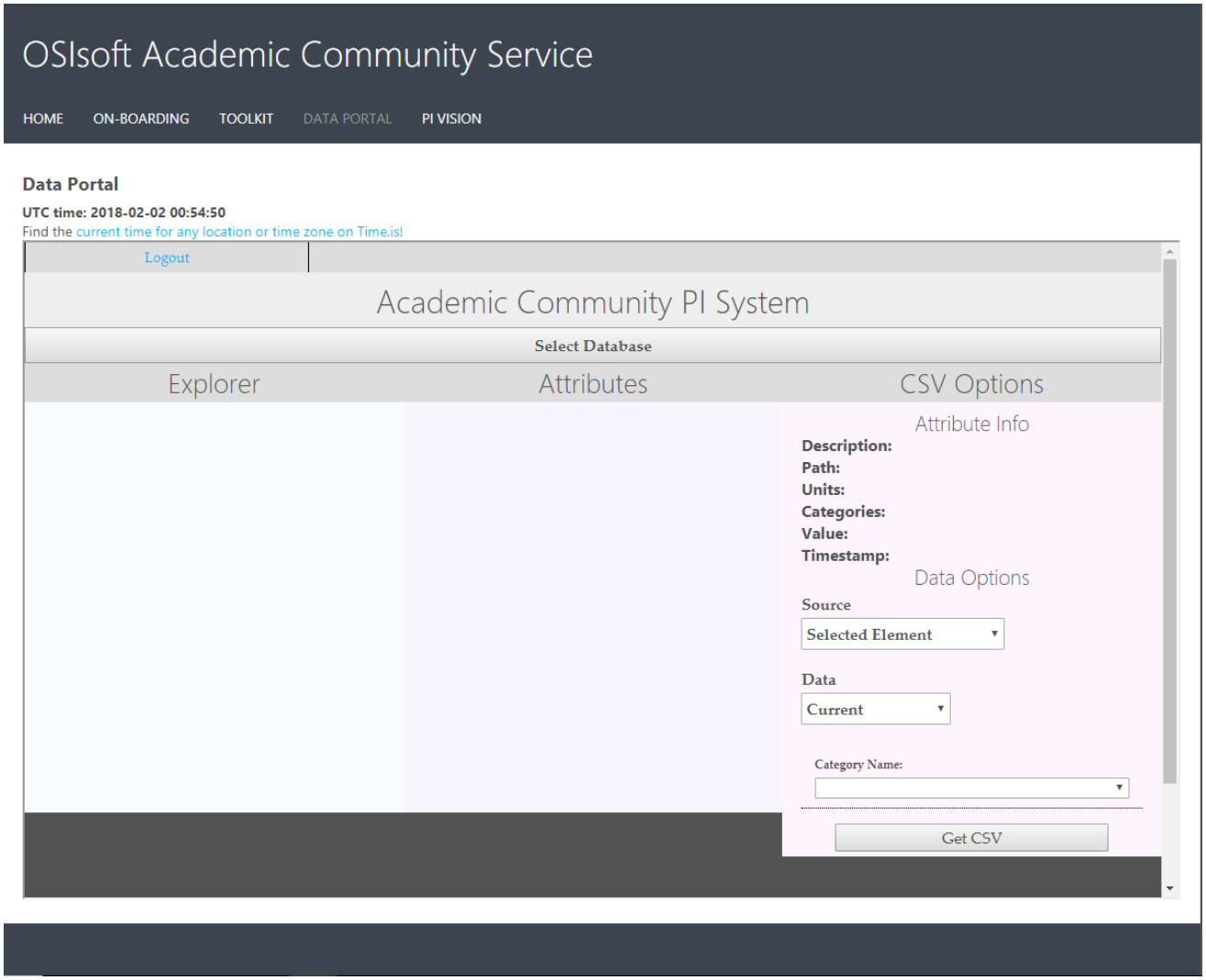
Connecting to the CSV controller requires a URL to the controller which is,

```
controller.url <- "https://academicpi.osisoft.com:444/piwebapi/Csv/ElementInterpolated"
```

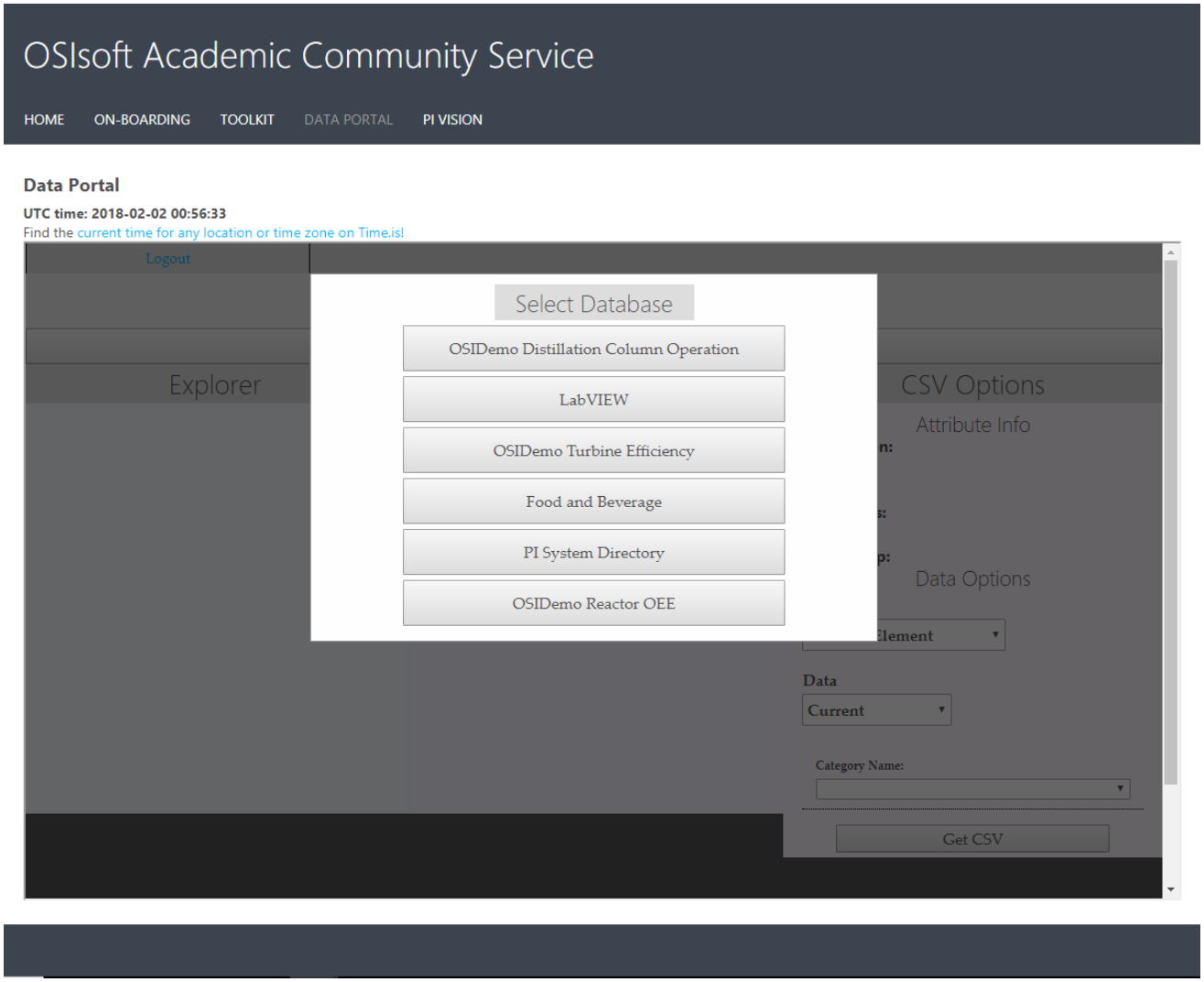
Also required is the element path to the desired asset.

Getting Element Path

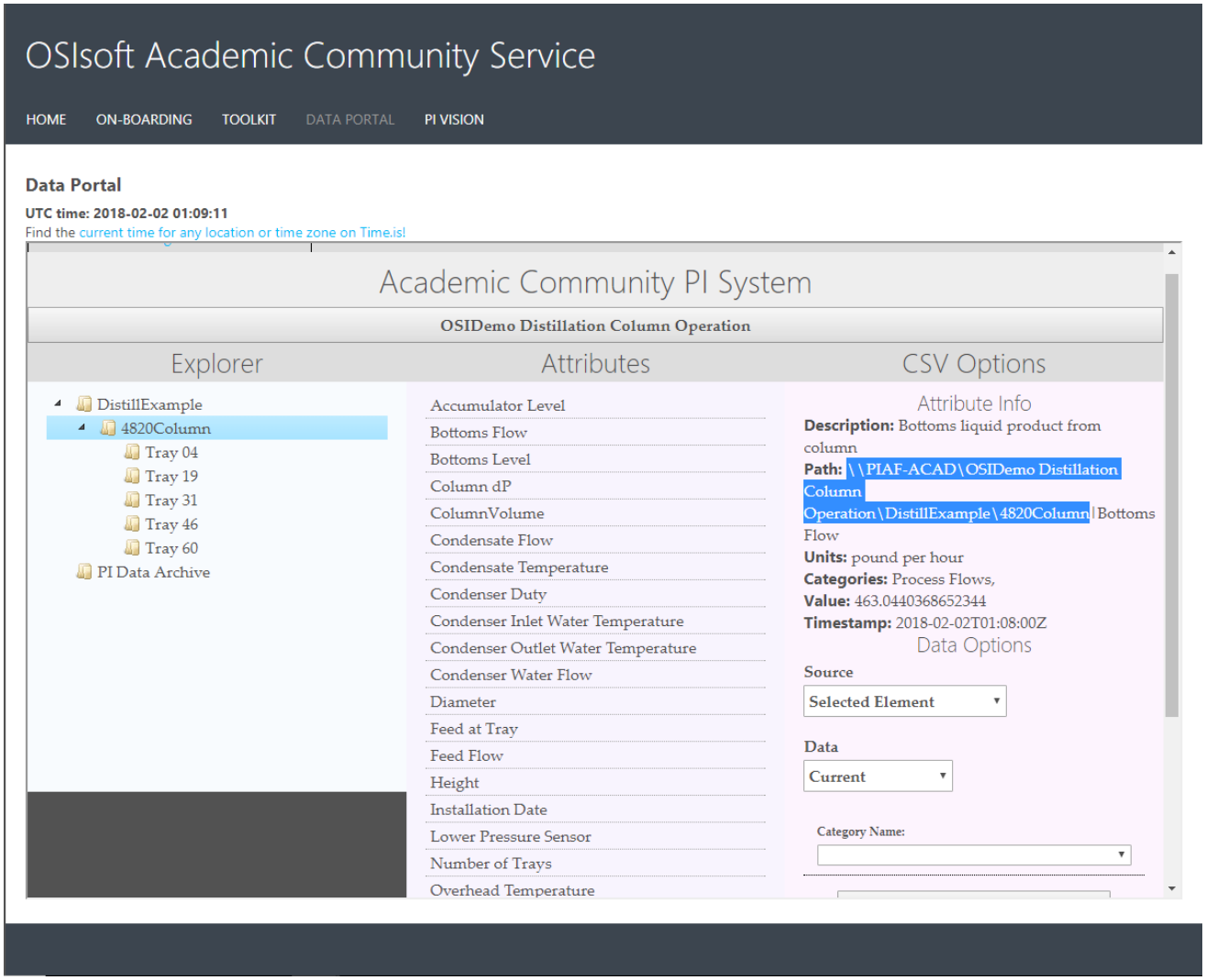
The element path can be found in the Data Portal tool on the ACS webpage. After logging in to the data portal you'll see the following window.



Select the database containing your assets. In this example we are using the "OSIDemo Distillation Column Operation"



Under the left panel titled "Explorer" select the desired asset. When you hover over one of the Attributes in the center panel the attribute info will appear in the right panel titled "CSV Options". The element path can be seen in the "Path" field in the attribute info. The path that we require is everything before the vertical bar, `|`, character. See the highlighted text in the image below.



We must also add `?path=` to the beginning of the element path before combining it with the controller URL. Using the distillation column the following is an example of the required element path.

```
element.path <- "?path=\\\\PIAF-ACAD\\OSIDemo Distillation Column
Operation\\DistillExample\\4820Column"
```

Optional parameters

The following parameters can be appended to the element path to request specific data from the PI Web API.

startTime

The starting time of the data collection can be specified, and if no start time is specified the default time of 24 hours before current time will be used. So, for example, if we want the data starting at 5 AM on January 30, 2018 we can append `startTime=01/30/2018 05:00:00` to the end of the element path and join it with an ampersand in between.

```
element.path <- "?path=\\\\PIAF-ACAD\\OSIDemo Distillation Column
Operation\\DistillExample\\4820Column&startTime=01/30/2018 05:00:00"
```

To see more valid time formats visit the time format help section

Note: Start time and end time parameters are considered to be in the user's local time

endTime

Similar to the start time parameter, we can also specify the end time of the data, and if no end time is specified the current time will be used as it's default value. So, if we want the data collected between 5 AM January 30, 2018 and 2 PM January 30, 2018 we can append the end time the same way we did the start time.

```
element.path <- "?path=\\\\PIAF-ACAD\\OSIDemo Distillation Column
Operation\\DistillExample\\4820Column&startTime=01/30/2018 05:00:00&endTime=01/30/2018 14:00:00"
```

interval

The interval parameter can be used to request data from the PI Web API at regularly spaced timestamps. If no interval is specified the default value used is one hour. Adding to our distillation column example we can specify that we want the event data every 10 minutes between 5 AM and 2 PM on January 30, 2018.

```
element.path <- "?path=\\\\PIAF-ACAD\\OSIDemo Distillation Column
Operation\\DistillExample\\4820Column&startTime=01/30/2018 05:00:00&endTime=01/30/2018
14:00:00&interval=10m"
```

nameFilter

We can specify which attributes we would like returned by the PI Web API using the name filter parameter. If no name filter is given, data for all of the attributes will be returned. If we wanted all of the types of flow data we can use `nameFilter=*flow*`. This would specify that we want data from every attribute that contains the word flow in the name. We can then append the parameter in the same way we did with the previous ones.

```
element.path <- "?path=\\\\PIAF-ACAD\\OSIDemo Distillation Column
Operation\\DistillExample\\4820Column&startTime=01/30/2018 05:00:00&endTime=01/30/2018
14:00:00&interval=10m&nameFilter=*flow*"
```

We can then combine the controller URL and element path using R's `paste` function as follows.

```
url <- paste(controller.url, element.path, sep = "")
```

The `sep` parameter of the paste function is the character string that will separate the controller URL and element path. In this case we do not want anything in between the two, so we pass an empty string as its value.

Next, we have to encode the URL which replaces any invalid URL characters such as empty spaces, backslashes, etc., and we do this by using R's `URLencode` function.

```
encoded.url <- URLencode(url)
```

Now we can get the data from the PI Web API with the encoded URL using the previously mentioned `getURL` function.

```
data.request <- getURL(encoded.url,
                       ssl.verifypeer=FALSE,
                       ssl.verifyhost=FALSE,
                       userpwd = "USERNAME:PASSWORD",
                       httpauth=1L)
```

The value returned by `getURL` is the CSV file in the form of a character string.

Time Zone

When we made the data request to the PI Web API it used our local time zone when retrieving the date, but the timestamps that are returned by the PI Web API are in the UTC time zone. We can convert the timestamps to our local time zone using R's built-in functions.

Data Processing

Now, that we have obtained the CSV from the PI web API we need to process the data into something useful like a data frame. To do this we can us the `read.csv` function as follows.

```
DataFrame <- read.csv(text = data.string)
```

Because the data was returned by the `getURL` function as a character string we must pass the data string into the `read.csv` function as a text parameter. Simply calling the `read.csv` function as follows will result in an error because the function would expect a file name.

```
DataFrame <- read.csv(data.string)
```

In the data frame's current state the timestamps are stored as character strings. We need to convert the timestamp strings into a more useful data type.

Time Formatting

R has date-time classes which we can use to extract information from the timestamps. If, for instance, we only need the date or hour of the timestamp, R has functions which can extract the information from the date-time class. To convert the character string to a date-time class we can call on R's `strptime` function.

```
DataFrame$Timestamp <- strptime(DataFrame$Timestamp, format = "%m/%d/%Y %H:%M:%S", tz = "UTC")
```

The `strptime` function needs to know what the format of the timestamp string is in, so we have to pass a parameter giving the format. The PI Web API returns the timestamps in the MM/DD/YYYY HH:MM:SS format. More information on the format parameter can be found in R's documentation for the `strptime` function.

Time Zone Conversion

When requesting data from the PI Web API the request is made in the local time zone, but the timestamps are returned in UTC. After converting the timestamp from a string to a date-time class we can then change the time zone from UTC to the local time zone.

```
DataFrame$Timestamp <- as.POSIXct(DataFrame$Timestamp)
DataFrame$Timestamp <- as.POSIXlt(DataFrame$Timestamp, tz = Sys.timezone())
```

Plotting

R has a built-in function which can plot our data, we just have to specify our x and y values and any modifications we want to add to the graph. In R we can reference columns in data frames using the column header name. If we wanted the steam flow data from our data frame we would we would call our data frame followed by the `$` symbol and then the header name. This would look like,

```
DataFrame$Steam.Flow
```

The following is a plot of the Steam Flow over our requested time range with an x-axis label of "Timestamp" and we specified we wanted a line graph. For more information about the parameters that can be given see R's documentation of the `plot` function.

```
plot(x=DataFrame$Timestamp, y=DateFrame$Steam.Flow, xlab="Timestamp", ylab="Flow Rate",
main="Steam Flow", type="l")
```