OSIsoft®

USERS CONFERENCE 2016

April 4-8, 2016 | San Francisco

TRANSFORM YOUR WORLD

The Tao of Qi
OSIsoft's New Martial Art

Presented by **Rhys Kirk, The Genius Group**
Laurent Garrigues, OSIsoft

The OSIsoft Qi Grading Syllabus

7th Kyu: **What is OSIsoft Qi?**

6th Kyu: **REST and the ways of the Qi Client.**

5th Kyu: **Hello World.**

4th Kyu: **Time series data.**

3rd Kyu: **Complex Data Types.**

2nd Kyu: **Depth Indexed Data.**

1st Kyu: **Advanced OSIsoft Qi.**

1st Dan: **Becoming an OSIsoft Qi expert.**

My current OSIsoft Qi grade

7th Kyu: **What is OSIsoft Qi?**

6th Kyu: **REST and the ways of the Qi Client.**

5th Kyu: **Hello World.**

4th Kyu: **Time series data.**

3rd Kyu: **Complex Data Types.**

2nd Kyu: **Depth Indexed Data.**

1st Kyu: **Advanced OSIsoft Qi.**

1st Dan: **Becoming an OSIsoft Qi expert.**

Why am I not a black belt?

1st Kyu: **Advanced OSIsoft Qi.**

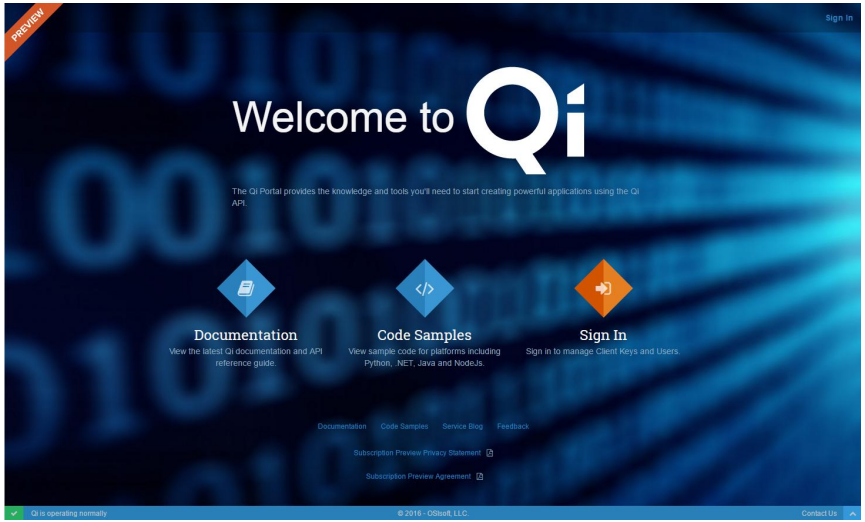
- Evolving platform.
- Currently a beta product.
- Takes time and experience.
- More ideas to test.

1st Dan: **Becoming an OSIsoft Qi expert.**

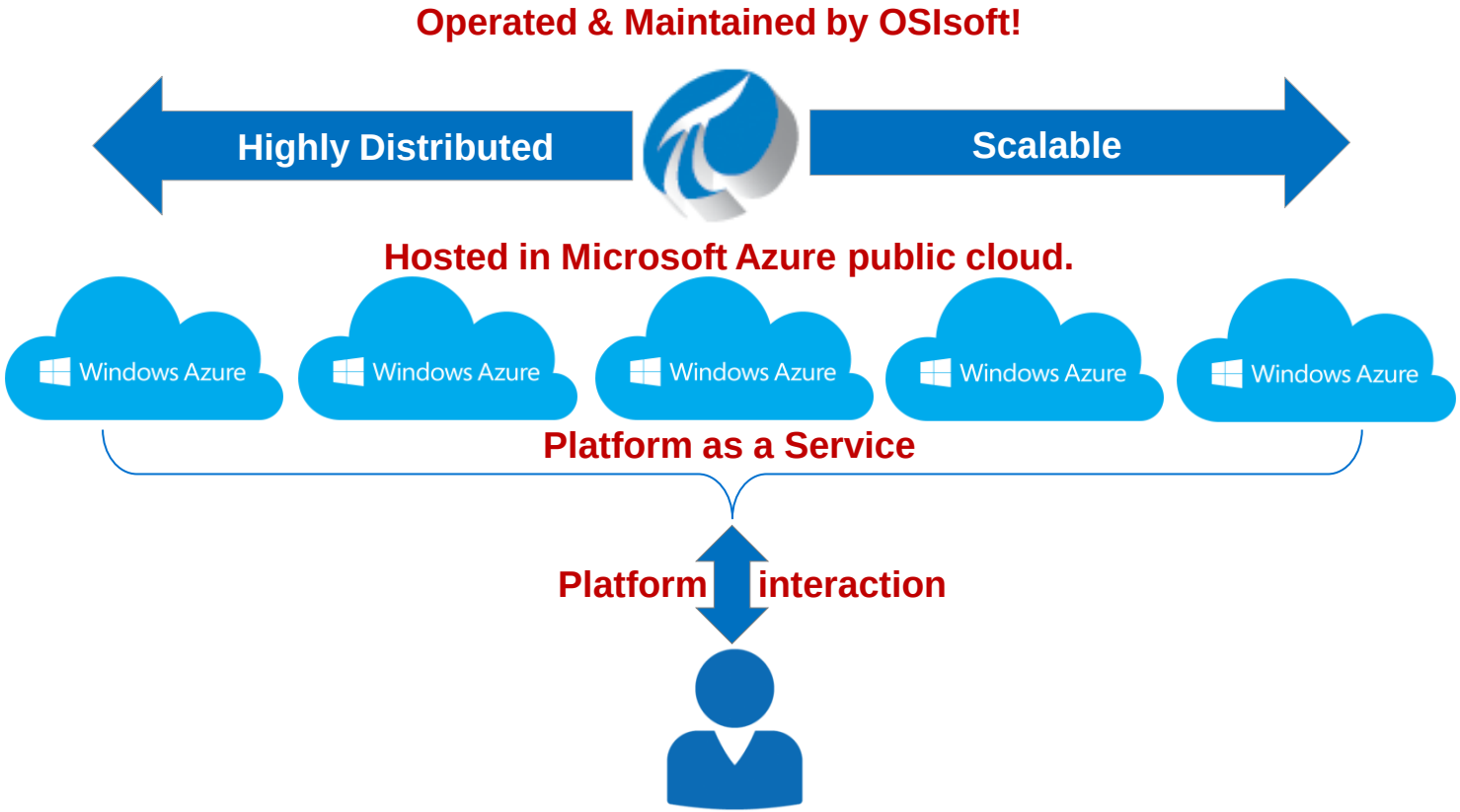
7th Kyu: **What is OSIsoft Qi?**

https://qi.osisoft.com/

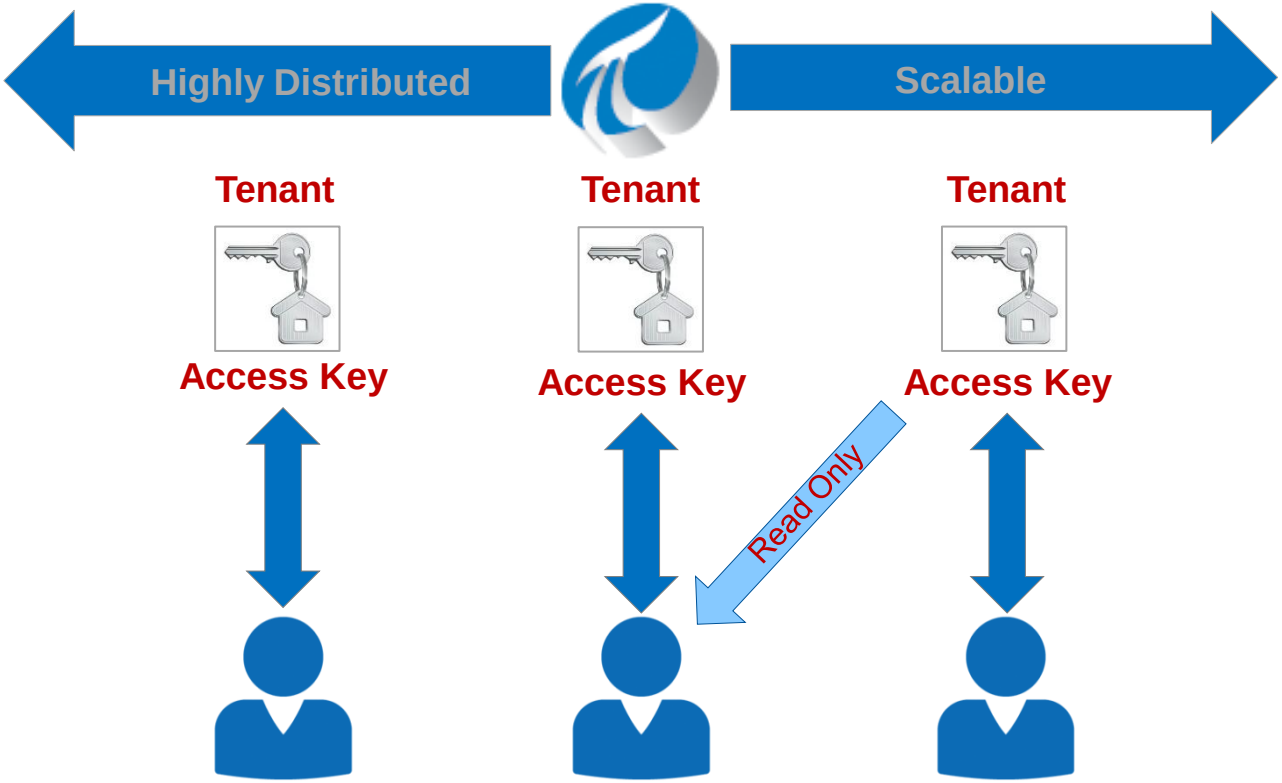
Qi is a **cloud-based** highly flexible sequential data historian that can be used to *store*, *retrieve* and *analyze* **data**.



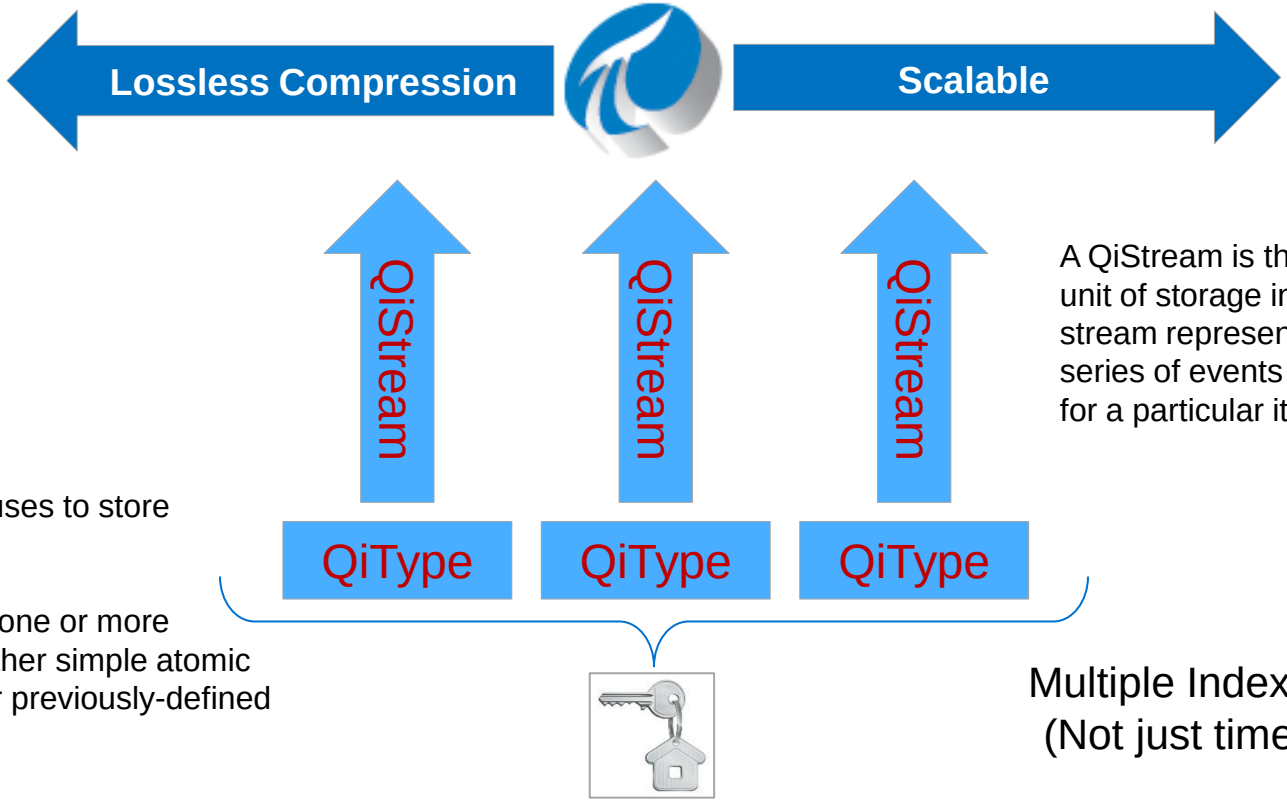
7th Kyu: **What is OSIsoft Qi?**



7th Kyu: **What is OSIsoft Qi?**



7th Kyu: **What is OSIsoft Qi?**



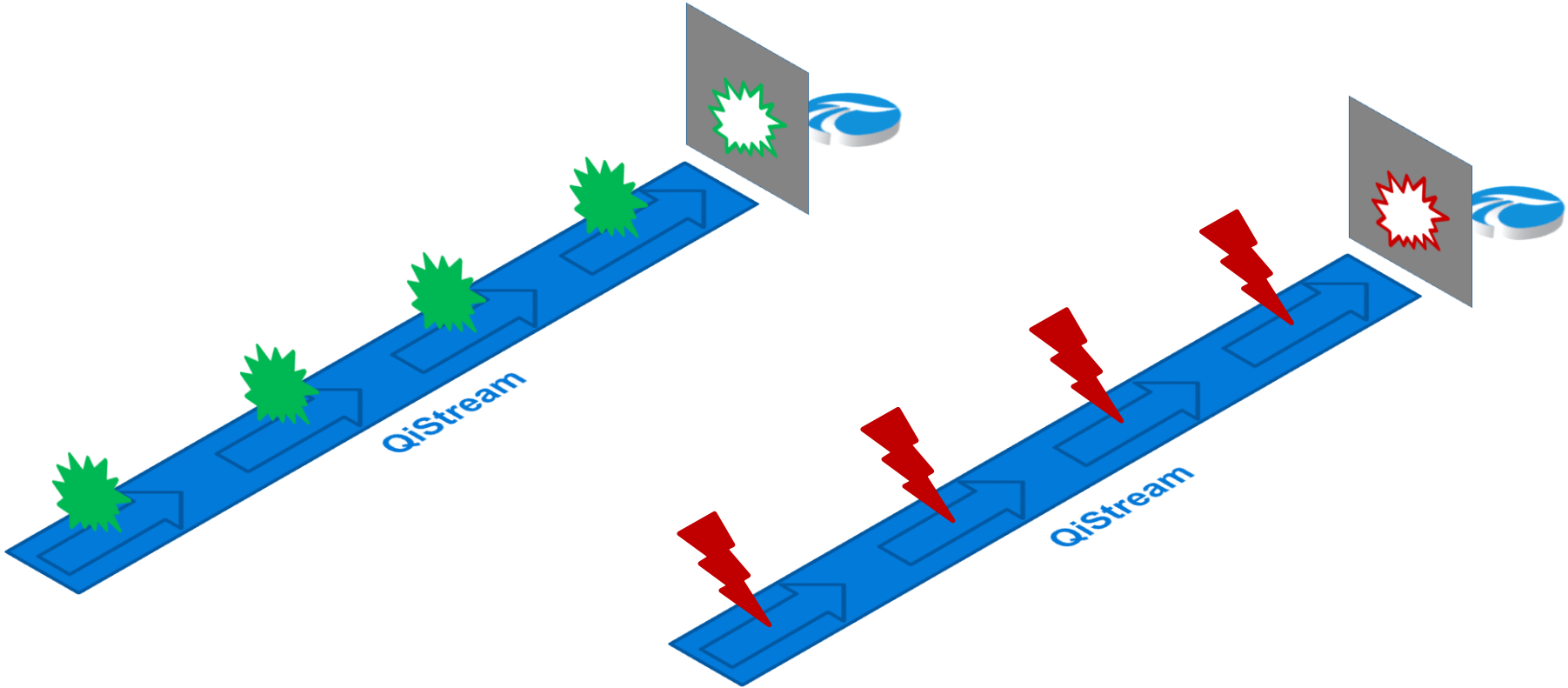
A QiStream is the fundamental unit of storage in Qi. Each stream represents an ordered series of events or observations for a particular item of interest.

A QiType is what Qi uses to store definable data types.

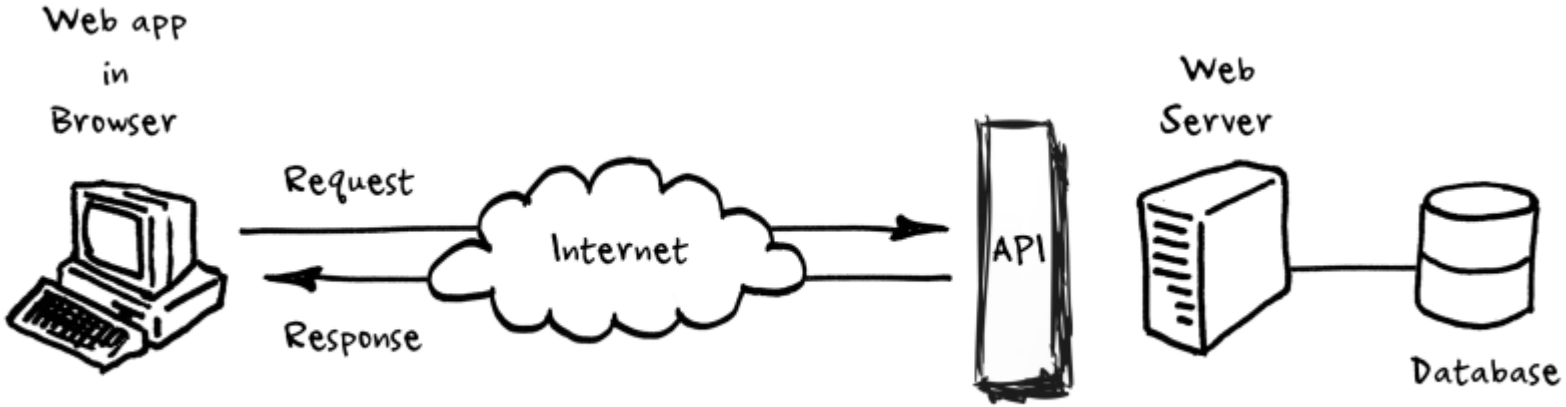
A QiType consists of one or more properties that are either simple atomic types (e.g. integer) or previously-defined QiTypes.

Multiple Indexes
(Not just time!)

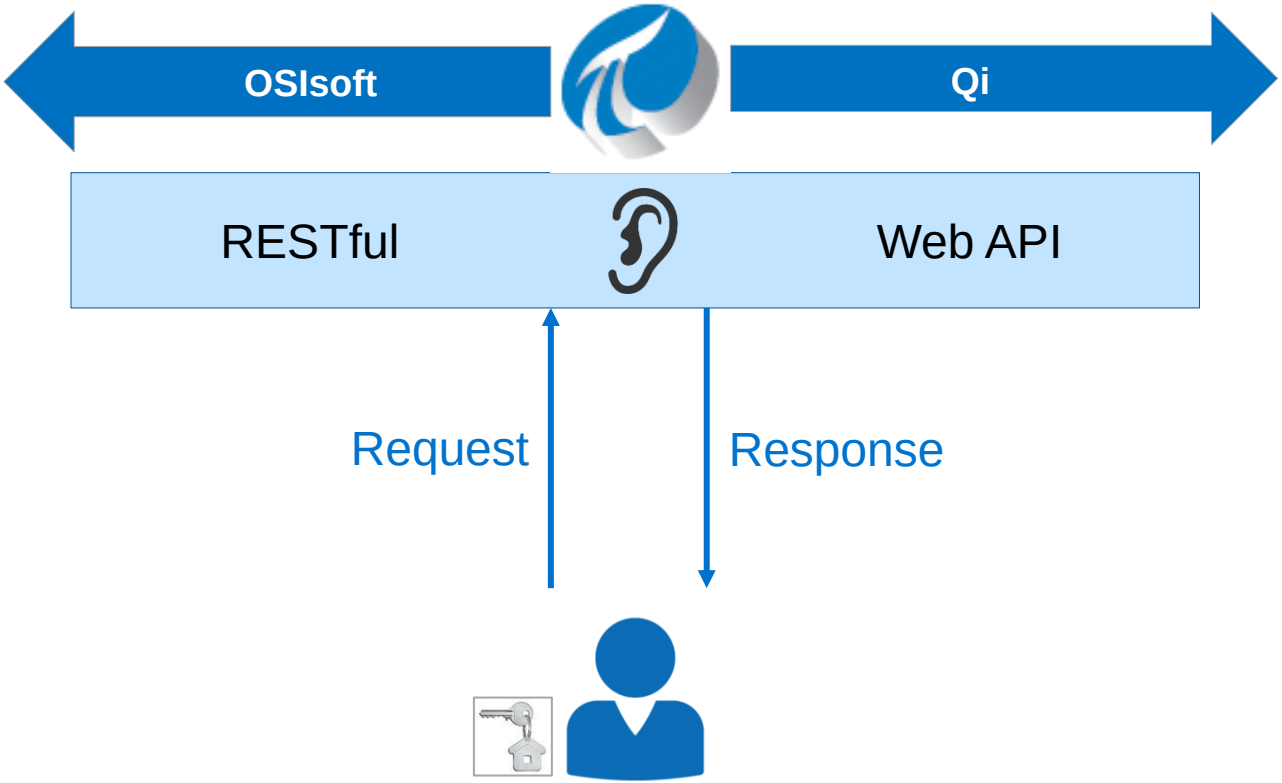
7th Kyu: **What is OSIsoft Qi?**



6th Kyu: **REST and the way of the Qi client**



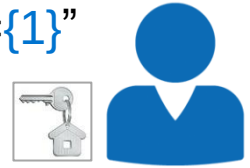
6th Kyu: **REST and the way of the Qi client**



6th Kyu: **REST and the way of the Qi client**

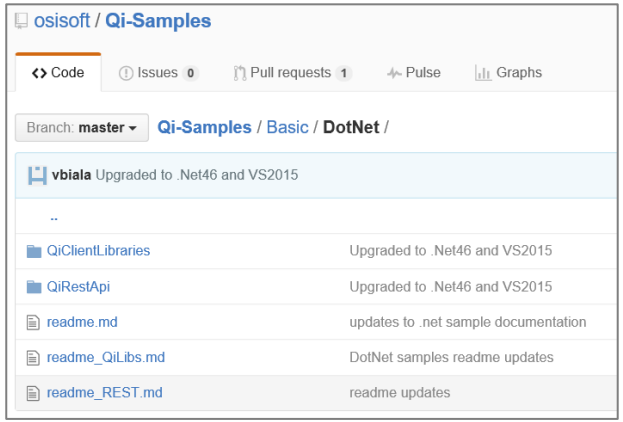
- "https://qi-data.osisoft.com:3380"
- “/Qi/Streams/{streamid}/"
- “/Data/GetWindowValues?startIndex={0}&endIndex={1}”

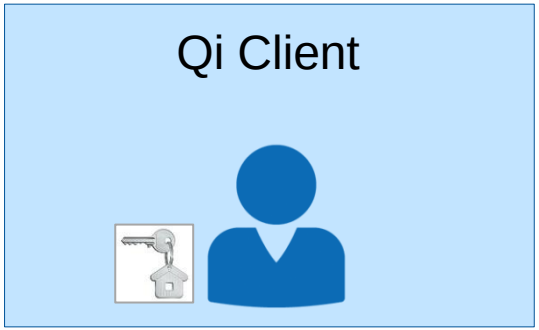
Request



6th Kyu: **REST and the way of the Qi client**

6th Kyu: **REST and the way of the Qi client**





NodeJs

.NET

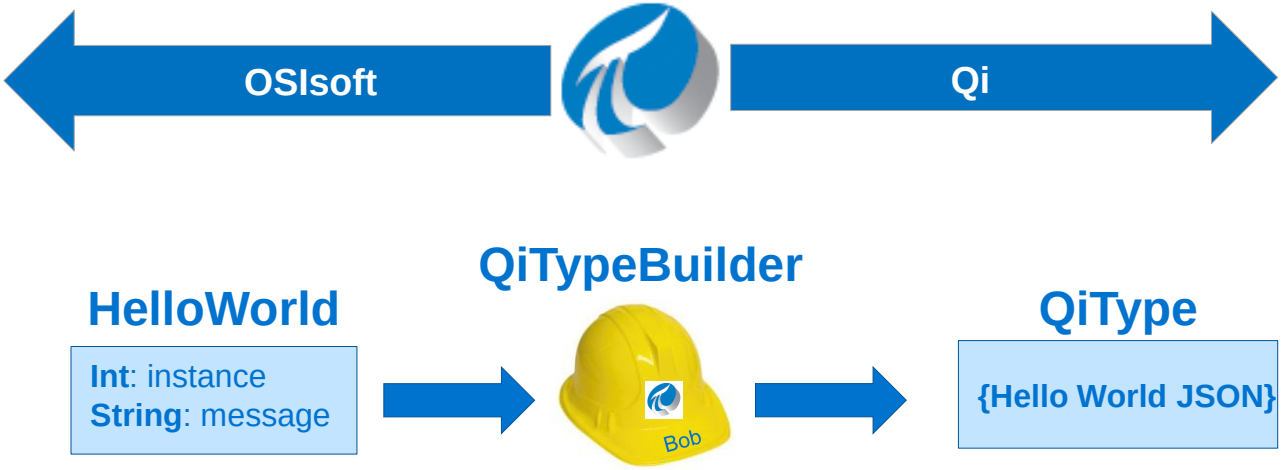
Java

Python

5th Kyu: **Hello World**

- Create a VERY basic “Hello World” object.
- Build a QiType from that object.
- Summon a QiStream for our “Hello Worlds” to sail down.
- Set sail to the OSIsoft Qi storage.

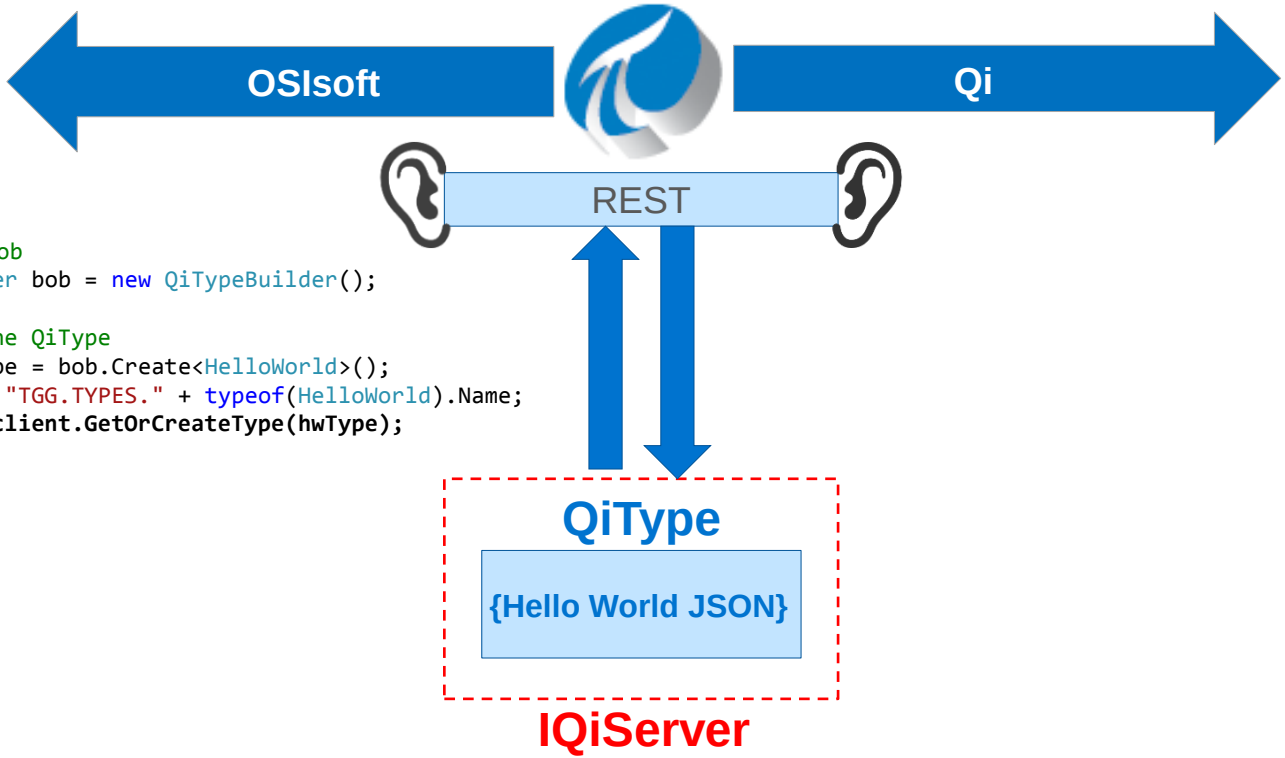
5th Kyu: **Hello World**




```
class HelloWorld
    {
        [Key]
        public int instance;
        public string message;
}
```

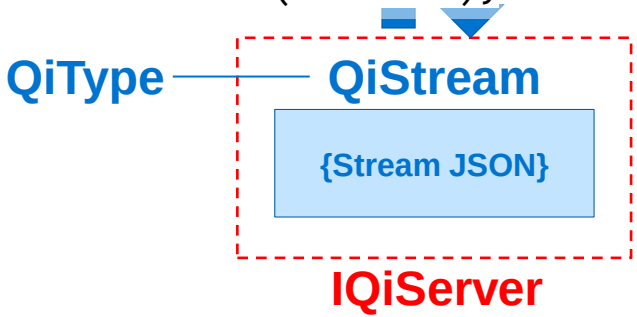
```
// Employ Bob
QiTypeBuilder bob = new QiTypeBuilder();
// Create the QiType
QiType hwType = bob.Create<HelloWorld>();
```

5th Kyu: **Hello World**



5th Kyu: **Hello World**


```
// Create the QiStream
QiStream hwStream = new QiStream() { Id = "TGG.STREAMS." + typeof(HelloWorld).Name,
TypeId = hwType.Id };
hwStream = qiclient.GetOrCreateStream(hwStream);
```



5th Kyu: **Hello World**


```
// Say Hello LOTS of times
List<HelloWorld> hellos = new List<HelloWorld>();

for (int i=0; i< 1000; i++)
   hellos.Add(new HelloWorld() { instance = i, message = "Hello...is there an
echo in here?" });

qiclient.UpdateValues(hwStream.Id, hellos);
```

5th Kyu: **Hello World**


```
var helloWorlds =
qiclient.GetWindowValues<HelloWorld>(hwStream.Id, “1", "1000");

foreach(HelloWorld helloWorld in helloWorlds)
        Console.WriteLine(helloWorld.ToString());
```

```
Key = 976, Message = Hello...is there an echo in here?
Key = 977, Message = Hello...is there an echo in here?
Key = 978, Message = Hello...is there an echo in here?
Key = 979, Message = Hello...is there an echo in here?
Key = 980, Message = Hello...is there an echo in here?
Key = 981, Message = Hello...is there an echo in here?
Key = 982, Message = Hello...is there an echo in here?
Key = 983, Message = Hello...is there an echo in here?
Key = 984, Message = Hello...is there an echo in here?
Key = 985, Message = Hello...is there an echo in here?
Key = 986, Message = Hello...is there an echo in here?
Key = 987, Message = Hello...is there an echo in here?
Key = 988, Message = Hello...is there an echo in here?
Key = 989, Message = Hello...is there an echo in here?
Key = 990, Message = Hello...is there an echo in here?
Key = 991, Message = Hello...is there an echo in here?
Key = 992, Message = Hello...is there an echo in here?
Key = 993, Message = Hello...is there an echo in here?
Key = 994, Message = Hello...is there an echo in here?
Key = 995, Message = Hello...is there an echo in here?
Key = 996, Message = Hello...is there an echo in here?
Key = 997, Message = Hello...is there an echo in here?
Key = 998, Message = Hello...is there an echo in here?
Key = 999, Message = Hello...is there an echo in here?
```

5th Kyu: **Hello World**


```
helloWorlds = qiclient.GetWindowValues<HelloWorld>(hwStream.Id, “1", "1000",
QiBoundaryType.Inside, "instance mod 2 eq 0");

foreach (HelloWorld helloWorld in helloWorlds)
     Console.WriteLine(helloWorld.ToString());
```

5th Kyu: **Hello World**

Hello World 1 2 3 4 5 6 7 8 9

OSIsoft **Qi**

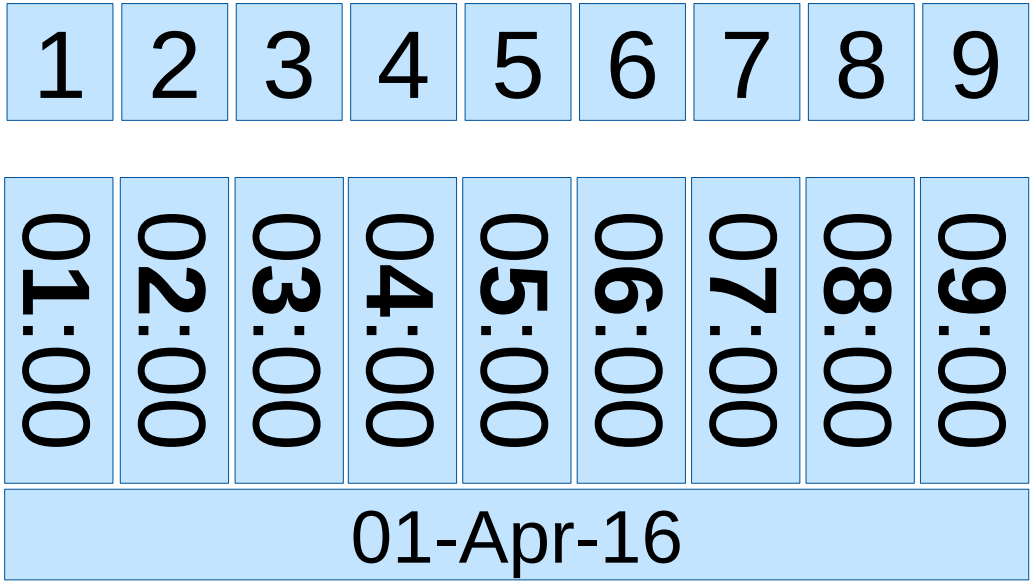
```
Key = 952, Message = Hello...is there an echo in here?
Key = 954, Message = Hello...is there an echo in here?
Key = 956, Message = Hello...is there an echo in here?
Key = 958, Message = Hello...is there an echo in here?
Key = 960, Message = Hello...is there an echo in here?
Key = 962, Message = Hello...is there an echo in here?
Key = 964, Message = Hello...is there an echo in here?
Key = 966, Message = Hello...is there an echo in here?
Key = 968, Message = Hello...is there an echo in here?
Key = 970, Message = Hello...is there an echo in here?
Key = 972, Message = Hello...is there an echo in here?
Key = 974, Message = Hello...is there an echo in here?
Key = 976, Message = Hello...is there an echo in here?
Key = 978, Message = Hello...is there an echo in here?
Key = 980, Message = Hello...is there an echo in here?
Key = 982, Message = Hello...is there an echo in here?
Key = 984, Message = Hello...is there an echo in here?
Key = 986, Message = Hello...is there an echo in here?
Key = 988, Message = Hello...is there an echo in here?
Key = 990, Message = Hello...is there an echo in here?
Key = 992, Message = Hello...is there an echo in here?
Key = 994, Message = Hello...is there an echo in here?
Key = 996, Message = Hello...is there an echo in here?
Key = 998, Message = Hello...is there an echo in here?
```

4th Kyu: **Time Series Data**

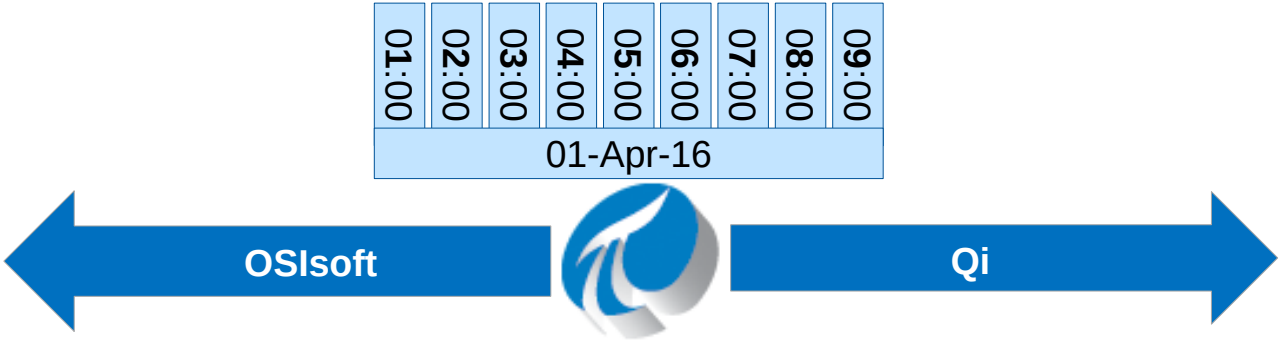
- Store some Time Series data in the same way as Hello World.
- Store future data too.
- Look at some variations of Time Series data.
- Build a utility to scrape BMRS (UK Power Data) data from the web.

4th Kyu: **Time Series Data**

Hello World

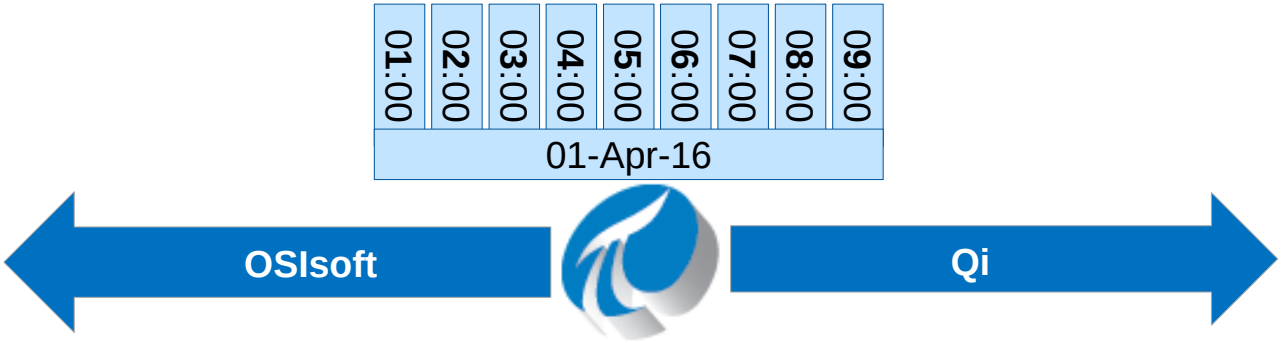


4th Kyu: **Time Series Data**




```
class HelloWorldByTime
    {
        [Key]
        public DateTime received { get; set; }
        public string message { get; set; }
}
```

4th Kyu: **Time Series Data**

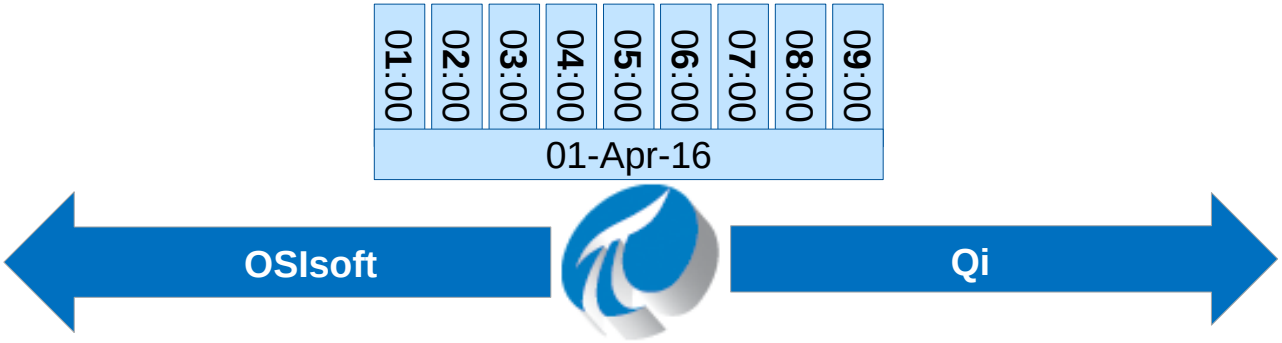



```
List<HelloWorldByTime> hellos = new List<HelloWorldByTime>();

for (int i = 0; i < 1000; i++)
        hellos.Add(new HelloWorldByTime()
{ received = DateTime.Now.AddHours(i), message = "Hello, what time is it?" });

qiclient.UpdateValues(hwStream.Id, hellos);
```

4th Kyu: **Time Series Data**

```
QiType TGG.TYPES.HelloWorldByTime created.
QiStream OSIsoft.Qi.QiStream created with QiType TGG.TYPES.HelloWorldByTime.
HelloWorld object sent to Qi.
HelloWorlds retrieved = 2

Received = 23/03/2016 11:39:21, Message = Hello wonderful world of Qi
Received = 23/03/2016 11:40:02, Message = Hello wonderful world of Qi

Just shouted Hello at Qi 1000 times.

HelloWorlds retrieved = 1001
```

4th Kyu: **Time Series Data**



Time Series

```
helloWorlds = qiclient.GetWindowValues<HelloWorldByTime>(hwStream.Id,
"01-Apr-2016", "08-Apr-2016");
```

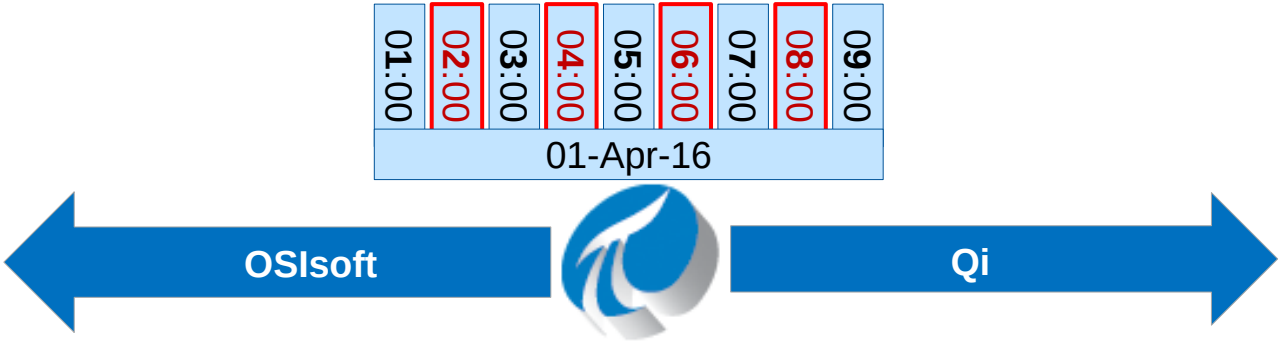
By Integer Sequence

```
helloWorlds = qiclient.GetWindowValues<HelloWorld>(hwStream.Id,
"0", "1000");
```

4th Kyu: **Time Series Data**

```
Received = 03/05/2016 12:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 13:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 14:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 15:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 16:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 17:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 18:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 19:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 20:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 21:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 22:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 23:40:31, Message = Hello...is there an echo in here?
Received = 04/05/2016 00:40:31, Message = Hello...is there an echo in here?
```

4th Kyu: **Time Series Data**



Time Series

```
helloWorlds = qiclient.GetWindowValues<HelloWorldByTime>(hwStream.Id,
"01-Apr-2016", “08-Apr-2016", QiBoundaryType.Inside,
"hour(received) mod 2 eq 0");
```

By Integer Sequence

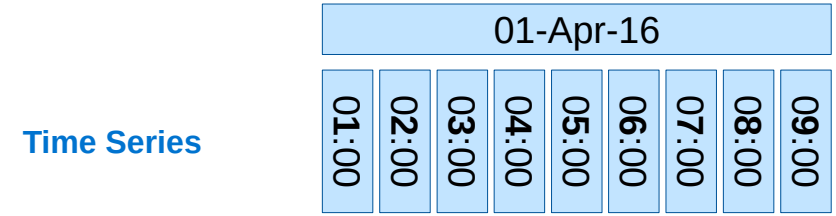
```
helloWorlds = qiclient.GetWindowValues<HelloWorld>(hwStream.Id, “1",
"1000", QiBoundaryType.Inside, "instance mod 2 eq 0");
```

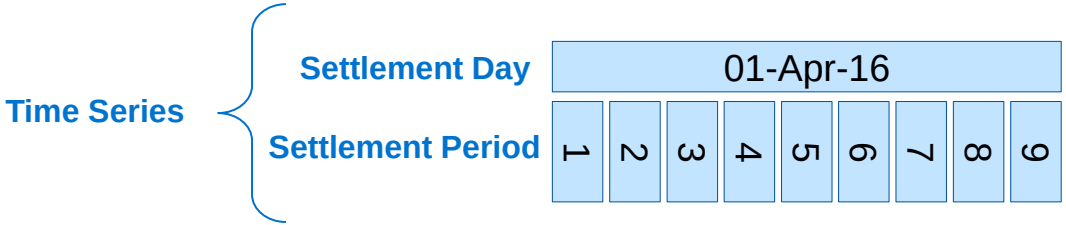
4th Kyu: **Time Series Data**

```
Received = 03/05/2016 22:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 23:40:31, Message = Hello...is there an echo in here?
Received = 04/05/2016 00:40:31, Message = Hello...is there an echo in here?

(Even) HelloWorlds retrieved = 500
```

```
Received = 02/05/2016 16:40:31, Message = Hello...is there an echo in here?
Received = 02/05/2016 18:40:31, Message = Hello...is there an echo in here?
Received = 02/05/2016 20:40:31, Message = Hello...is there an echo in here?
Received = 02/05/2016 22:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 00:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 02:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 04:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 06:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 08:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 10:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 12:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 14:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 16:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 18:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 20:40:31, Message = Hello...is there an echo in here?
Received = 03/05/2016 22:40:31, Message = Hello...is there an echo in here?
Received = 04/05/2016 00:40:31, Message = Hello...is there an echo in here?
```

4th Kyu: **Time Series Data**

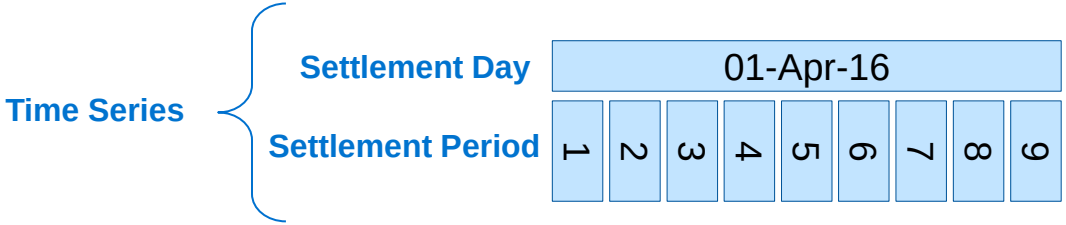




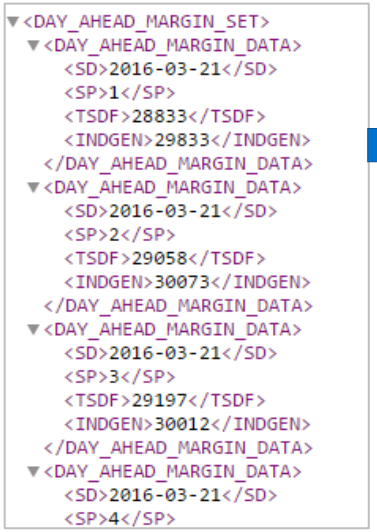
UK Power Network

Each day is divided up into 48 half hour settlement period.

4th Kyu: **Time Series Data**



4th Kyu: **Time Series Data**



```
▼<DAY_AHEAD_MARGIN_SET>
 ▼<DAY_AHEAD_MARGIN_DATA>
    <SD>2016-03-21</SD>
    <SP>1</SP>
    <TSDF>28833</TSDF>
    <INDGEN>29833</INDGEN>
  </DAY_AHEAD_MARGIN_DATA>
 ▼<DAY_AHEAD_MARGIN_DATA>
    <SD>2016-03-21</SD>
    <SP>2</SP>
    <TSDF>29058</TSDF>
    <INDGEN>30073</INDGEN>
  </DAY_AHEAD_MARGIN_DATA>
 ▼<DAY_AHEAD_MARGIN_DATA>
    <SD>2016-03-21</SD>
    <SP>3</SP>
    <TSDF>29197</TSDF>
    <INDGEN>30012</INDGEN>
  </DAY_AHEAD_MARGIN_DATA>
 ▼<DAY_AHEAD_MARGIN_DATA>
    <SD>2016-03-21</SD>
    <SP>4</SP>
```

```
public class DayAheadMarginData
   {
        [Key]
        public DateTime SD { get; set; }
        [Key]
        public int SP { get; set; }
        public int TSDF { get; set; }
        public int INDGEN { get; set; }
}
```

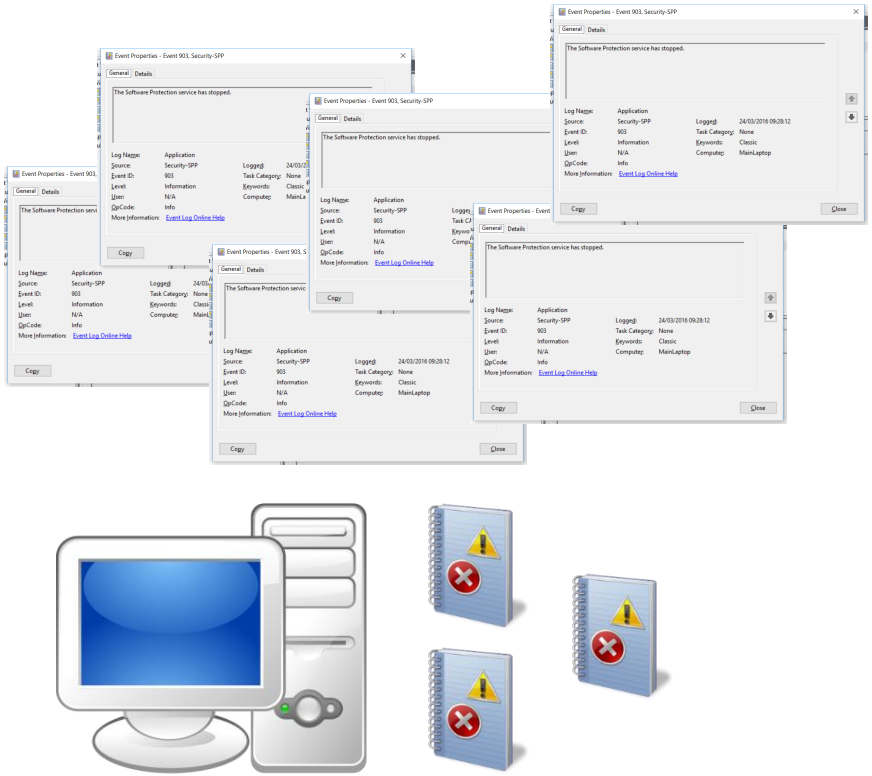
```
[XmlRootAttribute("DAY_AHEAD_MARGIN_SET")]
    public class DayAheadMarginDataCollection
    {
        [XmlElement("DAY_AHEAD_MARGIN_DATA")]
        public DayAheadMarginData[] AllDayAheadMarginData { get; set; }
    }
```

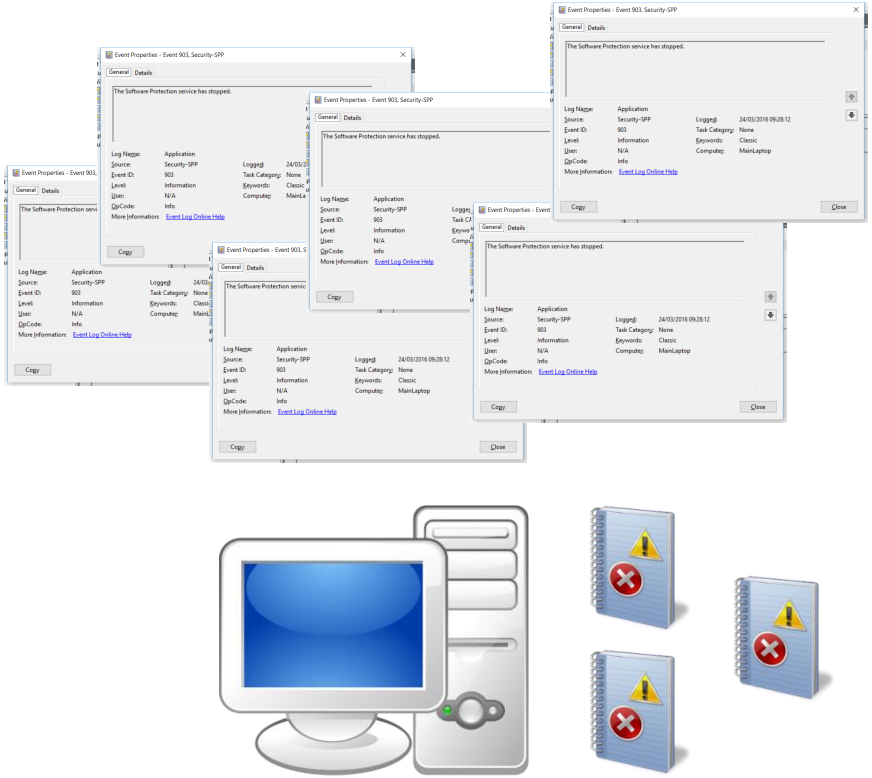
```
XmlTextReader reader; XmlSerializer serializer;
reader = new XmlTextReader("http://www.bmreports.com/bsp/additional/soapfunctions.php?element=ddam&submit=Invoke");
serializer = new XmlSerializer(typeof(DayAheadMarginDataCollection));
var damdCollection = ((DayAheadMarginDataCollection)serializer.Deserialize(reader)).AllDayAheadMarginData;

evtType = typeBuilder.Create<DayAheadMarginData>();
evtType.Id = "TieQi.Type." + typeof(DayAheadMarginData).Name;
evtType = QiClient.GetOrCreateType(evtType);
```

4th Kyu: **Time Series Data**

```
streamName = "TieQi.Stream." + typeof(DayAheadMarginData).Name;
evtStream = new QiStream();
evtStream.Id = streamName;
evtStream.Name = streamName;
evtStream.TypeId = evtType.Id;
evtStream = QiClient.GetOrCreateStream(evtStream);

if (damdCollection != null)
{
    if (damdCollection.Count() > 0)
    {
        try
        { QiClient.UpdateValues(streamName, damdCollection); }
        catch (QiQueryException QiError)
        { string err = QiError.Message; }
        catch (Exception e)
        { string err = e.Message; }
    }
}
```

3rd Kyu: **Complex Data Types**

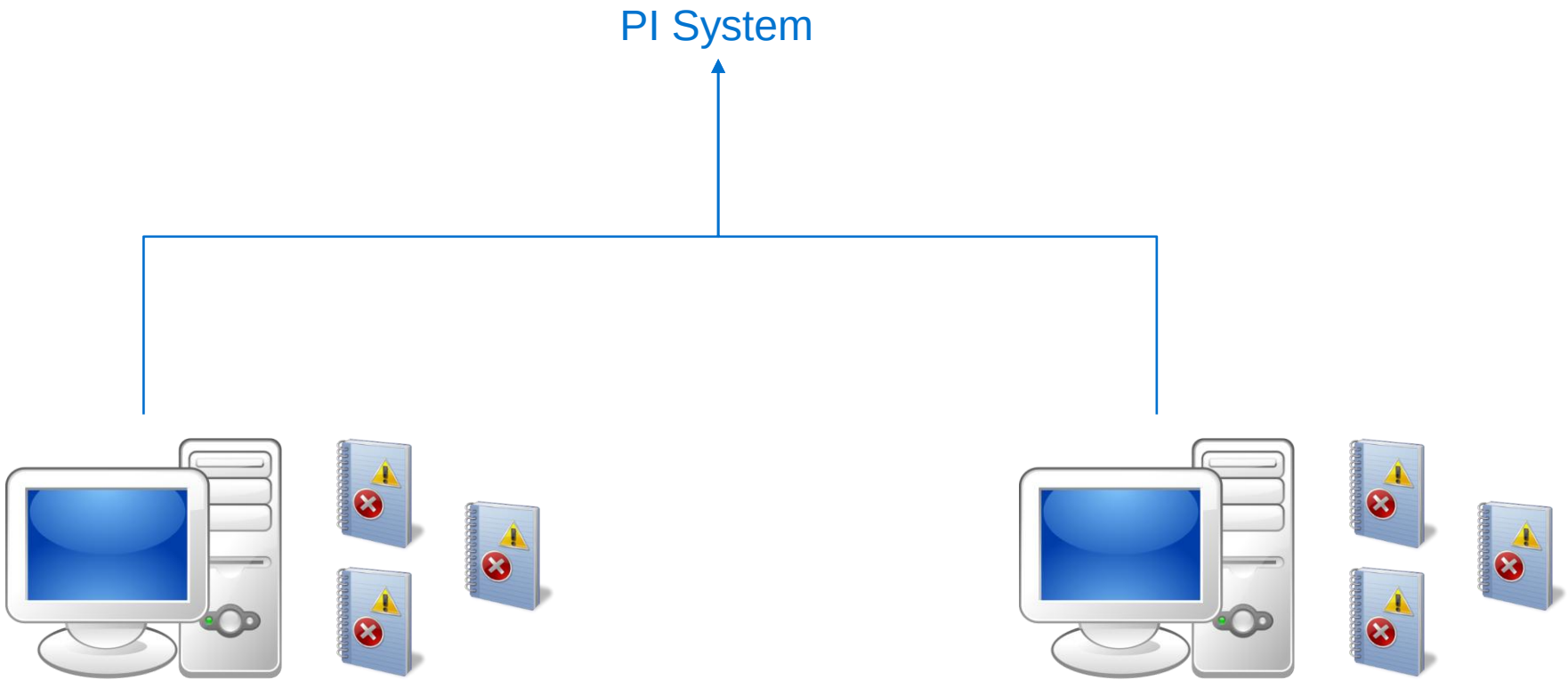




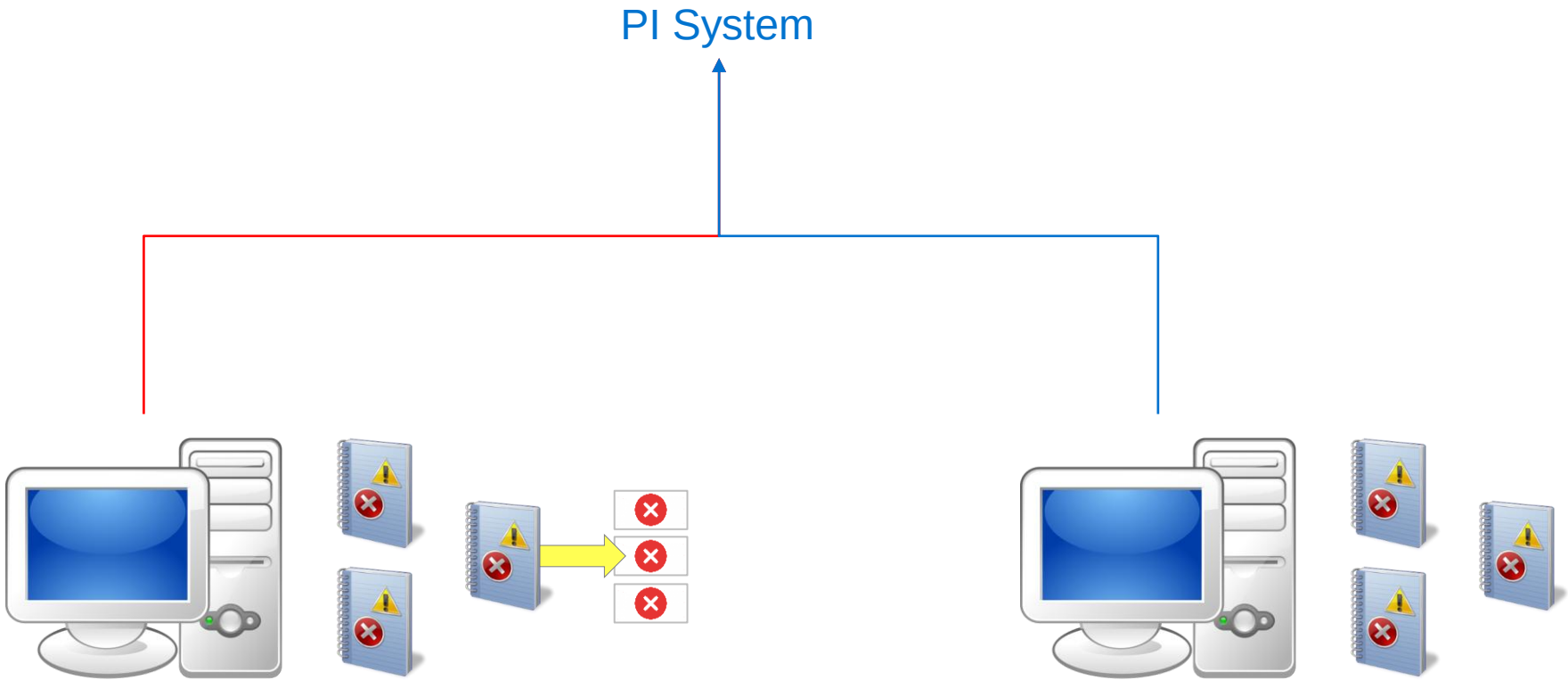
3rd Kyu: **Complex Data Types**

- Define a class with a compound index.
- Build some code to scrape an Event Log.
- Put all those Event Log Entries into Qi.
- Have a look through those Events via Qi.
- Find some interesting Events via Qi.

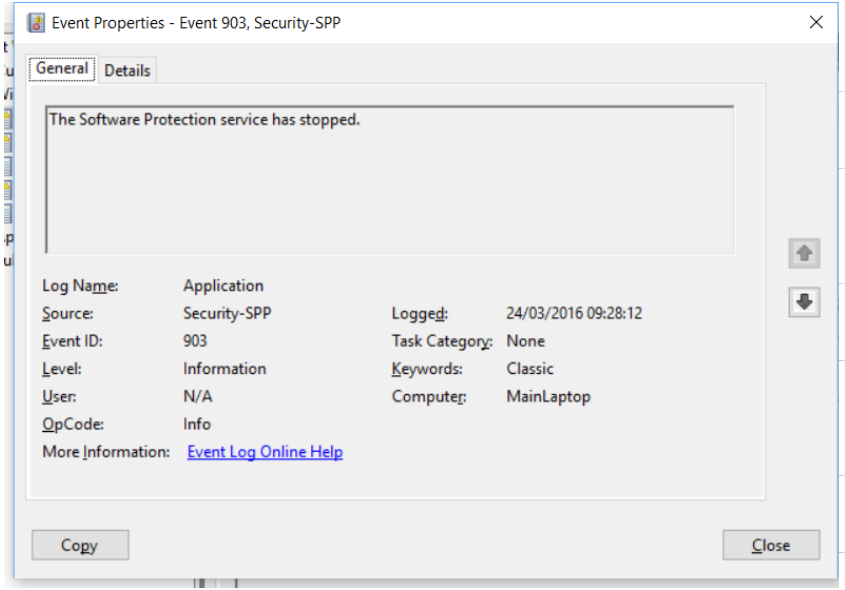
3rd Kyu: **Complex Data Types**



3rd Kyu: **Complex Data Types**



3rd Kyu: **Complex Data Types**




```
class ScrapedEventLogEntry
   {
        public string Category { get; set; }
        public short CategoryNumber { get; set; }
        public byte[] Data { get; set; }
        public EventLogEntryType EntryType { get; set; }
        [Key]
        public int Index { get; set; }
        public long InstanceId { get; set; }
        public string MachineName { get; set; }
        public string Message { get; set; }
        public string[] ReplacementStrings { get; set; }
        public string Source { get; set; }
        [Key]
        public DateTime TimeGenerated { get; set; }
        public DateTime TimeWritten { get; set; }
        public string UserName { get; set; }
        public Guid ClientId { get; set; }
```

3rd Kyu: **Complex Data Types**

```
class EventLogScraper
    {
        public static List<ScrapedEventLogEntry> ScrapeApplicationLog()
        {
            List<ScrapedEventLogEntry> entries = new List<ScrapedEventLogEntry>();

            Guid clientId = Guid.Parse("C40B9E52-81DA-44C3-91B3-DB478FA6D2C9");
            EventLog appLog = new EventLog("Application");
            foreach(EventLogEntry appEntry in appLog.Entries)
                entries.Add(new ScrapedEventLogEntry(appEntry, clientId));

            return entries;
        }

    }
```

3rd Kyu: **Complex Data Types**

Why couldn’t “Bob the QiType Builder” use the EventLogEntry class…

- Event Log Entry class inherits from Component. Helper properties will cause a circular reference.
- EventLogEntry class is a sealed class (no public constructor), which will cause issues during deserialization.
- I needed some extra properties, such as ClientId, to store in my QiType.

3rd Kyu: **Complex Data Types**


```
// Employ Bob
QiTypeBuilder bob = new QiTypeBuilder();

// Create the QiType
QiType logType = bob.Create<ScrapedEventLogEntry>();
logType.Id = "TGG.TYPES." + typeof(ScrapedEventLogEntry).Name;
logType.Name = logType.Id;
logType = qiclient.GetOrCreateType(logType);

// Create the QiStream
QiStream logStream = new QiStream() { Id = "TGG.STREAMS." +
typeof(ScrapedEventLogEntry).Name, TypeId = logType.Id };
logStream = qiclient.GetOrCreateStream(logStream);
```

```
QiType TGG.TYPES.ScrapedEventLogEntry created.
QiStream TGG.STREAMS.CLIENTID.ScrapedEventLogEntry created with QiType TGG.TYPES.ScrapedEventLogEntry.
```

3rd Kyu: **Complex Data Types**


```
List<ScrapedEventLogEntry> entries = EventLogScraper.ScrapeApplicationLog();

Console.WriteLine("Just received " + entries.Count + " Event Log Entries from
the Application Log.");
Console.WriteLine("Now I am going to give them to Qi...");

qiclient.UpdateValues(logStream.Id, cutentries);

Console.WriteLine("...done.");
```

```
QiType TGG.TYPES.ScrapedEventLogEntry created.
QiStream TGG.STREAMS.CLIENTID.ScrapedEventLogEntry created with QiType TGG.TYPES.ScrapedEventLogEntry.
Just receieved 24146 Event Log Entries from the Application Log.
Now I am going to give them to Qi...
...done.
```

3rd Kyu: **Complex Data Types**

Did I write my code any different to my simple index example, or my time series data example?

No.

QiType => QiStream => UpdateValues()

3rd Kyu: **Complex Data Types**


```
[Key]
public int Index { get; set; }
[Key]
public DateTime TimeGenerated { get; set; }
```

```
var startIndex = new Tuple<int, DateTime>(0, DateTime.Now.AddDays(-1));
var endIndex = new Tuple<int, DateTime>(50000, DateTime.Now);

var logEntries = qiclient.GetWindowValues<ScrapedEventLogEntry, int,
DateTime>(logStream.Id, startIndex, endIndex);
```

3rd Kyu: **Complex Data Types**


```
QiType TGG.TYPES.ScrapedEventLogEntry created.
QiStream TGG.STREAMS.CLIENTID.ScrapedEventLogEntry created with QiType TGG.TYPES.ScrapedEventLogEntry.
Just receieved 24146 Event Log Entries from the Application Log.
Now I am going to give them to Qi...
...done.

Just received 500 from Qi.
```

```
Time = 24-Mar-2016 09:27:44, Message = Reconciliation completed for the following store: C:\Users\aaa\Documents\Outlook Files\rhys@rjk.solutions.pst. Stat
: 276, Version: 15.0.4805.1000.
Time = 24-Mar-2016 09:27:52, Message = Reconciliation completed for the following store: C:\Users\aaa\AppData\Local\Microsoft\Outlook\rhys@thegeniusgroup.
ed: 2, Compared: 1126, Version: 15.0.4805.1000.
Time = 24-Mar-2016 09:28:12, Message = Successfully scheduled Software Protection service for re-start at 2016-03-25T09:16:12Z. Reason: RulesEngine.
Time = 24-Mar-2016 09:28:12, Message = The Software Protection service has stopped.
```

3rd Kyu: **Complex Data Types**


```
var logLastValue = qiclient.GetLastValue<ScrapedEventLogEntry>(logStream.Id);

Console.WriteLine("Last scraped event log entry = " + logLastValue.ToString());
```

```
Last scraped event log entry = Time = 24-Mar-2016 11:45:50, Message = ANManager.OnAbort[]: Unexpected error during operation. Terminating the PI Analysis
vice.
```

3rd Kyu: **Complex Data Types**


```
var logLastValue = qiclient.GetLastValue<ScrapedEventLogEntry>(logStream.Id);

Console.WriteLine("Last scraped event log entry = " + logLastValue.ToString());
```

```
Last scraped event log entry = Time = 24-Mar-2016 11:45:50, Message = ANManager.OnAbort[]: Unexpected error during operation. Terminating the PI Analysis
vice.
```

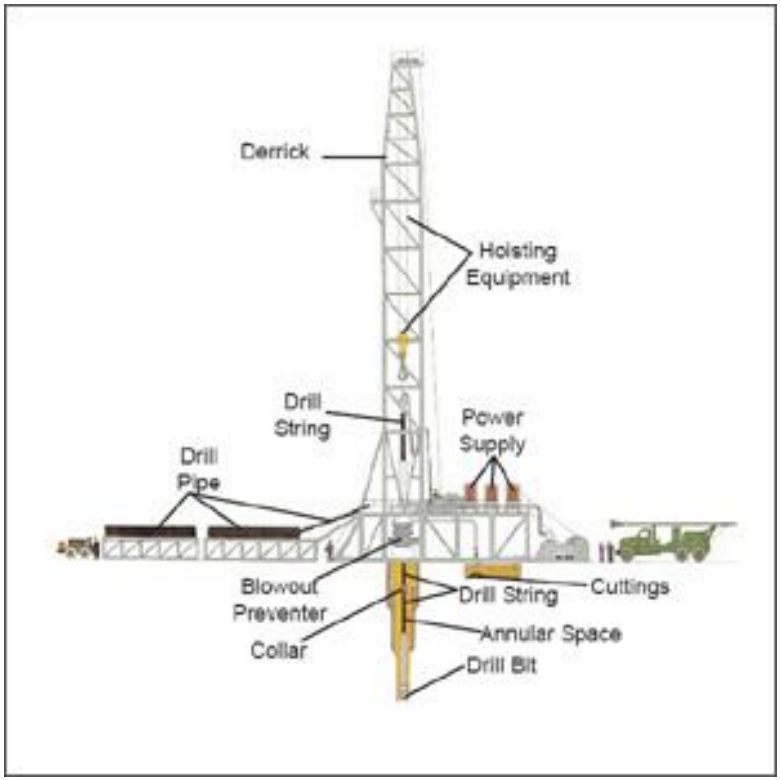
3rd Kyu: **Complex Data Types**


```
logEntries = qiclient.GetWindowValues<ScrapedEventLogEntry, int,
DateTime>(logStream.Id, startIndex, endIndex, QiBoundaryType.Inside,
"Source eq 'PI Analysis Service' AND EntryType eq 1");

Console.WriteLine("Issues with PI Analysis Service detected = " +
logEntries.Count());
```

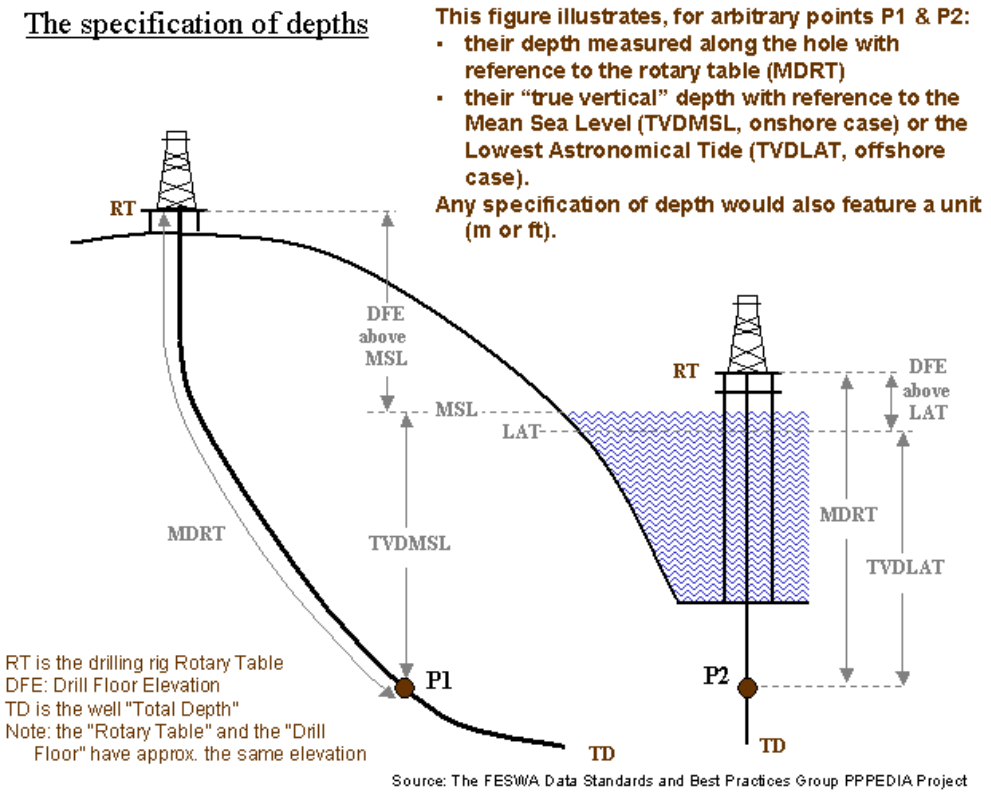
3rd Kyu: **Complex Data Types**


```
Issues with PI Analysis Service detected = 2
Time = 24-Mar-2016 11:45:50, Message = ANServiceThread.RunControlWorkflow[]:
System.ServiceModel.CommunicationException: Cannot connect to server 'MAINLAPTOP'. ---> System.InvalidOperationException: The PI AF
 by the server name being incorrect, the SQL services not being running, or the SQL network protocols not being enabled.
   at OSIsoft.AF.PISystem.CheckServerConnectError(PISystem system, Int32 errNumber)
   at OSIsoft.AF.Support.AFProxy.CheckGetSystemVersions(Boolean isSQLDatabaseCompatible, Int32 errNumber, Int32[] compatibilityNumbe
   at OSIsoft.AF.Support.AFSerialProxy.GetSystemVersions(Int32[]& compatibilityNumbers, String[]& versionStrings)
   at OSIsoft.AF.Support.AFProxy.Reconnect(Boolean autoPrompt, Boolean raiseEvents, AFConnectionProtocol protocol, String host, Int3
   --- End of inner exception stack trace ---
   at OSIsoft.AF.Support.AFProxy.Reconnect(AFCollectiveMember member, Boolean autoPrompt, Boolean raiseEvents, AFConnectionPreferenc
   at OSIsoft.AF.Support.AFProxy.Reconnect()
   at OSIsoft.AF.Support.AFSerialProxy.Call(String rpcName, ProxyDelegate codeBlock)
   at OSIsoft.AF.Support.AFSerialProxy.GetDatabaseList(DateTime sinceTime, AFSortField sortField, AFSortOrder sortOrder, Int32 maxCo
   at OSIsoft.AF.AFDatabases.LoadObjects(Int32 page, Boolean fullReload)
   at OSIsoft.AF.AFCollection`1.Load(Boolean force)
   at OSIsoft.AF.AFCollection`1.GetCount(AFCollectionMode& mode)
   at OSIsoft.AF.PISystem.get_ConfigurationDatabase()
   at OSIsoft.AN.ANGlobalConfigurationStore.TryGetStoreRootElement(AFElement& storeRootElement, Boolean permitRefresh)
   at OSIsoft.AN.ANGlobalConfigurationStore.TryGetValue(String propertyName, String& propertyValue)
   at OSIsoft.AN.Service.ANServiceConfigurationManager.LoadConfigurationFromStore()
   at OSIsoft.AN.Service.ANManagerConfiguration.GetServiceConfigurationManager()
   at OSIsoft.AN.Service.ANManager.OnStart()
   at OSIsoft.AN.ANServiceThread.RunControlWorkflow()
Time = 24-Mar-2016 11:45:50, Message = ANManager.OnAbort[]: Unexpected error during operation. Terminating the PI Analysis service.
```

2nd Kyu: **Depth Indexed Data**



2nd Kyu: **Depth Indexed Data**



WITS0

Time Cohesion

Depth Index

PI System

WITSML

Replication

2nd Kyu: **Depth Indexed Data**

- Build a Depth Log class.
- Store the depth data in Qi.
- Query the depth data in a numerous ways directly in Qi.
- Day dream about how Qi could be a WITSML data stream.

2nd Kyu: **Depth Indexed Data**


```
<log uidWell="W-12" uidWellbore="B-01" uid="f34a">
  <nameWell>6507/7-A-42</nameWell>
  <nameWellbore>A-42</nameWellbore>
  <name>L001</name>
  <serviceCompany>Baker Hughes INTEQ</serviceCompany>
  <runNumber>12.3</runNumber>
  <creationDate>2001-06-18T13:20:00.000</creationDate>
  <description>Drilling Data Log</description>
  <indexType>measured depth</indexType>
  <startIndex uom="m">499</startIndex>
  <endIndex uom="m">509.01</endIndex>
  <stepIncrement uom="m">0</stepIncrement>
  <direction>increasing</direction>
  <indexCurve columnIndex="1">Mdepth</indexCurve>
  <nullValue>-999.25</nullValue>
  <logParam index="1" name="MRES" uom="ohm.m" description="Mud Resistivity">1.25</logPar
  <logParam index="2" name="BDIA" uom="in" description="Bit Diameter">12.25</logParam>
  <logCurveInfo uid="lci-1">
    <mnemonic>Mdepth</mnemonic>
    <classWitsml>measured depth of hole</classWitsml>
    <unit>m</unit>
    <mnemAlias>md</mnemAlias>
    <nullValue>-999.25</nullValue>
    <minIndex uom="m">499</minIndex>
    <maxIndex uom="m">509.01</maxIndex>
    <columnIndex>1</columnIndex>
    <curveDescriptio
    <sensorOffset uo
    <traceState>raw<
    <typeLogData>dou
  </logCurveInfo>
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logCurveInfo uid=
  <logData>
    <data>
      499,498.99,1.25,0,1.45,3.67,11.02,187.66,0.29,116.24,0.01,0.05,0.01,0,886.03,1089.99,1.11,14.67,0.29,1.12,1.11
    </data>
    <data>
      500.01,500,1.9,0.01,1.42,9.94,11.32,185.7,0.29,116.24,0.01,0.01,0.01,0,795.19,973.48,1.11,14.67,0.29,0.95,1.11
    </data>
    <data>
      501.03,501.02,2.92,0.02,1.41,20.46,11.62,184.23,0.29,120,0.01,0.01,0.01,0,796.68,956.25,1.11,14.67,0.29,0.83,1.11
    </data>
```



```
<description>Drilling Data Log</description>
<indexType>measured depth</indexType>
<startIndex uom="m">499</startIndex>
<endIndex uom="m">509.01</endIndex>
<stepIncrement uom="m">0</stepIncrement>
<direction>increasing</direction>
<indexCurve columnIndex="1">Mdepth</indexCurve>
<nullValue>-999.25</nullValue>
```



```
<logCurveInfo uid="lci-21">...</logCurveInfo>
<logData>
  <data>
    499,498.99,1.25,0,1.45,3.67,11.02,187.66,0.29,116.24,0.01,0.05,0.01,0,886.03,1089.99,1.11,14.67,0.29,1.12,1.11
  </data>
  <data>
    500.01,500,1.9,0.01,1.42,9.94,11.32,185.7,0.29,116.24,0.01,0.01,0.01,0,795.19,973.48,1.11,14.67,0.29,0.95,1.11
  </data>
  <data>
    501.03,501.02,2.92,0.02,1.41,20.46,11.62,184.23,0.29,120,0.01,0.01,0.01,0,796.68,956.25,1.11,14.67,0.29,0.83,1.11
  </data>
  <data>
```

2nd Kyu: **Depth Indexed Data**

```
class DepthLog
   {
       public string wellId { get; set; }
       public string wellBoreId { get; set; }
       [Key]
       public double measuredDepth { get; set; }
       [Key]
       public DateTime timeReceived { get; set; }
       public double trueVerticalDepth { get; set; }
       public double rateOfPenetration { get; set; }
       public double weightOnBit { get; set; }
       public double rpm { get; set; }
       public double bitRpm { get; set; }
       public double mudFlow { get; set; }
       public double mudTemp { get; set; }
       public double mudPressure { get; set; }
   }
```

- Need to know the Well being drilled.
- The Well Bore.
- The depth at which measurements are relevant.
- The time at which the depth was received.

2nd Kyu: **Depth Indexed Data**

```
public static List<DepthLog> RandomDepthData(string wellId, string wellBoreId, double startingDepth)
        {
            List<DepthLog> depths = new List<DepthLog>();

            Random r = new Random();
            for (int i = 0; i<= 200; i++)
            {
                DepthLog d = new DepthLog()
                {
                    wellId = wellId,
                    wellBoreId = wellBoreId,
                    measuredDepth = startingDepth + i + r.NextDouble(),
                    trueVerticalDepth = startingDepth - 1 + r.NextDouble(),
                    rateOfPenetration = r.NextDouble() * 10,
                    weightOnBit = r.NextDouble() * 100,
                    rpm = r.NextDouble() * 1000,
                    bitRpm = r.NextDouble() * 50,
                    mudFlow = r.NextDouble() * 5,
                    mudTemp = r.NextDouble() * 50,
                    mudPressure = r.NextDouble() * 7,
                    timeReceived = DateTime.Now
                };
                depths.Add(d);
            }
            return depths;
        }
```

2nd Kyu: **Depth Indexed Data**

```
 // Employ Bob
QiTypeBuilder bob = new QiTypeBuilder();

// Create the QiType
QiType depthType = bob.Create<DepthLog>();
depthType.Id = "TGG.TYPES.WITSML13." + typeof(DepthLog).Name;
depthType.Name = depthType.Id;
depthType = qiclient.GetOrCreateType(depthType);

// Create the QiStream
QiStream depthStream = new QiStream() { Id = "TGG.STREAMS.WITSML13.CLIENTID." + typeof(DepthLog).Name, TypeId =
depthType.Id };
depthStream = qiclient.GetOrCreateStream(depthStream);
```

```
QiType TGG.TYPES.WITSML13.DepthLog created.
QiStream TGG.STREAMS.WITSML13.CLIENTID.DepthLog created with QiType TGG.TYPES.WITSML13.DepthLog.
Just receieved 201 depths from Drilling Rig 01
Now I am going to give them to Qi...
...done.
```

2nd Kyu: **Depth Indexed Data**

```
String wellName = "Well-001";
String[] wellBores = { "Wellbore-001", "Wellbore-002" };
List<DepthLog> depths001 = Drilling.RandomDepthData(wellName, wellBores[0], 500);
```

```
var startIndex = new Tuple<double, DateTime>(450, DateTime.MinValue);
var endIndex = new Tuple<double, DateTime>(1250, DateTime.MaxValue);

var depthEntries = qiclient.GetWindowValues<DepthLog, double,
DateTime>(depthStream.Id, startIndex, endIndex);
```

2nd Kyu: **Depth Indexed Data**

```
Just received 201 from Qi.
500.229074969063,501.064891116724,502.540300810496,503.637438149023,504.593885155206,505.253359684839,506.005838147833,507.525106757192,508.019556798981,509
648,512.34249950356,513.791143734377,514.590270922329,515.387975444732,516.222235683455,517.644486059735,518.953537826405,519.70687032943,520.292942689868,5
772,524.155064383594,525.160516298451,526.250947065768,527.156432032658,528.583514318142,529.546221330085,530.594962682386,531.989204388107,532.995780984869
5135926,536.663803084131,537.336311708361,538.119310760461,539.789482048615,540.249151345924,541.576078800753,542.073734296986,543.431991289571,544.91253302
16444266362,548.584998660062,549.649073569406,550.571507688878,551.832248657398,552.404249598926,553.079084854144,554.841660733727,555.118956495132,556.94904
59.937268787966,560.8476039408,561.048048252728,562.513461169095,563.836162082775,564.652346411092,565.080104563423,566.851333530085,567.864987822187,568.875
,571.641810898968,572.553475328048,573.120694142357,574.735392340801,575.987481184298,576.070326671968,577.13372963906,578.580659625856,579.604476576487,580
683,583.06922746127,584.688706809044,585.770241660425,586.908476688856,587.244895719571,588.862849462713,589.152521052935,590.431297588363,591.004987298048,5
67859,595.515813136248,596.222634352382,597.77607352509,598.557135802487,599.016688315206,600.849217725847,601.948877933411,602.940877891118,603.156010228748
31403501,607.13688438299,608.197196377533,609.203125923501,610.756230184695,611.464219021361,612.990510973144,613.352949761484,614.561655293015,615.521044468
3951269213,619.628743509589,620.443321787493,621.449898166792,622.299777169386,623.447595079638,624.650909641595,625.419109337227,626.092563503465,627.920785
0.096151618332,631.663091008395,632.379537570001,633.704926253159,634.012876656844,635.120865378119,636.094548888549,637.392062395994,638.917436377573,639.35
4,642.566230761617,643.196344934961,644.972742416883,645.088855884545,646.658762307213,647.718201089054,648.215293937929,649.90048463638,650.264891638544,651
8428,654.293861082426,655.589545809007,656.084533249067,657.016226638582,658.284406078181,659.619366210242,660.954735572429,661.116930928136,662.865040770203
62413394,666.910472475882,667.803587742058,668.063780739002,669.747296414686,670.494271263245,671.953482013174,672.242177767326,673.222459062106,674.62611504
570180985876,678.94224008822,679.635792421007,680.712746064976,681.023438958928,682.819812888661,683.188481309539,684.387914290832,685.231857205383,686.24928
89.731920153243,690.067487221708,691.681403480322,692.297571852942,693.634323266165,694.236168429365,695.189166269819,696.118515926003,697.39142172243,698.03
7,
```

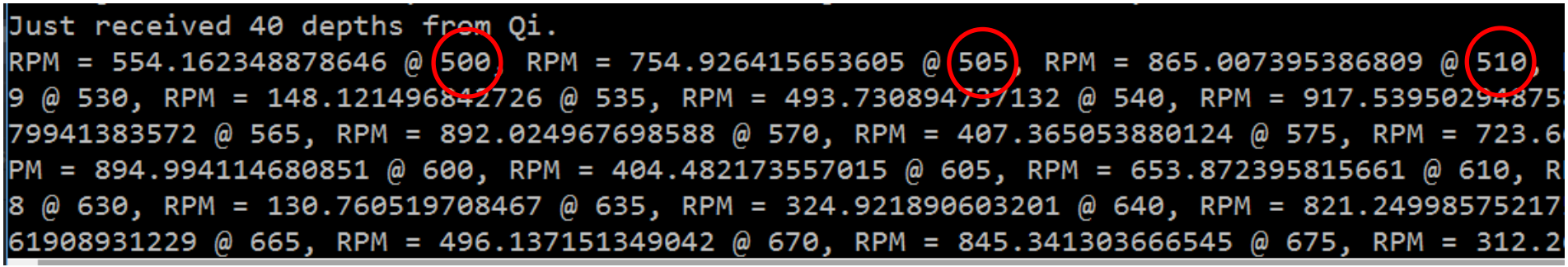
2nd Kyu: **Depth Indexed Data**

```
startIndex = new Tuple<double, DateTime>(500, DateTime.MinValue);
endIndex = new Tuple<double, DateTime>(540, DateTime.MaxValue);

depthEntries = qiclient.GetWindowValues<DepthLog, double,
DateTime>(depthStream.Id, startIndex, endIndex, QiBoundaryType.Inside,
"wellId eq '" + wellName + "' AND wellBoreId eq '" + wellBores[0] + "'");
```

```
Just received 40 depths from Qi.
RPM = 554.162348878646 @ 500.229074969063, RPM = 778.129657161482 @ 501
.593885155206, RPM = 696.313446246234 @ 505.253359684839, RPM = 595.709
533176231 @ 509.166618236418, RPM = 758.818988575982 @ 510.680529594738
 RPM = 551.805983554481 @ 514.590270922329, RPM = 420.523585481813 @ 51
8.953537826405, RPM = 12.229472870114 @ 519.70687032943, RPM = 129.3296
809009 @ 523.01392733772, RPM = 201.75524531014 @ 524.155064383594, RPM
 = 31.3827535283671 @ 528.583514318142, RPM = 181.402012790275 @ 529.54
5780984869, RPM = 503.51042230777 @ 533.892162854267, RPM = 290.7421329
9752 @ 537.336311708361, RPM = 941.41022299482 @ 538.119310760461, RPM
```

2nd Kyu: **Depth Indexed Data**

```
startIndex = new Tuple<double, DateTime>(500, DateTime.MinValue);
endIndex = new Tuple<double, DateTime>(700, DateTime.MinValue);

var depthIntervals = qiclient.GetIntervals<DepthLog, double, DateTime>
(depthStream.Id, startIndex, endIndex, 40);
```



2nd Kyu: **Depth Indexed Data**

```
depthIntervals = qiclient.GetIntervals<DepthLog, double, DateTime>
(depthStream.Id, startIndex, endIndex, 40, "rpm gt 800");
```

2nd Kyu: **Depth Indexed Data**

- Using OSIsoft Qi for such depth indexed projects would have been a breeze.

- Qi can store all the depth data and provide quick mechanisms for extracting just the required data.

- Different facades can be built on top of Qi to serve up data in a variety of protocols.

QiStreamBehavior

The QiStreamBehavior object determines how data-read operations are performed when an index to be read falls between, before, or after stream data in the stream.

Interpolation

When read methods affected by QiStreamBehavior (as shown above) are given an index that occurs between two values in a stream, the Mode object determines which values are retrieved.

QiStreamBehavior - Interpolation

Mode	Operation	PI Comparison
Default	Continuous	
Continuous	Interpolates the data using previous and next index values	
StepwiseContinuousLeading	Returns the data from the previous index	PrevEvent(…)
StepwiseContinuousTrailing	Returns the data from the next index	NextEvent(…)
Discrete	Returns ‘null’	ExactTime(…)

QiStreamBehavior - Extrapolation

In addition to interpolation settings, stream behavior is also used to define how the stream extrapolates data. ExtrapolationMode acts as a master switch to determine whether extrapolation occurs and at which end of the data. When defined, ExtrapolationMode works with the Mode to determine how a stream responds to requests for an index that precedes or follows all of the data in the stream.

ExtrapolationMode with *Mode*=StepwiseContinuousLeading

ExtrapolationMode	Enumeration value	Index before data	Index after data
All	0	Returns first data value	Returns last data value
None	1	Return null	Return null
Forward	2	Returns first data value	Return null
Backward	3	Return null	Returns last data value

1st Kyu: **Advanced OSIsoft Qi**

- Overriding the behaviour mode of individual properties of a QiType.
- Streaming high volume data into Qi.
- Building connectors to visualization tools & data analysis tools.
- Building stores / facades on top of Qi to serve up data in other protocols (e.g. WITSML).
- Studying even more to get closer to that elusive Qi Black Belt.

https://qi.osisoft.com/ QiSupport@osisoft.com

Contact Information

Speaker's Name

rhys@thegeniusgroup.co.uk

CEO and Founder

The Genius Group

Laurent Garrigues

lgarrigues@osisoft.com

SaaS Program Manager

OSIsoft, LLC

Questions

Please wait for the **microphone** before asking your questions

State your **name & company**

Please remember to…

Complete the Online Survey for this session

http://ddut.ch/osisoft

감사합니다

谢谢

Danke

Merci

Gracias

Thank You

ありがとう

Спасибо

Obrigado

OSIsoft®

USERS CONFERENCE 2016

April 4-8, 2016 | San Francisco

TRANSFORM YOUR WORLD