Q&A

Oct 2018

Can it connect to database used for NEO reports?

- Yes it can connect to the same database as NEO.
- It uses the same code as NEO.
- As part of the installation process you need to get a compatible version of NEO configured on that machine then uses its settings file to install in NEOpoint. NEO can then be uninstalled.
- Can connect to any number of databases. Supported types are Oracle, SQLServer, MySQL

Can it update automatically on a 30min basis?

- All reports in NEOpoint automatically update within a couple of seconds of new data coming in.
- NEOpoint uses “SignalR” long polling that gives sub second responses while only polling every minute. (Server holds onto request until data changes)
- Demo showing 5min report changing on 5min interval.

Can it alarm or have alerts for certain criteria?

- It has alerts for a wide range of criteria, e.g. pre5min price > X etc. Alert variables are easily added so you can pretty much create alerts on anything.
- Demo: NEOPoint (NP) Alert page. Note: NP does not allow creation of Equation type Alerts but they will however run in NP.
- Equations support these operators : +, -, *, /, >, <, <=, >=, =, !=, abs(). Can access multiple columns from Alert Variable and previous value
- Sample Alert Equations:
- abs('Region Price 5min'.NSW1["Price 5min"] - 'Region Price 5min'.NSW1["Price 5min"].1) > 5
- 'Region Price30min'["NSW1"] > (3 * 'STTM Ex ante market price'["SYD"])

Is it possible to have calculations conducted in a database etc and the output be published into a NEO Point report (instead of being done and displayed in spreadsheets)?

- Of course reports can use any number of complex SQL queries, plus you can then process the results in NEOpoint using built-in Python scripting.
- Sample Python script on next slide shows Python script that calculates distribution of spot prices.
- Demo: NEOpoint tabular report and dashboard.

```python
# Copyright (C) Intelligent Energy Systems
import System
import IES.Mercury
import MercuryUtilities #Script\MercuryUtilities
import IES.Utility

from IES.Mercury import *
from MercuryUtilities import *
from IES.Utility import EventLevel

# Name:       PostRun
# Description:     Creates a distribution of NEM spot prices based on the following Price categories
#           <50, 50-100, 100-300, >300.
# Input:       report - IES.Mercury.Reports.ReportProvider object.  Provides access to two IMercuryObjects. One representing report and the other representing a set or report variables.
# Remarks:
def PostRun(report):
    dataObject = report.DataObject
    eventLog = IES.Mercury.MercurySystem.MercuryStatic.Eventlog
    if dataObject.Count < 2:
        eventlog.AddEvent("Number of variables should be one at least to run this script", report.ReportObject.Name, EventLevel.Error)
        return
    sectionIn = dataObject[0]

    sectionsOut = GetCagegoriesSections(sectionIn)
    #remove existing sections except for the section representing axis
    dataObject.RemoveAt(0)

    for i in range(sectionsOut.Count):
        dataObject.Add(sectionsOut[i])

# Name:       GetCagegorySections
# Description:     Separates values in sectionIn into different categories and calculates the percentage of each category, then creates
#           sections for each category with 2 columns and populates one row of each section.
# Input:       sectionIn - input data.  Should contain at least two columns.
# Remarks:       The second column in sectionIn is used to calculate percentages and construct a StackedBar chart.
def GetCagegoriesSections(sectionIn):
    categoryCol = IES.Mercury.MercuryColString()
    categories = ["<50$/MWh", "50-100$/MWh", "100-300$/MWh", ">300$/MWh"]
    #specify one section for each category
    sectionsOut = CreateStringCategoriesSections("(%)", categories, Charts.ChartType.StackedBar)
    barLabel = "% half and hours graph"
    count = int(sectionIn[1].Count)

    for i in range(count):
        price = float(sectionIn[1][i])

        if price < 50.0:
            sectionOut = sectionsOut[0]
        elif price < 100.0:
            sectionOut = sectionsOut[1]
        elif price <= 300.0:
            sectionOut = sectionsOut[2]
        else:
            sectionOut = sectionsOut[3]
        if sectionOut[1].Count == 0:
            sectionOut[1].Add(1.0)
            sectionOut[0].Add(barLabel)
        else:
            sectionOut[1][0] = float(sectionOut[1][0]) + 1.0

    #normalize
    for i in range(sectionsOut.Count):
        if sectionsOut[i][1].Count > 0: #some sections may not have been populated
            sectionsOut[i][1][0] = float(sectionsOut[i][1][0]) * 100 / count
    return sectionsOut
```

Is it possible to have key NEO Point reports accessed via iphone etc?

- Yes. You can either use the NEOpoint mobile app or use a browser to access the NEOmobile web page that is specifically designed to work with mobile phones.
- All reports can be access via the desktop, pad or phone views. Note that “dashboards” are not supported in Mobile view.
- Demo: show NEOpoint in Mobile view.