Instruction Manual

Exaquantum API Reference Manual

IM 36J04A14-01E

YOKOGAWA

IM 36J04A14-01E

25th Edition Issue 2

Highlights

The Highlights section gives details of the changes made since the previous issue of this document.

Summary of Changes

This is the 25th Edition of the document.

Detail of Changes

The changes are as follows.

Chapter/Section/Page	**Change**
Chapter 26, Chapter 27	Added chapters 26 and 27 – Reading Exaquantum Data with C# and Python using the COM API
Chapter 28	Added chapter 28 – An introduction to OPC UA, requirements for connection and how to use UaExpert with it.
Chapter 29, Chapter 30	Added chapters 29 and 30 – Reading Exaquantum Data with C# and Python using the OPC UA API

Exaquantum Document Set

The documents available for Exaquantum are:

- **Exaquantum General Specification (GS 36J04A10-01E)**
- **Exaquantum Technical Information (TI 36J04A10-01E)**
- **Exaquantum/PIMS User's Manual (IM 36J04A11-01E)**
- **Exaquantum/Explorer User's Manual Volume 1 General Information (IM 36J04A12-01E)**
- **Exaquantum/Explorer User's Manual Volume 2 Custom Controls (IM 36J04A12-02E)**
- **Exaquantum/Explorer User's Manual Volume 3 Microsoft Excel Reports (IM 36J04A12-03E)**
- **Exaquantum/Explorer User's Manual Volume 4 Advanced Configuration (IM 36J04A12-04E)**
- **Exaquantum Installation Guide (IM 36J04A13-01E)**
- **Exaquantum API Reference Manual (IM 36J04A14-01E)**
- **Exaquantum Engineering Guide Volume 1 Administration (IM 36J04A15-01E)**
- **Exaquantum Engineering Guide Volume 2 Network Configuration (IM 36J04A15-02E)**
- **Exaquantum Engineering Guide Volume 3 Support Tools (IM 36J04A15-03E)**
- **Exaquantum Engineering Guide Volume 4 Web Authoring (IM 36J04A15-04E)**
- **Exaquantum Engineering Guide Volume 5 PI Connection (IM 36J04A15-05E)**

Table of Contents

This page intentionally left blank

Chapter 1 Introduction

This document serves as a reference document describing the Exaquantum Application Programming Interfaces (APIs).

Two types of Interface are provided:

- COM interface
- .NET interface.

The Summary section below describes what is available for each type.

Note: The COM interface and .NET interface are only supported for applications running as an x86 process. Applications that run on an x64 operating system must also be run as an x86 process under WOW64.

The COM Interfaces are standard COM Automation Interfaces and are available to Windows developers who develop applications in languages such as Microsoft Visual Basic (VB) and Microsoft Visual C++ (VC++).

The .NET Interfaces are available to any .NET 2.0 languages, such as Microsoft C# and Microsoft VB .NET.

Note: Prior to Exaquantum R2.60 the .NET Interfaces are available to any .NET 1.1 languages.

The definitions of the COM APIs given in this document are in a VB style, but the APIs are available to other languages including VC++.

This document is arranged in two broad sections, a section giving examples of each of the data exchanges available via the APIs and a reference section giving details of the APIs.

1.1. Summary

APIs are available for the following:

COM APIs:

- ♦ Reading the latest RTDB Item values in both a synchronous and asynchronous way
- ♦ Reading the history of RTDB Item values in both a synchronous and asynchronous way
- ♦ Writing to RTDB Item values
- ♦ Reading the history of Alarm and Event data in both a synchronous and asynchronous way.
- ♦ Reading and writing item values through RBNS (Role Based Namespace)
- ♦ Conducting recalculation and re-aggregation
- ♦ Setting Tags and Function Blocks offline

OPC Server API's:

- ♦ Tag Browsing
- ♦ Synchronous Reading and Writing of Live Data
- ♦ Asynchronous Reading and Writing of Live Data
- ♦ Synchronous Reading and Writing of History Data

NET APIs:

- ♦ Management of Production Calendar and Tag Templates
- ♦ Management of Tags and Folders

Chapter 2 Using the Exaquantum APIs

2.1. COM API

The Exaquantum COM APIs can be used on any PC that has either the Exaquantum Server or Client components installed.

Access to the APIs is via the ‘QuantumAutomation 1.0 Type Library’ installed as '…\Yokogawa\Exaquantum PIMS\System\QuantumAutomation.dll’. In VB first make a reference to this type library in your project before trying to use the API functions.

Notes.

- This type library exposes more functionality than is described in this document. These additional functions are not supported for this release of Exaquantum.
- Only the 32 bit version of the API should be used. There is no support for a 64 bit version.

2.2. OPC Server API

The Exaquantum OPC Server APIs can be used on any PC that has the OPC Server client libraries installed.

2.3. .NET API

The Exaquantum .NET APIs can be used on any PC that has either the Exaquantum Server or Client components installed, the Exaquantum .NET API uses the .Net 2.0 Framework.

Access to the API is via the following two libraries:

- ♦ Yokogawa.Exa.Exaquantum.Common.dll – defines structures and enumerations
- ♦ Yokogawa.Exa.Exaquantum.dll – defines interfaces and methods

These are located under the '…\Yokogawa\Exaquantum PIMS\System’ folder.

This page intentionally left blank

Chapter 3 Introduction to the COM API Example Chapters

The following chapters give simple examples of how to perform the key operations available via the API. The examples are taken from three VB projects that are installed with the Exaquantum documentation, as follows:

Table 3-1

Install Place	**Project Contents**
…\Documentation\SampleCode\QDataAccessSample\QDALive	Examples of reading and writing the latest Item value, both synchronously and asynchronously.
…\Documentation\SampleCode\QDataAccessSample\QDAHistory	Examples of reading Item value and Alarm and Event history, both synchronously and asynchronously.
…\Documentation\SampleCode\QDataAccessSample\QDAMultiServer	Examples of reading the latest and historical Item values from multiple Exaquantum servers.

This page intentionally left blank

Chapter 4 Common Activities

This chapter explains the common activities required for all API calls.

4.1. Accessing the API Objects

Before a client program can use any API it must first get an Exaquantum Session object. The Session object allows access to all the other API functions.

It is good programming practice for each client to obtain a single Session object, and keep hold of this for the entire time that access to Exaquantum API functions is required. This will help clients respond to the various events available on Session, such as NetworkError and QuantumShutdown.

NOTE: From Exaquantum R2.30 the Session and Session2 objects are superseded by Session3.

An Example of how to obtain and use the Session (or Session2 or Session3) Object:

```
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objAEAccess As QUANTUMAUTOMATIONLib.QAEAccess
Dim objQualityHelper As QUANTUMAUTOMATIONLib.QQualityHelper

' Get a session object
Set objSession = New QUANTUMAUTOMATIONLib.Session

' Get a DataAccess object.
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess

' Get the Browser Object from Session.
Set objBrowser = objSession.Browser

' Give the Session Pointer to the DataAccess object.
objDataAccess.SetSession objSession

' Get an Alarm & Event Access object.
Set objAEAccess = New QUANTUMAUTOMATIONLib.QAEAccess

' Give the Session Pointer to the Alarm & Event Access object.
objAEAccess.SetSession objSession

' Get a QualityHelper object
Set objQualityHelper = New QUANTUMAUTOMATIONLib.QQualityHelper

' Do some work ..


' I've finished so tidy up.
Set objDataAccess = Nothing
Set objAEAccess = Nothing
Set objBrowser = Nothing
Set objSession = Nothing
Set objQualityHelper = Nothing
```

4.2. Accessing Multiple Exaquantum Servers

In order to access multiple Exaquantum servers a client must first create an Exaquantum Session2 object for each server. The Session2 object allows the client to specify the machine name of the Exaquantum server to connect to using its ServerName property. ServerName may only be set once for a Session2 object. Unlike the single-server Session object, a Session2 object can be set to connect to an Exaquantum server that is not currently running. A client can test whether the Exaquantum server is available by calling the Session2 IsConnected() method.

It is good programming practice for each client to obtain a single Session2 object for each Exaquantum server, and retain these for the entire time that access to Exaquantum API functions is required. This will help clients respond to the various events available on Session2, such as QuantumShutdown and QuantumAvailable.

An Example of how to obtain and use the Session2 Object:

```
Dim objSession1 As QUANTUMAUTOMATIONLib.Session2
Dim objSession2 As QUANTUMAUTOMATIONLib.Session2
Dim objDataAccess1 As QUANTUMAUTOMATIONLib.QDataAccess
Dim objDataAccess2 As QUANTUMAUTOMATIONLib.QDataAccess
Dim objBrowser1 As QUANTUMAUTOMATIONLib.Browse2
Dim objBrowser2 As QUANTUMAUTOMATIONLib.Browse2
' Get a Session2 object for the first server.
Set objSession1 = New QUANTUMAUTOMATIONLib.Session2

' Tell Session2 which Exaquantum server to connect to.
objSession1.ServerName = "Server1"

' Get a Session2 object for the second server.
Set objSession2 = New QUANTUMAUTOMATIONLib.Session2

' Tell Session2 which Exaquantum server to connect to.
objSession2.ServerName = "Server2"

' Check that both Exaquantum servers are running.
If Not objSession1.IsConnected() Or Not objSession2.IsConnected() Then
    Exit Sub
End If

' Get a DataAccess object for the first server.
Set objDataAccess1 = New QUANTUMAUTOMATIONLib.QDataAccess

' Give the Session Pointer to the DataAccess object.
objDataAccess1.SetSession objSession1

' Get a DataAccess object for the second server.
Set objDataAccess2 = New QUANTUMAUTOMATIONLib.QDataAccess

' Give the Session Pointer to the DataAccess object.
objDataAccess2.SetSession objSession2

' Get a Browse object to the first server
Set objBrowser1 = objSession1.Browser

' Get a Browse object to the second server
Set objBrowser2 = objSession2.Browser
```

4.3. Resolving a Path to an Item Id

In order to read or write data to Items it is first necessary to obtain an Item Id. An Item Id is a 64 bit number represented as an 8 element byte array that is used to uniquely identify an Item in the RTDB.

An Item Id can be obtained from an Item Path by using the Browser PathToMetaData method as follows:

```
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim sItemsArray(1) As String
Dim lNumItems As Long
Dim Count As Long
Dim vItemID As Variant
Dim rsBrowseResults As ADODB.Recordset

' Get a session object
Set objSession = New QUANTUMAUTOMATIONLib.Session

' Get the Browser Object from Session.
Set objBrowser = objSession.Browser

' Define the Items required by their paths.
sItemsArray(0) = "Root.Tag1.Value"
sItemsArray(1) = "Root.Tag1.Aggregations.Hour.Mean.Value"
lNumItems = 2

' Resolve the paths to Item Ids
Set rsBrowseResults = objBrowser.PathToMetaData(sItemsArray, lNumItems,
brDetail)

' Run through the Recordset returned from Browse and add check which
' is valid.
For Count = 1 To lNumItems
   vItemID = rsBrowseResults!ItemID
   If Not IsNull(vItemID) Then
      ' Do something with the Item ID..

   Else
      MsgBox "Path is invalid:" & sItemsArray(Count - 1)
   End If
   rsBrowseResults.MoveNext
Next Count
```

4.4. Browsing the Role based namespace (RBNS)

Browsing of the namespace allows for all tags (or function blocks) under a specific folder to be listed, so they may be used by an application:

```
Dim sPath As String
Dim sFilter As String
Dim sItemPath As String
Dim objRBNSBrowse As QUANTUMAUTOMATIONLib.RBNSBrowseEx
Dim rsRBNSBrowseResults As ADODB.Recordset

Set objRBNSBrowse = New QUANTUMAUTOMATIONLib.RBNSBrowseEx

' Specify the retrieved folder
sPath = "MyNamespace.Folder1"
' Specify the filter
sFilter = "*"

' Get the tags in the specified folder
Set rsRBNSBrowseResults = objRBNSBrowse.Browse(sPath, sFilter, brDetail)


' Set cursor to the first record
rsRBNSBrowseResults.MoveFirst


' Iterate through the returned Recordset
While (rsRBNSBrowseResults.EOF <> True)
    sItemPath = rsRBNSBrowseResults!Path

    ' Add the code using sItemPath here

    rsRBNSBrowseResults.MoveNext
Wend
```

4.5. Obtaining an OPCAEPumpHistId for a specific OPC Gateway

In order to read Alarm and Event data it is first necessary to obtain an OPCAEPumpHistId. An OPCAEPumpHistId is a 64-bit number represented as an 8-element byte array that is used to uniquely identify the history stream for a particular OPC Gateway.

An OPCAEPumpHistId can be obtained from the QConfig database by executing a query to the OPCServer table using ADO as follows:

```
Dim ConnectStr As String
Dim objConnection As ADODB.Connection
Dim objRecordset As ADODB.Recordset
Dim vAEId As Variant

' Make the Connect String
Set objConnection = New ADODB.Connection
ConnectStr = "Provider=sqloledb;"                        'Provider
ConnectStr = ConnectStr + "Network Library=dbmssocn;"   'Network Library
ConnectStr = ConnectStr + "Server=;"                     'Server Name
(Local)
ConnectStr = ConnectStr + "database=QConfig;"            'Database Name
ConnectStr = ConnectStr + "Trusted_Connection=yes"      'Make it trusted
'connection

'Connect to the database
objConnection.Open ConnectStr

' Run the SQL Script and get the recordset
Set objRecordset = New ADODB.Recordset
objRecordset.LockType = adLockOptimistic
objRecordset.Open "SELECT OPCAEPumpHistId FROM OPCServer WHERE Name = 'OPC
Gateway 1'", objConnection

'Check whether the Recordset is returned from query
If objRecordset.EOF Then
    MsgBox "Invalid OPC Gateway name"
    Exit Sub
End If

' Get the OPCAEPumpHistId (first row only for this example)
vAEId = objRecordset!OPCAEPumpHistId

'Check whether the OPC Gateway Alarm & Events Id is found
If Not IsNull(vAEId) Then
' Do something with the Alarm & Event ID...

Else
    MsgBox "OPCAEPumpHistId is Null"
    Exit Sub
End If
```

This page intentionally left blank

Chapter 5 Reading the Latest Item Value

The latest, or most resent, value of an Item can be read in a one off, synchronous way or continuously updated with changes to the latest value in an asynchronous way. In both cases, reading an Item value is a two-stage process; first resolve the path to the Item required to an Item ID, then use the DataAccess object to return the data.

5.1. Synchronous Read

This type of read is used when a one off request for the latest value is required. If the client must be notified on an event basis each time the Item is updated then an Asynchronous read should be used.

This example reads the current value for two items:

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim rsBrowseResults As ADODB.Recordset
Dim sItemsArray(1) As String
Dim lNumItems As Long
Dim Count As Long
Dim vItemID As Variant
Dim vData As Variant
Dim vValue1, vValue2 As Variant
Dim lQuality1, lQuality2 As Long
Dim dTimeStamp1, dTimeStamp2 As Date

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser

' Create new Exaquantum Data Access object
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess

' Define the Items required by their paths.
sItemsArray(0) = "Root.Tag1.Value"
sItemsArray(1) = "Root.Tag1.Aggregations.Hour.Mean.Value"
lNumItems = 2

' Resolve the paths to Item Ids
Set rsBrowseResults = objBrowser.PathToMetaData(sItemsArray, lNumItems, brDetail)

' Run through the Recordset returned from Browse and add the Item Ids
' to the DataAccess collection.
For Count = 1 To lNumItems
   vItemID = rsBrowseResults!ItemID
   If Not IsNull(vItemID) Then
      ' Add the item to the DataAccess collection
      objDataAccess.Add vItemID
      rsBrowseResults.MoveNext
   Else
      MsgBox "Path is invalid:" & sItemsArray(Count - 1)
      Exit Sub
   End If
Next Count

' Get the latest values of all Items in the collection.
vData = objDataAccess.ReadValue
```

```
' Get VQT from the Variant Array for each Item
‘ The order is not necessarily the order in which
‘ they were added in.
vValue1 = vData(1, 0)(0, 0)     ‘ Value
lQuality1 = vData(1, 0)(1, 0)   ‘ Quality
dTimeStamp1 = vData(1, 0)(2, 0) ‘ Timestamp
vValue2 = vData(1, 1)(0, 0)     ‘ Value
lQuality2 = vData(1, 1)(1, 0)   ‘ Quality
dTimeStamp2 = vData(1, 1)(2, 0) ‘ Timestamp
```

5.2. Asynchronous Read

This type of read is used when the client must be notified on an event basis each time the Item is updated.

Note: For this type of read, the Data Access object must be defined 'With Events' as follows:

```
Dim WithEvents objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
```

This example is kept updated with the current value for two items. The first piece of code registers the items and the second piece of code defines an event routine that will receive the data:

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim rsBrowseResults As ADODB.Recordset
Dim sItemsArray(1) As String
Dim lNumItems As Long
Dim Count As Long
Dim vItemID As Variant

' Define the Items required by their paths.
sItemsArray(0) = "Root.Tag1.Value"
sItemsArray(1) = "Root.Tag1.Aggregations.Hour.Mean.Value"
lNumItems = 2

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser

' Create new Exaquantum Data Access object
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess' Resolve the
paths to Item Ids
Set rsBrowseResults = objBrowser.PathToMetaData(sItemsArray, lNumItems, brDetail)

' Run through the Recordset returned from Browse and add the Item Ids
' to the DataAccess collection.
For Count = 1 To lNumItems
   vItemID = rsBrowseResults!ItemID
   If Not IsNull(vItemID) Then
      ' Add the item to the DataAccess collection
      objDataAccess.Add vItemID
      rsBrowseResults.MoveNext
   Else
      MsgBox "Path is invalid:" & sItemsArray(Count - 1)
      Exit Sub
   End If
Next Count

' Call Execute on DataAccess object to read data asynchronously
objDataAccess.Execute
```

This piece of code shows how to define an event routine that will receive the data. This will be called once for each Item registered with the initial value then again on an item by item basis as an item changes:

```
Private Sub objDataAccess_OnData(ByVal ItemID As Variant, ByVal
DataReturnType As QUANTUMAUTOMATIONLib.qdaDataReturnType, ByVal Data As
Variant)
Dim vValue As Variant
Dim lQuality As Long
Dim dTimeStamp As Date
Dim vItemID As Variant

 ' Read the Item Id for which this data is sent
 vItemID = ItemID

 ' Get VQT from the Variant Array for this Item.
 vValue = Data(0, 0)     ‘ Value
 lQuality = Data(1, 0)   ‘ Quality
 dTimeStamp = Data(2, 0) ‘ Timestamp

 End Sub
```

This page intentionally left blank

Chapter 6 Reading the History of Item Values

The history of an item, for a given time period, can be read in a one off, synchronous way or in an asynchronous way. In both cases, reading an Item value is a two-stage process; first resolve the path to the Item required to an Item ID, then use the DataAccess object to return the data.

6.1. Synchronous Read

This type of read is used when a client can be suspended while the read request is being satisfied. If the client must continue execution and receive data as it is ready via an event, then an Asynchronous read should be used.

This example reads the values for an item from History for the last 5 minutes:

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim rsBrowseResults As ADODB.Recordset
Dim sItem As String
Dim lNumItems As Long
Dim Count As Long
Dim vItemID As Variant
Dim vData As Variant
Dim vValue As Variant
Dim lQuality As Long
Dim dTimeStamp As Date
Dim dQueryTime As Date

' Define the Items required by their paths.
sItem = "Root.Tag1.Value"
lNumItems = 1

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser

' Create new Exaquantum Data Access object
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess

' Resolve the paths to Item Id
Set rsBrowseResults = objBrowser.PathToMetaData(sItem, lNumItems,
brDetail)

' Using the Recordset returned from Browse add the Item Id
' to the DataAccess collection.
vItemID = rsBrowseResults!ItemID
If Not IsNull(vItemID) Then
   ' Add the item to the DataAccess collection
   objDataAccess.Add vItemID
Else
   MsgBox "Path is invalid:" & sItem
   Exit Sub
End If

' Lets query for the last 5 minutes
dQueryTime = DateAdd("n", -5, Now)

' Set the QueryTimes on DataAccess object
```

```
objDataAccess.SetQueryTimes , dQueryTime

' Get the latest values of all Items in the collection.
vData = objDataAccess.ReadValue

' Loop through all the VQTs returned and store them
If Not IsNull(vData) Then
    ' Loop through the entire array and store the data
    ' NB: In this sample only the last data in the Array is stored when we
are out of the loop
    For Count = 0 To UBound(vData(1, 0), 2)
        vValue = vData(1, 0)(0, Count)      ' Value
        lQuality = vData(1, 0)(1, Count)    ' Quality
        dTimeStamp = vData(1, 0)(2, Count)  ' Timestamp
    Next Count
End If
```

6.2. Asynchronous Read

This type of read is used when client must continue execution and receive data as it is ready via an event.

Note: For this type of read, the Data Access object must be defined 'With Events' as follows:

```
Dim WithEvents objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
```

This example receives values for an item from history for the past 5 minutes. The first piece of code registers the items and the second piece of code defines an event routine that will receive the data:

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim rsBrowseResults As ADODB.Recordset
Dim sItem As String
Dim lNumItems As Long
Dim vItemID As Variant
Dim dQueryTime As Date

' Define the Items required by their paths.
sItem = "Root.Tag1.Value"
lNumItems = 1

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser

' Create new Exaquantum Data Access object
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess
' Resolve the paths to Item Ids
Set rsBrowseResults = objBrowser.PathToMetaData(sItem, lNumItems,
brDetail)

' Using the Recordset returned from Browse add the Item Id
' to the DataAccess collection.
vItemID = rsBrowseResults!ItemID
If Not IsNull(vItemID) Then
   ' Add the item to the DataAccess collection
   objDataAccess.Add vItemID
Else
   MsgBox "Path is invalid:" & sItem
   Exit Sub
End If

' Lets query for the last 5 minutes
dQueryTime = DateAdd("n", -5, Now)

' Set the QueryTimes on DataAccess object
objDataAccess.SetQueryTimes , dQueryTime

' Call Execute on DataAccess object to read data asynchronously
objDataAccess.Execute
```

This piece of code shows how to define an event routine that will receive the data. This will be called for multiple times until DataReturnType is qdaEnd for each Item registered:

```
Private Sub objDataAccess_OnData(ByVal ItemID As Variant, ByVal
DataReturnType As QUANTUMAUTOMATIONLib.qdaDataReturnType, ByVal Data As
Variant)
Dim vValue As Variant
Dim lQuality As Long
Dim dTimeStamp As Date
Dim vItemID As Variant
Dim Count As Long


' Read the Item Id for which this data is sent
vItemID = ItemID


' Loop through the entire array and store the data
' NB: In this sample only the last data in the Array is stored
' when we are out of the loop
If Not IsNull(Data) Then       ' Ignore notification if Data is NULL
    For Count = 0 To UBound(Data, 2)
        vValue = Data(0, Count)       ‘ Value
        lQuality = Data(1, Count)     ‘ Quality
        dTimeStamp = Data(2, Count)   ‘ Timestamp
    Next Count
End If

' Check whether more data to come
If DataReturnType = qdaNew Then
    MsgBox "First notification received"
ElseIf DataReturnType = qdaMoreRows Then
    MsgBox "More notifications to be received"
ElseIf DataReturnType = qdaEnd Then
    MsgBox "Last notification received"
End If
```

This page intentionally left blank

Chapter 7 Setting a Filter and Reading the Results

The filter allows you to reduce the amount of raw tag data returned by three methods:

- ♦ Resolution filtering algorithm - Splitting the overall time period into smaller individual periods for which minimum and maximum values will be returned.
- ♦ Aggregation calculations - Apply some pre-processing to summarize the data using defined statistical functions.
- ♦ Simple Query Filter – returns only points that match a value and/or quality filter.

This method is defined on both the IQDataAccess and IQDataAccessMulti interfaces. Refer to section 18.12 SetFilterParameters Method for more details.

7.1. Resolution filtering algorithm

For one item, the input parameters to the algorithm are:

- ♦ Number of periods required (for example the number of horizontal pixels)
- ♦ Start time of data
- ♦ End Time of data

The algorithm then works on the resulting data, as follows:

1 A query is issued covering the duration between the two input parameters; ‘Start time of data’ and the ‘End time of data’.

2 The data returned by the query is “time-sliced” into periods. The number of periods being determined by the ‘Number of periods’ input parameter.

3 For each period, the points with the Maximum and Minimum values are extracted. These points will be sent to the client application. The Timestamps sent will be those associated with the Maximum and Minimum values in that period.

4 If a bounding value is requested, an extra period is created whose boundary time is the start time of the query. The only point contained within this extra period is the bounding value.

<u>NOTES</u>

1. If there is only one point within a period, then this point is sent to the client.
2. If there are no points within a period, then no data is sent to the client for this period.
3. For string data, an attempt will be made to convert them to a DOUBLE. If it succeeds, these data will be included in the processing for minimum and maximum.
4. For any period, if there is a point with quality bad or uncertain shutdown, and whose value is either:
   - ♦ greater than any maximum value with a good quality,
   - ♦ less than a minimum value with a good quality,

   then this point is returned in addition to the max and /or min points having good quality. This means up to four points can be returned from one period, two with good quality, and two with bad or uncertain shutdown quality.
5. Data having a secondary quality of deleted, irrespective of the primary quality, will be deleted by the filtering. A client receiving raw data never gets these points.

7.2. Aggregation calculations

The following forms of aggregation calculations are supported.

Table 7-1

Type of data	**Types of Aggregation calculations**
Continuous data	Sum, Mean, Minimum, Maximum, StdDev, Spot
Discrete data	Status count, OnTime

7.2.1. Aggregation types

This section describes the method used to calculate each types of aggregation.

7.2.1.1. Mean

The calculation of the Mean is based on the difference between the timestamp of the current input notification and the equivalent timestamp form the previous input notification as follows:

$\text{Mean} = ((\text{Mean} * T_{\text{total}}) + ((Vn\text{-}1) * (Tn - Tn\text{-}1)))/ (T_{\text{total}} + (Tn - Tn\text{-}1))$

Where:

Mean – Is the accumulating mean.

T_{total} – Is the accumulating time for the input being valid – i.e. its data quality was either GOOD or Uncertain.

Vn-1 – Is the value of the input at the last time notification.

Tn-1 – Is the timestamp of the input at the last time notification.

Tn – Is the timestamp of the input at this time notification.

7.2.1.2. Minimum

Determines the minimum value within the required period data. The timestamp indicates when the minimum value was recorded

7.2.1.3. Maximum

Determines the maximum value within the required period data. The timestamp indicates when the maximum value was recorded.

7.2.1.4. Standard deviation

The standard deviation builds upon the Mean calculation shown above:

$\text{SumOfXSquared} = \text{SumOfXSquared} + ((V_{n-1})^2 * (T_n - T_{n-1}))$

$\text{StdDev} = \text{Sqrt}(((\text{SumOfXSquared}) - (T_{\text{total}} * \text{Mean}^2)) / T_{\text{total}} - 1)$

7.2.1.5. Summation

There are two ways used to calculate the Summation that depend on the TimeFactor configured for the aggregation calculation:

- ♦ When TimeFactor > 0, Integration summation is used.
- ♦ When TimeFactor = 0, Simple summation is used.

These two methods are explained below.

7.2.1.5.1. TimeFactor > 0 (Integration Summation)

The time difference between the timestamp of the current input notification from the timestamp of the previous input notification is used to calculate the contribution towards the summation.

$\text{Sum} = \text{Sum} + (V_{n-1}) * (T_n – T_{n-1})$

Where:

- ♦ Sum – Is the accumulating summation.
- ♦ V_{n-1} – Is the value of the input at the final requirement time.
- ♦ T_{n-1} – Is the timestamp of the input at the final requirement time or the timestamp of the start of the current aggregation period if no notifications have been received during the current period. T is measured in units of 100 nanoseconds.
- ♦ T_n– Is the timestamp of the input at this data.

At the end of the aggregation period, the summation is calculated as:

Summation = Sum / (10,000,000 * TimeFactor)

7.2.1.5.2. TimeFactor = 0 (Simple Summation)

At the beginning of the aggregation period the internal summation is reset to zero. As each input notification arrives, it is added to the summation. The summation result is the accumulated summation at the end of the aggregation period.

7.2.1.6. Spot value

By default, the value of the input at the end of the aggregation period. The value at the beginning of the period may be selected by setting registry value HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\Server\AggSpotValueAtPeriodStart to 1.

7.2.1.7. Count

Within the aggregation period under consideration, this represents the number of times the input notification value changes from a value not equal to the ‘On State’ value, to a value equal to the ‘On State’ value.

7.2.1.8. On Time

The number of seconds the input value has been equal to the ‘On State’ value within the aggregation period.

7.3. Simple Query Filter

The simple query filter allows points to only include in the return data if they match value and / or a quality filter(s); the two types can be combined if required. The query can be applied to both raw data and calculated (ad-hoc) aggregation data.

7.3.1.1. Value Filter

This comprises one or two simple conditional expressions containing numerical operators; in the latter case, the expressions are joined by a logical operator.

The filter takes the form:

(<op1> value1) <lop> (<op2> value2)

Where:

- <op1> and <op2> are one of the following numerical operators: =, !=, >, <, >=, <=
- <lop> is one of the logical operators AND or OR.

Examples of value filters:

- Tag value > 100
- Tag value > 100 and Tag value < 120

7.3.1.2. Quality Filter

The quality filter is defined by a bitmask:

- Good = 8
- Uncertain = 4
- Bad = 2
- Assumed = 1

If no quality bitmask is defined, points with all qualities are returned.

7.4. Example code

The following two examples of code demonstrate how to use the SetFilterParameters method to set the properties for a filter, and then read the results.

7.4.1. Intrinsic using single server Data Access

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim rsBrowseResults As ADODB.Recordset
Dim sItem As String
Dim lNumItems As Long
Dim Count As Long
Dim vItemID As Variant

' Dimension the filter parameters to return interpolated data
Dim vFilterParameters as Variant
Dim vFilterParms(1) As Variant
Dim lFilterId as Long
Dim lNumPeriods as Long

' Dimension variable to receive the data
Dim vData As Variant
Dim vValue as Variant
Dim vQuality as Variant
Dim vTimestamp as Variant

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser
' Create new Exaquantum Data Access object
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess

' Items are requested by using Intrinsic Namespace Access strings, i.e.
Root.???
sItem = "Root.Tag1.Value"
lNumItems = 1

' Resolve the paths to Item Id
Set rsBrowseResults = objBrowser.PathToMetaData(sItem, lNumItems,
brDetail)

' Using the Recordset returned from Browse add the Item Id
' to the DataAccess collection.
vItemID = rsBrowseResults!ItemID
If Not IsNull(vItemID) Then
   ' Add the item to the DataAccess collection
   objDataAccess.Add vItemID
Else
   MsgBox "Path is invalid:" & sItem
   Exit Sub
End If

' Set time range
objDataAccess.SetQueryTimes StartTime:=CDate("01/26/2023 10:00:00")

' Set filter parameters to return interpolated data
lFilterId = 1
lNumPeriods = 200
```

```
vFilterParms(0) = lFilterId
vFilterParms(1) = lNumPeriods
vFilterParameters = vFilterParms
objDataAccess.SetFilterParameters vFilterParameters

' Read filtered data for specified time range
vData = objDataAccess.ReadValue()

'Iterate through the returned data retrieving the history values
Dim lNumberOfPoints As Long
Dim lPoint As Long

lNumberOfPoints = UBound(vData(1, 0), 2)
For lPoint = 0 To lNumberOfPoints
    vValue = vData(1, 0)(0, lPoint)
    vQuality = vData(1, 0)(1, lPoint)
    vTimestamp = vData(1, 0)(2, lPoint)
    …
Next

Set objDataAccess = Nothing
```

7.4.2. RBNS using Multi-Server Interface

```
Dim objMDA As QUANTUMAUTOMATIONLib.QDataAccessMulti
Dim sItem As String

' Dimension the filter parameters to return interpolated data
Dim vFilterParameters as Variant
Dim vFilterParms(1) As Variant
Dim lFilterId as Long
Dim lNumPeriods as Long

‘ Dimension variable to receive the data
Dim vData as Variant
Dim vValue as Variant
Dim vQuality as Variant
Dim vTimestamp as Variant

' Items are requested by using Role-Based Namespace Access strings
sItem = “MyNamespace.Tag1.Value”

' Create new Exaquantum QDataAccessMulti object
Set objMDA = New QUANTUMAUTOMATIONLib.QDataAccessMulti
objMDA.Add sItem

' Set time range from yesterday to now
objMDA.SetQueryTimes StartTime:= “now – 1 day”

' Set filter parameters to return interpolated data
lFilterId = 1
lNumPeriods = 200
vFilterParms(0) = lFilterId
vFilterParms(1) = lNumPeriods
vFilterParameters = vFilterParms
objMDA.SetFilterParameters vFilterParameters


' Read filtered data from yesterday to now
vData = objMDA.ReadValue()

'Iterate through the returned data retrieving the history values
Dim lNumberOfPoints As Long
Dim lPoint As Long

lNumberOfPoints = UBound(vData(1, 0), 2)
For lPoint = 0 To lNumberOfPoints
    vValue = vData(1, 0)(0, lPoint)
    vQuality = vData(1, 0)(1, lPoint)
    vTimestamp = vData(1, 0)(2, lPoint)
    …
Next

Set objMDA = Nothing
```

This page intentionally left blank

Chapter 8 Writing Item Values

Writing values into an item can be done using the DataAccess object. First resolve the path to an Item Id, and then use DataAccess object to write values.

```
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim objDataAccess As QUANTUMAUTOMATIONLib.QDataAccess
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objQualityHelper As QUANTUMAUTOMATIONLib.QQualityHelper
Dim rsBrowseResults As ADODB.Recordset
Dim vItemID As Variant
Dim lNumItems As Long
Dim vValue As Variant
Dim lQuality As Long
Dim sItem As String

' Define the item that is to be written
sItem = "Root.Tag1.Value"
lNumItems = 1

' Create Session, Browser, Data Access and Quality objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser
Set objDataAccess = New QUANTUMAUTOMATIONLib.QDataAccess
Set objQualityHelper = New QUANTUMAUTOMATIONLib.QQualityHelper

' Give the Session Pointer to the DataAccess object.
objDataAccess.SetSession objSession

' Resolve the paths to Item Id
Set rsBrowseResults = objBrowser.PathToMetaData(sItem, lNumItems,
brDetail)

' Read the Item Id from the recordset
vItemID = rsBrowseResults!ItemID

'If the Item Id is NULL, then the defined item does not exist
If IsNull(vItemID) Then
    MsgBox "Invalid Path: " & sItem
    Exit Sub
End If

' define the VQT to be written into Item
vValue = CSng(100)  ' Value of data type single
' Use QualityHelper function to fetch raw quality definition
lQuality = objQualityHelper.RawQuality(qdaQualityGood)      ' Quality good

' Write into DataAccess
‘ Leave TimeStamp empty so that Server time is used
objDataAccess.WriteValue vValue, vItemID, lQuality
```

This page intentionally left blank

Chapter 9 Accessing Exaquantum Server(s) through RBNS

Session2 object is an explicit way of accessing multiple Exaquantum servers. Instead of managing multiple servers yourself, you can use QDataAccessMulti object to access multiple Exaquantum servers transparently through RBNS (Role Based Namespace) mechanism.

Accessing Exaquantum data through RBNS also provides a high security by way of the RBNS configuration. For example one user can read an item but another user may not be able to read the same item.

QDataAccessMulti is built on top of QDataAccess and Session2 objects, and the same principle applies to QDataAccessMulti object as to these two. One difference is that QDataAccessMulti methods use path names and not item identifiers to identify items.

9.1. Reading Data

```
Dim objMDA As QDataAccessMulti

' Items are requested by using Role-Based Namespace Access strings
Dim item1 As String
Dim item2 As String

item1 = "MyNamespace.Tag1.Value"
item2 = "MyNamespace.Folder1.Tag1.Value"
…

' Create new Exaquantum QDataAccessMulti object
Set objMDA = New QDataAccessMulti
objMDA.Add item1
objMDA.Add item2

Dim data as Variant
data = objMDA.ReadValue()

' Retrieve VQT for first Item
Dim value1 as Variant
Dim quality1 As Long
Dim timestamp1 As Date
value1 = data(1, 0)(0, 0)
quality1 = data(1, 0)(1, 0)
timestamp1 = data(1, 0)(2, 0)

' Get yesterdays value of item 1
objMDA.SetQueryTimes SpotTime:= "now - 1 day"
data = objMDA.ReadValue()
value11 = data(1, 0)(0, 0)

' Get values from yesterday to now for item 1
objMDA.SetQueryTimes StartTime:= "now - 1 day", EndTime:= "now"
data = objMDA.ReadValue()
'Iterate through the returned data retrieving the history values
Dim lNumberOfPoints As Long
lNumberOfPoints = UBound(data(1, 0), 2)
Dim nPoint As Long
For nPoint = 0 To lNumberOfPoints
    value1 = data(1, 0)(0, nPoint)
    quality1 = data(1, 0)(1, nPoint)
    timestamp1 = data(1, 0)(2, nPoint)
    …
```

```
Next

Set objMDA = Nothing
```

9.2. Writing Data

```
Dim objMDA As QDataAccessMulti

' Items are requested by using Role-Based Namespace Access strings
Dim item1 As String
Dim item2 As String

item1 = “MyNamespace.Tag1.Value”
item2 = “MyNamespace.Folder1.Tag1.Value”
…

' Create new Exaquantum QDataAccessMulti object
Set objMDA = New QdataAccessMulti

' Start registering new current data to write
Dim value1 as Variant
value1 = 123.0
objMDA.AddWrite item1, value1
value1 = “hello”
objMDA.AddWrite item2, value1

' Write new values to server
Dim vStatus as Variant
vStatus = objMDA.ExecuteWrite(“The Reason”)

Set objMDA = Nothing
```

Chapter 10 Reading the History of Alarm and Events

The history of Alarm and Events, for a given time period, can be read in a one off, synchronous way or in an asynchronous way. In both cases, reading an Alarm and Event data is a two-stage process; first obtain the history stream OPCAEPumpHistID for a particular OPC Gateway, then use the AEAccess object to return the data.

10.1. Synchronous Read

This type of read is used when a client can be suspended while the read request is being satisfied. If the client must continue execution and receive data as it is ready via an event, then an Asynchronous read should be used.

This example reads the Alarm and Events for a particular OPC Gateway from History:

```
Dim ConnectStr As String
Dim objConnection As ADODB.Connection
Dim objRecordset As ADODB.Recordset
Dim Count As Long
Dim i As Long
Dim vAEId As Variant
Dim Events As Variant
Dim vOPCAEPumpHistID As Variant
Dim sSource As String
Dim sMessage As String
Dim dTimeStamp As Date
Dim lSequenceNo As Long
Dim lEventCategory As Long
Dim lSeverity As Long
Dim lCookie As Long
Dim sConditionName As String
Dim sSubConditionName As String
Dim lChangeMask As Long
Dim lNewState As Long
Dim lConditionQuality As Long
Dim bAckRequired As Boolean
Dim dActiveTime As Date
Dim sActorID As String
Dim lNoOfAttributes As Long
Dim vEventAttributesArray As Variant
Dim vEventAttribute As Variant
Dim dQueryTime As Date
Dim objAEAccess As QUANTUMAUTOMATIONLib.QAEAccess

' Make the Connect String
Set objConnection = New ADODB.Connection
ConnectStr = "Provider=sqloledb;"                           'Provider
ConnectStr = ConnectStr + "Network Library=dbmssocn;"   'Network Library
ConnectStr = ConnectStr + "Server=;"                       'Server
Name(Local)
ConnectStr = ConnectStr + "database=QConfig;"             'Database Name
ConnectStr = ConnectStr + "Trusted_Connection=yes"        'Make it trusted
'connection

'Connect to the database
objConnection.Open ConnectStr

' Run the SQL Script and get the recordset
Set objAEAccess = New QUANTUMAUTOMATIONLib.QAEAccess
Set objRecordset = New ADODB.Recordset
objRecordset.LockType = adLockOptimistic
objRecordset.Open "SELECT OPCAEPumpHistId FROM OPCServer WHERE Name = 'OPC
Gateway 1'", objConnection

'Check whether the Recordset is returned from query
If objRecordset.EOF Then
    MsgBox "Invalid OPC Gateway name"
    Exit Sub
End If

' Get the OPCAEPumpHistId (first row only for this example)
vAEId = objRecordset!OPCAEPumpHistId

'Check whether the OPC Gateway Alarm & Events Id is found
If Not IsNull(vAEId) Then
    ' Add the AE Id into the AEAccess collection
    objAEAccess.Add vAEId
Else
    MsgBox "OPCAEPumpHistId is Null"
    Exit Sub
End If

' Lets query for the last 5 minutes
dQueryTime = DateAdd("n", -5, Now)

' Set the QueryTimes on AEAccess object
objAEAccess.SetQueryTimes dQueryTime

' Now read the values
Events = objAEAccess.ReadValue

' Check to see if there are any data returned
If Not IsNull(Events) Then
    ' Loop through the entire array and store the AE data
    ' NB: In this sample only the last data in the Array is stored
    ' when we are out of the loop
    For Count = 0 To UBound(Events, 2)
        vOPCAEPumpHistID = Events(0, Count)             ' OPCAEPumpHistID
        sSource = Events(1, Count)                     ' Source
        sMessage = Events(2, Count)                    ' Message
        dTimeStamp = Events(3, Count)                  ' TimeStamp
        lSequenceNo = Events(4, Count)                 ' SequenceNumber
        lEventCategory = Events(5, Count)              ' EventCate
        lSeverity = Events(6, Count)                   ' Severity
        lCookie = Events(7, Count)                     ' Cookie
        sConditionName = Events(8, Count)              ' ConditionName
        sSubConditionName = Events(9, Count)           ' SubConditionName
        lChangeMask = Events(10, Count)                ' ChangeMask
        lNewState = Events(11, Count)                  ' NewState
        lConditionQuality = Events(12, Count)          ' ConditionQuality
        bAckRequired = Events(13, Count)               ' AckRequired
        dActiveTime = Events(14, Count)                ' ActiveTime
        sActorID = Events(15, Count)                   ' ActorID
lNoOfAttributes = Events(16, Count)          ' Number Of EventAttributes
        vEventAttributesArray = Events(17, Count)    ' EventAttributes in
an Array

        ' Loop through all the EventAttributes and store them
        For i = 0 To lNoOfAttributes - 1
            vEventAttribute = vEventAttributesArray(i)
        Next i
    Next Count
End If
```

10.2. Asynchronous Read

This type of read is used when client must continue execution and receive data as it is ready via an event.

Note: For this type of read, the AEAccess object must be defined 'With Events' as follows:

```
Dim WithEvents objAEAccess As QUANTUMAUTOMATIONLib.QAEAccess
```

This example receives the Alarm and Events for a particular OPC Gateway from History. The first piece of code registers the Alarm and Event HistID, and the second piece of code defines an event routine that will receive the data:

```
Dim vAEId As Variant
Dim vData As Variant
Dim vAEData As Variant
Dim lQuality As Long
Dim dTimeStamp As Date
Dim ConnectStr As String
Dim objConnection As ADODB.Connection
Dim objRecordset As ADODB.Recordset
Dim dQueryTime As Date

' Make the Connect String
Set objConnection = New ADODB.Connection
ConnectStr = "Provider=sqloledb;"                        'Provider
ConnectStr = ConnectStr + "Network Library=dbmssocn;"   'Network Library
ConnectStr = ConnectStr + "Server=;"                     'Server
Name(Local)
ConnectStr = ConnectStr + "database=QConfig;"            'Database Name
ConnectStr = ConnectStr + "Trusted_Connection=yes"      'Make it trusted
connection

'Connect to the database
objConnection.Open ConnectStr

' Run the SQL Script and get the recordset
Set objAEAccess = New QUANTUMAUTOMATIONLib.QAEAccess
Set objRecordset = New ADODB.Recordset
objRecordset.LockType = adLockOptimistic
objRecordset.Open "SELECT OPCAEPumpHistId FROM OPCServer WHERE Name = 'OPC
Gateway 1'", objConnection

'Check whether the Recordset is returned from query
If objRecordset.EOF Then
    MsgBox "Invalid OPC Gateway name"
    Exit Sub
End If

' Get the OPCAEPumpHistId (first row only for this example)
vAEId = objRecordset!OPCAEPumpHistId

'Check whether the OPC Gateway Alarm & Events Id is found
If Not IsNull(vAEId) Then
    ' Add the AE Id into the AEAccess collection
    objAEAccess.Add vAEId
Else
```

```
        MsgBox "OPCAEPumpHistId is Null"
        Exit Sub
    End If

    ' Lets query for the last 5 minutes
    dQueryTime = DateAdd("n", -5, Now)

    ' Set the QueryTimes on AEAccess object
    objAEAccess.SetQueryTimes dQueryTime

    ' Call Execute on AEAccess to read data asynchronously
    objAEAccess.Execute
```

This piece of code shows how to define an event routine that will receive the data. This will be called multiple times until EventReturnType is qdaEnd:

```
Private Sub objAEAccess_OnAE(ByVal EventReturnType As
QUANTUMAUTOMATIONLib.qeaEventReturnType, ByVal Events As Variant)
Dim vOPCAEPumpHistID As Variant
Dim sSource As String
Dim sMessage As String
Dim dTimeStamp As Date
Dim lSequenceNo As Long
Dim lEventCategory As Long
Dim lSeverity As Long
Dim lCookie As Long
Dim sConditionName As String
Dim sSubConditionName As String
Dim lChangeMask As Long
Dim lNewState As Long
Dim lConditionQuality As Long
Dim bAckRequired As Boolean
Dim dActiveTime As Date
Dim sActorID As String
Dim lNoOfAttributes As Long
Dim vEventAttributesArray As Variant
Dim vEventAttribute As Variant
Dim Count As Long
Dim i As Long

' Check if the data is NULL
If Not IsNull (Events) Then
    ' Loop through the entire array and store the AE data
    ' NB: In this sample only the last data in the Array is stored
    ' when we are out of the loop
    For Count = 0 To UBound(Events, 2)
        vOPCAEPumpHistID = Events(0, Count)             ' OPCAEPumpHistID
        sSource = Events(1, Count)                  ' Source
        sMessage = Events(2, Count)                 ' Message
        dTimeStamp = Events(3, Count)               ' TimeStamp
        lSequenceNo = Events(4, Count)              ' SequenceNumber
        lEventCategory = Events(5, Count)           ' EventCate
        lSeverity = Events(6, Count)                ' Severity
        lCookie = Events(7, Count)                  ' Cookie
        sConditionName = Events(8, Count)           ' ConditionName
        sSubConditionName = Events(9, Count)        ' SubConditionName
        lChangeMask = Events(10, Count)             ' ChangeMask
        lNewState = Events(11, Count)               ' NewState
        lConditionQuality = Events(12, Count)       ' ConditionQuality
        bAckRequired = Events(13, Count)            ' AckRequired
        dActiveTime = Events(14, Count)             ' ActiveTime
        sActorID = Events(15, Count)                ' ActorID
lNoOfAttributes = Events(16, Count)          ' Number Of EventAttributes
        vEventAttributesArray = Events(17, Count)    ' EventAttributes in
an Array

        ' Loop through all the EventAttributes and store them
        For i = 0 To lNoOfAttributes - 1
            vEventAttribute = vEventAttributesArray(i)
        Next i

    Next Count

End If

' Check whether more data to come
If EventReturnType = qeaNew Then
    MsgBox "First notification received"
ElseIf EventReturnType = qeaMoreEvents Then
    MsgBox "More notification to be received"
ElseIf EventReturnType = qeaEnd Then
    MsgBox "Last notification received"
End If

End Sub
```

This page intentionally left blank

Chapter 11 Conducting Recalculation and Re-aggregation

The recalculation and re-aggregation is a mechanism to update Exaquantum calculated tags and aggregation result items after raw data have been changed that would affect dependent calculated tags and aggregation result items.

The operation consists of two major steps:

1 To update raw data with an existing data write mechanism, e.g. Data Write API

2 To execute recalculation and re-aggregation by this new Recalculation API

This chapter covers the second step mentioned above.

11.1. Concept of Recalculation

The recalculations are based upon "Recalculation Jobs" defined by the user and stored in a database for re-use.

For each job the user must specify a unique name and a set of input Items – these are the Items for which updated data has been previously written to the Exaquantum Historian.

The user may also optionally de-select one or more dependent items to be excluded from the recalculation.

For this version of Exaquantum the following scope will be applied:

- ♦ Only dependent aggregations and scripted aggregations will be recalculated.
- ♦ Recalculation can only be requested for dependent aggregations after having changed the aggregation result items on which they depend. [See Note 1]
- ♦ All dependent aggregations will be recalculated – the user cannot deselect any items.
- ♦ Recalculation will be synchronous. No notifications on progress status will be sent to the user as the time is expected to be very short (a few seconds).
- ♦ All result items set in a scripted aggregation will be recalculated, but only those that are aggregation result items will have their dependent aggregations recalculated.
- ♦ Number of aggregation result items must be small, e.g. a few. Also the time span to recalculate should be short, e.g. a few hours. [See Note 2]

Note 1: Future release of Exaquantum will allow specifying raw data that will affect referencing calculated tags as well as base and dependent aggregations.

Note 2: There is no hard figure; the actual number depends on the system work load. But keep it to a minimum.

When requesting recalculation for an aggregation result item, the PercentTimeGood item may not be selected – this will be updated automatically when other aggregation results are recalculated.

If an aggregation result uses the Changed property of the PercentTimeGood item (as shown below) then it will not be recalculated. This is because the PercentTimeGood value is calculated internally and should not be written to explicitly.

```
if [Root.Tag1.Aggregations.Hour.PercentTimeGood.Value].Changed then
    [Result.Aggregations.Hour.Mean.Value] = <n>
    [Result.Aggregations.Hour.PercentTimeGood.Value] = <n>
end if
```

A recalculation operation cannot be carried out for:

- ♦ A period that lies within an archive. - If the start time is within an archive period and the end time is valid then the data within the valid online period will be recalculated. The Systems Events Viewer will report a failed recalculation.
- ♦ A period before the time that an item was created. - If the start time is before the creation of the item and the end time is valid then the data within the valid online period will be recalculated. The Systems Events Viewer will report a failed recalculation.

11.2. Recalculation Engine

The Exaquantum Recalculation Engine is implemented as a free-threaded COM server (QRecalcEngine.exe). It has a similar architecture to Quantum.exe with a single RecalcBroker object responsible for:

- ♦ Saving RecalculationJob definitions in a database
- ♦ Determining the input and output dependencies of Items requested for recalculation
- ♦ Reading data from Historian to initialize input items
- ♦ Starting up all Item, Aggregation and Calculation objects required for the recalculation
- ♦ Providing locking so that only one recalculation can be performed at once
- ♦ Driving the recalculation and building an information of recalculated values for inspection by the user
- ♦ Writing recalculated data points to the Historian
- ♦ Logging recalculation details to audit trail tables
- ♦ Forwarding updated current aggregation estimated results to the Exaquantum RTDB to update “live” aggregations.

Commit of a recalculation is synchronous. The method does not return to the user until all updated data has been written into history and any updates to live aggregations have been made to the RTDB.

11.3. Introduction to Recalculation API

The client accesses the API through the RecalcJob property of the Exaquantum Session2 or Session3[1] object. Only members of the following groups are allowed to access Recalculations

Legacy Model: QAdministratorGroup or QDataWriteGroup

Standard Model: QTM_MAINTENANCE or QTM_DATA_WRITE

Typical scenarios are described below.

User creates a new Recalc Job:

- Call NewRecalcJobDefinition to create a new job.
- Call the SetRecalcInputItemIds method to specify the Items whose data will be changed.

User re-runs an existing Recalc Job:

- Set the RecalcJobName property to specify which job to run.

Then in both cases:

- Set the time range for recalculation by calling SetRecalcTimes.
- Call Recalc to request recalculation – a Recordset of data values that will be written is returned for review.
- User either calls Commit to save the recalculated data to Exaquantum Historian or calls Cancel to abandon the recalculation.

A recalculation is “active” from the beginning of a call on the Recalc() method until the end of a call on the Commit() or Cancel() methods. Only one recalculation may be active at one time. If a user releases a RecalcJob object that they have called the Recalc() method on without calling Commit() or Cancel(), the recalculation will be automatically cancelled.

Users may still configure other recalculation job definitions while a recalculation is active. Only one user may access a given job definition at one time.

[1] The Session3 object must be used if the API code is to be run on an Exaquantum Server with the following DCOM settings:

Default Authentication Level: None

Default Impersonation Level: Identify

11.4. Notes on Minimum and Maximum aggregation results

In the case where the user wishes to write a new timestamp as well as a new value for a Minimum or Maximum aggregation result, the user must also update the existing Minimum or Maximum point with a secondary quality of DELETED. If this results in a change to the timestamp of recalculated dependent Minimum or Maximum points, the Recalculation Engine will set the secondary quality of old dependent points to DELETED.

If the user writes an aggregation result item at a timestamp which is not on the aggregation period boundary for any result other than a Minimum or Maximum, the point will be ignored.

11.5. Sample code

11.5.1. User changes a single hourly aggregation result

Production calendar dependencies look as follows:

Raw -> Hour -> Day -> Month

New data points written:

Table 11-1

ItemPath	Value	Quality	Timestamp
Root.Tag1.Aggregations. Hour.Mean.Value	118	Good	#1/26/2003 9:00:00 AM#

```
    Dim objSession As QUANTUMAUTOMATIONLib.Session2
    Dim objRecalcJob As QUANTUMAUTOMATIONLib.IQRecalcAuto
    Dim lRecalcSessionID As Long
    Dim objItemValuesRS As ADODB.Recordset

    ' Get a Recalculation Job handler object
    Set objSession = New QUANTUMAUTOMATIONLib.Session2
    Set objRecalcJob = objSession.RecalcJob

    ' Create a new recalculation job definition
    objRecalcJob.NewRecalcJobDefinition "Tag1Hourly", "The reason"

    ' Define the items to be recalculated for this job

objRecalcJob.SetRecalcInputItems("Root.Tag1.Aggregations.Hour.Mean.Value")

    ' Specify the time for recalculation – current time is #1/27/2003
9:00:00 AM#
    objRecalcJob.SetRecalcTimes #1/26/2003 9:00:00 AM#

    ' Do the recalculation
    lRecalcSessionID = objRecalcJob.Recalc(False, objItemValuesRS)

    ' Inspect the results
    ….
    ' Commit the recalculation results to history
    objRecalcJob.Commit lRecalcSessionID
```

Example of item values recordset showing points that may be affected:

Table 11-2

Item ID	Item Path	Timestamp	Value	Quality	Prev. Value	Prev. Quality	Previous Timestamp
14	Root.Tag1.Aggregations. Day.Mean.Value	#1/27/2003 00:00:00#	120	Good	119	Good	#1/27/2003 00:00:00#
21	Root.Tag1.Aggregations. Month.Mean.Value	#1/27/2003 00:00:00#	100	Good Estimated	Null	Null	Null

11.5.2. User uses an existing recalculation job definition

The plan is to change the same hourly aggregation result item for more than one hour within a day.

New data points written:

Table 11-3

ItemPath	Value	Quality	Timestamp
Root.Tag1.Aggregations.Hour.Mean.Value	118	Good	#1/27/2003 9:00:00 AM#
Root.Tag1.Aggregations.Hour.Mean.Value	120	Good	#1/27/2003 10:00:00 AM#

```
    Dim objSession As QUANTUMAUTOMATIONLib.Session2
    Dim objRecalcJob As QUANTUMAUTOMATIONLib.IQRecalcAuto
    Dim lRecalcSessionID As Long
    Dim objItemValuesRS As ADODB.Recordset

    ' Get a Recalculation Job handler object
    Set objSession = New QUANTUMAUTOMATIONLib.Session2
    Set objRecalcJob = objSession.RecalcJob

    ' Use existing recalculation job definition
    objRecalcJob.RecalcJobName = "Tag1Hourly"

    ' Specify a time range for recalculation – current time is #2/01/2003
9:00:00 AM#
    objRecalcJob.SetRecalcTimes , #1/27/2003 9:00:00 AM#, #1/27/2003
10:00:00 AM#

    ' Do the recalculation
    lRecalcSessionID = objRecalcJob.Recalc(False, objItemValuesRS)

    ' Inspect the results
    ….
    ' User decides to cancel the recalculation
    objRecalcJob.Cancel lRecalcSessionID
```

Example of item values recordset showing points that may be affected:

Table 11-4

Item ID	Item Path	Timestamp	Value	Quality	Prev. Value	Prev. Quality	Previous Timestamp
14	Root.Tag1.Aggregations. Day.Mean.Value	#1/28/2003 00:00:00#	120	Good	121	Good	#1/28/2003 00:00:00#
21	Root.Tag1.Aggregations. Month.Mean.Value	#2/01/2003 00:00:00#	100	Good	101	Good	#2/01/2003 00:00:00#

11.5.3. User changes two hourly aggregation result items that feed a scripted aggregation

Production calendar dependencies look as follows:

Raw -> Hour -> Day -> Month

Scripted aggregation calculation generates Hour’s aggregated value as in:

(Result.Aggregations.Hour.Mean.Value =

Root.Tag3.Aggregations.Hour.Mean.Value +

Root.Tag4.Aggregations.Hour.Mean.Value)

New data points written:

Table 11-5

ItemPath	Value	Quality	Timestamp
Root.Tag3.Aggregations.Hour.Mean.Value	118	Good	#1/26/2003 9:00:00 AM#
Root.Tag4.Aggregations.Hour.Mean.Value	230	Good	#1/26/2003 9:00:00 AM#

```
Dim objSession As QUANTUMAUTOMATIONLib.Session2
Dim objRecalcJob As QUANTUMAUTOMATIONLib.IQRecalcAuto
Dim lRecalcSessionID As Long
Dim objItemValuesRS As ADODB.Recordset
Dim ItemPaths(2) as Variant

' Get a Recalculation Job handler object
Set objSession = New QUANTUMAUTOMATIONLib.Session2
Set objRecalcJob = objSession.RecalcJob

' Set the item paths for the items to recalculate
ItemPaths(0) = "Root.Tag3.Aggregations.Hour.Mean.Value"
ItemPaths(1) = "Root.Tag4.Aggregations.Hour.Mean.Value"

' Create a new recalculation job definition
objRecalcJob.NewRecalcJobDefinition "CalcHourly", "The reason"
```

```
        ' Define the items to be recalculated for this job
        objRecalcJob.SetRecalcInputItems ItemPaths

        ' Specify the time for recalculation – current time is #2/01/2003
    9:00:00 AM#
        objRecalcJob.SetRecalcTimes #1/26/2003 9:00:00 AM#

        ' Do the recalculation
        lRecalcSessionID = objRecalcJob.Recalc( False, objItemValuesRS)

        ' Inspect the results
        ….
        ' Commit the recalculation results to history
        objRecalcJob.Commit lRecalcSessionID
```

Example of item values recordset showing points that may be affected:

Table 11-6

Item ID	Item Path	Timestamp	Value	Quality	Prev. Value	Prev. Quality	Previous Timestamp
9	Root.Tag3.Aggregations. Day.Mean.Value	#1/28/2003 00:00:00#	120	Good	121	Good	#1/28/2003 00:00:00#
6	Root.Tag3.Aggregations. Month.Mean.Value	#2/01/2003 00:00:00#	100	Good	101	Good	#2/01/2003 00:00:00#
70	Root.Tag4.Aggregations. Day.Mean.Value	#1/28/2003 00:00:00#	220	Good	208	Good	#1/28/2003 00:00:00#
77	Root.Tag4.Aggregations. Month.Mean.Value	#2/01/2003 00:00:00#	210	Good	200	Good	#2/01/2003 00:00:00#
84	Root.Calc.Aggregations. Hour.Mean.Value	#1/26/2003 09:00:00#	320	Good	318	Good	#1/26/2003 09:00:00#
88	Root.Calc.Aggregations. Day.Mean.Value	#1/28/2003 00:00:00#	340	Good	329	Good	#1/28/2003 00:00:00#
92	Root.Calc.Aggregations. Month.Mean.Value	#2/01/2003 00:00:00#	310	Good	301	Good	#2/01/2003 00:00:00#

11.5.4. User changes multiple hourly Minimum or Maximum aggregation result

Production calendar dependencies look as follows:

Raw -> Hour -> Day -> Month

New data points written (current time is #2/01/2003 9:00:00 AM#):

Table 11-7

ItemPath	Value	Quality	Timestamp
Root.Tag1.Aggregations.Hour.Minimum.Value	118	Good	#1/26/2003 8:45:00 AM#
Root.Tag1.Aggregations.Hour.Minimum.Value	120	Bad | Deleted	#1/26/2003 8:50:00 AM#
Root.Tag1.Aggregations.Hour.Minimum.Value	116	Good	#1/26/2003 9:30:00 AM#
Root.Tag1.Aggregations.Hour.Minimum.Value	122	Bad | Deleted	#1/26/2003 9:40:00 AM#

(Sample code is not shown.)

Example of item values recordset showing points that may be affected:

Table 11-8

Item ID	Item Path	Timestamp	Value	Quality	Prev. Value	Prev. Quality	Previous Timestamp
14	Root.Tag1.Aggregations. Day.Minimum.Value	#1/26/2003 09:00:00#	120	Bad | Deleted	120	Good	#1/26/2003 09:00:00#
14	Root.Tag1.Aggregations. Day.Minimum.Value	#1/26/2003 10:00:00#	116	Good	Null	Null	#1/26/2003 09:00:00#
21	Root.Tag1.Aggregations. Month.Minimum.Value	#1/27/2003 00:00:00#	116	Good	120	Good	#1/27/2003 00:00:00#

Chapter 12 Setting Tags and Function Blocks Offline

Tags and Function Blocks may be set Offline or Online using the NamespaceBuilder object that is obtained from the Session, Session2 or Session3[2] object. First resolve the path of the Tag or Function Block to a Namespace Id using the Browser object, and then use the NamespaceBuilder object to set the Tag or Function Block Offline or Online.

```
Dim objSession As QUANTUMAUTOMATIONLib.Session
Dim objNamespaceBuilder As QUANTUMAUTOMATIONLib.INamespaceBuilder2
Dim objBrowser As QUANTUMAUTOMATIONLib.Browse2
Dim rsBrowseResults As ADODB.Recordset

Dim vNamespaceIDs(1) As Variant
Dim lNumNamespaceItems As Long
Dim vValue As Variant
Dim lQuality As Long
Dim sTagPaths(1) As String
Dim vStatusArray As Variant
Dim sUserName As String
Dim sMachine As String

' Create Session and Browser objects
Set objSession = New QUANTUMAUTOMATIONLib.Session
Set objBrowser = objSession.Browser

' Define the Tags and Function Blocks to be set Offline
sTagPaths(0) = "Root.OPCTag1"
sTagPaths(1) = "Root.OPCTag2"
lNumNamespaceItems = 2

' Resolve the paths to Item Id
Set rsBrowseResults = objBrowser.PathToMetaData(sTagPaths,
lNumNamespaceItems, brDetail)

' Read the Namespace Ids from the recordset
For i = 0 To lNumNamespaceItems - 1
     vNamespaceIDs(i) = rsBrowseResults!NamespaceId
    ' If the Namespace Id is NULL, then the defined path does not exist
    If IsNull(vNamespaceIDs(i)) Then
        MsgBox "Invalid Path: " & sTagPaths(i)
        Exit Sub
    End If
    rsBrowseResults.MoveNext
Next

' Get a NamespaceBuilder object from Session
Set objNamespaceBuilder = objSession.NamespaceBuilder
```

[2] The Session3 object must be used if the API code is to be run on an Exaquantum Server with the following DCOM settings:

Default Authentication Level: None

Default Impersonation Level: Identify

```
' Set User and Machine parameters
sUserName = “MyUser”
Set sMachine = “MyMachine”
' Tell NamespaceBuilder to set the Tag or Function Block Offline
objNamespaceBuilder.SetOfflineState True, vNamespaceIDs,
lNumNamespaceItems, sUserName, sMachine, vStatusArray

' Check the returned status array for errors
For i = 0 To lNumNamespaceItems - 1
       If vStatusArray(i) > 0 Then
         MsgBox "Not a Tag or Function Block: " & sTagPaths(i)
       End If
Next
```

Chapter 13 Introduction to the API Reference Chapters

The following chapters provide the definition of the Exaquantum APIs. Each Object is dealt with in a separate chapter. The definitions are given in a VB style.

This page intentionally left blank

Chapter 14 Session Object API Reference

The Session object controls the connection from the client to the server.

14.1. Browser Property

14.1.1. Description

Returns a Browser object.

14.2. NamespaceBuilder Property

14.2.1. Description

Returns a NamespaceBuilder object used for setting Tags and Function Blocks Offline and Online. This object can only be obtained if the client is a member of the following group:

Legacy Model: QAdministratorGroup.

Standard Model: QTM_Maintenance

14.3. NetworkError Event Method

14.3.1. Description

This method is called back on the client when the network connection fails between client and server.

Clients cannot recover the connection to Exaquantum after this failure without first closing the connection and re-establishing it by destroying and recreating the Session object.

14.3.2. Signature

```
Event NetworkError()
```

14.4. QuantumShutdown Event Method

14.4.1. Description

This method is called back on the client when the Exaquantum Server processes are shut down.

Clients cannot recover the connection to Exaquantum after this failure without first closing the connection and re-establishing it by destroying and recreating the Session object.

14.4.2. Signature

```
Event QuantumShutdown(bErrored As Boolean)
```

14.4.3. Parameters

bErrored is False if the Exaquantum processes shutdown normally (by stopping the Service) or True if the processes shutdown abnormally.

This page intentionally left blank

Chapter 15 Session2 and Session3 Object API Reference

The Session2 and Session3 objects control the connection from the client to a server in a multi-server environment. Session3 should be used in place of the Session and Session2 objects. Session and Session2 are maintained for backward compatibility. Existing applications using Session and Session2 will continue to function. Session3 is used in exactly the same way as Session2. They support the same properties and methods as Session in addition to the following:

15.1. ServerName Property

15.1.1. Description

ServerName is a string property allowing the client to specify the machine name of the Exaquantum server to connect to. This property must be set before calling any other method on the Session2 object. If not set, it will default to using the DesignatedServer in the same way as the Session object.

ServerName may only be set once – an attempt to reset it, or to set it after calling any other method, will raise an error.

15.2. IsConnected Method

15.2.1. Description

This method may be used to determine whether the Exaquantum server that the Session2 object is set to connect to is available. It returns True if the Exaquantum server is running or False if the server is shutdown or the network is unavailable.

15.2.2. Signature

```
 Public Function IsConnected() As Boolean
```

15.3. QuantumAvailable Event Method

15.3.1. Description

This method is called back on the client when the Exaquantum server that the Session2 object is set to connect to has been unavailable but is now connected.

15.3.2. Signature

```
Event QuantumAvailable()
```

15.4. ConnectState Method

15.4.1. Description

A read-only property which returns the current connection state of an Exaquantum client to the Exaquantum server.

15.4.2. Signature

```
Public Function ConnectState() As qsmConnectState
```

15.4.3. Parameters

The connection state is an enumeration:

Table 15-1

Enumeration	**Value**	**Meaning**
qsmConnected	0	Client is successfully connected
qsmNotRunning	1	The Exaquantum Server is not in a running state
qsmShutdown	2	The Exaquantum Server has been shut down
qsmNetworkDown	3	A network error has been detected – only set after initial successful connection has been lost
qsmConnectFailed	4	Connection to the Exaquantum server failed – possibly a security or network problem

Chapter 16 Browser Object API Reference

The Browse object gives access to the Exaquantum Namespace information. This information is used to locate a particular Tag, Folder, Function Block or Item in the hierarchy as displayed in the Data Selector in the Tag Editor or Explorer.

16.1. Browse Method

16.1.1. Description

Return a Recordset of metadata information for the contents of a single leaf in the namespace.

16.1.2. Signature

```
Public Function Browse(Path As String, _
    Optional Filter As String = "*", _
    Optional Fields As brFields = brSummary) As ADODB.Recordset
```

16.1.3. Return Value

A Recordset will be returned populated with metadata information for the selected **Path**. The information will be returned in alphabetic order based on the *Name* column. The columns returned will be defined by the **Fields** parameter.

16.1.4. Parameters

Path specifies the leaf of the namespace to use e.g. "Root.Folder1".

Filter specifies a text filter to apply to the names of the leaves. This allows the client to reduce the number of rows returned. The filter allows wildcards specified with '*'. Filter will be ignored for browsing below a tag.

Fields allows the fields returned to be selected by the enumeration **brFields** as described below.

Table 16-1 Enumeration brSummary=1

Column Name	Data Type	Comments
Name	nvarchar(32)	The name for this leaf.
Id	int	The id for this leaf. This is typically the column ExtId from table namespace. See below for details.
ClassName	nvarchar(32)	The class name from the column Name in table Class.

Table 16-2 Enumeration brDetail=2

Column Name	Data Type	Comments
Name	nvarchar(32)	The name for this leaf.
Id	int	The id for this leaf. This is typically the column ExtId from table namespace. See below for details.
ClassName	nvarchar(32)	The class name from the column Name in table Class.
ItemDataTypeId	tinyint	For Items this returns the data type from the column DataTypeId in table Item.
ItemId	binary(8)	For Items this returns the Item ID which is required when using the DataAccess object.
NamespaceId	int	The id for this leaf from the Namespace table. If this leaf is not stored in the namespace table, i.e. it is detail below the tag, then NULL is returned.
IsShortcut	bit	Indicates that this leaf is a shortcut.

16.1.5. Id returned in Recordset

The Id returned depends on the class of the leaf and whether it exists above or below a tag in the hierarchy as described below:

Table 16-3

Class	Example	Id returned
Tag or Function block	Root.Tag1	The ExtId column from the Namespace table. This matches the relevant row in the Tag or FunctionBlock tables Id column.
Folders above the level of a tag	Root.Folder1	The ExtId column from the Namespace table. This also matches the Id column of the Namespace table.
Item	Root.Tag1.Value	The Id column from the Item table.
‘Aggregations’ folder	Root.Tag1.Aggregations	NULL
Aggregations or tags below aggregations	Root.Tag1.Aggregations. Hour or Root.Tag1.Aggregations. Hour.Mean	NULL

16.2. PathToMetadata Method

16.2.1. Description

Return a Recordset of metadata information for objects in the namespace hierarchy as defined by a list of one or more paths.

16.2.2. Signature

```
Public Function PathToMetadata(Paths() As String, _
    NumPaths As Long, _
    Optional Fields As brFields = brSummary) As ADODB.Recordset
```

16.2.3. Return Value

A Recordset will be returned populated with metadata information for the selected **Paths**. The information will be returned in the same order as the entries in the **Paths** array.

Fields allows the fields returned to be selected by the enumeration **brFields** as described in section 16.1.

If a Path is invalid then all fields in the Recordset will be NULLs.

16.2.4. Parameters

Paths specifies a set of leaves in the namespace to use e.g. “Root.Folder1”.

NumPaths specifies the number of entries in the **Paths** array.

Fields allows the fields returned to be selected by the enumeration **brFields** as described above.

16.3. ItemIdToPath Method

16.3.1. Description

Return a Recordset of paths for Items in the namespace hierarchy as defined by a list of one or more Ids.

If more than one path exists due to a shortcut then return just one of the possible paths.

16.3.2. Signature

```
Public Function ItemIdToPath(Ids() As Long, _
    NumIds As Long) As ADODB.Recordset
```

16.3.3. Return Value

A Recordset will be returned populated with a single column ‘Path’ for the selected **Ids**. The information will be returned in the same order as the entries in the **Ids** array.

If an Id is invalid then the corresponding row in the Recordset will be NULL.

Recordset shape:

Table 16-4

Column Name	**Data Type**	**Comments**
Path	nvarchar(256)	The Path for the corresponding Id or NULL if the Id is invalid.

16.3.4. Parameters

Ids specifies a set of Ids as defined by the **Id** column of the **Item** Table.

NumIds specifies the number of entries in the **Ids** array.

16.4. PathToIds Method

16.4.1. Description

Returns VARIANT array of 8 byte item ids for supplied array of item paths. If paths are invalid, ids are returned as VT_NULL.

```
Public Function PathToIds ( _
    NamespaceId As Long, _
    RemainingPaths() As String, _
    NumPaths As Long, _
    Optional FolderFilter As String = "*", _
    Optional FolderFilterAdditive As Boolean = TRUE, _
    Optional FBFilter As String = "*", _
    Optional FBFilterAdditive As Boolean = TRUE, _
    Optional FBTagFilter As String = "*", _
    Optional FBTagFilterAdditive As Boolean = TRUE, _
    Optional TagFilter As String = "*", _
    Optional TagFilterAdditive As Boolean = TRUE, _
    Optional ShowAggregations As Boolean = TRUE) As VARIANT
```

16.4.2. Return Value

A VARIANT, containing an array of 8 byte item ids, will be returned for the selected **NamespaceId**, **RemainingPaths, HiRange** and **LowRange**. The information will be returned in the same order as the entries in the **RemainingPaths** array.

16.4.3. Parameters

NamespaceId Specifies the id in the namespace to start parsing the path from.

RemainingPaths Specifies the leaf of the namespace to use below the specified NamespaceId e.g. “Folder1.Folder2”.

NumPaths Specifies the number of entries in the RemainingPaths arrays.

Various filters may be specified to allow the client to reduce the number of rows returned. Filters allow wildcards specified with ‘*’. Each filter has an associated flag to specify whether the filter is additive (only matching leaves are returned) or subtractive (all matching leaves are excluded)

FolderFilter Specifies a text filter to apply to the names of folders.

FolderFilterAdditive Specifies whether the FolderFilter is additive or subtractive.

FBFilter Specifies a text filter to apply to the names of Function Blocks.

FBFilterAdditive Specifies whether the FBFilter is additive or subtractive.

FBTagFilter Specifies a text filter to apply to the names of Tags within Function Blocks.

FBTagFilterAdditive Specifies whether the FBTagFilter is additive or subtractive.

TagFilter Specifies a text filter to apply to the names of Tags outside Function Blocks.

TagFilterAdditive Specifies whether the TagFilter is additive or subtractive.

ShowAggregations Specifies whether to return metadata for Aggregations below a Tag.

16.5. WritePlotLimits Method

16.5.1. Description

This method controls the writing of the plot limits of specified items.

16.5.2. Signature

```
Public Sub WritePlotLimits(   NamespaceID as Long, _
      RemainingPaths() as Variant, _
      lNumPaths as long, _
      PlotRanges() as Variant )
```

16.5.3. Parameters

Table 16-5

Parameter	**Description**
NamespaceID	Specifies the ID in the namespace under which the paths in RemainingPaths start from.
RemainingPaths	Specifies the leaf of the namespace to use below the specified NamespaceID. For example, “Folder1.Folder2”.
lNumPaths	Number of paths in RemainingPaths
PlotRanges	Contains a 2 column SAFEARRAY of doubles; column 0 contains lowranges, column 1 hiranges. Should contain the same number of rows as there are ItemIDs.

16.5.4. Errors

E_INVALIDARG

Either:

- The specified number of IDs does not concur with the size of either of the input arrays.
- Either of the input arrays are of the wrong data type.

Chapter 17 RBNSBrowseEx Object API Reference

The RBNSBrowseEx object enables you to access to the information defined with the Exaquantum RBNS. This information is used to find the position of tree-structured tags, function blocks, or items displayed with the Tag Editor or Exaquantum/Explorer Data Selector. Also, if RBNS is not defined in Exaquantum, this object performs the same function as the Browser object does.

17.1. Browse Method

17.1.1. Description

Return a Recordset of metadata information for the contents of a single leaf in the namespace.

17.1.2. Signature

```
Public Function Browse(Path As String, _
    Optional Filter As String = "*", _
    Optional Fields As brFields = brSummary) As ADODB.Recordset
```

17.1.3. Return Value

A Recordset will be returned populated with metadata information for the selected **Path**. The information will be returned in alphabetic order based on the *Name* column. The columns returned will be defined by the **Fields** parameter.

17.1.4. Parameters

Path specifies the leaf of the namespace to use e.g. “Root.Folder1”.

Filter specifies a text filter to apply to the names of the leaves. This allows the client to reduce the number of rows returned. The filter allows wildcards specified with ‘*’. Filter will be ignored for browsing below a tag.

Fields allows the fields returned to be selected by the enumeration **brFields** as described below.

Table 17-1 Enumeration brSummary=1

Column Name	Data Type	Comments
Name	nvarchar(32)	The name for this leaf.
ClassName	nvarchar(32)	The class name from the column **Name** in table **Class**.

Table 17-2 Enumeration brDetail=2

Column Name	Data Type	Comments
Name	nvarchar(32)	The name for this leaf.
ClassName	nvarchar(32)	The class name from the column **Name** in table **Class**.
ItemDataTypeId	Tinyint	For Items this returns the data type from the column **DataTypeId** in table **Item**.
Path	nvarchar(256)	The full path for specifying a leaf.
ServerId	Int	Exaquantum server ID
ServerName	nvarchar(32)	Exaquantum server name

17.1.5. Id returned in Recordset

The Id returned depends on the class of the leaf and whether it exists above or below a tag in the hierarchy as described below:

Table 17-3

Class	Example	Id returned
Tag or Function block	Root.Tag1	The ExtId column from the Namespace table. This matches the relevant row in the Tag or FunctionBlock tables Id column.
Folders above the level of a tag	Root.Folder1	The ExtId column from the Namespace table. This also matches the Id column of the Namespace table.
Item	Root.Tag1.Value	The Id column from the Item table.
‘Aggregations’ folder	Root.Tag1.Aggregations	NULL
Aggregations or tags below aggregations	Root.Tag1.Aggregations.Hour or Root.Tag1.Aggregations.Hour.Mean	NULL

17.2. PathToMetadata Method

17.2.1. Description

Return a Recordset of metadata information for objects in the namespace hierarchy as defined by a list of one or more paths.

17.2.2. Signature

```
Public Function PathToMetadata(Paths() As String, _
    NumPaths As Long, _
    Optional Fields As brFields = brSummary) As ADODB.Recordset
```

17.2.3. Return Value

A Recordset will be returned populated with metadata information for the selected **Paths**. The information will be returned in the same order as the entries in the **Paths** array.

Fields allows the fields returned to be selected by the enumeration **brFields** as described in section 17.1.

If a Path is invalid then all fields in the Recordset will be NULLs.

17.2.4. Parameters

Paths specifies a set of leaves in the namespace to use e.g. "Root.Folder1".

NumPaths specifies the number of entries in the **Paths** array.

Fields allows the fields returned to be selected by the enumeration **brFields** as described above.

This page intentionally left blank

Chapter 18 QDataAccess Object API Reference

The DataAccess Object allows Item data to be read and written.

Items of interests are specified by Item Ids, these can be resolved from the Item Path using the Browser object. Before an Item can be read its Id must first be added to a collection maintained by QDataAccess. After filling the collection, clients select the function they require either for the whole collection (typical) or for individual items in the collection.

All date/time parameters and results are in local time. This behavior may be overridden by calling the SetProcessOptions method setting the bUTC parameter to True (see section 18.10).

18.1. Add Method

18.1.1. Description

Add an Item Id to the collection of Items of interest. On add, if a request for data is current, data for the new ItemID will be fetched automatically.

18.1.2. Signature

```
Sub Add (ItemID As Variant)
```

18.1.3. Parameters

ItemId - specifies the Item of interest. This should be an 8 byte array as returned by the browser.

18.2. Cancel Method

18.2.1. Description

Cancels the current request and stops acquisition of live data.

18.2.2. Signature

```
Sub Cancel ()
```

18.3. Count Property

18.3.1. Description

Read the number of Items in the collection of Items of interest.

18.3.2. Signature

```
Property Count () As Long
```

18.4. Execute Method

18.4.1. Description

Starts the asynchronous acquiring of data using the current time and processing options set for the QDataAccess Object.

Data is returned via the OnData event.

18.4.2. Signature

```
Sub Execute (Optional WithChanges As Boolean = True, _
    Optional ReturnDataAsReady As Boolean = True )
```

18.4.3. Parameters

WithChanges - When requesting *latest data*, if set FALSE then only a single event will be fired to the client for each Item. The default is to notify the client whenever the value of any Item in the collection changes.

ReturnDataAsReady - When requesting historical data, if set FALSE then all the values for each Item will be returned in a single event. The default is to send data as soon as it is available to allow a more responsive user interface.

18.5. Item Method

18.5.1. Description

Return an Item Id from the collection.

18.5.2. Signature

```
Function Item (Index As Variant) as Variant
```

18.5.3. Parameters

Index is the position in the collection.

18.5.4. Return Value

The 8 byte array representing the Item Id.

18.6. OnData Event Method

18.6.1. Description

This method is called back on the client for asynchronous events initiated by Execute().

18.6.2. Signature

```
Event OnData (ItemId As Variant, _
      DataReturnType As qdaDataReturnType, _
      Data As Variant)
```

18.6.3. Parameters

ItemId is the Id of the Item for which data is arriving.

DataReturnType One of enumeration qdaDataReturnType :

qdaNew=0 – This is new data for this item. This is always the type for the first notification for an item after Execute(), and for live data.

qdaMoreRows=1 – For historical data, this is additional data to append to previously received data.

qdaEnd=2 – For historical data, this is the final data for this Item.

Data - A 2-dimensional array of *Variants* with the following format:

Variant Value	*Long* Quality	*Date* Timestamp

where live data has only one row and historical data has one row for each time point.

Note: **Data** may be NULL and clients should check for this. This indicates that no data is returned for this firing of the event.

18.7. OnError Event Method

18.7.1. Description

This method is called back on the client for asynchronous requests if an error occurs.

18.7.2. Signature

```
Event OnError(Number As Long, _
      Description As String, _
      Source As String, _
      HelpFile As String, _
      HelpContext As Long)
```

18.7.3. Parameters

Number - is the Quantum error number.

Description - is the error text.

Source - is the component that raised the error.

HelpFile - is the name of the HelpFile containing further information on the error.

HelpContext - is the Help context number.

18.8. ReadValue Method

18.8.1. Description

Perform a synchronous read for values of items using the current time and processing options set for the QDataAccess Object.

Data may be read for all items in the QDataAccess collection or for one specific item.

Data is always fetched for the whole collection and stored in the local cache. Subsequent calls to ReadValue retrieve data from the cache. If the query times are not set, updating live data is fetched so that the value in the cache is always up to date.

18.8.2. Signature

```
Function ReadValue (Optional ItemID As Variant) As Variant
```

18.8.3. Parameters

ItemID -An Item internal ID as an 8-byte array. If not supplied then defaults to fetch data for all items in the collection.

Note: Specifying **ItemID** as Null or Empty is equivalent to not supplying it.

18.8.4. Return Value

A 2-dimensional array of *Variants* with the following format:

Variant ItemID	*Variant* VQT

one row for each Item, where ItemID is the Item Id and VQT is a 2-dimensional *Variant* array with the format:

Variant Value	*Long* Quality	*Date* Timestamp

with one row for each data point.

Note: The Return Value may be NULL and clients should check for this. This indicates that no data is returned for this query.

18.8.5. Errors

E_Q_HISTORYREQUESTFAIL – General error indicating that the request for historical data failed. Normally accompanied by associated errors from QHistorian

E_Q_DAUNEXPECTEDERROR – An unexpected data access error occurred.

E_Q_HISTORYTIMEOUT – Request for historical data timed out.

E_Q_DATAACCESS_INVALIDFILTERPARAMETERS – Request for historical data contains one or more invalid parameters.

E_Q_SCHEDULERCREATEFAIL – Request for historical replay data failed to create a scheduler thread.

E_Q_HISTORYREPLAYSCHEDULEFAIL – Request for historical replay data failed.

E_Q_HISTORYREPLAYFAIL – Request for historical replay data failed.

E_Q_STARTTIMEAFTEREND – Start time of request is after the end time.

E_Q_HISTORYCANCELFAIL – Failed to cancel a historical data request.

18.9. Remove Method

18.9.1. Description

Remove an Item Id from the collection of Items of interest.

18.9.2. Signature

```
Sub Remove (ItemID As Variant)
```

18.9.3. Parameters

ItemId - specifies the Item of interest. This should be an 8 byte array as returned by the browser.

18.10. SetProcessOptions Method

18.10.1. Description

Allows the processing of item data to be specified.

Cancels any current query.

18.10.2. Signature

```
Sub SetProcessOptions(Optional UpdateRate As Variant, _
      Optional InterpMethod As qdaInterpMethod, _
      Optional EdgeValueOptions As qdaEdgeValueOptions, _
      Optional TimeIncrement As Variant, _
      Optional ResampleInterval As Variant, _
      Optional bUTC As Boolean = False)
```

18.10.3. Parameters

UpdateRate - specifies the maximum rate at which the client is to be notified of data changes. If not specified the default is 5 seconds (must be specified as DATE.) The DATE is converted to an interval by subtracting the earliest date that can be represented by the DATE data type, i.e. midnight on 30th December 1899. This allows VB assignments such as #00:00:05# for 5 seconds, or #00:02:00# for 2 minutes.

Note: Specifying **UpdateRate** as Null or Empty is equivalent to not supplying it.

InterpMethod - determines how the raw historized data should be processed. If not specified then defaults to qdaPreviousForward.

Otherwise one from the enumeration qdaInterpMethod:

qdaDefault=0 – the default (qdaPreviousForward)

qdaNextBackward=1 – Cast the next value backwards.

qdaPreviousForward=2 – Cast the previous value forwards.

qdaLinear=3 – Linear interpolation between the previous and next values.

EdgeValueOption - determines how data matching the start and end times of a historical query should be processed. If not specified then defaults to qdaIncludeEdge.

Otherwise one from the enumeration qdaEdgeValueOptions:

qdaIncludeEdge=0 – Include values at specified start and end times by using the propagating data using the selected interpolation method.

qdaIncludeBoundingValues =1 – Include bounding value i.e. the data point immediately before the requested start time.

TimeIncrement - If this is specified, a timed request for history data will be made at the UpdateRate, incrementing the SpotTime or StartTime by this amount. If latest data is requested, "polling" is simulated by firing the latest value to the Client at the UpdateRate (must be specified as DATE). See UpdateRate for details of how the DATE type is interpreted as an interval.

Note: Specifying **TimeIncrement** as Null or Empty is equivalent to not supplying it.

ResampleInterval - If this is specified, history data will be resampled to provide exactly one point per ResampleInterval (Must be specified as DATE.). See UpdateRate for details of how the DATE type is interpreted as an interval.

Timestamps for all items except Minimum and Maximum Aggregation result items are returned as the fixed ResampleInterval time. If an Aggregation result item is resampled at its aggregation period, Exaquantum shutdown points within a period are automatically screened out.

Minimum and Maximum Aggregation result items show the timestamp of the actual minimum or maximum value within the ResampleInterval. Any Minimum or Maximum Aggregation result item data with "Deleted" secondary quality is screened out.

Note: Specifying **ResampleInterval** as Null or Empty is equivalent to not supplying it.

bUTC – If specified as True the timestamps returned will be in UTC. The timestamp that is specified in SetQueryTime(), WriteValue(), should be in UTC as well. The default is False.

18.11. SetQueryTimes Method

18.11.1. Description

Allows the period or point in time to be specified for the items of interest.

Cancels any current query.

18.11.2. Signature

```
 Sub SetQueryTimes(Optional SpotTime As Variant, _
         Optional StartTime As Variant, _
         Optional EndTime As Variant)
```

18.11.3. Parameters

SpotTime - if specified, return data at this time otherwise return live data (must be specified as a DATE).

Note: Specifying **SpotTime** as Null or Empty is equivalent to not supplying it.

StartTime & **EndTime** - If only StartTime is specified then return data from this time to the latest data. If both StartTime and EndTime are specified then return data over this period (must be specified as DATE).

Note 1: Specifying **StartTime** or **EndTime** as Null or Empty is equivalent to not supplying them.

Note 2: If the period of time specified by StartTime and EndTime falls wholly or partially in the future (with respect to the Exaquantum Server clock), then additional data points will be introduced in the data returned by DataAccess with a data quality of Bad/Not Available. The points will be introduced for all times greater than the Exaquantum Server clock.

18.12. SetFilterParameters Method

This method is defined on both the IQDataAccess and IQDataAccessMulti interfaces.

18.12.1. Description

This method may be called before performing a read of history data to specify a filter to pre-process the data. Any current query is cancelled.

18.12.2. Signature

```
Sub SetFilterParameters( Optional FilterParameters As Variant )
```

18.12.3. Parameters

FilterParameters - An optional VARIANT holding an array of filter parameters. If specified, the VARIANT should hold an array of VARIANTs, with the first array member being the long identity of the filter algorithm and subsequent parameters being algorithm-dependent.

If called with no arguments any existing filter parameters are unset.

18.12.4. Parameters Definition, for Ad-hoc (Calculated Aggregations)

The following table shows the details of the parameters:

Table 18-1

		Array parameters		
Filter Algorithm ID	**Description**	**Element Number**	**Contents**	**Data type**
1	Interpolated data	• 0	Id = 1	Long integer
		• 1	Number of periods	Long integer
2	Mean	• 0	Id = 2	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
3	Summation	• 0	Id = 3	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
		• 3	Sum Time Factor (0 to 3600 seconds). Only used if element 4 is FALSE.	Date
		• 4	Use Differential Summation	Boolean
		• 5	Reset Value. Only used if element 4 is TRUE.	Double

		Array parameters		
Filter Algorithm ID	**Description**	**Element Number**	**Contents**	**Data type**
		• 6	Maximum number of Decimal places. Only used if element 4 is TRUE.	Integer
		• 7	Percentage Difference. Only used if element 4 is TRUE.	Double
4	Maximum	• 0	Id = 4	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
5	Minimum	• 0	Id = 5	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
6	Standard Deviation	• 0	Id = 6	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
7	Spot Value	• 0	Id = 7	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
8	Count	• 0	Id = 8	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
		• 3	‘On State’ value	Variant
9	OnTime	• 0	Id = 9	Long integer
		• 1	Interval Period (1 second upwards) 0 = Aggregation of whole period	Long integer
		• 2	Partial aggregation support option flag (0/1)	Boolean
		• 3	‘On State’ value	Variant

18.12.5. Parameters Definition, for Simple Query Filter

Table 18-2 shows the details of the parameters:

Table 18-2

Element Number	Data Type	Description
0	Long integer	12 (Data Query Algorithm ID)
1	VARIANT	Quality bitmask, or NULL if no selection is to be applied to qualities. See Table 18-6 for values.
2	VARIANT	Contains array of Value parameters (see Table 18-3 for details), or NULL / 0 if no selection is to be applied to values. The number of parameters is controlled by the SAFEARRAY dimensions.
3	VARIANT	Contains ad-hoc (calculated) aggregations parameters (see Table 18-1 for details), or NULL / 0, if query is to be run against raw data.

Table 18-3 Value VARIANT array structure

Element Number	Data Type	Description
0	VARIANT	Operator 1 (op1>) – Any data type that can be converted to a short integer. See Table 18-4 for values.
1	VARIANT	Value 1 (value1) – stored as a variant
2	VARIANT	Logical operator – Any data type that can be converted to a short integer. See Table 18-5 for values.
3	VARIANT	Operator 2 (op2>) - Any data type that can be converted to a short integer. See Table 18-4 for values.
4	VARIANT	Value 2 (value2) – stored as a variant

Table 18-4 Arithmetic Operator Values

Value	Arithmetic operator
1	=
2	!=
3	>=
4	<=
5	>
6	<

Table 18-5 Values for Logical Operator

Value	Logical operator
1	AND
2	OR

Table 18-6 Values for Quality Bitmask

Value	Quality
1	Assumed
2	Bad
4	Uncertain
8	Good

18.13. SetSession Method

18.13.1. Description

This method may be called by a client to allow the QDataAccess object to share its Session connection. This is the recommended way to use QDataAccess.

If this method is not called before a request to fetch data, the QDataAccess object will create its own Session object.

18.13.2. Signature

```
Sub SetSession(pSession As Object)
```

18.13.3. Parameters

pSession - A pointer to the client's Session object

18.14. WriteValue Method

18.14.1. Description

This method may be called by a client to write Values into an Item.

18.14.2. Signature

```
Sub WriteValue (Value As Variant, _
      Optional Item As Variant, _
      Optional Quality As Variant, _
      Optional TimeStamp As Variant)
```

18.14.3. Parameters

Value - The new value to be written to the Item.

Item - The ItemId of the Item as an 8-byte array. This need not be specified if there is only a single item in the collection.

Note: Specifying **Item** as Null or Empty is equivalent to not supplying it.

Quality - The quality of the new value to be written to the Item. This defaults to GOOD if not specified. If specified, it must be long integer.

If the primary quality Assumed is written to an Item that is not Offline, the Item quality will remain unchanged. If a new value is written to an Offline Item, its secondary quality will always be set to Replaced.

Note: Specifying **Quality** as Null or Empty is equivalent to not supplying it.

Timestamp - The timestamp of the new value to be written to the Item. This defaults to the current time if not specified. If specified, it must be DATE.

Note: Specifying **Timestamp** as Null or Empty is equivalent to not supplying it.

18.15. WriteHistoryValue Method

18.15.1. Description

This method may be called by a client to write Values into the history of an Item. The method blocks until the write to history is complete. The new value is written directly to history even if the timestamp is more recent than the latest timestamp in the Item.

18.15.2. Signature

```
Sub WriteHistoryValue(Value As Variant, _
      Item As Variant, _
      Quality As Long, _
      TimeStamp As Date, _
      Optional WaitTimeout As Long)
```

18.15.3. Parameters

Value - The new value to be written to the Item.

Item - The ItemId of the Item as an 8-byte array.

Quality - The quality of the new value to be written to the Item as a long integer.

Timestamp - The timestamp of the new value to be written to the Item as a DATE.

WaitTimeout - The time in milliseconds to block before timing out. Default is 5 minutes.

Note: If a value less than or equal to zero is specified, the default is used.

This page intentionally left blank

Chapter 19 QDataAccessMulti Object API Reference

QDataAccessMulti object that is directly created by a client provides IQDataAccessMulti interfaces.

It provides access to Exaquantum Item data distributed across multiple servers. Both synchronous and asynchronous access is provided.

Only the methods detailed in this section are supported on this interface, all other methods should not be used.

In addition to the specific errors listed, any method may return the standard system errors E_INVALIDARG (for a parameter of incorrect type) or E_OUTOFMEMORY.

All date/time parameters and results are in local time, by default.. This behavior may be overridden by calling the SetProcessOptions method setting the bUTC parameter to True (see section 18.10).

19.1. Add Method

19.1.1. Description

The client adds paths to item data that it is interested in. Data is fetched for the entire collection when ReadValue() is called.

Internally QDataAccessMulti calls the RBNSBrowse object which browses the Role-Based Namespace to fetch the ServerID, ServerName and ItemID for each Item path

19.1.2. Signature

```
Sub Add(ItemPath as String)
```

19.1.3. Parameters

ItemPath - A full path to an Item from the Role-Based Namespace for this user

19.1.4. Errors

May return any of the errors defined for QDataAccess::Add()

19.2. Clear Method

19.2.1. Description

Cancels any current data request, removes all added paths and clears all internally allocated resources.

19.2.2. Signature

```
Sub Clear()
```

19.3. Cancel Method

19.3.1. Description

Cancels any current read data request but maintains list of added Items.

19.3.2. Signature

```
Sub Cancel ()
```

19.4. ReadValue Method

19.4.1. Description

Reads the values of all added items using the current time and processing options set for the QDataAccessMulti Object.

If no query is currently active, internally it creates a Session2 object for each group of Items and sets the ServerName for that group. It also creates a QDataAccess object for each group and passes in the Session2 object. It then calls SetQueryTimes and SetProcessOptions passing on its own settings to each QDataAccess object and finally calls Execute() on each QDataAccess object to obtain the data asynchronously. Subsequent calls to ReadValue() retrieve data from the QDataAccess cache. If the query times are not set, updating live data is fetched so that the value in the cache is always up to date.

19.4.2. Signature

```
Function ReadValue() As Variant
```

19.4.3. Parameters

Data - A 2-dimensional array of VARIANTs with the following format:

BSTR ItemPath	VARIANT VQT

One row for each Item, where ItemPath is the added Item Path and VQT is a 2-dimensional VARIANT array with the format:

VARIANT Value	long Quality	DATE Timestamp

One row for each data point.

N.B. Item Paths are returned in the same order in which they were added.

The array for return is built by calling QDataAccess::ReadValue() specifying an individual ItemID for each Item Path in the collection in order.

If an Exaquantum server is unavailable, Item data for that server will appear with quality BAD/NOTAVAILABLE – when the server is restarted live and updating history data will automatically be recovered.

19.4.4. Errors

Any of the errors defined for QDataAccess::ReadValue() may be returned

19.5. SetQueryTimes Method

19.5.1. Description

Allows the period or point in time to be specified for the items of interest.

Cancels any current read query.

If times are specified as type string or string by reference, internally QDataAccessMulti calls QTimeHelper::ParseTimes() to validate the time formats.

19.5.2. Signature

```
Sub SetQueryTimes( _
       Optional SpotTime as Variant, _
       Optional StartTime as Variant, _
       Optional EndTime as Variant)
```

19.5.3. Parameters

SpotTime - If specified, return data at this time otherwise return the *latest data.*

If latest data is requested, the client will be notified when the data changes. (Must be specified in one of the formats described below.)

StartTime & **EndTime** - If only StartTime is specified then return data from this time to the latest data.

If both StartTime and EndTime are specified then return data over this period. (Must be specified in one of the formats described below.)

19.5.4. Errors

E_Q_STARTTIMEAFTEREND - Start time is later than end time (from QDataAccess).

IDS_ERR_InvalidTime - Invalid time format specified (from QTimeHelper)

19.6. SetReportTimes Method

19.6.1. Description

If times are specified to SetQueryTimes() as type string or string by reference, internally QDataAccessMulti passes the Report Times to QTimeHelper::SetReportTimes() followed by calling QTimeHelper::ParseTimes() to validate the time formats.

19.6.2. Signature

```
Sub SetReportTimes ( _
       Optional ReportStart as Variant, _
       Optional ReportEnd as Variant, _
       Optional SpotTime as Variant)
```

19.6.3. Parameters

ReportStart - Defines the time specification to be substituted if “ReportStart” is referenced in times specified to SetQueryTimes(). If not specified, defaults to “Now”.

ReportEnd - Defines the time specification to be substituted if “ReportEnd” is referenced in times specified to SetQueryTimes(). If not specified, defaults to “Now”.

SpotTime - Defines the time specification to be substituted if “SpotTime” is referenced in times specified to SetQueryTimes(). If not specified, defaults to “Now”.

19.6.4. Errors

IDS_ERR_InvalidTime - Invalid time format specified (from QTimeHelper)

19.6.5. Valid Time formats

Times may be specified to QDataAccessMulti in the following syntax (in BNF):

```
TimeSpec ::= ((<ABSOLUTEDATE>)  |  (<NOW> (+ | -) <number> <TIMELBL>)) |
( <PRODCALDATE> + <number> (<PRODCALPERIOD>)) | (<RANGE>) |
(<RESERVED> (+ | -) <number> <TIMELBL>))
```

<ABSOLUTEDATE> ::= System date format including time – always taken as client local machine time unless the bUTC flag is set in SetProcessOptions()

e.g. #31/12/00 01:00:00#

<NOW> ::= Current server time (taken from the Designated Server)

<number ::= (0..9)

<TIMELBL> ::= (S... | M... | H... | D...) – abbreviations for Seconds, Minutes, Hours, Days

e.g. “now – 1 hour”

< PRODCALDATE > ::= System date format excluding time

< PRODCALPERIOD >::= A valid Production Calendar Period name (as configured on the Designated Server)

e.g. “31/12/00 + 1 (SHIFT)”

<RANGE> ::= Where the StartTime is specified using the Production Calendar, “RANGE” may be used to specify the EndTime as the end of the specified Production Calendar Period.

`<RESERVED> ::= (NOW | REPORTSTART | REPORTEND | SPOTTIME)`

19.7. SetProcessOptions Method

19.7.1. Description

Allows the processing of the raw item data to be specified.

Cancels any current read query.

19.7.2. Signature

```
Sub SetProcessOptions( _
       Optional UpdateRate As Variant, _
       Optional InterpMethod As qdaInterpMethod = 0, _
       Optional EdgeValueOptions As qdaEdgeValueOptions = 0, _
       Optional TimeIncrement as Variant, _
       Optional ResampleInterval as Variant, _
       Optional bUTC As Boolean, _
       Optional TZIndex As Long = -1)
```

19.7.3. Parameters

UpdateRate - Specifies the maximum rate at which the client is to be notified of data changes. The default is 5 seconds. (Must be specified as DATE.)

InterpMethod – See Section 18.10 for detail

EdgeValueOptions - One of:

qdaIncludeEdge=0 – Include values at specified times. (default)

qdaIncludeBoundingValues =1 – Include bounding values. [See Note 1]

TimeIncrement - If this is specified, a timed request for history data will be made at the UpdateRate, incrementing the SpotTime or StartTime by this amount. If *latest data* is requested, “polling” is simulated by firing the latest value to the Client at the UpdateRate. (Must be specified as DATE.)

ResampleInterval - If this is specified, history data will be resampled to provide exactly one point per ResampleInterval. (Must be specified as DATE.)

BUTC - If set TRUE, timestamps specified in SetQueryTimes are interpreted as UTC and timestamps are returned in UTC. Defaults to FALSE where times are specified and returned as local time.

TZIndex - A long integer uniquely identifying the client Time Zone. If specified, time conversions to and from UTC will be performed using the specified Time Zone rather than the current Time Zone.

This allows conversions between UTC and local time to be performed on the server when the server and client are running in different time zones.

If the UTC flag is set TRUE, the value of TZIndex will be ignored and no local time conversions will be performed.

TZIndex will usually be obtained by calling method GetTZIndex() on QTimeHelper.

Note : Bounding values are returned when the requested times do not exactly lie on points stored in history. They are the first point before the start time and the last point after the end time. If Bounding Values are requested, data is returned directly from QHistorian and may in some circumstances comprise only one data point.

19.8. SetFilterParameter Method

19.8.1. Description

This method may be called before performing a read of history data to specify a filter to pre-process the data. Any current query is cancelled. For full details see section 18.12 SetFilterParameters Method.

19.9. AddWrite Method

19.9.1. Description

Writes the specified value, quality and timestamp to the specified Item.

Internally QDataAccessMulti queries the RBNSBrowse object which browses the Role Based Namespace to obtain ServerID, ServerName and ItemID. It groups Items according to Server.

19.9.2. Signature

```
Sub AddWrite (ItemPath as String, _
       Value As Variant, _
       Optional Quality As Variant, _
       Optional Timestamp as Variant)
```

19.9.3. Parameters

ItemPath - A fully specified path to the Item through the Role-Based Namespace.

Value - The new value to be written to the Item.

Quality - The quality of the new value to be written to the Item. Defaults to GOOD. If specified, must be long integer.

Timestamp - The timestamp of the new value to be written to the Item. Defaults to the current time. If specified, must be DATE.

19.9.4. Errors

Any error defined for IQItemWriter::Add() may be returned

E_Q_INVALIDPATH - The specified path does not exist or the user does not have access rights to view the namespace

E_Q_NOWRITEACCESS - The user does not have write access to the Item

19.10. ExecuteWrite Method

19.10.1. Description

Called by client to write all registered data to Exaquantum servers.

Internally creates a QItemWriter object for each Server group of Items and calls Add one or more times, and Execute on each. The returned status arrays are amalgamated for return to the client.

19.10.2. Signature

```
Function ExecuteWrite(Optional Reason As String = "") As Variant
```

19.10.3. Parameters

Reason - An optional string describing why the data is being written. Must be specified for CFR Part 11 compliance.

The Variant returned contains a 2-dimensional array of ItemPath/Status Code where the StatusCode is the "worst" return code for any value written to the item, i.e.

S_OK = all successful,

S_FALSE = at least one value not written to real-time database for this item,

<0 = error, one or more values not written to this item.

19.10.4. Errors

Any error defined for IQItemWriter::Execute() may be returned

19.11. CancelWrite Method

19.11.1. Description

Clears any Item data entered by the AddWrite() method before Execute() has been called.

19.11.2. Signature

```
Sub CancelWrite ()
```

19.12. Remove Method

19.12.1. Description

This method removes paths to specified items added using the Add() method.

19.12.2. Signature

```
Sub Remove (ItemPath As String)
```

19.12.3. Parameters

ItemPath - A full path to an Item from the Role-Based Namespace for this user

19.12.4. Errors

- May return any of the errors defined for QDataAccess::Remove()
- If called when object is in intrinsic mode, will return E_INVALIDARG

19.13. Execute Method

19.13.1. Description

Starts acquiring data asynchronously using current time and processing options set for the QDataAccessMulti object.

19.13.2. Signature

```
Sub Execute ( optional WithChanges As Boolean = TRUE, _
        optional ReturnDataAsReady As Boolean = TRUE )
```

19.13.3. Parameters

WithChanges - When requesting latest data, if set FALSE then only a single event will be fired to the client for each Item. The default is to notify the client whenever the value of any Item in the collection changes.

ReturnDataAsReady - When requesting historical data, if set FALSE then all the values for each Item will be returned in a single event. The default is to send data as soon as it is available to allow a more responsive user interface.

19.13.4. Errors

Can return any of the errors defined for:

- IQDataAccess::Add()
- IQDataAccess::Execute()

19.14. OnData Event Method

19.14.1. Description

This method is called back on the client for asynchronous events initiated by Execute().

19.14.2. Signature

```
Event OnData (ItemPath As String, _
      DataReturnType As qdaDataReturnType, _
      Data As Variant)
```

19.14.3. Parameters

ItemPath is the full path to an Item from the Role-Based Namespace for this user.

DataReturnType One of enumeration qdaDataReturnType :

qdaNew=0 – This is new data for this item. This is always the type for the first notification for an item after Execute(), and for latest data.

qdaMoreRows=1 – For trend data, this is additional data to append to previously received data.

qdaEnd=2 – For historical data, this is the final data for this Item.

Data - A 2-dimensional array of *Variants* with the following format:

Variant Value   *Long* Quality   *Date* Timestamp

where spot data has only one row and trend data has one row for each time point.

Note: **Data** may be NULL and clients should check for this. This indicates that no data is returned for this firing of the event.

19.15. OnError Event Method

19.15.1. Description

This method is called back on the client for asynchronous requests if an error occurs.

19.15.2. Signature

```
Event OnError(Number As Long, _
      Description As String, _
      Source As String, _
      HelpFile As String, _
      HelpContext As Long)
```

19.15.3. Parameters

Number - is the Quantum error number.

Description - is the error text.

Source - is the component that raised the error.

HelpFile - is the name of the HelpFile containing further information on the error.

HelpContext - is the Help context number.

This page intentionally left blank

Chapter 20 QAEAccess Object API Reference

The AEAccess Object allows Alarm and Event data history to be read for one or more OPC Gateway.

OPC Gateways of interest are specified by OPCAEPump Ids, these can be resolved the OPC Gateway name by retrieving the column OPCAEPumpHistId from the table OPCServer in the QConfig SQL Server database. Before Alarm and Event history can be read for a specific OPC Gateway, its OPCAEPumpHistId must first be added to a collection maintained by QAEAccess. After filling the collection, clients select the function they require either for the whole collection (typical) or for individual items in the collection.

20.1. Add Method

20.1.1. Description

Add an OPCAEPumpHistId to the collection of OPC Gateways of interest. On add, if a request for data is current, data for the new OPCAEPumpHistId will be fetched automatically.

20.1.2. Signature

```
Sub Add (AEPumpID As Variant)
```

20.1.3. Parameters

AEPumpId - specifies the OPC Gateway of interest. This should be an 8 element byte array as returned from column OPCAEPumpHistId from the table OPCServer.

20.2. Cancel Method

20.2.1. Description

Cancels the current request.

20.2.2. Signature

```
Sub Cancel ()
```

20.3. ClearFilterOptions Method

20.3.1. Description

Clears any filters set up by SetFilterOptions().

20.3.2. Signature

```
Sub ClearFilterOptions ()
```

20.4. Count Property

20.4.1. Description

Read the number of OPC Gateways in the collection of OPC Gateways of interest.

20.4.2. Signature

```
Property Count () As Long
```

20.5. Execute Method

20.5.1. Description

Starts the asynchronous acquiring of data using the current time, filter and processing options set for the QAEAccess Object.

Data is returned via the OnAE event.

Note: If the user calls the Execute Method before the previous Execute operation is complete/canceled, it returns an error.

20.5.2. Signature

```
Sub Execute(Optional ReturnEventsAsReady As Boolean = True)
```

20.5.3. Parameters

ReturnEventsAsReady - When requesting historical data, if set FALSE then all the values for each OPC Gateway will be returned in a single event. The default is to send data as soon as it is available to allow a more responsive user interface.

20.6. Item Method

20.6.1. Description

Return a OPCAEPumpHistId from the collection.

20.6.2. Signature

```
Function Item(Index As Variant) as Variant
```

20.6.3. Parameters

Index - is the position in the collection.

20.6.4. Return Value

The 8 element byte array representing the OPCAEPumpHistId.

20.7. OnAE Event Method

20.7.1. Description

This method is called back on the client for asynchronous events initiated by Execute().

20.7.2. Signature

```
Event OnAE(EventReturnType As qaeEventReturnType, _
      Events As Variant)
```

20.7.3. Parameters

EventReturnType - One of enumeration qaeEventReturnType from:

qeaNew=0 – This is new data for this OPCAEPumpHistId. This is always the type for the first notification for an OPCAEPumpHistId after Execute(). This is not used if Execute is called with ReturnEventsAsReady=False.

qeaMoreEvents=1 – This is additional data to append to previously received data. This is not used if Execute is called with ReturnEventsAsReady=False.

qeaEnd=2 – This is the final piece of data for this OPCAEPumpHistId.

Events   A 2-dimensional array of *Variant* with the following format:

- *Variant* OPCAEPumpHistId
- *String* Source
- *String* Message
- *Date* Timestamp
- *Long* SequenceNo
- *Long* Category
- *Long* Severity
- *Long* Cookie
- *String* ConditionName
- *String* Sub Condition Name
- *Long* Change Mask
- *Long* New State
- *Long* Condition Quality
- *Boolean* AckRequired
- *Date* Active Time
- *String* Actor ID
- *Long* NumberOfAttributes
- *Variant* ATTRIBUTES

one row for each Alarm/Event.

The Attributes column contains a 1-dimensional array of *Variant* attributes, the number and order of which are configured in the QConfig database for the particular Event Category.

The data is returned in time order from the oldest to the newest event with records from different OPC Gateways interleaved as necessary.

Note: **Events** may be NULL and clients should check for this. This indicates that no data is returned for this firing of the event.

Note: A Conditional event will include all attributes listed.

A Simple event will only include the attributes listed from Source to Cookie, all other attributes will be NULL.

A Tracking event will only include the attributes listed from Source to Cookie, plus Actor ID. All other attributes will be NULL.

Each event, regardless of type, will return values for OPCAEPumpHistId, NumberOfAttributes and ATTRIBUTES.

This information is as defined by the OPC Foundation for the OPC Alarm and Event Interface Specification.

20.8. OnError Event Method

20.8.1. Description

This method is called back on the client for asynchronous requests if an error occurs.

20.8.2. Signature

```
Event OnError(Number As Long, _
      Description As String, _
      Source As String, _
      HelpFile As String, _
      HelpContext As Long)
```

20.8.3. Parameters

Number - is the Quantum error number.

Description - is the error text.

Source - is the component that raised the error.

HelpFile - is the name of the HelpFile containing further information on the error.

HelpContext - is the Help context number.

20.9. ReadValue Method

20.9.1. Description

Perform a synchronous read for values of Alarm and Events data using the current time, filter and processing options set for the QAEAccess Object.

Data is read for all OPCAEPumpHistIds in the QAEAccess collection.

Note: If the user calls the ReadValue Method during asynchronous operations (Execute request) is in progress, it returns an error.

20.9.2. Signature

```
Function ReadValue () As Variant
```

20.9.3. Return Value

This is a 2-dimensional *Variant* array with the format:

- *Variant* OPCAEPumpHistId
- *String* Source
- *String* Message
- *Date* Timestamp
- *Long* SequenceNo
- *Long* Category
- *Long* Severity
- *Long* Cookie
- *String* ConditionName
- *String* Sub Condition Name
- *Long* Change Mask
- *Long* New State
- *Long* Condition Quality
- *Boolean* AckRequired
- *Date* Active Time
- *String* Actor ID
- *Long* NumberOfAttributes
- *Variant* ATTRIBUTES

One row for each Alarm/Event.

The Attributes column contains a 1-dimensional array of *Variant* attributes, the number and order of which are configured in the QConfig database for the particular Event Category.

The data is returned in time order from the oldest to the newest event with records from different OPC Gateways interleaved as necessary.

Note: The Return Value may be NULL and clients should check for this. This indicates that no data is returned for this query.

Note: A Conditional event will include all attributes listed.

A Simple event will only include the attributes listed from Source to Cookie, all other attributes will be NULL.

A Tracking event will only include the attributes listed from Source to Cookie, plus Actor ID. All other attributes will be NULL.

Each event, regardless of type, will return values for OPCAEPumpHistId, NumberOfAttributes and ATTRIBUTES.

This information is as defined by the OPC Foundation for the OPC Alarm and Event Interface Specification.

20.10. Remove Method

20.10.1. Description

Remove an OPCAEPumpHistId from the collection of OPC Gateways of interest.

20.10.2. Signature

```
Sub Remove(AEPumpID As Variant)
```

20.10.3. Parameters

AEPumpId - specifies the OPC Gateway of interest. This should be an 8 element byte array from column OPCAEPumpHistId from the table OPCServer.

20.11. SetFilterOptions Method

20.11.1. Description

Allows filtering of the Alarm and Event data that will be returned to be specified.

Cancels any current query.

20.11.2. Signature

```
 Sub SetFilterOptions(
        FilterField As qdaAEFilterField, _
        FilterOperator As qdaAEFilterOperator, _
        varMatch As Variant)
```

20.11.3. Parameters

FilterField - is one of enumeration qdaAEFilterFields:

qdaAEFilterMessage=2 – Filter on the Message field.

qdaAEFilterGeneric=3 – Use the varMatch parameter as a valid SQL string to be applied directly to the SQL query.

FilterOperator - is one of enumeration qdaAEFilterOperator:

qdaAEFilterLikeOperator=1 – Match field specified in FilterField with value specified in varMatch allowing wildcard characters '*'.

varMatch - is the filter operator parameter. May contain the wildcard “*” to match many or no characters and the wildcard “?” to match any single character.

Both or neither filter options may be set.

The filters are applied in the form “<alarm/event query> WHERE (message LIKE <MsgFilter>) AND (<Filter>)”. For example, to return events that have a Severity of 500 and a Data Value of more than 20 use the string “(Severity = 500) AND (DataValue > ‘20’)”.

It is important that this field is set correctly as no validation is carried out. The names of the fields for comparison can be obtained from the event storage tables in the Historian database.

20.12. SetProcessOptions Method

20.12.1. Description

Allows the processing of Alarm and Event data to be specified.

Cancels any current query.

20.12.2. Signature

```
Sub SetProcessOptions(Optional UpdateRate As Variant, _
      Optional ActiveTime As Variant, _
      Optional MaximumEvents As Long = -1)
```

20.12.3. Parameters

UpdateRate - specifies the rate at which history should be polled to check for new alarms/events (must be specified as DATE). The DATE is converted to an interval by subtracting the earliest date that can be represented by the DATE data type i.e. midnight on 30th December 1899. This allows VB assignments such as #00:00:05# for 5 seconds or #00:02:00# for 2 minutes.

Note: Specifying **UpdateRate** as Null or Empty is equivalent to not supplying it.

ActiveTime - specifies the time band before the previous EndTime which should be re-queried to catch any late-arriving events (must be specified as DATE). See UpdateRate for details of how the DATE data type is interpreted as an Interval.

Note: Specifying **ActiveTime** as Null or Empty is equivalent to not supplying it.

MaximumEvents - specifies the maximum number of Alarm and Event records that will be returned to the client. Processing begins at the specified start time, with data returned to the client for subsequent times until this limit is reached. At this point any additional events are discarded and not returned to the client. This option allows the client to limit the number of events returned in the case where this may be very large. MaximumEvents is ignored if an UpdateRate is specified.

20.13. SetQueryTimes Method

20.13.1. Description

Allows the period to be specified for the portion of Alarm and Event history of interest.

Cancels any current query.

20.13.2. Signature

```
 Sub SetQueryTimes(Optional StartTime As Variant, _
       Optional EndTime As Variant)
```

20.13.3. Parameters

StartTime & **EndTime** - If only StartTime is specified, then return data from this time to the latest data. If both StartTime and EndTime are specified, then return data over this period (must be specified as DATE).

Note: Specifying **StartTime** or **EndTime** as Null or Empty is equivalent to not supplying them.

20.14. SetSession Method

20.14.1. Description

This method may be called by a client to allow the QAEAccess object to share its Session connection. This is the recommended way to use QAEAccess.

If this method is not called before a request to fetch data, the QAEAccess object will create its own Session object.

20.14.2. Signature

```
Sub SetSession (pSession As Object)
```

20.14.3. Parameters

pSession - A pointer to the client's Session object.

This page intentionally left blank

Chapter 21 QQualityHelper Object API Reference

The QQualityHelper object can be used to interpret the 32 bit Raw data quality code for an Item.

The Raw Data Quality of an Item encodes both Primary and Secondary qualities. The QQualityHelper object has enumeration qdaQuality and qdaSecondaryQuality to determine the Primary and Secondary Data Qualities respectively.

21.1. IsBad Method

21.1.1. Description

Tests whether the Primary Data Quality is Bad based on the Raw Data Quality passed in.

21.1.2. Signature

```
Function IsBad(RawQuality As Long) as Boolean
```

21.1.3. Parameters

RawQuality - specifies the Raw Data Quality.

21.1.4. Return Value

Returns a TRUE or FALSE.

21.2. IsGood Method

21.2.1. Description

Tests whether the Primary Data Quality is Good based on the Raw Data Quality passed in.

21.2.2. Signature

```
Function IsGood(RawQuality As Long) as Boolean
```

21.2.3. Parameters

RawQuality - specifies the Raw Data Quality.

21.2.4. Return Value

Returns TRUE or FALSE.

21.3. IsUncertain Method

21.3.1. Description

Tests whether the Primary Data Quality is Uncertain based on the Raw Data Quality passed in.

21.3.2. Signature

```
Function IsUncertain(RawQuality As Long) as Boolean
```

21.3.3. Parameters

RawQuality - specifies the Raw Data Quality.

21.3.4. Return Value

Returns TRUE or FALSE.

21.4. IsAssumed Method

21.4.1. Description

Tests whether the Primary Data Quality is Assumed, based on the Raw Data Quality passed in.

21.4.2. Signature

```
Function IsAssumed(RawQuality As Long) as Boolean
```

21.4.3. Parameters

RawQuality - specifies the Raw Data Quality.

21.4.4. Return Value

Returns TRUE or FALSE.

21.5. Quality Method

21.5.1. Description

Returns an enumeration for the Primary Data Quality based on the Raw Data Quality passed in.

21.5.2. Signature

```
Function Quality(RawQuality As Long) as qdaQuality
```

21.5.3. Parameters

RawQuality- specifies the Raw Quality for an Item.

21.5.4. Return Value

One of enumerated qdaQuality is returned:

qdaQualityGood=0 – The Primary Quality is Good

qdaQualityUncertain=1 – The Primary Quality is Uncertain

qdaQualityBad=2 – The Primary Quality is Bad

qdaQualityAssumed=3 – The Primary Quality is Assumed

qdaQualityUnknown=999– The Primary Quality is Unknown

21.6. QualityText Method

21.6.1. Description

Returns the Primary Quality Text as a string based on the Raw Data Quality passed in.

21.6.2. Signature

```
Function QualityText(RawQuality As Long) as String
```

21.6.3. Parameters

RawQuality - specifies the Raw Data Quality for an Item.

21.6.4. Return Value

Returns Primary Raw Data Quality as a string.

21.7. SecondaryQuality Method

21.7.1. Description

Returns an enumeration for the Secondary Data Quality based on the Raw Data Quality passed in.

21.7.2. Signature

```
Function SecondaryQuality(RawQuality As Long) As qdaSecondaryQuality
```

21.7.3. Parameters

RawQuality - specifies the Raw Data Quality for an Item.

21.7.4. Return Value

One of enumerated qdaSecondaryQuality is returned.

qdaSecondaryQualityNone=0 – None

qdaSecondaryQualityBadCalc=1 – Bad calculation result.

qdaSecondaryQualityEstimated=2 – Estimated aggregation result.

qdaSecondaryQualityOffline=3 – Item offline.

qdaSecondaryQualityOfflineArch=4 – Data not available because archive is offline.

qdaSecondaryQualityNotAvail=5 – Data not available for the specified time.

qdaSecondaryQualityShutdown=6 – Item shutdown marker.

qdaSecondaryQualityOPCError=7 – OPC Communications error.

qdaSecondaryQualityTimeChange=8 – Aggregations have detected a time change.

qdaSecondaryQualityBeginCatchup=9 – Historical data was requested from OPC Server.

qdaSecondaryQualityDeleted=10 – Deleted.

qdaSecondaryQualityReplaced=11 – Offline Item has been manually overwritten.

qdaSecondaryQualityUnknown=9 – Unknown.

21.8. SecondaryQualityText Method

21.8.1. Description

Returns the Secondary Quality Text as a string based on the Raw Data Quality passed in.

21.8.2. Signature

```
Function SecondaryQualityText(RawQuality As Long) As String
```

21.8.3. Parameters

RawQuality - specifies the Raw Data Quality for an Item.

21.8.4. Return Value

Returns Secondary Data Quality as a string.

21.9. RawQuality Method

21.9.1. Description

Returns the Raw Data Quality based on the enumerations qdaQuality and qdaSecondaryQuality passed in.

21.9.2. Signature

```
Function RawQuality(Quality As qdaQuality, _
     Optional SecondaryQuality As qdaSecondaryQuality) As Long
```

21.9.3. Parameters

Quality - specifies the enumerated Primary Quality for an Item.

SecondaryQuality - specifies the enumerated Secondary Quality for an Item. If not supplied then assumes

21.9.4. Return Value

Returns the Raw Data Quality based on the combination of Primary Quality and Secondary Quality passed in.

21.10. QualityTexts Property

21.10.1. Description

Returns a pointer to a collection of Primary Quality Text strings.

This may be used by a Client to present a list of possible Primary Qualities to a user.

21.10.2. Signature

```
Property QualityTexts () As Object
```

21.11. SecondaryQualityTexts Property

21.11.1. Description

Returns a pointer to a collection of Secondary Quality Text strings.

This may be used by a Client to present a list of possible Secondary Qualities to a user.

21.11.2. Signature

```
Property SecondaryQualityTexts () As Object
```

Chapter 22 Recalculation API Reference

User will use IQRecalcAuto interfaces to conduct recalculation.

For any optional parameters, VARIANT types of VT_ERROR, VT_EMPTY or VT_NULL will be accepted as missing parameters.

For any method, the standard automation errors such as E_INVALIDARG or E_OUTOFMEMORY may be returned in addition to any method-specific errors.

Additional general errors may be raised by the recalculation engine – these will also be logged in the Application EventLog to aid in error diagnosis.

NOTE: The Session3 object must be used to obtain a RecalcJob object if the API code is to be run on an Exaquantum Server with the following DCOM settings:

Default Authentication Level: None

Default Impersonation Level: Identify

General errors

E_RECALC_UNEXPECTED (58432) - An unexpected error has occurred - generally indicates that an exception has been thrown.

E_RECALC_ERROR (58433) - An internal error has occurred during processing – further details should be available in the Application EventLog.

22.1. NewRecalcJobDefinition method

22.1.1. Description

The user calls this method to create a new recalculation job definition. If successful, the RecalcJobName property is set to the new job definition.

22.1.2. Signature

```
Sub NewRecalcJobDefinition ( _
    ByVal Name As String, _
    Optional ByVal Reason As String = "")
```

22.1.3. Parameters

Name - A unique name to identify this recalculation job. May be up to 256 characters.

Reason - The reason for this recalculation. May be up to 2000 characters.

22.1.4. Errors

E_JOBEXISTS (58435) - A recalculation job already exists with the specified name

E_INVALIDSTRING (58450) - String parameter supplied is either greater than maximum allowed length or empty.

22.2. SetRecalcInputItems method

22.2.1. Description

The user calls this method after creating a new RecalcJobDefinition to specify the items whose values have been updated in history and whose dependent items should be recalculated. If changes to the list of items are required, the whole list should be re-specified. If a failure status code is returned for any Item, this Item is not added to the RecalcInputItem table.

22.2.2. Signature

```
Function SetRecalcInputItems (ByVal ItemPaths As Variant) As Variant
```

22.2.3. Parameters

ItemPaths - May be a single ItemPath as a BSTR, or a SAFEARRAY of ItemPaths to specify multiple items.

For R2.01.50 only aggregation result items may be specified. An error is returned if any other item type is specified.

The return value is a single HRESULT defining the status of the requested ItemPath, or if an array of ItemPaths is specified, a SAFEARRAY of HRESULTS, one for each ItemID in the input array.

22.2.4. Item Status Codes to be returned

S_OK - The Item is valid and was successfully added as an input Item for this RecalcJobDefinition

E_ITEMNOTFOUND (58440) - The Item does not exist

E_INVALIDITEMTYPE (58441) - The Item is not a valid type for recalculation. This error is returned if a reference data Item is specified.

For R2.01.50 this error is returned if the Item is not an Aggregation Result Item.

E_DUPLICATEITEM (58451) - This Item has already been specified.

E_DEPENDENTITEM (58452) - This Item is a dependent item of another specified Recalc Input Item.

22.2.5. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

22.3. DeleteRecalcJobDefinition method

22.3.1. Description

The user calls this method to delete a specified job definition.

22.3.2. Signature

```
Sub DeleteRecalcJobDefinition (ByVal JobName As String)
```

22.3.3. Parameters

RecalcJobName - A unique name to identify an existing recalculation job to delete. May be up to 256 characters.

22.3.4. Errors

E_JOBNOTFOUND (58436) - No recalculation job was found with the specified name

E_JOBINUSE (58437) - This recalculation job is in use by another user

22.4. RecalcJobName property

22.4.1. Description

The user sets this property to specify an existing job definition to use. Only one user may use a given recalculation job definition at one time.

22.4.2. Signature

```
Property Let RecalcJobName (ByVal Name As String)
Property Get RecalcJobName () As String
```

22.4.3. Parameters

Name - A unique name to identify this recalculation job. May be up to 256 characters. When setting the property, any existing JobName is cleared. The property may be set to an empty string.

22.4.4. Errors

E_JOBNOTFOUND (58436) - No recalculation job was found with the specified name

E_JOBINUSE (58437) - This recalculation job is in use by another user

22.5. SetRecalcTimes method

22.5.1. Description

The user calls this method to set the time range for recalculation. Either SpotTime should be specified for a single point, or StartTime and optionally EndTime to recalculate a range of data. If StartTime is specified and EndTime is not specified, recalculation will continue to the current systemtime.

22.5.2. Signature

```
Sub SetRecalcTimes( _
     Optional ByVal SpotTime As Variant, _
     Optional ByVal StartTime As Variant, _
     Optional ByVal EndTime As Variant)
```

22.5.3. Parameters

SpotTime - The local timestamp of a single updated point to use to recalculate dependent items. If specified, must be type VT_DATE. If SpotTime is specified, StartTime and EndTime will be ignored.

StartTime - The inclusive local start time of a range of data points to use to recalculate dependent items. If specified, must be type VT_DATE.

EndTime - The inclusive local end time of a range of data points to use to recalculate dependent items. If specified, must be type VT_DATE. If not specified and StartTime is specified, EndTime is taken to be the current system time. If SpotTime is specified, StartTime and EndTime will be ignored.

22.5.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

E_STARTAFTEREND (58442) - StartTime specified is later than EndTime

E_FUTURETIME (58443) - Future time is specified

22.6. Recalc method

22.6.1. Description

The user calls this method to perform the recalculation. A Recordset of recalculated values is returned for inspection. The user is not allowed to modify the recalculated values – they can only Commit all values to history or Cancel the recalculation. The QConfig database RecalcSession table is populated to indicate that a recalculation is active and no other recalculation may be performed from the beginning of the Recalc() method until completion of a Commit() or Cancel() call for this recalculation.

If any error occurs during recalculation, the QConfig database RecalcSession table is cleared of data. Additional error information will be logged to the Windows Application Event Log.

22.6.2. Signature

```
Function Recalc( _
     Optional ByVal bAutoCommit = False, _
     ItemValueRS As Variant) As Long
```

22.6.3. Parameters

bAutoCommit - Default value is false.

Set to VARIANT_TRUE to specify that Commit() of data should be performed automatically when recalculation is complete.

ItemValuesRS - If AutoCommit is not specified, an ADO Recordset of recalculated Item values is returned so that the user can inspect the data that will be written to history and decide whether to Commit or Cancel.

Recordset fields:

- ItemID - long
- ItemPath - BSTR
- Timestamp - DATE (as local time)
- Value - VARIANT
- Quality - long
- PreviousValue - VARIANT (may be NULL)
- PreviousQuality - long (may be NULL)
- PreviousTimestamp - DATE (may be NULL)

Note: Only one value will be returned per Item at a given timestamp.

PreviousValue, PreviousQuality and PreviousTimestamp will be NULL where a previous point did not exist in history, e.g. for a current Aggregation estimated value.

Where a previous value does exist, PreviousTimestamp will only be different from the new timestamp when the timestamp of an Aggregation Minimum or Maximum has changed.

The **return value** is a unique ID assigned for this calculation and referenced in the recalculation audit logs.

22.6.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

E_Q_RECALCINPROGRESS (58444) - Recalculation in progress

E_Q_NORECALCITEMS (58445) - No recalculation input items are defined for the current RecalcJobDefinition

E_Q_NORECALCTIMES (58446) - No times have been specified for the recalculation. This error can only be returned before a recalculation job has been successfully run and committed. For subsequent recalculations the last committed run times will be used if no new times are specified.

E_ITEMNOTFOUND (58440) - The Item does not exist

E_INVALIDITEMTYPE (58441) - The Item is not a valid type for recalculation. This error is returned if a reference data Item is specified.

For R2.01.50 this error is returned if the Item is not an Aggregation Result Item.

E_NOITEMSUPDATED (58453) - No item values were updated by this recalculation

22.7. Commit method

22.7.1. Description

The user calls this method to write all recalculated data to history. Any updated "live" aggregation results are notified to the Exaquantum RTDB. Control is not returned to the user until data has been written to Historian and the QConfig database RecalcSession table is cleared of data.

If one or more current aggregation values are recalculated and a new "live" aggregation period completes between the Recalc and the Commit, an error is returned and the recalculation must be re-executed.

22.7.2. Signature

```
Sub Commit (ByVal RecalcSessionID As Long)
```

22.7.3. Parameters

RecalcSessionID - The unique ID assigned for this recalculation.

22.7.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

E_Q_NORECALC (58447) - No recalculation is waiting for commit.

E_Q_INVALIDSESSIONID (58448) - This RecalcSessionID is not in progress or does not belong to this RecalcJob object.

E_Q_LIVEAGGPERIODPASSED (58449) - A live aggregation period has expired since the recalculation was run. The recalculation must be re-executed.

E_Q_ITEMSNOTCOMMITTED (58454) - One or more recalculated points were not successfully written to History. See the Application Event Log for details of failure.

22.8. Cancel method

22.8.1. Description

The user calls this method to terminate the current recalculation session and discard all recalculated data. The QConfig database RecalcSession table is cleared of data. The Cancel() method may only be called after a successful Recalc() which is not committed.

22.8.2. Signature

```
Sub Cancel (ByVal RecalcSessionID As Long)
```

22.8.3. Parameters

RecalcSessionID - The unique ID assigned for this recalculation.

22.8.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

E_Q_NORECALC (58447) - No recalculation is in progress.

E_Q_INVALIDSESSIONID (58448) - This RecalcSessionID is not in progress or does not belong to this RecalcJob object.

22.9. RecalcInputItems property

22.9.1. Description

A read-only property to allow the user to view the input items defined for the current job.

22.9.2. Signature

```
Property Get RecalcInputItems () As Variant
```

22.9.3. Parameters

InputItemsRS - An ADO Recordset containing a list of all the Recalc Input Items defined for the current RecalcJob.

Recordset fields are:

- ItemID - long
- ItemPath - BSTR

22.9.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

22.10. RecalcJobDefinition property

22.10.1. Description

A read-only property returning details about the current job.

22.10.2. Signature

```
Property Get RecalcJobDefinition () As Variant
```

22.10.3. Parameters

The **return value** is a variant containing an ADO Recordset containing results from a query of the QConfig RecalcJobDefinition table for the current job.

22.10.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

22.11. Reason property

22.11.1. Description

The user may set this property to alter the Reason associated with the current RecalcJobName at any time after creation of the Recalculation Job Definition.

22.11.2. Signature

```
Property Let Reason (ByVal Reason As String)
Property Get Reason () String
```

22.11.3. Parameters

Reason - The reason for this recalculation. May be up to 2000 characters. The property may be set to an empty string.

22.11.4. Errors

E_NOJOB (58438) - The RecalcJobName property has not been set.

E_INVALIDSTRING (58450) - String parameter supplied is greater than maximum allowed length.

22.12. GetDependentItems method

22.12.1. Description

This method may be called without setting the RecalcJobDefinition property to determine all Items that are immediately dependent upon the specified Item ID. This would typically be called by a user-interface application to display Items affected by changing a highlighted Item path.

22.12.2. Signature

```
Function GetDependentItems (ByVal ItemPath As String) As Variant
```

22.12.3. Parameters

ItemPath - A single Item path.

The **return value** is a variant containing an ADO Recordset listing all Items dependent upon the specified ItemID.

Recordset fields:

- ♦ ItemID
- ♦ ItemPath
- ♦ ItemTypeID (aggregation or calculation)

  3 = aggregation

  4 = calculation

22.12.4. Errors

E_ITEMNOTFOUND (58440) - The Item does not exist

22.13. GetRecalcJobs method

22.13.1. Description

This method may be called without setting the RecalcJobDefinition property to return a recordset of all configured Recalc Job Definitions.

22.13.2. Signature

```
Function GetRecalcJobs () As Variant
```

22.13.3. Parameters

The **return value** is a variant containing an ADO Recordset listing all configured Recalculation Job Definitions, ordered in descending time order by LastRunDate.

Recordset fields:

- Name
- CreationDate
- Reason
- LastRunDate
- LastRecalcStartTime
- LastRecalcEndTime
- LastRunUserId
- LastMachineName
- LastCommitStatus

Chapter 23 Exaquantum OPC Server API

23.1. Introduction

This chapter describes aspects of the interfaces supported by the Exaquantum OPC Server component.

A comprehensive description of the full OPC interface is available in the documentation made available by the OPC foundation. The documents can be downloaded from the OPC Foundation Web sites at: http://www.opcfoundation.org/

The interfaces supported by the Exaquantum OPC are listed in the following section.

The rest of the chapter provides extra information concerning the Exaquantum OPC server implementation and highlights specific behavior and any differences from the published standard.

23.2. OPC Compliance

23.2.1. OPC HDA Server

23.2.1.1. Custom Interface

The supported OPC HDA interfaces Version 1.1 are:

- IOPCCommon Version 1.0
- IOPCHDA_Server
- IOPCHDA_Browser
- IOPCHDA_SyncRead
- IOPCHDA_SyncUpdate

The following OPC HDA interfaces are not supported:

- IOPCHDA_SyncAnnotations
- IOPCHDA_AsyncRead
- IOPCHDA_AsyncUpdate
- IOPCHDA_AsyncAnnotations
- IOPCHDA_Playback

- The supported OPC HDA automation interfaces Version 1.0 are:

Table 23-1

Object Name	Method / Property Name	Description
OPCHDAServer	OPCHDAItems Property	HDA Item Property
	Connect	Connect
	Disconnect	disconnect
OPCHDAItem	Parent Property	Parent group object
	Count Property	Number of connection
	AddItem	Add one item
	AddItems	Add many items
	Remove	Remove one item
	RemoveAll	Remove all items
	SyncInsertReplace	In case of no data then data insert, in case of data had already set then data replace
	SyncDeleteRaw	Delete date range

23.2.2. OPC DA Server

The following custom interfaces are supported, partially compliant with OPC DA Version 2.05A.

- IOPCCommon Version 1.0
- IOPCServer
- IOPCItemMgt
- IOPCGroupStateMgt
- IOPCSyncIO
- IOPCAsyncIO2
- IOPCItemProperties (partial, LookupItemIDs is not implemented)
- IOPCBrowseServerAddressSpace (partial, BrowseAccessPaths is not implemented)

The following OPC DA interfaces are not supported:

- IOPCServerPublicGroups
- IPersistFile
- IOPCPublicGroupStateMgt

23.3. IOPCCommon Interface

Exaquantum supports only the System Default LocaleID.

This method ignores the entered locale and returns S_OK.

23.3.1. IOPCCommon::GetLocaleID

Exaquantum OPCHDA server behavior:

Exaquantum returns the System Default LocaleID.

23.3.2. IOPCCommon:: QueryAvailableLocaleIDs

Exaquantum OPCHDA server behavior:

This method returns a single LocaleID: the System Default LocaleID.

23.3.3. IOPCCommon::GetErrorString

Exaquantum OPCHDA server behavior:

This method calls the ::FormatMessage API to obtain an error description for standard Windows errors or for Exaquantum OPC HDA Server-specific errors. The only OPC-defined errors and information codes returned by the Exaquantum OPC HDA server are:

- OPC_UNKNOWNITEMID
- OPC_E_INVALIDHANDLE
- OPC_E_UNKNOWNATTRID
- OPC_E_NOT_AVAIL
- OPC_S_MOREDATA
- OPC_S_NODATA
- OPC_S_CURRENTVALUE

Calling this method with any other OPC-defined code will return E_INVALIDARG.

23.3.4. IOPCCommon::SetClientName

Exaquantum OPCHDA server behavior:

This method is ignored – Exaquantum returns S_OK.

23.4. Historical Data Access

A stand-alone component, QOPCHDAServer.exe, supports OPC Historical Data Access (HDA) to Exaquantum. There are two main HDA server objects which may be used, differing only in the behavior of the IOPCHDAServer_SyncRead::ReadProcessed() method when the HDA_INTERPOLATIVE aggregate is requested (see Section 23.6 for further details).The creatable version dependent ProgIds for the HDA server objects are:

QOPCHDAServer.HDAServer.1

which returns processed data with up to 4 points per interval when OPCHDA_INTERPOLATIVE is requested. This behavior is optimized for graphical trend displays to include minimum and maximum values for each interval but does not conform to the OPC HDA 1.2 specification.

And

QOPCHDAServer.HDAServer2.1

which returns a single spot value at the start of each sample interval. This conforms to the OPC HDA 1.2 Specification.

A third object is implemented which returns filtered data rather than raw data to allow more efficient operation for OPC clients that could require this behavior. See IOPCHDA_SyncRead::ReadRaw. The creatable version dependent ProgId for the HDA server object is:

QOPCHDAServer.HDAServerEx.1

In the following pages, the Exaquantum-specific behavior is noted for each method. For a fuller discussion of OPC HDA functionality, and definition of interface behavior, refer to the OPC Foundation document, "OPC Historical Data Access Custom Interface Standard Version 1.1".

23.4.1. IOPCHDA_Server::GetItemAttributes

Exaquantum behavior:

Exaquantum can return data for the following six attributes:

- OPCHDA_ITEMID
- OPCHDA_DATA_TYPE
- OPCHDA_DESCRIPTION
- OPCHDA_ENG_UNITS
- OPCHDA_NORMAL_MAXIMUM
- OPCHDA_NORMAL_MINIMUM

Exaquantum OPC HDA Server browsing supports filtering on Item path (OPCHDA_ITEMID) only.

23.4.2. IOPCHDA_Server::GetAggregates

Exaquantum behavior:

Exaquantum can return data for the following aggregates only:

- OPCHDA_INTERPOLATIVE
- OPCHDA_TIMEAVERAGE
- OPCHDA_STDEV
- OPCHDA_MINIMUM
- OPCHDA_MAXIMUM
- OPCHDA_MINIMUMACTUALTIME
- OPCHDA_MAXIMUMACTUALTIME
- OPCHDA_START
- OPCHDA_END
- OPCHDA_RANGE
- OPCHDA_PERCENTGOOD
- OPCHDA_TOTAL

23.4.3. IOPCHDA_Server::GetHistorianStatus

Exaquantum behavior:

Status is returned as OPCHDA_UP if the Exaquantum server is running and HDAServer has successfully connected, otherwise OPCHDA_DOWN.

StartTime is returned as FILETIME 0 (1st January 1601).

MaxReturnValues is set to zero as the number of values returned is limited only by available memory.

StatusString is always returned as an empty string.

23.4.4. IOPCHDA_Server::GetItemHandles

Exaquantum behavior:

This method expects the ItemIDs will be specified as an array of full Exaquantum Item Path names, for example:

- Root.Folder1.Function Block1.Tag1.Value
- Root.Folder1.Function Block1.Tag1.Description
- Root.Folder1.Function Block1.Tag1.HighEng
- Root.Folder1.Function Block1.Tag1.LowEng
- Root.Folder1.Function Block1.Tag1.Units
- Root.Folder1.Function Block1.Tag1.Aggregations.Hour.Mean.Value
- Root.Folder1.Function Block1.Tag1.Aggregations.Day.Minimum.Value

Server handles returned are an array of corresponding Exaquantum Item IDs as unique long integers. The Server Handles are ordered corresponding to the input ItemIDs array.

Any Item path not found will have Server Handle returned as 0 and the corresponding Error set to the standard OPC error OPC_E_UNKNOWNITEMID.

Returns E_Q_OPCHDA_NOTCONNECTED if Exaquantum server is not running.

23.4.5. IOPCHDA_Server::ReleaseItemHandles

Exaquantum behavior:

This method removes the association of the specified Exaquantum Item ID with a previously supplied Client Handle. If the Exaquantum ItemID is not associated with a registered Client Handle, the corresponding error is set to the standard OPC error OPC_E_INVALIDHANDLE

If GetItemHandles() is called multiple times for the same Item path, the server will not remove the server handle until ReleaseItemHandles() has been called a corresponding number of times. Subsequent calls to read methods succeed until the server handle is removed.

23.4.6. IOPCHDA_Server::ValidateItemIDs

Exaquantum behavior:

This method expects the ItemIDs will be specified as an array of full Exaquantum Item Path names, for example:

Root.Folder1.Function Block1.Tag1.Value

Any Item path not found will have the corresponding Error set to the standard OPC error OPC_E_UNKNOWNITEMID.

Returns E_Q_OPCHDA_NOTCONNECTED if Exaquantum server is not running.

23.4.7. IOPCHDA_Server::CreateBrowse

Exaquantum behavior:

Exaquantum OPC browsing presents a flattened view of the Exaquantum folder hierarchy. HDA browsing supports only a single filter on full Item Path name.

If a filter other than OPCHDA_ITEMID is specified, or more than one filter is specified, the corresponding Error code is set to the standard OPC error OPC_W_NOFILTER.

The single filter operator may be specified as any one of:

- OPCHDA_EQUAL
- OPCHDA_LESS
- OPCHDA_LESSEQUAL
- OPCHDA_GREATER
- OPCHDA_GREATEREQUAL
- OPCHDA_NOTEQUAL

The single filter value must be supplied as a string with optional wildcards, for example, Root.Folder1.*

The list of item paths returned includes all raw items matching the specified filter. Reference item paths (Description, Units, HighEng and LowEng) are not returned.

Wildcards supported are:

*	Zero or more characters in that position
?	One character at that position

The Exaquantum OPC Server does not support # as a wildcard due to the possible valid inclusion of the # character as part of an item path. The pattern [0-9] can be used to provide the same functionality.

In addition Microsoft SQL Server pattern matching may be used:

%	Zero or more characters in that position
_	One character at that position
[]	Any single character within the specified range or set, e.g. [0-9], [abc]
[^]	Any single character not within the specified range or set, e.g. [^abc]

Aggregation result items may be returned by changing the registry value:

```
HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\OPC
Server\BrowseAggregations
```

- Type=DWORD
- Default Value = 0 (do not return aggregation)
- A Non-zero value will include aggregation result items matching the filter.

23.4.8. IOPCHDA_Server::DeleteAtTime

Exaquantum behavior:

Archived data cannot be deleted.

This method expects the time parameter to be supplied in local time.

23.5. IOPCHDA_Browser interface

This is a required interface implemented by the HDABrowser object that is obtained through the IOPCHDA_Server::CreateBrowse() method.

23.5.1. IOPCHDA_Browser::GetEnum

Exaquantum behavior:

If BrowseType of OPCHDA_BRANCH is requested, the enumerator is set to NULL and the return code is S_FALSE.

For all other values of BrowseType the returned enum set contains a list of the full paths of all historized Value or Aggregation result Items that match the filter criterion supplied in the IOPCHDA_Server::CreateBrowse() method, starting at the current browse position. Item paths are in ascending alphabetical order. The initial browse position is always set to the first alphabetical Item path.

23.5.2. IOPCHDA_Browser::ChangeBrowsePosition

Exaquantum behavior:

This method has no effect as Exaquantum presents a flat address space for browsing. The return code is set to S_OK.

23.5.3. IOPCHDA_Browser::GetItemID

Exaquantum behavior:

As the enumerated string set obtained from the GetEnum() method already contains the full Item path, a copy of the NodeName is made to the output ItemID parameter. No check for validity is made.

23.5.4. IOPCHDA_Browser::GetBranchPosition

Exaquantum behavior:

This method sets the output BranchPos parameter to the hard coded string “Root”.

23.6. IOPCHDA_SyncRead interface

This is a required interface implemented by the HDAServer object. The only methods implemented are ReadRaw(), ReadProcessed() and ReadAttribute() – all other methods return E_NOTIMPL.

23.6.1. IOPCHDA_SyncRead::ReadRaw

Exaquantum behavior:

Exaquantum never returns trailing bound values.

The OPCHDA_QUALITY bits (bits 16-31) of the quality DWORD are always set to OPCHDA_RAW, or OPCHDA_NODATA if not available. The Data Access quality bits are returned directly from Exaquantum (bits 0-15). If the Exaquantum primary quality is ASSUMED (i.e. the tag has been set offline) bits 0-15 are set to OPC_QUALITY_UNCERTAIN.

If data does not exist in the requested time period, a valid point is returned with Data Access quality set to BAD and the error return is S_OK.

Exaquantum does not support calculation of StartTime which must therefore always be specified.

If a spot time of “now” is requested latest data is always returned.

Returns E_Q_OPCHDA_NOTCONNECTED if Exaquantum server is not running.

In most cases, if the HDAServerEx object is used, a call to ReadRaw() is converted to a call to ReadProcessed(), in order to return filtered data. However, when dwNumValues is specified to be less than or equal to 8, raw data is returned. To give as close a representation of the trend data as possible, the number of points requested will be returned as minimum and maximum values. OPC_S_MOREDATA will never be returned.

23.6.2. IOPCHDA_SyncRead::ReadProcessed

Exaquantum behavior:

Exaquantum only supports the following aggregates.:

OPCHDA_INTERPOLATIVE

OPCHDA_TIMEAVERAGE

OPCHDA_STDEV

OPCHDA_MINIMUM

OPCHDA_MAXIMUM

OPCHDA_MINIMUMACTUALTIME

OPCHDA_MAXIMUMACTUALTIME

OPCHDA_START

OPCHDA_END

OPCHDA_RANGE

OPCHDA_PERCENTGOOD

OPCHDA_TOTAL

If a spot time is requested the ResampleInterval is ignored. If a spot time of “now” is requested latest data is always returned.

OPCHDA_INTERPOLATIVE

For the HDAServer and HDAServerEx objects, if a time range is specified the ResampleInterval must be less than the requested time range and greater than 0, otherwise the error E_INVALIDARG is returned. If OPCHDA_INTERPOLATIVE is requested, the Exaquantum OPC HDA server requests filtered History data with the number of periods set to the number of ResampleIntervals within the requested start and end times. Historian returns a minimum and a maximum value for each ResampleInterval with the raw data timestamps. If no point is returned at the requested start time the timestamp of the leading bounding value is modified to the requested start time. If no point is returned at the requested end time a point is added at the end time with value and quality cast forward from the previous point. Exaquantum never sets the OPCHDA_EXTRADATA quality flag.

The OPCHDA_QUALITY bits (bits 16-31) of the quality DWORD are always set to OPCHDA_RAW, or OPCHDA_NODATA if not available. The Data Access quality bits are returned directly from Exaquantum (bits 0-15). If the Exaquantum primary quality is ASSUMED (i.e. the tag has been set offline) bits 0-15 are set to OPC_QUALITY_UNCERTAIN.

The HDAServer2 object returns the Exaquantum Spot value at start of each interval (equivalent to requesting the OPCHDA_START aggregate).

Aggregate types other than OPCHDA_INTERPOLATIVE

The OPCHDA_QUALITY bits (bits 16-31) of the quality DWORD are always set to OPCHDA_CALCULATED, or OPCHDA_NODATA if not available. All raw data with GOOD or UNCERTAIN quality is included in the calculation. Data points with BAD quality are ignored. If the Exaquantum primary quality is ASSUMED (i.e. the tag has been set offline) bits 0-15 are set to OPC_QUALITY_UNCERTAIN.

Returns E_Q_OPCHDA_NOTCONNECTED if Exaquantum server is not running.

23.6.3. IOPCHDA_SyncRead::ReadAtTime

Exaquantum behavior:

An optional method – Exaquantum returns E_NOTIMPL.

23.6.4. IOPCHDA_SyncRead::ReadModified

Exaquantum behavior:

An optional method – Exaquantum returns E_NOTIMPL.

23.6.5. IOPCHDA_SyncRead::ReadAttribute

Exaquantum behavior:

Exaquantum never returns trailing bound values.

If no attribute data is available for a requested item, the error code for each requested attribute is set to OPC_E_UNKNOWNITEMID.

If a specific attribute is not available for a requested item, the error code for that attribute is set to OPC_E_NOT_AVAIL.

If an attribute is requested that is not supported by Exaquantum, the error code for that attribute is set to OPC_E_UNKNOWNATTRID.

Only current data is available for attributes OPCHDA_ITEMID and OPCHDA_DATA_TYPE.

Returns E_Q_OPCHDA_NOTCONNECTED if Exaquantum server is not running.

23.7. IOPCHDA_SyncUpdate Interface

This is an interface implemented by the HDAServer object. Only the QueryCapabilities(), InsertReplace(), DeleteAtTime(), and DeleteRaw() methods are implemented. All other methods return E_NOTIMPL.

23.7.1. IOPCHDA_SyncUpdate::QueryCapabilities

Exaquantum behavior:

Exaquantum returns the following bit sums.

- OPCHDA_INSERTREPLACECAP=0x04
- OPCHDA_DELETERAWCAP=0x08
- OPCHDA_DELETEATTIMECAP=0x10

23.7.2. IOPCHDA_SyncUpdate::Insert

Exaquantum behavior:

Method is not implemented– Exaquantum returns E_NOTIMPL.

23.7.3. IOPCHDA_SyncUpdate::Replace

Exaquantum behavior:

Method is not implemented – Exaquantum returns E_NOTIMPL.

23.7.4. IOPCHDA_SyncUpdate::InsertReplace

Exaquantum behavior:

The OPC specification states that the qualities passed in should be the raw OPC qualities, not the OPC HDA quality flags, but to assist clients which may simply pass back in data read via the IOPCHDA_SyncRead interface, any secondary quality flags will be stripped before passing to Exaquantum.

Nothing can be written to data that has been archived.

Writing data before the start of History of an item.

This is controlled by the registry key

HKEY_LOCAL_MACHINE\Software\WOW6432NodeDoe\Quantum\Server\Historian\WriteBeforeTagGenerationFlag

0 –Writing of data before the start of item history is disabled

1 –Writing of data before the start of item history is enabled

Scope of Writing of data.

This is determined by the value of the registry key:

HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum/Server\BulkWriteUpdateLatestValues (DefaultValue = 1)

This registry key accepts values as defined in Table 23-2.

Table 23-2

Value	Behavior
0	Data is only written to history. This setting should be used when importing historical data into Exaquantum.
1	The last data point supplied for each item is written to the item live value if its timestamp is later than the previous live value. This is the default behavior for backward compatibility with Exaquantum R2.30. This option may be used when OPC HDA is being used to import latest data into Exaquantum and no calculations are configured for the items for which data is being imported.
2	All data points supplied for each item are written to the item live value if their timestamp is later than the previous live value. This option may be used for improved performance when OPC HDA is being used to import latest data into Exaquantum. To improve performance Write Audit Logging should also be disabled by setting registry value: HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\Server\WriteAuditLogging = 0

These options offer flexibility in the way in which the interface is used. Ordinarily the default value of 1 will be used. Consultation with Yokogawa for specific applications may result in other values being applied. This setting applies to all connections to the OPC HDA interface.

Data points with timestamps earlier than the item live value will always be written directly to history. Each data point will either be written to history or to the Exaquantum RTDB, not both.

23.7.5. IOPCHDA_SyncUpdate::DeleteAtTime

Exaquantum behavior:

Archived data cannot be deleted.

This method expects the time parameter to be supplied in local time.

23.7.6. IOPCHDA_SyncUpdate::DeleteRaw

Exaquantum behavior:

Archived data cannot be deleted.

23.8. Exaquantum OPC HDA Service

This service is provided to allow continuous running of the QOPCHDAServer.exe process for attachment by clients that do not have launch permissions.

If the service is entered as a key under the Quantum\Services registry key, the service will be started automatically after Exaquantum is started and stopped on shutdown.

23.8.1. Sample registry file

```
Windows Registry Editor Version 5.00


[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Quantum\Services\1]
@="Exaquantum OPC HDA"
"HaltStartup"=dword:00000000


Note:
- If web server is already used, please add the following registry.
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Quantum\Services\2]
@="Exaquantum OPC HDA"
"HaltStartup"=dword:00000000


- If PI connection is already used, please refer to "Autostart setting
for Exaquantum OPC-HDA server/Exaquantum OPC-HDA PIFORAE SERVER" in
Exaquantum Engineering Guide Volume 5 – PI Connection (IM36J04A15-05E)
```

23.9. Data Access

The Exaquantum OPC DA server is implemented as an Exaopc cassette running within the ZOPDA.exe Exaopc server process. Availability of this component is controlled by the Exa Bossd service which is normally started automatically. The ProgID of the OPC Server object is:

- ♦ Yokogawa.ExaopcDAEXQ.1

Descriptions of interface behavior are as defined in the OPC Foundation document “OPC Data Access Custom Interface Specification Version 2.05A” which should be read for a fuller discussion of OPC DA functionality. Exaquantum-specific behavior is noted for each method.

The Exaquantum OPC DA server caches all requested item paths and holds metadata for each item for enhanced performance. This cache is monitored and managed periodically. Changes in the namespace are reflected if necessary. The frequency of monitoring is defined by the registry value:

```
HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\Client\NamespaceC
heckPeriod
```

- ♦ Type = DWORD
- ♦ DefaultValue = 10000 (milliseconds)

23.9.1. IOPCServer::AddGroup

Exaquantum behavior:

No Exaquantum-specific behavior defined.

DeadBand is ignored for Items that have no Engineering Range configured.

23.9.2. IOPCServer::GetErrorString

Exaquantum behavior:

This method should not be called from within a callback method (for example, OnDataChange) or the server may be locked.

23.9.3. IOPCServer::GetGroupByName

Exaquantum behavior:

No Exaquantum-specific behavior defined.

23.9.4. IOPCServer::GetStatus

Exaquantum behavior:

No Exaquantum-specific behavior defined.

23.9.5. IOPCServer::RemoveGroup

Exaquantum behavior:

No Exaquantum-specific behavior defined.

23.9.6. IOPCServer::CreateGroupEnumerator

Exaquantum behavior:

The Exaquantum OPC DA server does not support Public Groups.

23.10. IOPCItemProperties interface

23.10.1. IOPCItemProperties::QueryAvailableProperties

This is a required interface on the OPCServer object. It is used to return selected metadata for items.

The LookupItemIDs method is not implemented.

Exaquantum behavior:

Exaquantum supports the following properties:

Table 23-3

Symbolic name	Property ID	Data Type	Description
OPC_PROP_CDT	1	VT_I2	Item Canonical Data Type (VARTYPE)
OPC_PROP_VALUE	2	VT_VARIANT	Latest Item Value – the type of the value returned is as defined by Item Canonical Data Type
OPC_PROP_QUALITY	3	VT_I2	Latest Item Quality
OPC_PROP_TIME	4	VT_DATE	Latest Item Timestamp
OPC_PROP_RIGHTS	5	VT_I4	Item Access Rights – always set to OPC_READABLE | OPC_WRITEABLE
OPC_PROP_SCANRATE	6	VT_R4	Server Scan Rate – always set to 1000 milliseconds
OPC_PROP_UNIT	100	VT_BSTR	EU Units
OPC_PROP_DESC	101	VT_BSTR	Item Description
OPC_PROP_HIEU	102	VT_R8	High EU – the highest value likely to be read in normal operation
OPC_PROP_LOEU	103	VT_R8	Low EU – the lowest value likely to be read in normal operation

Some items may not have Engineering Units and/or Engineering Range defined.

Exaquantum supports only the following data types:

- ♦ VT_I2
- ♦ VT_I4
- ♦ VT_R4
- ♦ VT_R8
- ♦ VT_BSTR

23.10.2. IOPCItemProperties::GetItemProperties

Exaquantum behavior:

If the Exaquantum server is unavailable, individual item error codes are set to E_FAIL and the method returns S_FALSE.

23.10.3. IOPCItemProperties::LookupItemIDs

Exaquantum behavior:

This method is not implemented for Exaquantum – E_NOTIMPL is returned.

23.11. IOPCBrowseServerAddressSpace interface

This optional interface on the OPCServer object is partially supported.

23.11.1. IOPCBrowseServerAddressSpace:: QueryOrganization

Exaquantum behavior:

Exaquantum supports only a FLAT address space.

23.11.2. IOPCBrowseServerAddressSpace:: ChangeBrowsePosition

Exaquantum behavior:

Exaquantum supports only a FLAT address space but returns E_FAIL if called.

23.11.3. IOPCBrowseServerAddressSpace:: BrowseOPCItemIDs

Exaquantum behavior:

Exaquantum only filters on ItemPath – datatype and access rights filters are ignored.

If a filter value is specified, it must be supplied as a string to match ItemPath with optional wildcards, for example:

```
Root.Folder1.*
```

The list of item paths returned includes all raw items matching the specified filter. Reference item paths (Description, Units, HighEng and LowEng) are not returned.

Wildcards supported are:

*	Zero or more characters in that position
?	One character at that position

The Exaquantum OPC Server does not support # as a wildcard due to the possible valid inclusion of the # character as part of an item path. The pattern [0-9] can be used to provide the same functionality.

In addition Microsoft SQL Server pattern matching may be used:

%	Zero or more characters in that position
_	One character at that position
[]	Any single character within the specified range or set, e.g. [0-9], [abc]
[^]	Any single character not within the specified range or set, e.g. [^abc]

Aggregation result items may be returned by changing the registry value:

```
HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\OPC Server\BrowseAggregations
```

- Type=DWORD
- Default Value = 0 (do not return aggregation)
- A Non-zero value will include aggregation result items matching the filter.

23.11.4. IOPCBrowseServerAddressSpace:: GetItemID

Exaquantum behavior:

As the enumerated string set obtained from the GetEnum() method already contains the full Item path, a copy of the ItemDataID string is made to the output ItemID parameter. No check for validity is made.

23.11.5. IOPCBrowseServerAddressSpace:: BrowseAccessPaths

Exaquantum behavior:

This method is not implemented for Exaquantum – E_NOTIMPL is returned.

23.12. IOPCItemMgt interface

The IOPCItemMgt required interface on the OPCGroup object allows a client to add, remove and control the behavior of items in a group.

23.12.1. IOPCItemMgt::AddItems

Exaquantum behavior:

Exaquantum always returns OPC_E_UNKNOWNITEMID rather than OPC_E_INVALIDITEMID if an invalid item path is specified.

23.12.2. IOPCItemMgt::ValidateItems

Exaquantum behavior:

Exaquantum always returns OPC_E_UNKNOWNITEMID rather than OPC_E_INVALIDITEMID if an invalid item path is specified.

23.12.3. IOPCItemMgt::RemoveItems

Exaquantum behavior:

Exaquantum returns S_OK rather than S_FALSE if one or more invalid handles is specified.

23.12.4. IOPCItemMgt::SetActiveState

Exaquantum behavior:

Exaquantum returns S_OK rather than S_FALSE if one or more invalid handles is specified.

23.12.5. IOPCItemMgt::SetClientHandles

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.12.6. IOPCItemMgt::SetDatatypes

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.12.7. IOPCItemMgt::CreateEnumerator

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.13. IOPCGroupStateMgt interface

The required IOPCGroupStateMgt interface allows the client to manage the overall state of the group. Primarily this allows changes to the update rate and active state of the group. This interface is implemented within Exaopc.

23.13.1. IOPCGroupStateMgt::GetState

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.13.2. IOPCGroupStateMgt::SetState

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.13.3. IOPCGroupStateMgt::SetName

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.13.4. IOPCGroupStateMgt::CloneGroup

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.14. IOPCSyncIO interface

IOPCSyncIO allows a client to perform synchronous read and write operations to a server. The operations will run to completion.

23.14.1. IOPCSyncIO::Read

Exaquantum behavior:

If the Exaquantum primary quality for an item is ASSUMED (i.e. the tag has been set offline), the returned quality is set to OPC_QUALITY_UNCERTAIN.

If the Exaquantum server is unavailable, individual item error codes are set to E_FAIL and the method returns S_FALSE.

23.14.2. IOPCSyncIO::Write

Exaquantum behavior:

The user running the EXA Bossd service must be a member of the following groups in order to be able to write to Exaquantum items.

Legacy Model: QDataWrite Group or QAdministratorGroup

Standard Model: QTM_DATA_WRITE or QTM_MAINTENANCE

If the Exaquantum server is unavailable, individual item error codes are set to E_FAIL and the method returns S_FALSE.

23.15. IOPCAsyncIO2 interface

IOPCAsyncIO2 allows a client to perform asynchronous read and write operations to a server. The operations will be 'queued' and the function will return immediately so that the client can continue to run. Each operation is treated as a 'transaction' and is associated with a transaction ID. As the operations are completed, a callback will be made to the IOPCDataCallback in the client. The information in the callback will indicate the transaction ID and the results of the operation.

23.15.1. IOPCAsyncIO2::Read

Exaquantum behavior:

If the Exaquantum primary quality for an item is ASSUMED because the tag has been set offline, the returned quality is set to OPC_QUALITY_UNCERTAIN.

If the Exaquantum server is unavailable, individual item error codes are set to E_FAIL and the method returns S_FALSE.

23.15.2. IOPCAsyncIO2::Write

Exaquantum behavior:

The user running the EXA Bossd service must be a member of the following groups in order to be able to write to Exaquantum items.

Legacy Model: QDataWriteGroup or QAdministratorGroup

Standard Model: QTM_DATA_WRITE or QTM_MAINTENANCE

If the Exaquantum server is unavailable, individual item error codes are set to E_FAIL and the method returns S_FALSE.

23.15.3. IOPCAsyncIO2::Refresh2

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.15.4. IOPCAsyncIO2::Cancel2

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.15.5. IOPCAsyncIO2::SetEnable

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.15.6. IOPCAsyncIO2::GetEnable

Exaquantum behavior:

No Exaquantum-specific behavior is defined.

23.16. IOPCDataCallback client callback interface

This interface will be called on a client which is connected through the OPC Group object IOPCDataCallback connection point, as a result of changes in the data of the group (OnDataChange) and also as a result of calls to the IOPCAsyncIO2 interface.

23.16.1. IOPCDataCallback::OnDataChange

Exaquantum behavior:

If the Exaquantum primary quality for an item is ASSUMED, because the tag has been set offline, the returned quality is set to OPC_QUALITY_UNCERTAIN.

23.16.2. IOPCDataCallback::OnReadComplete

Exaquantum behavior:

If the Exaquantum primary quality for an item is ASSUMED, because the tag has been set offline, the returned quality is set to OPC_QUALITY_UNCERTAIN.

23.17. Exaquantum Server-specific error codes

The Exaquantum OPC Server components return some server-specific error codes. Errors are also logged to the Application Event Log for debugging purposes.

The table below lists some of the most common errors and their possible causes:

Table 23-4

Error Message	Probable cause
Event log error only: FAILED TO CONNECT TO EXAQUANTUM Event Log and returned error: **EXAQUANTUM IS UNAVAILABLE**	Either: The Exaquantum server is not running. The user running the client process is not a member of QUserGroup.
Event log error only: FAILED TO CREATE SESSION EVENT SINK OR REGISTER FOR EXAQUANTUM SESSION EVENTS	The user running the client process is not a member of QUserGroup.
READ OF HISTORY DATA TIMED OUT	Either: The Exaquantum historian is very loaded and the data read request was cancelled – see Qhistorian messages in EventLog for details. The read request has taken longer than the timeout specified in the registry under: HKEY_LOCAL_MACHINE\Software\WOW6432Node\Quantum\Client\QDataAccessSyncReadTimeout This timeout is specified in milliseconds and may be changed – it will be re-read when QOPCHDAServer is re-started.

Error Message	**Probable cause**
TIME STRING HAS INVALID FORMAT	The string representation of time specified in the OPCHDA_TIME structure is not correct. See section 23.6 IOPCHDA_SyncRead interface for further details.
INVALID REQUEST PARAMETERS - AT LEAST TWO OF STARTTIME, ENDTIME AND NUMBEROFVALUES MUST BE SPECIFIED. STARTTIME MUST ALWAYS BE SPECIFIED FOR EXAQUANTUM.	As error message
RESAMPLEINTERVAL MUST BE LESS THAN REQUESTED TIME RANGE AND GREATER THAN 0.	Invalid parameters supplied for ReadProcessed().
UNSUPPORTED AGGREGATE REQUESTED	Exaquantum can return data for the following aggregates only: OPCHDA_INTERPOLATIVE OPCHDA_TIMEAVERAGE OPCHDA_STDEV OPCHDA_MINIMUM OPCHDA_MAXIMUM OPCHDA_MINIMUMACTUALTIME OPCHDA_MAXIMUMACTUALTIME OPCHDA_START OPCHDA_END OPCHDA_RANGE OPCHDA_PERCENTGOOD OPCHDA_TOTAL

This page intentionally left blank

Chapter 24 Exaquantum.NET API

The Exaquantum .NET API is a suite of Components that operate under the .NET 2.0 Framework. The API connects with Exaquantum via the Common .Net Library (see table below). Functionality provided in the Admin Tools UI, can be executed programmatically, providing the Developer with:

- ♦ Management of Production Calendar information
- ♦ Management of Tag Templates
- ♦ Management of Folders and Tags

In addition, the API provides programmatic control over the Startup and Shutdown of the Exaquantum Server.

Note 1: Prior to Exaquantum R2.60 the .NET Interfaces are available to any .NET 1.1 language. Applications created for a releases of exaquantum prior to R2.60 will need to be recompiled against the .Net Framework Version 2.0 or later and reference the Exaquantum R2.60 or R2.70 libraries. Failure to do this will result in the Application failure to function correctly.

Note 2: All .Net applications must be compiled for the x86 CPU in order for the Exaquantum .NET API to function correctly. Failure to do this when running the application on an x64 system will result in the application running as an x64 process which is not supported.

Note 3 : The application which uses Exaquantum .NET library in the previous revision are out of scope for correct execution. Required to use the latest Exaquantum .Net Library.

Access to the API is performed through the following libraries:

Table 24-1

Library Name	**Description**
Yokogawa.Exa.Exaquantum.Common	Enumerations and Structures are defined.
Yokogawa.Exa.Exaquantum	Interfaces and Methods are defined.

The Exaquantum .NET API files are as follows:

<Exaquantum Installation Folder>\System\Yokogawa.Exa.Exaquantum.Common.dll

<Exaquantum Installation Folder>\System\ Yokogawa.Exa.Exaquantum.dll

The following table shows details of the Exaquantum.NET API Library.

Table 24-2

Interface Name	Method / Property Name	Description
Session	ServerName	Setting and acquisition of connected server name
ExaquantumAPI	Session	Setting and acquisition of Session
	AggregationPeriodManger	Acquisition of AggregationPeriodManager
	TemplateManager	Acquisition of TemplateManager
	TagManager	Acquisition of TagManager
	Operation	Acquisition of Operation
AggregationPeriodManager	GetAggregationPeriod	Acquisition of production calendar
	AddAggregationPeriod	Addition of production calendar
	ModifyAggregationPeriod	Modification of production calendar
	DeleteAggregationPeriod	Deletion of production calendar
	CheckAggregationPeriod	Check of production calendar
	QueryAggregationPeriod	Acquisition of production calendar list
TemplateManager	GetTagTemplate	Acquisition of tag template
	AddTagTemplate	Addition of tag template
	ModifyTagTemplate	Modification of tag template
	DeleteTagTemplate	Deletion of tag template
	CheckTagTemplate	Check of tag template
	QueryTagTemplate	Acquisition of tag template list

Interface Name	**Method / Property Name**	**Description**
TagManager	GetTag	Acquisition of tag
	AddTag	Addition of tag
	ModifyTag	Modification of tag
	DeleteTag	Deletion of tag
	CheckTag	Check of tag
	FindTag	Search of tag
	GetFolder	Acquisition of folder list
	AddFolder	Addition of folder
	ModifyFolder	Modification of folder
	DeleteFolder	Deletion of folder
	CheckFolder	Check of folder
Operation	Startup	Start-up
	Status Changed	Event that occurs status changed
	Shutdown	Shutdown
	OnShutdown Event	Event that occurs after shutdown completion
	IsMonitoring	Status Change Monitoring Flag

24.1. Installation Types for .NET API

The .NET API is installed by default on:

- Exaquantum Server
- Combined Server

It is not installed on

- Exaquantum Web Server
- Exaquantum Client

To install the Exaquantum .NET API on an Exaquantum Client or Web Server, please refer to Chapter 9 “Installing the .NET API on an Exaquantum Client or Web Server” in the “Exaquantum Installation Guide (IM 36J04A13-01E)”.

24.2. Exaquantum .NET API

The Exaquantum .NET API manages connections and provides the properties to return various Interfaces.

Note: users must be in QAdministratorGroup to gain access to the .NET API.

24.2.1. Session Property

24.2.1.1. Description

Gets/sets a Session object from/on the Exaquantum API. The session object gives the API:

- Access to remote Exaquantum servers
- Monitoring of the connection to Exaquantum.

24.2.1.2. Signature

```
Session Session{get;set;}
```

24.2.2. AggregationPeriodManager Property

24.2.2.1. Description

Get the Production Calendar Manager API

24.2.2.2. Signature

```
AggregationPeriodManager AggregationPeriodManager{get;}
```

24.2.3. TemplateManager Property

24.2.3.1. Description

Get the Template Manager API

24.2.3.2. Signature

```
TemplateManager TemplateManager{get;}
```

24.2.4. TagManager Property

24.2.4.1. Description

Get the Tag Manager API

24.2.4.2. Signature

```
TagManager TagManager{get;}
```

24.2.5. Operation Property

24.2.5.1. Description

Get the Operation API

24.2.5.2. Signature

```
Operation Operation{get;}
```

24.2.6. Sample Code

This section gives example of how to access the Exaquantum API objects.

```
using Yokogawa.Exa.Exaquantum;
using Yokogawa.Exa.Exaquantum.Common;

ExaquantumAPI api = null ;
Session QSession = null ;
TagManager Qtm = null;
TemplateManager Tm = null;
AggregationPeriodManager Ap = null ;
Operation op = null ;

api = new ExaquantumAPI();

QSession = api.Session ;
QSession.ServerName = "MyServer" ;
Qtm = api.TagManager ;
Tm = api.TemplateManager ;
Ap = api.AggregationPeriodManager ;
op = api.Operation ;
```

24.3. Session Interface

Setting and acquisition of connected server name.

24.3.1. ServerName Property ()

24.3.1.1. Signature

```
string ServerName{get;set;}
```

24.4. AggregationPeriodManager interface

Allows management of the Production Calendar.

Namespace: Yokogawa.Exa.Exaquantum.

24.4.1. GetAggregationPeriod Method

24.4.1.1. Description

This returns the Production Calendar Period detail, in the output QAggregationPeriod() object, corresponding to the supplied handles.

24.4.1.2. Signature

```
GetAggregationPeriod(int handles, _
       out QAggregationPeriod[] aggregationPeriod, _
       out int[] errors)
```

24.4.1.3. Parameters

Handles - Array of IDs

AggregationPeriod - Array of QAggregationPeriod Objects

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

24.4.2. AddAggregationPeriod Method

24.4.2.1. Description

Adds one or more Production Calendar Periods. If a Production Calendar already exists, an error occurs.

24.4.2.2. Signature

```
AddAggregationPeriod (QAggregationPeriod[] aggregationPeriods, _
       out int[] handles, _
       out int[] errors)
```

24.4.2.3. Parameters

AggregationPeriods - Array of QAggregationPeriod object(s) to be added

Handles - Array of returned handles

Errors - Array of errors, corresponds to each element in the **AggregationPeriods** array.

Note: In the instance where two or more elements have the same name in AggregationPeriods, the **EQ_E_DUPLICATE_NAME** error is returned for all elements having this condition, except for the initial one.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_PERIOD - Invalid Period

EQ_E_INVALID_OFFSET - Invalid Offset

EQ_E_INVALID_PARENTHANDLE - Invalid Parent Handle

EQ_E_DUPLICATE_NAME - Duplicate in **AggregationPeriods** `array`

EQ_E_ITEMEXIST – Aggregation period already exists

24.4.2.4. API Exception

ArgumentOutOfRangeException – The bounds of the **AggregationPeriods** array are incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.4.2.5. ModifyAggregationPeriod Method

24.4.2.6. Description

Allows modification of one or more Production Calendar periods. When the target production calendar period does not exist, an error occurs.

24.4.2.7. Signature

```
ModifyAggregationPeriod (int[] handles, _

     in QAggregationPeriod[] aggregationPeriods, _

     out int[] errors)
```

24.4.2.8. Parameters

Handles - Array of the production calendar period handle to be modified

AggregationPeriods - Array of the QAggregationPeriod object(s) to be modified

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_PERIOD - Invalid Period

EQ_E_INVALID_OFFSET - Invalid Offset

EQ_E_INVALID_PARENTHANDLE - Invalid Parent Handle

EQ_E_DUPLICATE_NAME - Duplicate Name

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_NAME - Duplicate in **AggregationPeriods** `array`

EQ_E_ITEMEXIST – Aggregation period already exists

24.4.2.9. API Exception

ArgumentOutOfRangeException – The bounds of the **AggregationPeriods** or **Handles** array are incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.4.3. DeleteAggregationPeriod Method

24.4.3.1. Description

Allows deletion of the specified Production Calendar(s).

24.4.3.2. Signature

```
DeleteAggregationPeriod(int[] handles, _
   out int[] errors)
```

24.4.3.3. Parameter

Handles - Array of the production calendar period handle(s) to be deleted.

Errors - Array of errors – corresponds to each handle

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_HANDLE - Duplicate Handle

EQ_E_INUSE – The production calendar period is used with the tag template.

24.4.3.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Handles** array are incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.4.4. CheckAggregationPeriod Method

24.4.4.1. Description

Validates the contents of the QAggregationPeriod array. The result of each array entry is returned in the **errors** array.

24.4.4.2. Signature

```
CheckAggregationPeriod(QAggregationPeriod[] aggregationPeriods, _
   out int[] errors)
```

24.4.4.3. Parameter

AggregationPeriods - Array of the QAggregationPeriod object(s) to be checked

Errors - Array of errors, corresponds to each handle.

Note: In the instance where two or more elements have the same name in AggregationPeriods, the **EQ_E_DUPLICATE_NAME** error is returned for all elements having this condition, except for the initial one.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_PERIOD - Invalid Period

EQ_E_INVALID_OFFSET - Invalid Offset

EQ_E_INVALID_PARENTHANDLE - Invalid Parent Handle

EQ_E_DUPLICATE_NAME - Duplicate in **AggregationPeriods** `array`

EQ_E_ITEMEXIST – Aggregation period already exists

24.4.4.4. API Exception

ArgumentOutOfRangeException – The bounds of the **AggregationPeriods** array are incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.4.5. QueryAggregationPeriod Method

24.4.5.1. Description

Returns a list of all of the Production Calendars.

24.4.5.2. Signature

```
QueryAggregationPeriod(out int[] handles, _
   out QAggregationPeriod[] aggregationPeriod)
```

24.4.5.3. Parameter

Handles - Array of production calendar period handle(s)

AggregationPeriod - Array of the production calendar periods

24.4.6. Sample Code

This section gives example of how to use the AggregationPeriodManager methods().

24.4.6.1. GetAggregationPeriod

```
AggregationPeriodManager Agg =
       new ExaquantumAPI().AggregationPeriodManager;

QAggregationPeriod[] aggregationPeriod;
int[] handles = new int[2];
int[] errors;

// get
handles[0] = 97;
handles[1] = 1;
Agg.GetAggregationPeriods(handles, out aggregationPeriod, out errors);
```

24.4.6.2. AddAggregationPeriod

```
AggregationPeriodManager Agg =
       new ExaquantumAPI().AggregationPeriodManager;
QAggregationPeriod[] ParaOfAgPd =
       new QAggregationPeriod[2];
int[] handles;
int[] errors;

ParaOfAgPd[0].Name = "aaa"; // Name
ParaOfAgPd[0].Description = ""; // Comments
ParaOfAgPd[0].Period = 2; // 2 Seconds
// For a list of values you can use, please refer to the items in the
  structure of QAggregationPeriod.
ParaOfAgPd[0].Offset = 1; // 1 Second - The value must be below that of
the Period value.
ParaOfAgPd[0].ParentHandle = QAggregationPeriod.InstantValue; // Instant
value
// When specifying a customization for a ParentHandle, after acquiring a
  handle by QueryAggregationPeriod or similar means, please define that
  handle.

ParaOfAgPd[1].Name = "bbb";
ParaOfAgPd[1].Description = "";
ParaOfAgPd[1].Period = 3; //3 Seconds
ParaOfAgPd[1].Offset = 2;
ParaOfAgPd[1].ParentHandle = QAggregationPeriod.InstantValue;
 // Instant value
// Adds a calendar.
Agg.AddAggregationPeriod(ParaOfAgPd, out handles, out errors);
```

24.4.6.3. AddAggregationPeriod/CheckAggregationPeriod/ModifyAggregationPeriod/DeleteAggregationPeriod

```
AggregationPeriodManager Agg =
       new ExaquantumAPI().AggregationPeriodManager;

// parameter set
QAggregationPeriod[] ParaOfAgPd= new QAggregationPeriod[1];
int[] handles;
int[] errors;

int i = 0;
ParaOfAgPd[i].Name              = "ccc";
ParaOfAgPd[i].Description       = "";
ParaOfAgPd[i].Period            = 2;
ParaOfAgPd[i].Offset            = 1;
ParaOfAgPd[i].ParentHandle      = 0;

// add calendar
Agg.AddAggregationPeriod(ParaOfAgPd, out handles, out errors);

i = 0;
ParaOfAgPd[i].Name              = "ddd";
ParaOfAgPd[i].Description       = "";
ParaOfAgPd[i].Period            = 2;
ParaOfAgPd[i].Offset            = 2;
ParaOfAgPd[i].ParentHandle      = 0;

// check
Agg.CheckAggregationPeriod(ParaOfAgPd, out errors);

// change calendar
Agg.ModifyAggregationPeriod(handles, ParaOfAgPd, out errors);

// delete
Agg.DeleteAggregationPeriod(handles, out errors);
```

24.4.6.4. QueryAggregationPeriod

```
AggregationPeriodManager Agg =
       new ExaquantumAPI().AggregationPeriodManager;

int[] Handles;
QAggregationPeriod[] AggPeriods;

Agg.QueryAggregationPeriod(out Handles, out AggPeriods);
```

24.5. TemplateManager interface

The TemplateManager allows management of the Tag Templates.

Namespace: Yokogawa.Exa.Exaquantum.

24.5.1. GetTagTemplate Method

24.5.1.1. Description

The Tag Template(s) for the specified handle(s) is returned – the error **EQ_E_INVALID_HANDLE** is returned if the handle does not exist.

24.5.1.2. Signature

```
GetTagTemplate(int[] handles, _
  out QTagTemplate[] tagTemplates,_
  out int[] errors)
```

24.5.1.3. Parameter

Handles - Array of TagTemplate handles

tagTemplates - Array of QTagTemplates

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

24.5.2. AddTagTemplate Method

24.5.2.1. Signature

```
AddTagTemplate(QTagTemplate[] tagTemplates, _
 out int[] handles, _
 out int[] errors)
```

24.5.2.2. Parameter

TagTemplates - Array of QTagTemplates

Handles - Array of handles for templates that get added

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_TAGTYPE - Invalid Tag Type

EQ_E_INVALID_DATATYPE - Invalid Data Type

EQ_E_INVALID_AGGREGATIONTYPE - Invalid Aggregation Type

EQ_E_INVALID_HISTORIANINTERVAL - Invalid Historian Interval

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_RANGE - Invalid Range

EQ_E_INVALID_PERIOD_ID - Invalid Period ID

EQ_E_INVALID_OPCGATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPCPERIOD - Invalid OPC Period

EQ_E_INVALID_DEADBAND - Invalid Deadband

EQ_E_INVALID_CALCFLAG - Invalid Flag of Aggregation supplied by calculation

EQ_E_INVALID_AGGREGATIONINFO - Invalid Aggregation Calculations

EQ_E_DUPLICATE_NAME - Duplicate in **TagTemplates** `array`

EQ_E_ITEMEXIST – Tag Template already exists

24.5.2.3. API Exception

ArgumentOutOfRangeException – The bounds of the **TagTemplates** array are incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.5.3. ModifyTagTemplate Method

24.5.3.1. Description

Modify one or more Tag Templates. If the template is already in use, then a new version of the template is added. This new version will be applied when the tag(s) that are built to the template are rebuilt (or upgraded).

24.5.3.2. Signature

```
ModifyTagTemplate(int[] handles, _
 QTagTemplate[] tagTemplates, _
 out int[] errors)
```

24.5.3.3. Parameter

Handles - Array of handles

TagTemplates - Array of tag template information

Errors - Array of errors, corresponds to each handle

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_TAGTYPE - Invalid Tag Type

EQ_E_INVALID_DATATYPE - Invalid Data Type

EQ_E_INVALID_AGGREGATIONTYPE - Invalid Aggregation Type

EQ_E_INVALID_HISTORIANINTERVAL - Invalid Historian Interval

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_RANGE - Invalid Range

EQ_E_INVALID_PERIOD_ID - Invalid Period ID

EQ_E_INVALID_OPCGATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPCPERIOD - Invalid OPC Period

EQ_E_INVALID_DEADBAND - Invalid Deadband

EQ_E_INVALID_CALCFLAG - Invalid Flag of Aggregation supplied by calculation

EQ_E_INVALID_AGGREGATIONINFO - Invalid Aggregation Calculations

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_NAME - Duplicate in **TagTemplates** `array`

EQ_E_ITEMEXIST – Tag Template already exists

EQ_E_INUSE – One or more parameters cannot be changed, as the template has been used to create tags (for example tag type).

24.5.3.4. API Exception

ArgumentOutOfRangeException – The bounds of the **TagTemplates** or **Handles** array are incorrect

ArgumentNullException – One or more of the parameters is NULL.

24.5.4. DeleteTagTemplate Method

24.5.4.1. Description

Delete an existing tag template. This will fail if the template is in use.

24.5.4.2. Signature

```
DeleteTagTemplate (int[] handles, _
   out int[] errors)
```

24.5.4.3. Parameter

Handles - Array of Tag Template Handle(s) to be deleted

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_HANDLE - Duplicate Handle

EQ_E_INUSE - TagTemplate is in use.

24.5.4.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Handles** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.5.5. CheckTagTemplate Method

24.5.5.1. Description

This method is used for the verification of tag template

24.5.5.2. Signature

```
CheckTagTemplate (QTagTemplate[] tagTemplates, _
   out int[] errors)
```

24.5.5.3. Parameter

TagTemplates - Array of Tag Templates

Errors - Array of errors, corresponds to each element in the **tagTemplates** array**.**

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_TAGTYPE - Invalid Tag Type

EQ_E_INVALID_DATATYPE - Invalid Data Type

EQ_E_INVALID_AGGREGATIONTYPE - Invalid Aggregation Type

EQ_E_INVALID_HISTORIANINTERVAL - Invalid Historian Interval

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_RANGE - Invalid Range

EQ_E_INVALID_PERIOD_ID - Invalid Period ID

EQ_E_INVALID_OPCGATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPCPERIOD - Invalid OPC Period

EQ_E_INVALID_DEADBAND - Invalid Deadband

EQ_E_INVALID_CALCFLAG - Invalid Flag of Aggregation supplied by calculation

EQ_E_INVALID_AGGREGATIONINFO - Invalid Aggregation Calculations

EQ_E_DUPLICATE_NAME - Duplicate in **TagTemplates** `array`

EQ_E_ITEMEXIST – Tag Template already exists

24.5.5.4. API Exception

ArgumentOutOfRangeException – The bounds of the **tagTemplates** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.5.6. QueryTagTemplate Method

24.5.6.1. Description

A list of TagTemplates is returned.

24.5.6.2. Signature

```
QueryTagTemplate(out int[] handles, _
   out QTagTemplate[] tagTemplates)
```

24.5.6.3. Parameter

Handles - Array of TagTemplate Handles

tagTemplates - Array of TagTemplates

24.5.6.4. API Exception

ApplicationException – This is thrown if an error occurred during the reading of the Tag Templates. The detail of the error is described in the Message property of the exception.

24.5.7. SampleCode

This section gives some examples on how to use the TemplateManager methods.

24.5.7.1. GetTagTemplate

```
TemplateManager templateManager = new ExaquantumAPI().TemplateManager;

QTagTemplate[] TaT;
int[] handles = new int[]{1};
int[] errors;

templateManager.GetTagTemplate(handles, out TaT, out errors);
```

24.5.7.2. AddTagTemplate/CheckTagTemplate/ModifyTagTemplate/DeleteTagTemplate

```
TemplateManager templateManager = new ExaquantumAPI().TemplateManager;

QTagTemplate[] AddTemp = new QTagTemplate[3]; // Creates 3 templates.
int[] handles;
int[] errors;
int[] APid = new int[1];
QAggregationInfo[] AInfo = new QAggregationInfo[1];

APid[0] = QAggregationPeriod.HourlyReportHandle;
 // This is acquired by the QueryAggregationPeriod API or similar means.

AInfo[0].AggregationCalcType = AggregationCalcType.AGGREGATION_MAX;
 // Sets the maximum value limit.
AInfo[0].Parameter = null;

AddTemp[0].Name = "CALC-R4"; // Sets the name to "CALC-R4".
AddTemp[0].TagType = TagType.TAG_TYPE_CALC; // Calculations tag
AddTemp[0].DataType = DataType.DATA_TYPE_FLOAT; // Use a real number.
AddTemp[0].AggregationType = AggregationType.CONTINUOUS; // Continuous
AddTemp[0].RawHistorianInterval =
 RawHistorianIntervalType.RAW_INTERVAL_ONCHANGE;
 // Interval for save on change
AddTemp[0].EngUnitsFlag = false; // Do not use engineering units.
AddTemp[0].EngRangeFlag = false; // Do not use end range value.
```

```
AddTemp[0].DescriptionFlag = true; // Use comments. (Non-functional parameter.
Even if "false" is specified, it will not function.)
AddTemp[0].AggregationPeriodIDs = APid;
 // Arrangement by limit cycle IDs. int[]
AddTemp[0].WriteToOPCGatewayFlag = false;
 // Will not write to the OPC gateway. (Except for the OPC tags, everything
 will become false.)
AddTemp[0].OPCPeriod = OPCPeriodType.OPC_PERIOD_NOUSE;
 // OPC cycles. Except for the OPC tags, everything will become
 OPC_PERIOD_NOUSE.
AddTemp[0].PercentDeadband = QTagTemplate.PercentDeadbandNoUse;
 // Deadband. Except for the OPC tags, it will be specified this way.
AddTemp[0].AggregationSuppliedByCalculationFlag = new bool[]{false};
 // Flag for finding a deadline by calculation.
AddTemp[0].AggregationInfo = AInfo;

QAggregationInfo[] AggInfoOPC = new QAggregationInfo[3];
AggInfoOPC[0].AggregationCalcType = AggregationCalcType.AGGREGATION_MAX;
AggInfoOPC[0].Parameter = null;
AggInfoOPC[1].AggregationCalcType = AggregationCalcType.AGGREGATION_MIN;
AggInfoOPC[1].Parameter = null;
AggInfoOPC[2].AggregationCalcType = AggregationCalcType.AGGREGATION_SUM;
AggInfoOPC[2].Parameter = new object[]{null, true, 10, 10.0D, 0.1D};
//

AddTemp[1].Name = "OPC-R8"; // Sets the name to "OPC-R8".
AddTemp[1].TagType = TagType.TAG_TYPE_OPC; // OPC Tag
AddTemp[1].DataType = DataType.DATA_TYPE_DOUBLE; // Use a double real number.
AddTemp[1].AggregationType = AggregationType.CONTINUOUS; // Continuous
AddTemp[1].RawHistorianInterval = RawHistorianIntervalType.RAW_INTERVAL_1SEC;
 // Saves every second.
AddTemp[1].EngUnitsFlag = true; // Use engineering units.
AddTemp[1].EngRangeFlag = true; // Use end range value.
AddTemp[1].DescriptionFlag = true;
 // Use comments. (Non-functional parameter. Even if "false" is specified, it
 will not function.)
AddTemp[1].AggregationPeriodIDs = APid;
 // Arrangement by deadline cycle ID. int[]
AddTemp[1].WriteToOPCGatewayFlag = true; // Write to the OPC gateway.
AddTemp[1].OPCPeriod = OPCPeriodType.OPC_PERIOD_1SEC;
 // The OPC cycle period is set to 1 second.
AddTemp[1].PercentDeadband = 0.01F;
 // Sets the deadband to 0.01%. If set to 0, there will be no deadband.
AddTemp[1].AggregationSuppliedByCalculationFlag = new bool[]{false};
 // Flag for finding a deadline by calculation. Except for the calculation
 tag, everything is false.
AddTemp[1].AggregationInfo = AggInfoOPC;

AddTemp[2].Name = "MAN-STR"; // Sets the name to "MAN-STR".
AddTemp[2].TagType = TagType.TAG_TYPE_MAN; // Manual tag
AddTemp[2].DataType = DataType.DATA_TYPE_STRING; // A character string
AddTemp[2].AggregationType = AggregationType.DESCRETE; // Non-continuous
AddTemp[2].RawHistorianInterval =
        RawHistorianIntervalType.RAW_INTERVAL_ONCHANGE;
 // Time interval on change
AddTemp[2].EngUnitsFlag = true; // Use engineering units.
AddTemp[2].EngRangeFlag = false;
 // Do not use an end range value with character string data.
AddTemp[2].DescriptionFlag = true;
```

```
  // Use comments. (Non-functional parameter. Even if "false" is specified, it
  will not function.)
AddTemp[2].AggregationPeriodIDs = null;
  // Arrangement by deadline cycle ID. int[]
AddTemp[2].WriteToOPCGatewayFlag = false; // Will not write to the OPC gateway.
AddTemp[2].OPCPeriod = OPCPeriodType.OPC_PERIOD_NOUSE;
  // Will not be used like CAL.
AddTemp[2].PercentDeadband = QTagTemplate.PercentDeadbandNoUse; // Deadband.
AddTemp[2].AggregationSuppliedByCalculationFlag = null; //
AddTemp[2].AggregationInfo = null; //

templateManager.AddTagTemplate(AddTemp, out handles, out errors);

AddTemp[0].EngUnitsFlag = true;
AddTemp[0].EngRangeFlag = true;

AggInfoOPC[2].AggregationCalcType = AggregationCalcType.AGGREGATION_SUM;
AggInfoOPC[2].Parameter = new
object[]{SummationTimeFactorType.TIME_FACTOR_HOUR, false, null, null, null};

templateManager.CheckTagTemplate(AddTemp, out errors);

templateManager.ModifyTagTemplate(handles, AddTemp, out errors);

templateManager.DeleteTagTemplate(handles, out errors);
```

24.5.7.3. QueryTagTemplate

```
TemplateManager templateManager = new ExaquantumAPI().TemplateManager;

int[] handles;
QTagTemplate[] TaT;

templateManager.QueryTagTemplate(out handles, out TaT);
```

24.6. TagManager interface

Allows management of the Tag and Folders.

Namespace: Yokogawa.EXA.Exaquantum

24.6.1. GetTag Method

24.6.1.1. Description

Get one or more Tags under the specified path(s).

24.6.1.2. Signature

```
GetTag(string[] paths, _
 out int[] handles, _
 out QTag[] tags, _
 out int[] errors)
```

24.6.1.3. Parameter

Paths - Array of full tag path(s).

Handles - Array of tag handles; these correspond to the specified paths.

Tags - Array of tag information.

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_PATH - Invalid Path

24.6.2. AddTag Method

24.6.2.1. Description

Adds one or more tags. Where folders do not exist, they are created prior to adding of the tags..

24.6.2.2. Signature

```
AddTag (QTag[] tags, _
 out int[] handles, _
 out int[] errors)
```

24.6.2.3. Parameter

Tags - Array of Tag Information.

Handles - Array of corresponding tag handles.

Errors - Array of errors – correspond to each handle and each tags.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_PATH - Invalid Path

EQ_E_INVALID_TEMPLATENAME – no Template Name is specified

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_LOWENG - Invalid Lower limit of engineering range

EQ_E_INVALID_HIGHENG - Invalid Higher limit of engineering range

EQ_E_INVALID_RANGE - Invalid engineering range

EQ_E_INVALID_VALUE - Invalid Initial Value

EQ_E_INVALID_QUALITY - Invalid Quality

EQ_E_INVALID_OPC_GATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPC_ITEMID - Invalid OPC Item ID

EQ_E_INVALID_SCRIPT - Invalid Script

EQ_E_ANALYZE - Analyze Error

EQ_E_DUPLICATE_NAME - Duplicate in **Tags** `array`

EQ_E_ITEMEXIST – Tag already exists

24.6.2.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Tags** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.3. ModifyTag Method

24.6.3.1. Description

Modification of one or more tags. If the tag template is to be changed, the only case where this will succeed is if the new template has the same tag type and data type to which the tag is currently built.

24.6.3.2. Signature

```
ModifyTag (int[] handles, _
  QTag[] tags,_
  out int[] errors)
```

24.6.3.3. Parameter

Handles - Array of tag handles

Tags - Array of tag information

Errors - Array of errors, corresponds to each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_PATH - Invalid Path

EQ_E_INVALID_TEMPLATENAME - Invalid Template Name

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_LOWENG - Invalid Lower limit of engineering range

EQ_E_INVALID_HIGHENG - Invalid Higher limit of engineering range

EQ_E_INVALID_RANGE - Invalid engineering range

EQ_E_INVALID_VALUE - Invalid Initial Value

EQ_E_INVALID_QUALITY - Invalid Quality

EQ_E_INVALID_OPC_GATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPC_ITEMID - Invalid OPC Item ID

EQ_E_INVALID_SCRIPT - Invalid Script

EQ_E_ANALYZE - Analyze Error

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_HANDLE - Duplicate Handle

EQ_E_DUPLICATE_NAME - Duplicate in **Tags** `array`

EQ_E_ITEMEXIST – Tag already exists

24.6.3.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Tags** or **Handles** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.4. DeleteTag Method

24.6.4.1. Description

Deletion of one or more tags.

24.6.4.2. Signature

```
DeleteTag (int[] handles, _
     out int[] errors, _
     bool force)
```

24.6.4.3. Parameter

Handles - Array of tag handles

Errors - Array of errors, corresponds to each handle.

Force - If this flag is true, then the tag(s) will be deleted, even if they are being referenced by other tags (for example, by a calculation tag script).

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_HANDLE - Duplicate Handle

EQ_E_INUSE – Tag is referenced by calculation tag, and cannot be deleted.

24.6.4.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Handles** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.5. CheckTag Method

24.6.5.1. Description

This method is used for the verification of the content of the QTag array.

24.6.5.2. Signature

```
CheckTag (QTag[] tags, _
 out int[] errors)
```

24.6.5.3. Parameter

Tags - Array of tag information

Errors - Array of errors, corresponds to each tag.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_PATH - Invalid Path

EQ_E_INVALID_TEMPLATENAME - Invalid Template Name

EQ_E_INVALID_DESCRIPTION - Invalid Description

EQ_E_INVALID_UNITS - Invalid Units

EQ_E_INVALID_LOWENG - Invalid Lower limit of engineering range

EQ_E_INVALID_HIGHENG - Invalid Higher limit of engineering range

EQ_E_INVALID_RANGE - Invalid engineering range

EQ_E_INVALID_VALUE - Invalid Initial Value

EQ_E_INVALID_QUALITY - Invalid Quality

EQ_E_INVALID_OPC_GATEWAY - Invalid OPC Gateway

EQ_E_INVALID_OPC_ITEMID - Invalid OPC Item ID

EQ_E_INVALID_SCRIPT - Invalid Script

EQ_E_ANALYZE - Analyze Error

EQ_E_DUPLICATE_NAME - Duplicate in **Tags** `array`

EQ_E_ITEMEXIST – Tag already exists

24.6.5.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Tags** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.6. FindTag Method

24.6.6.1. Description

Returns a list of tags that match the specified filter(s).

24.6.6.2. Signature

```
FindTag (QTagFilter[] TagFilters, _
     out int[] handles, _
     out QTag[] tags)
```

24.6.6.3. Parameter

Filters - Array of one of more QTagFilters.

*: represents a character string.

?: represents a character.

[]: represents a character string in a specified character range: [0-9],[abc]

Handles - Array of tag handles

Tags – Array of tag information

24.6.6.4. API Exception

ArgumentOutOfRangeException - The bounds of the **TagFilters** array is incorrect.

ArgumentException – One or more of the filter parameters is set incorrectly.

ApplicationException – This is thrown if an unexpected error occurred during the processing of the **FindTag** method. The detail of the error is described in the Message property of the exception.

24.6.7. GetFolder Method

24.6.7.1. Description

Returns a list of folder information, for the specified path(s).

24.6.7.2. Signature

```
GetFolder (string[] paths, _
     out int[] handles, _
     out QFolder[] folders, _
     out int[] errors)
```

24.6.7.3. Parameter

Paths - Array of paths

Handles – Array of folder handles

Folders - Array of folder information.

Errors - Array of errors, corresponds to each path, handle and folder.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_PATH - Invalid Path

24.6.8. AddFolder Method

24.6.8.1. Description

Adds one or more folders.

24.6.8.2. Signature

```
AddFolder (QFolder[] folders, _
 out int[] handles, _
 out int[] errors)
```

24.6.8.3. Parameter

Folders - Array of folder information

Handles - Array of folder handles.

Errors - Array of errors, corresponds to each folder and each handle.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_PATH - Invalid Path

EQ_E_DUPLICATE_NAME - Duplicate in **Folders** `array`

EQ_E_ITEMEXIST – Folder already exists

24.6.8.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Folders** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.9. ModifyFolder Method

24.6.9.1. Description

Modifies one or more folders.

24.6.9.2. Signature

```
ModifyFolder (int[] handles, _
 QFolder[] folders, _
 out int[] errors)
```

24.6.9.3. Parameter

Handles - Array of folder handles

Folders - Array of folder information

Errors - Array of errors, corresponds to each handle and each folder.

Errors Returned

- **S_OK** - Succeeded
- **EQ_E_INVALID_NAME** - Invalid Name
- **EQ_E_INVALID_PATH** - Invalid Path
- **EQ_E_INVALID_HANDLE** - Invalid Handle
- **EQ_E_DUPLICATE_HANDLE** - Duplicate Handle
- **EQ_E_DUPLICATE_NAME** - Duplicate in **Folders** `array`
- **EQ_E_ITEMEXIST** – Folder already exists

24.6.9.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Folders** or **Handles** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.10. DeleteFolder Method

24.6.10.1. Description

Deletes one or more folders.

24.6.10.2. Signature

```
DeleteFolder (int[] handles, _
       out int[] errors, _
       bool force)
```

24.6.10.3. Parameter

Handles - Array of folder handles

Errors - Array of errors – correspond to each handle.

Force - If this flag is true, tags under the folder will be deleted even if they are being referenced by other tags.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_HANDLE - Invalid Handle

EQ_E_DUPLICATE_HANDLE - Duplicate Handle

EQ_E_INUSE – A tag being referenced exists under the folder.

24.6.10.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Handles** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.11. CheckFolder Method

24.6.11.1. Description

This method is used for the verification of the content of the QFolder array.

24.6.11.2. Signature

```
CheckFolder (QFolder[] folders, _
       out int[] errors)
```

24.6.11.3. Parameter

Folders - Array of folder information

Errors - Array of errors, corresponds to each folder.

Errors Returned

S_OK - Succeeded

EQ_E_INVALID_NAME - Invalid Name

EQ_E_INVALID_PATH - Invalid Path

EQ_E_DUPLICATE_NAME - Duplicate in **Folders** `array`

EQ_E_ITEMEXIST – Folder already exists

24.6.11.4. API Exception

ArgumentOutOfRangeException – The bounds of the **Folders** array is incorrect.

ArgumentNullException – One or more of the parameters is NULL.

24.6.12. Sample Code

This section gives some examples of sample code, showing how to use the TagManager API.

24.6.12.1. Sample Code for the management of Tags

24.6.12.1.1. AddTag/DeleteTag

```
TagManager tagManager = new ExaquantumAPI().TagManager;

QTag[] tags = new QTag[3];
int[] handles;
int[] errors;

// tag create
//OPC R8
int i                    = 0;
tags[i].Name             = "OPC-TAG";
tags[i].Path             = "Root.Test.OPC";
tags[i].TemplateName     = "OPC-R8";
tags[i].Description      = "";
tags[i].Units            = "";
tags[i].LowEng           = 0;
tags[i].HighEng          = 0;
tags[i].OPCGateWay       = "Gateway1";
tags[i].OPCItemID        = "F001.PV";
tags[i].LockStatus       = false;
tags[i].Online           = true;

//CALC R4
i                        = 1;
tags[i].Name             = "CALC-TAG";
tags[i].Path             = "Root.Test.CAL";
tags[i].TemplateName     = "CALC-R4";
tags[i].Description      = "float";
tags[i].Units            = "l";
tags[i].LowEng           = 0;
tags[i].HighEng          = 9999;
tags[i].InitQuality     = QualityType.qdaQualityGood;
tags[i].Script          = "[Result] = 1"
tags[i].LockStatus      = false;
tags[i].Online          = true;

//CAL String
i                       = 2;
tags[i].Name            = "MAN-TAG";
tags[i].Path            = "Root.Test.Man";
tags[i].TemplateName    = "MAN-STR";
tags[i].Description     = "";
tags[i].Units           = "";
tags[i].LowEng          = 0;
tags[i].HighEng         = 0;
tags[i].InitValue       = 0;
tags[i].InitQuality     = QualityType.qdaQualityGood;
tags[i].LockStatus      = false;
tags[i].Online          = true;

// add
tagManager.AddTag(tags, out handles, out errors);

// delete
tagManager.DeleteTag(handles, out errors, false);
```

24.6.12.1.2. GetTag/CheckTag/ModifyTag

```
TagManager tagManager = new ExaquantumAPI().TagManager;

QTag[] tags;
int[] handles;
int[] errors;

string[] paths = new string[1];

paths[0] = "Root.Test.MAN-TAG";

// Get
tagManager.GetTag(paths, out handles, out tags, out errors);

tags[0].Name = "MAN-TAG1";
tags[0].InitValue = 0;

// Check
tagManager.CheckTag(tags, out errors);
if(errors[0] != 0)
{
       return; // error
}

// Modify
tagManager.ModifyTag(handles, tags, out errors);
```

24.6.12.1.3. FindTag

```
TagManager tagManager = new ExaquantumAPI().TagManager;

QTagFilter[] tagFilters = new QTagFilter[1];
QTag[] tags;
int[] handles;

tagFilters[0].TagFilterType = TagFilterType.TAG_FILTER_PATH;
tagFilters[0].Filter = "Root.Test.*";

tagManager.FindTag(tagFilters, out handles, out tags);
```

24.6.12.2. Sample Code for the management of Folders

24.6.12.2.1. GetFolder

```
TagManager tagManager = new ExaquantumAPI().TagManager;

string[] paths = new string[1];
QFolder[] folders;
int[] handles;
int[] errors;

paths[0] = "Root.Test";

tagManager.GetFolder(paths, out handles, out folders, out errors);
```

24.6.12.2.2. AddFolder/CheckFolder/ModifyFolder/DeleteFolder

```
TagManager tagManager = new ExaquantumAPI().TagManager;

// parameter set

QFolder[] f_folders = new QFolder[3];
int[] f_handles;
int[] f_errors;

// folder create
f_folders[0].Name       = "OPC";
f_folders[0].Path       = "Root.Test";
f_folders[0].Online     = true;
f_folders[1].Name       = "Man";
f_folders[1].Path       = "Root.Test";
f_folders[1].Online     = true;
f_folders[2].Name       = "CAL";
f_folders[2].Path       = "Root.Test";
f_folders[2].Online     = true;

tagManager.AddFolder(f_folders, out f_handles, out f_errors);

f_folders[0].Name       = "OPC-TEST";
// folder check
tagManager.CheckFolder(f_folders, out f_errors);

// folder modify
tagManager.ModifyFolder(f_handles, f_folders, out f_errors);

// folder delete
tagManager.DeleteFolder(f_handles, out f_errors, false);
```

24.7. Operation interface

The Operation object allows the control over the startup and shutdown of the Exaquantum server.

Namespace: Yokogawa.EXA.Exaquantum

NOTE: If the Operation interface is run from a remote client machine, the appropriate permissions should be configured. If not then the error “Access Denied” will be returned. The appropriate permissions can be configured by either of the following options:

a) The client application may be run as a local Administrator user exactly matching the original Administrator user on the Windows Server machine, i.e. the user name and password should match exactly. If the client application is run on a Windows Client OS, the user must be logged on as the local Administrator user.

b) On the Windows Server machine, change the Local Security Policy Security Option “User Account Control: Run all administrators in Admin Approval Mode” to “Disabled”. This disables all UAC functionality for the server machine.

24.7.1. Startup Method

24.7.1.1. Description

Startup Exaquantum. This is an asynchronous call.

24.7.1.2. Signature

```
Startup()
```

24.7.2. StatusChanged Event

24.7.2.1. Description

This event is fired on the completion of Exaquantum startup or shutdown.

24.7.2.2. Signature

```
Event EventHandler StatusChanged;
```

24.7.3. Shutdown Method

24.7.3.1. Description

Stop Exaquantum. This is an asynchronous call.

24.7.3.2. Signature

```
Shutdown()
```

24.7.4. Status Property

24.7.4.1. Description

Get the Exaquantum Service Status. The returned value is one of the following:

```
ContinuePending
Paused
PausePending
Running
StartPending
Stopped
StopPending
```

24.7.4.2. Signature

```
System.ServiceProcess.ServiceControllerStatus Status{get;}
```

24.7.5. IsMonitoring Property

24.7.5.1. Description

Set the notification of Exaquantum Service status change. In case of TRUE, notice Exaquantum Service status changed.

24.7.5.2. Signature

```
bool IsMonitoring(get; set;)
```

24.7.5.3. Sample Code for Startup and Shutdown

```
ExaquantumAPI m_api = new ExaquantumAPI();
Operation  m_ope;

public void StartUp()
{
 if(m_ope == null)
 {
  m_ope = m_api.Operation;
  m_ope.StatusChanged += new EventHandler(EventChange);
 }
 m_ope.Startup();
}

public void ShutDown()
{
 if(m_ope == null)
 {
  m_ope = m_api.Operation;
  m_ope.StatusChanged += new EventHandler(EventChange);
 }
 m_ope.Shutdown();
}

private void EventChange(object e, EventArgs arg) {
 Console.WriteLine(m_ope.Status.ToString());
}
```

24.8. Enumerations

This section defines the Enumerations user in the Exaquantum .NET API.

- TagType
- DataType
- AggregationType
- AggregationCalcType
- SummationTimeFactorType
- RawHistorianIntervalType
- OPCPeriodType
- TagFilterType
- QualityType

24.8.1. TagType Enumeration

Enumeration for the tag type

```
enum TagType{
    TAG_TYPE_MAN = 1,
    TAG_TYPE_OPC = 2,
    TAG_TYPE_CALC = 3,
    TAG_TYPE_SHORTCUT = 4
};
```

Table 24-3

Member name	**Description**
TAG_TYPE_OPC	OPC Tag
TAG_TYPE_MAN	Manual Tag
TAG_TYPE_CALC	Calc Tag
TAG_TYPE_SHORTCUT	Shortcut Tag

24.8.2. DataType Enumeration

Enumeration for the data type

```
enum DataType{
    DATA_TYPE_INTEGER = 2,
    DATA_TYPE_LONG = 3,
    DATA_TYPE_FLOAT = 4,
    DATA_TYPE_DOUBLE = 5,
    DATA_TYPE_STRING = 8
}
```

Table 24-4

Member name	**Description**
DATA_TYPE_INTEGER	Integer (VT_I2)
DATA_TYPE_LONG	Long (VT_I4)
DATA_TYPE_FLOAT	Float (VT_R4)
DATA_TYPE_DOUBLE	Double (VT_R8)
DATA_TYPE_STRING	String (VT_BSTR)

24.8.3. AggregationType Enumeration

Enumeration for the Aggregation Type

```
enum AggregationType{
    AGGREGATION_TYPE_CONTINUOUS,
    AGGREGATION_TYPE_DESCRETE
}
```

Table 24-5

Member name	**Description**
AGGREGATION_TYPE_CONTINUOUS	Continuous
AGGREGATION_TYPE_DISCRETE	Discrete

24.8.4. AggregationCalcType Enumeration

Enumeration for the aggregation calculation

```
enum AggregationCalcType{
    AGGREGATION_MAX = 4,
    AGGREGATION_MIN = 5,
    AGGREGATION_MEAN = 2,
    AGGREGATION_STDDEV = 6,
    AGGREGATION_SUM = 3,
    AGGREGATION_SPOT = 7,
    AGGREGATION_STATUSCOUNT = 8,
    AGGREGATION_ONSTATUS = 9
}
```

Table 24-6

Member name	**Description**
AGGREGATION_MAX	Maximum
AGGREGATION_MIN	Minimum
AGGREGATION_MEAN	Mean
AGGREGATION_STDDEV	Standard Deviation
AGGREGATION_SUM	Summation
AGGREGATION_SPOT	Spot Value
AGGREGATION_STATUSCOUNT	Status Count
AGGREGATION_ONSTATUS	State on Time

24.8.5. SummationTimeFactorType Enumeration

Enumeration for the Summation Time Factor.

```
enum SummationTimeFactorType{
    SUMMATION_TIME_FACTOR_NONE = 0,
    SUMMATION_TIME_FACTOR_SECOND = 1,
    SUMMATION_TIME_FACTOR_MINUTE = 60,
    SUMMATION_TIME_FACTOR_HOUR = 3600
}
```

Table 24-7

Member name	**Description**
SUMMATION_TIME_FACTOR_NONE	Time Factor is none
SUMMATION_TIME_FACTOR_SECOND	Time Factor is 1 second
SUMMATION_TIME_FACTOR_MINUTE	Time Factor is 1 minute
SUMMATION_TIME_FACTOR_HOUR	Time Factor is 1 hour

24.8.6. RawHistorianIntervalType Enumeration

Enumeration for the raw interval

```
enum RawHistorianIntervalType{
    RAW_INTERVAL_NONE = -1,
    RAW_INTERVAL_ONCHANGE = 0,
    RAW_INTERVAL_1SEC = 1000,
    RAW_INTERVAL_2SEC = 2000,
    RAW_INTERVAL_5SEC = 5000,
    RAW_INTERVAL_10SEC = 10000,
    RAW_INTERVAL_30SEC = 30000,
    RAW_INTERVAL_1MIN = 60000,
    RAW_INTERVAL_2MIN = 120000,
    RAW_INTERVAL_5MIN = 300000,
    RAW_INTERVAL_10MIN = 600000,
    RAW_INTERVAL_30MIN = 1800000,
    RAW_INTERVAL_1HOUR = 3600000
}
```

Table 24-8

Member name	Description
RAW_INTERVAL_NONE	Don’t write values to Historian
RAW_INTERVAL_ONCHANGE	Write a value when the value changes
RAW_INTERVAL_1SEC	Write a value at 1 second if the value has changed
RAW_INTERVAL_2SEC	Write a value at 2 seconds if the value has changed
RAW_INTERVAL_5SEC	Write a value at 5 seconds if the value has changed
RAW_INTERVAL_10SEC	Write a value at 10 seconds if the value has changed
RAW_INTERVAL_30SEC	Write a value at 30 seconds if the value has changed
RAW_INTERVAL_1MIN	Write a value at 1 minute if the value has changed
RAW_INTERVAL_2MIN	Write a value at 2 minutes if the value has changed
RAW_INTERVAL_5MIN	Write a value at 5 minutes if the value has changed
RAW_INTERVAL_10MIN	Write a value at 10 minutes if the value has changed
RAW_INTERVAL_30MIN	Write a value at 30 minutes if the value has changed
RAW_INTERVAL_1HOUR	Write a value at 1 hour if the value has changed

24.8.7. OPCPeriodType Enumeration

Enumeration for the OPC update period

```
enum OPCPeriodType{
    OPC_PERIOD_NONE = -1,
    OPC_PERIOD_1SEC = 1000,
    OPC_PERIOD_2SEC = 2000,
    OPC_PERIOD_5SEC = 5000,
    OPC_PERIOD_10SEC = 10000,
    OPC_PERIOD_30SEC = 30000,
    OPC_PERIOD_1MIN = 60000,
    OPC_PERIOD_2MIN = 120000,
    OPC_PERIOD_5MIN = 300000,
    OPC_PERIOD_10MIN = 600000,
    OPC_PERIOD_1HOUR = 3600000
}
```

Table 24-9

Member name	**Description**
OPC_PERIOD_NONE	Except OPC tag
OPC_PERIOD_1SEC	OPC update period is 1 second
OPC_PERIOD_2SEC	OPC update period is 2 seconds
OPC_PERIOD_5SEC	OPC update period is 5 seconds
OPC_PERIOD_10SEC	OPC update period is 10 seconds
OPC_PERIOD_30SEC	OPC update period is 30 seconds
OPC_PERIOD_1MIN	OPC update period is 1 minute
OPC_PERIOD_2MIN	OPC update period is 2 minutes
OPC_PERIOD_5MIN	OPC update period is 5 minutes
OPC_PERIOD_10MIN	OPC update period is 10 minutes
OPC_PERIOD_1HOUR	OPC update period is 1 hour

24.8.8. TagFilterType Enumeration

Enumeration for the tag filter

```
enum TagFilterType{
    TAG_FILTER_PATH = 1,
    TAG_FILTER_NAME = 2,
    TAG_FILTER_OPC_ITEM_ID = 3
    TAG_FILTER_GATEWAY = 4
}
```

Table 24-10

Member name	Description
TAG_FILTER_PATH	Filtering by path
TAG_FILTER_NAME	Filtering by tag name
TAG_FILTER_OPC_ITEM_ID	Filtering by OPC item ID
TAG_FILTER_GATEWAY	Filtering by OPC Gateway

24.8.9. QualityType Enumeration

Enumeration for tag quality

```
enum QualityType{
    QUALITY_BAD = 0,
    QUALITY_UNCERTAIN = 64,
    QUALITY_GOOD = 192
}
```

Table 24-11

Member name	Description
QUALITY_BAD	Bad
QUALITY_UNCERTAIN	Uncertain
QUALITY_GOOD	Good

24.9. Definition of Structures

This section details the content of the data structures that are used in the .NET API.

- ♦ QAggregationPeriod
- ♦ QTagTemplate
- ♦ QAggregationInfo
- ♦ QTag
- ♦ QFolder
- ♦ QTagFilter

24.9.1. QAggregationPeriod Structure

Structure of Production calendar period information.

Table 24-12

Member name	**Type**	**Range**	**Description**
Name	string	[1-32] characters	Production calendar period name
Description	string	[0-256] characters	Description of the aggregation period
Period	Int	!*1	Period in minutes
Offset	Int	!*2	Offset from the period boundary, in minutes
ParentHandle	Int	!*3	When making a customization the deadline source, after using QueryAggregationPeriod (or similar means) and receiving a handle, please define it. Instant value, hourly report, daily report, monthly report handles are defined fixed values. (*4)

*1: Must be one of the following: 2,3,4,5,6,10,12,15,20,30,60,90,120,180,240,360,480,720,1440

*2: Must be less than the Period

*3: See the following

List of effective value of aggregation source data handles

Table 24-13

Aggregation Period	**Effective value of aggregation source data handles**
Hour	Raw
Day	Raw, Hour, Custom

Aggregation Period	Effective value of aggregation source data handles
Month	Raw, Hour, Data, Custom
Custom	Raw

*4: See the following

Fixed values defined within the framework

Table 24-14

Fixed Value	Description
InstantValue	Instant Value Handle
HourlyReportHandle	Hourly Report Handle
DailyReportHandle	Daily Report Handle
MonthlyReportHandle	Monthly Report Handle
MinAggregationNameLength	Minimum operation calendar name length
MaxAggregationNameLength	Maximum operation calendar name length
MaxDescriptionLength	Maximum comment length
MinHourlyOffset	Minimum hourly offset
MaxHourlyOffset	Maximum hourly offset
MinDailyOffset	Minimum daily offset
MaxDailyOffset	Maximum daily offset
MinMonthlyOffset	Minimum monthly offset
MaxMonthlyOffset	Maximum monthly offset
MonthlyOffsetLastDay	Offset on the last day of the month
HourlyBasicPeriod	Hourly basic period
DailyBasicPeriod	Daily basic period
MonthlyBasicPeriod	Monthly basic period

24.9.2. QTagTemplate Structure

Structure for Tag Template operation

Table 24-15

Member name	**Type**	**Range**	**Description**
Name	String	[1-32] characters	Tag template name
Version	Int		Version
TagType	TagType		Tag type
DataType	DataType		Data type
AggregationType	AggregationType		Aggregation type
RawHistorianInterval	RawHistorianInterval Type		Raw Historian Interval
EngUnitsFlag	Bool		Engineering Unit
EngRangeFlag	Bool		Engineering range
DescriptionFlag	Bool	True	Tag Description
AggregationPeriodIDs	Int[]	*1	Array of Aggregation period ID(s) Set to NULL if no aggregations are used
WriteToOPCGateway Flag	Bool	*2	Writes to OPC gateway.
OPCPeriod	OPCPeriodType	*3	OPC update period ID
PercentDeadband	Float	*4	Deadband
AggregationSupplied _ ByCalculationFlag	Bool[]	*5	Aggregation is made with the computational expression.
AggregationInfo	QAggregationInfo[]		Detail of aggregation information

*1: Get valid value using QueryAggregationPeriod

*2: Always False when TagType is not OPC.

*3: Must be set to OPC_PERIOD_NONE when not OPC tag.

*4: Must be one of the following values: 0,0.01,0.1,0.2,0.5,1,1.5,2,5,100.
Except for the OPC tags, specify QTagTemplate.PercentDeadbandNoUse.

*5: The contents of AggregationSuppliedByCalculationFlag should correspond to the contents of AggregationPeriodIDs. If TagType is not CALC, the contents should be all FALSE. The content can be TRUE, only if the size is 1 (one).

24.9.2.1. List of Parameter used by each TagType

These are used by Tag Template

Table 24-16

Parameter Name	**Description**	**OPC Tag**	**MAN Tag**	**Calc Tag**
Name	Tag Template Name	X	X	X
TagType	Tag Type	X	X	X
DataType	Data Type	X	X	X
AggregationType	Aggregation Type	X	X	X
RawHistorianInterval	Raw Historian Interval	X	X	X
EngUnitsFlag	Engineering Units	XX*1	XX*1	XX*1
EngRangeFlag	Engineering Range	XX	XX	XX
DescriptionFlag	Descriptions	X*3	X*3	X*3
AggregationPeriods	Aggregation Periods	XX*2	XX*2	XX*2
Write to OPC Gateway	Writes to OPC gateway	XX	Blank	Blank
Period	OPC update period	X	Blank	Blank
Percent Deadband	Dead band	X	Blank	Blank
AggregationSuppliedByCalculation	Aggregation period determined with computed expressions	Blank	Blank	XX
AggregationInfo	Detailed aggregation information	XX*2	XX*2	XX*2

X:	Mandatory
XX:	Optional
Blank:	Not settable
*1:	Not settable when the DataType is string.
*2:	If AggregationPeriods has one or more elements (i.e. one or more aggregation periods defined), then AggregationInfo must also have one or more elements (one or more aggregation types defined).
*3:	It must always be true.

24.9.3. QAggregationInfo Structure

Structure of aggregation information.

Table 24-17

Member name	**Type**	**Range**	**Description**
Aggregatio n	AggregationCalcTyp e		Aggregation type.
Parameter	Object	*1	Parameters required for certain aggregations

Table 24-18

Aggregation Type	**Enum Value**	**Parameter**	**Continuous**	**Discrete**
Maximum	AGGREGATION_MAX	NULL	X	Blank
Minimum	AGGREGATION_MIN	VT_ NULL	X	Blank
Mean	AGGREGATION_MEAN	NULL	X	Blank
Standard Deviation	AGGREGATION_STDDEV	NULL	X	Blank
Summation	AGGREGATION_SUM	Object[5]:*2 [0] Summation Time Factor (Summation TimeFactor Type value) [1] Differential Summation Flag [2] Decimal Places [3] Maximum Value [4] Percentage Difference	X	Blank
Spot Value	AGGREGATION_SPOT	NULL	X	Blank
Status Count	AGGREGATION_ STATUSCOUNT	On Status *1	Blank	X
State on Time	AGGREGATION_ONSTAT US	On Status *1	Blank	X

*1: These must have the same value. The value must be a string, with a maximum of 32 characters

*2: This parameter must be a 5 element array of type Object.

24.9.4. QTag Structure

Structure used with Tag management functions.

Table 24-19

Member name	Type	Range	Description
Name	String	[1-32] characters	Tag name
Path	String	[1-256] characters (Including Name, where applicable)	Parent path of tag
TemplateName	String	[1-32]characters	Template name
Description	String	[0-256] characters	Tag Description
Units	String	[0-256] characters	Unit
LowEng	Double	*1	Engineering lower limit
HighEng	Double	*1	Engineering upper limit
InitValue	As appropriate for data type	*2	Initial value (Used in Man tag)
InitQuality	QualityType		Initial quality (Used in Man tag)
OPCGateWay	String	32 (*3)	OPC gateway name (Used in OPC tags)
OPCItemID	String	244(*3)	OPC item ID (Used in OPC tags)
Script	String	No HighRange limit(*4)	Script (Used in Calculation tags)
LockStatus	Bool		Lock status (Template is not reflected.)
Online	Bool		TRUE if tag is to be set online.

*1: Only applicable for numeric tags.

*2: Only applicable to Manual tags. When using a numeric data type, and the tag template calls for LowEng and HighEng, then the value must be between LowEng and HighEng, and within the range of the data type. When using a string data type, the value can be up to, and including 256 characters. For tag types other than Manual, this should be set to NULL.

*3: This is only applicable to OPC tags. For non-OPC tags, this should be set to NULL.

*4: This is only applicable to Calculated tags. For non-Calculated tags, this should be set to NULL.

24.9.4.1. List of Parameter used by each TagType

Table 24-20

Parameter Name	Description	OPC Tag	MAN Tag	Calc Tag
Name	Tag Name	X	X	X
Path	Parent Path	X	X	X
TemplateName	Template Name	X	X	X
Description	Description	XX	XX	XX
Units	Units	XX*1	XX*1	XX*1
LowEng	Low Range	XX*2	XX*2	XX*2
HighEng	High Range	XX*2	XX*2	XX*2
InitValue	Value	Blank	X	Blank
InitQuality	Data Quality	Blank	X*3	Blank
OPCGateway	OPC Gateway	X	Blank	Blank
OPCItemID	OPC Item ID	X	Blank	Blank
Script	Script	Blank	Blank	X
LockStatus	Lock Status	XX	XX	XX
Online	Online	XX	Blank	XX

*1: Only required when Engineering Units are defined by the Tag Template.

*2: Only required when Engineering Range is defined by the Tag Template.

*3: When adding a tag, the initial quality is always set to Good, regardless of what is specified. This value is used when modifying the tag.

24.9.5. QFolder Structure

Structure used for Folder management functions.

Table 24-21

Member name	Type	Range	Description
Name	String	[1-32] Characters	Folder name
Path	String	[1-256] Characters (Including folder name, where applicable)	Parent Path of Folder
Online	Bool	True or False	Online flag *1

*1: Exaquantum R2.20 always returns TRUE.

24.9.6. QTagFilter Structure

Structure used for retrieving tags

Table 24-22

Member name	Type	Range	Description
tagFilterType	QTagFilterType		Filter type
filter	String		Filtering conditions

24.10. Error Code List

Error code list API was defined following name space.

Yokogawa.Exa.Exaquantum.Common.ErrorCode

Table 24-23

No	Member name	Description
14	EQ_E_ANALYZE	Analyze Error
37	EQ_E_DUPLICATE_HANDLE	Duplicate handle(s)
13	EQ_E_DUPLICATE_NAME	Duplicate name in array parameter.
-1	EQ_E_FAIL	Failure
99	EQ_E_INTERNAL_ERROR	Internal Error
38	EQ_E_INUSE	One or more objects are being referenced
30	EQ_E_INVALID_AGGREGATIONINFO	Invalid Aggregation Calculation
25	EQ_E_INVALID_AGGREGATION_PERIOD	Invalid Aggregation Period
21	EQ_E_INVALID_AGGREGATIONTYPE	Invalid Aggregation Type
35	EQ_E_INVALID_ARGUMENT	Invalid Argument
29	EQ_E_INVALID_CALCFLAG	Invalid Calculation Flag
36	EQ_E_INVALID_CALLING_SP	Error calling an internal Stored Procedure
33	EQ_E_INVALID_CONSTRUCT	Error in Construct
20	EQ_E_INVALID_DATATYPE	Invalid DataType
28	EQ_E_INVALID_DEADBAND	Invalid DeadBand
4	EQ_E_INVALID_DESCRIPTION	Invalid Description
15	EQ_E_INVALID_HANDLE	Invalid Handle
7	EQ_E_INVALID_HIGHENG	Invalid High Range
22	EQ_E_INVALID_HISTORIANINTERVAL	Invalid Historian Interval
6	EQ_E_INVALID_LOWENG	Invalid Low Range
1	EQ_E_INVALID_NAME	Invalid Name
17	EQ_E_INVALID_OFFSET	Invalid Offset

No	**Member name**	**Description**
10	EQ_E_INVALID_OPC_GATEWAY	Invalid OPC Gateway
11	EQ_E_INVALID_OPC_ITEMID	Invalid ItemID
27	EQ_E_INVALID_OPCPERIOD	Invalid OPC Period
18	EQ_E_INVALID_PARENTHANDLE	Invalid Handle Parameter
2	EQ_E_INVALID_PATH	Invalid Path
16	EQ_E_INVALID_PERIOD	Invalid Period
9	EQ_E_INVALID_QUALITY	Invalid Quality
24	EQ_E_INVALID_RANGE	Invalid Range
12	EQ_E_INVALID_SCRIPT	Invalid Script
19	EQ_E_INVALID_TAGTYPE	Invalid TagType
34	EQ_E_INVALID_TEMPLATE	Invalid TagTemplate
3	EQ_E_INVALID_TEMPLATENAME	Invalid TagTemplate name
5	EQ_E_INVALID_UNITS	Invalid Units
8	EQ_E_INVALID_VALUE	Invalid Initial Value
26	EQ_E_INVALID_WRITE_OPCGATEWAY	Invalid Write To OPC Gateway flag
32	EQ_E_ITEMEXIST	This object already exists
39	EQ_E_OTHERS_ANALYSE_ERROR	Error caused by other analyze error
31	EQ_E_TRANSACTIONOPERATION	Transaction Error
0	S_OK	SUCCEED

This page intentionally left blank

Chapter 25 Accessing Exaquantum COM API using Visual Studio 2015

To work with COM objects in Visual Studio 2015 a RCW (Runtime-callable wrapper) must be created. This allows the clients written in .Net (such as Applications, Components) to think that they are calling .Net code. The RCW acts as a proxy between the managed and unmanaged (COM) code, handling all administrative tasks such as marshalling, lifetime management and so on.

When referencing a COM object in Visual Studio .Net, the RCW is automatically created in the background by the CLR (Common Language Runtime); this parses the COM type library (TypeLib), and produces a managed DLL whose metadata describes the methods that wrap the COM object's interface.

Once the RCW is created a managed application can now create instances of the object and use it as though it were a native managed type.

25.1. Using Exaquantum COM objects in Visual Studio

To use the Exaquantum COM API, references can be created to the appropriate Exaquantum COM objects. These objects (DLLs) can be found in the folder

<Installation Root Folder>\Yokogawa\Exaquantum PIMS\System.

Once the DLLs have been referenced, they can be used like any other DLL, managed or unmanaged. For more information on how to use the Exaquantum API's please see Chapter 2.

This page intentionally left blank

Chapter 26 Reading Exaquantum data with C# using the COM API

This section outlines examples of how to use the COM API for reading:

- Historical Data
- Live Data
- Alarm and Events Data

The COM API supports both 32-bit and 64-bit C# application types.

Interop.QuantumAutomationLib.dll must be added as a reference within the C# project to access the API functions. This is achieved by:

1. Running Visual Studio 2019
2. Right-clicking on ‘References’ from the Solution Explorer
3. Clicking ‘Add Reference’
4. Select the ‘Browse’ tab, and click the ‘Browse’ button
5. Navigate to ‘<Exaquantum Installation Directory>/System’ and select Interop.QuantumAutomationLib.dll

26.1. Reading the Latest Item Value

This is a Synchronous data read of live values, used when the latest value of a tag is required.

This example reads the current value, quality, and timestamp for one Item:

This process consists of the following steps:

Table 26-1

Step	Section Description	Section
1	Import Libraries to Project	26.1.1
2	Create qdaMulti Object	26.1.2
3	Set Spot Read Processing Options	26.1.3
4	Read Live Spot Data	26.1.4
5	Read Live Trend Data	26.1.5

26.1.1. Import Libraries to Project

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using QDAMULTILib;
```

26.1.2. Create qdaMulti object

```
//Create qdaMulti object
var qdaMulti = new QDataAccessMulti();
//Add tag name to read from
 qdaMulti.Add(tagName);
```

26.1.3. Set Spot Read Processing Options

```
qdaMulti.SetQueryTimes("NOW", null, null);
//Sets the options for processing raw data - default values for simplicity
qdaMulti.SetProcessOptions(null,
  qdaInterpMethod.qdaDefault,
 qdaEdgeValueOptions.qdaIncludeBoundingValues,
 null, null, false, -1);
```

26.1.4. Read Live Spot Data

```
//Read the current VQT values – Spot Read
var currentVqt = (Array)qdaMulti.ReadValue();
var value = (Array)currentVqt.GetValue(1, 0);

var v = value.GetValue(0, 0);
var q = value.GetValue(1, 0);
var t = value.GetValue(2, 0);

Console.WriteLine($"Current Value for {tagName}, Value {v}, Quality {q},
Timestamp {t}");
```

26.1.5. Read Live Trend Data

```
//Read Trend data from the previous 10 minutes
qdaMulti.SetQueryTimes(null, "NOW-10 MINUTES", "NOW");
//Read the Trend data
var prev10Vqt = (Array)qdaMulti.ReadValue();
var prev10value = (Array)prev10Vqt.GetValue(1, 0);

for(var i = 0; I < prev10value.GetLength(1); i++)
{
    var prev10v = prev10value.GetValue(0, i);
    var prev10q = prev10value.GetValue(1, i);
    var prev10t = prev10value.GetValue(2, i);
    Console.WriteLine($"Value 10 minutes ago for {tagName}, Value
{prev10v}, Quality {prev10q}, Timestamp {prev10t}");
}
```

26.2. Reading the History of Item Values

The History data of a tag can be read using a defined start and end time.

This example reads the value, quality, and timestamp for an Item from History, from 10 minutes before the current time to the current time.

This process consists of the following steps:

Table 26-2

Step	Section Description	Section
1	Import Libraries to Project	26.2.1
2	Create qdaMulti Object	26.2.2
3	Set History Processing Options	26.2.3
4	Read History Data	26.2.4

26.2.1. Import Libraries to Project

```
using System;
using QDAMULTILib;
```

26.2.2. Create qdaMulti Object

```
//Create qdaMulti object
var qdaMulti = new QDataAccessMulti();
//Add tag name to read from
qdaMulti.Add(tagName);
qdaMulti.SetQueryTimes(null,“NOW-20 MINUTES”,“NOW-10 MINUTES”,);
```

26.2.3. Set History Read Processing Options

```
//Sets the options for processing raw data - default settings for simplicity
qdaMulti.SetProcessOptions(null,
qdaInterpMethod.qdaDefault,
qdaEdgeValueOptions.qdaIncludeBoundingValues,
null, null, false, -1);
```

26.2.4. Read History Trend Data

```
//Read 10 minutes of Historical Trend data
 var readVqt = (Array)qdaMulti.ReadValue();
 var value = (Array)readVqt.GetValue(1, 0);

 for(var i = 0; I < prev10value.GetLength(1); i++)
 {
   var v = value.GetValue(0, 0);
   var q = value.GetValue(1, 0);
   var t = value.GetValue(2, 0);
   Console.WriteLine($"Value for {tagName}, Value {v}, Quality {q},
 Timestamp {t}");
 }
```

26.3. Reading the History of Alarm and Events

The history of Alarm and Events data for a specified OPC Server can be read using a defined start and end time.

This example reads the Alarm and Events for the user-specified OPC Gateway from History.

The process to read Alarm and Events data consists of the following steps:

Table 26-3

Step	Section Description	Section
1	Import Libraries to Project	26.3.1
2	Connect to Database	26.3.2
3	Execute Query to Retrieve History Stream ID	26.3.3
4	Create QAEAccess Object	26.3.4
5	Set Query Times	26.3.5
6	Read Alarm and Events Data	26.3.6

26.3.1. Import Libraries to Project

```
using System;
using Interop.QuantumAutomationLib;
using System.Data.OleDb;
```

26.3.2. Connect to Database

```
object pump = null;
//Set up the connection to the database
string ConnectStr = “Provider=sqloledb;”;
ConnectStr = ConnectStr + “Network Library=dbmssocn;”;
ConnectStr = ConnectStr + “Server=<Server Name>;”;
ConnectStr = ConnectStr + “database=Qconfig;”;
ConnectStr = ConnectStr + “Trusted_Connection=yes”;
```

26.3.3. Execute Query to Retrieve History Stream ID

```
string queryString = “SELECT OPCAEPumpHistId FROM OPCServer WHERE Name =
‘<OPC Server name>’”;
using (OleDbConnection connection = new OleDbConnection(ConnectStr))
    {OleDbCommand command = new OleDbCommand(queryString, connection);

     //Connect to the database and read the History Stream ID
     connection.Open();
     OleDbDataReader reader = command.ExecuteReader();
     while (reader.Read())
           {pump = reader[0];}
     reader.Close();}
```

26.3.4. Create QAEAccess Object

```
//Create the QAEAccess object
var QAEA = new QAEAccess();
QAEA.Add(pump);
```

26.3.5. Set the Query Times

```
//Set the query times
DateTime startTime = new DateTime(2023, 8, 9, 10, 0, 0);
DateTime endTime = new DateTime(2023, 8, 9, 11, 0, 0);
QAEA.SetQueryTimes(startTime, endTime);
```

26.3.6. Read Alarm and Events Data

```
          //Read the values returned
          var Events = QAEA.ReadValue();
          // Loop through the entire array and store the AE data
          // NB: In this sample only the last data in the Array is stored when
     we are out of the loop
          for (int Count = 0; Count < Events.Length - 1; Count++)
           {var vOPCAEPumpHistID = Events[0, Count];
           var sSource = Events[1, Count];
           var sMessage = Events[2, Count];
           var dTimeStamp = Events[3, Count];
           var lSequenceNo = Events[4, Count];
           var lEventCategory = Events[5, Count];
           var lSeverity = Events[6, Count];
           var lCookie = Events[7, Count];
           var sConditionName = Events[8, Count];
           var sSubConditionName = Events[9, Count];
           var lChangeMask = Events[10, Count];
           var lNewState = Events[11, Count];
           var lConditionQuality = Events[12, Count];
           var bAckRequired = Events[13, Count];
           var dActiveTime = Events[14, Count];
           var sActorID = Events[15, Count];
           var lNoOfAttributes = Events[16, Count];
           var vEventAttributesArray = Events[17, Count];

             // Loop through all the EventAttributes to print
             for (int i = 0; i <= lNoOfAttributes - 1; i++)
      {var vEventAttribute = vEventAttributesArray[i];
                 Console.WriteLine(vEventAttribute);}}
```

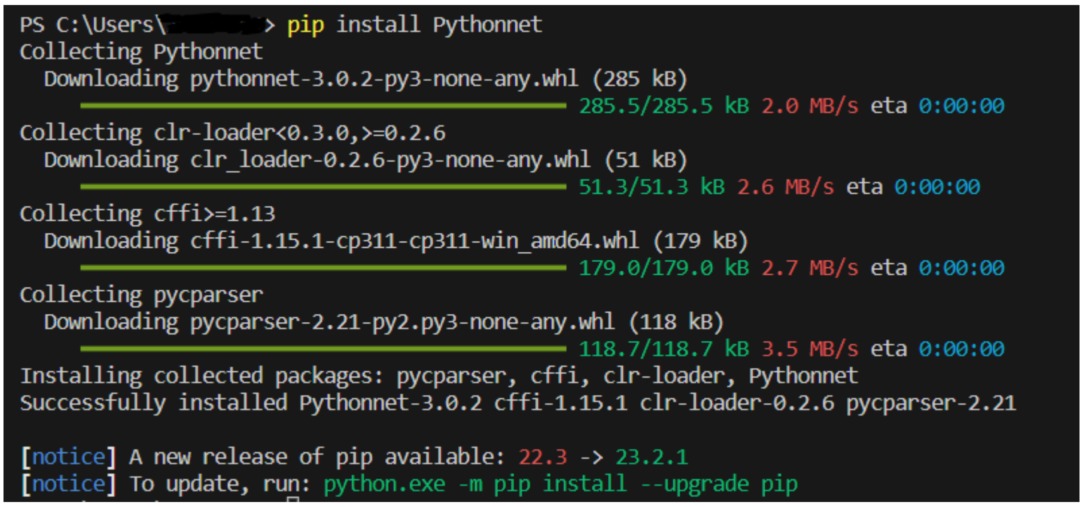
Chapter 27 Reading Exaquantum data with Python using the COM API

This section outlines examples of how to use the COM API for reading:

- Historical Data Synchronously
- Live Data Synchronously
- Alarm and Events Data

As a prerequisite, the Pythonnet package on python3 must be installed on the Exaquantum Client to run this script. Pythonnet grants integration with the .NET Common Language Runtime (CLR).

To install Pythonnet, enter 'pip install Pythonnet' in a PowerShell command prompt.



```
PS C:\Users\[redacted]> pip install Pythonnet
Collecting Pythonnet
  Downloading pythonnet-3.0.2-py3-none-any.whl (285 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 285.5/285.5 kB 2.0 MB/s eta 0:00:00
Collecting clr-loader<0.3.0,>=0.2.6
  Downloading clr_loader-0.2.6-py3-none-any.whl (51 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.3/51.3 kB 2.6 MB/s eta 0:00:00
Collecting cffi>=1.13
  Downloading cffi-1.15.1-cp311-cp311-win_amd64.whl (179 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.0/179.0 kB 2.7 MB/s eta 0:00:00
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.7/118.7 kB 3.5 MB/s eta 0:00:00
Installing collected packages: pycparser, cffi, clr-loader, Pythonnet
Successfully installed Pythonnet-3.0.2 cffi-1.15.1 clr-loader-0.2.6 pycparser-2.21

[notice] A new release of pip available: 22.3 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Figure 27-1 Installing Pythonnet with PowerShell

python3 can be installed via python.org/downloads/

27.1. Reading the Latest Item Value

This is a Synchronous data read of live values, used when the latest value of a tag is required.

This example reads the current value, quality, and timestamp for one Item:

This process consists of the following steps:

Table 27-1

Step	**Section Description**	**Section**
1	Import Libraries to Project	27.1.1
2	Import Assembly into Project	27.1.2
3	Create the QDAMulti Object	27.1.3
4	Add Item IDs to the QDAMulti Item Collection	27.1.4

5	Read Live Spot Data	27.1.5

27.1.1. Import Libraries to Project

```
import clr
import sys
```

27.1.2. Import Assembly into Project

```
# Set the path to the assembly directory
assemblydir = r″C:\Program Files (x86)\Yokogawa\Exaquantum PIMS\System″
sys.path.append(assemblydir)
clr.AddReference(‘Interop.QuantumAutomationLib’)

# Import contents of the assembly
from Interop.QuantumAutomationLib import *
```

27.1.3. Create the QDAMulti Object

```
QDAMulti = None
QDAMulti = QdataAccessMulti()
```

27.1.4. Add Item IDs to the QDAMulti Item Collection

```
# Create access strings
item_1 = ‘<Tag Path> ’
# Adding the Exaquantum Item IDs to the Data Access Item Collection
QDAMulti.Add(item_1)
```

27.1.5. Read Live Spot Data

```
# Read current values of all Items in the collection
itemData = QDAMulti.ReadValue()

# Retrieve Value Quality Timestamp (VQT) for item_1
item_1_Value = itemData[1,0][0,0]
item_1_Quality= itemData[1,0][1,0]
item_1_Timestamp = itemData[1,0][2,0]
```

27.2. Reading the History of Item Values

The History data of a tag can be read using a defined start and end time.

This example reads the value, quality, and timestamp for an Item from History, from 10 minutes before the current time to the current time.

This process consists of the following steps:

Table 27-2

Step	Section Description	Section
1	Import Libraries to Project	27.2.1
2	Import Assembly into Project	27.2.2
3	Create the QDAMulti Object	27.2.3
4	Add Item IDs to the QDAMulti Item Collection	27.2.4
5	Set Query Times For History Read	27.2.5
6	Read the History Data	27.2.6

27.2.1. Import Libraries to Project

```
import clr
import sys
```

27.2.2. Import Assembly into Project

```
# Set the path to the assembly directory
assemblydir = r″C:\Program Files (x86)\Yokogawa\Exaquantum PIMS\System″
sys.path.append(assemblydir)
clr.AddReference(‘Interop.QuantumAutomationLib’)

# Import contents of the assembly
from Interop.QuantumAutomationLib import *
```

27.2.3. Create the QDAMulti Object

```
QDAMulti = None
QDAMulti = QdataAccessMulti()
```

27.2.4. Add Item IDs to the QDAMulti Item Collection

```
# Create access strings
item_1 = '<Tag Path> '
# Adding the Exaquantum Item IDs to the Data Access Item Collection
QDAMulti.Add(item_1)
```

27.2.5. Set Query Times For History Read

```
# Get all values stored in history from yesterday to now for item_1
QDAMulti.SetQueryTimes(StartTime="now - 10 min", EndTime="now")
```

27.2.6. Read the History Trend Data Values

```
itemData = QDAMulti.ReadValue()
# Iterate through the returned data retrieving the history values
NumberOfPoints = len(itemData[1,0])//3

for nPoint in range(NumberOfPoints):
    itemValue = itemData[1,0][0,nPoint]
    itemQuality= itemData[1,0][1,nPoint]
    itemTimestamp = itemData[1,0][2,nPoint]
```

27.3. Reading the History of Alarm and Events

The history of Alarm and Events data for a specified OPC Server can be read using a defined start and end time.

This example reads the Alarm and Events for the user-specified OPC Gateway from History.

The process to read Alarm and Events data consists of the following steps:

Table 27-3

Step	Section Description	Section
1	Import Libraries to Project	27.3.1
2	Import Assembly into Project	27.3.2
3	Connect to Database	27.3.3
4	Retrieve History Stream ID From Database	27.3.4
5	Convert History Stream ID to a Byte[]	27.3.5
6	Set Configurations for Data Read	27.3.6
7	Create QAEAccess Object	27.3.7
8	Begin Alarm and Events Data Read	27.3.8

27.3.1. Import Libraries to Project

```
import clr
import sys
```

27.3.2. Import Assembly into Project

```
# Set the path to the assembly directory
assemblydir = r″C:\Program Files (x86)\Yokogawa\Exaquantum PIMS\System″
sys.path.append(assemblydir)
clr.AddReference(‘Interop.QuantumAutomationLib’)

# Import contents of the assembly
from Interop.QuantumAutomationLib import *
```

27.3.3. Connect to Database

```
# Connects to the database
mydb = pyodbc.connect(‶Driver={SQL Server};″
                      ‶Server=<Server Name>;″
                      ‶Database=Qconfig;″
                      ‶Trusted_Connection=yes;″)
cursor = mydb.cursor()
```

27.3.4. Retrieve History Stream ID From Database

```
cursor.execute(‶SELECT OPCAEPumpHistId FROM OPCserver WHERE Name = ‘<OPC
Server>’″)

row = cursor.fetchone()
row_as_list = [x for x in row]
b = row_as_list[0]
```

27.3.5. Convert History Stream ID to a Byte[]

```
# Here we Convert the ID into a Byte[] this Makes it Readable by the COM
API
c_sharp_bytes = Array[Byte](b)
```

27.3.6. Set Configurations for Data Read

```
# Set Start and End time for Alarm Data
Cstart =  DateTime(2023, 8, 8)
Cend =  DateTime(2023, 8, 9)
```

27.3.7. Create QAEAccess Object

```
QAEA = None
QAEA = QAEAccess()
QAEA.Add(c_sharp_bytes)
QAEA.SetQueryTimes(Cstart, Cend)
```

27.3.8. Begin Alarm and Events Data Read

```
itemData = QAEA.ReadValue()
# Here we Iterate through the A&E Data
for nPoint in range(len(itemData)):
    vOPCAEPumpHistID = itemData[0, nPoint]
    sSource= itemData[1,nPoint]
    sMessage = itemData[2, nPoint]
    dTimeStamp = itemData[3, nPoint]
    lSequenceNo = itemData[4, nPoint]
    lEventCategory = itemData[5, nPoint]
    lSeverity = itemData[6, nPoint]
    lcookie = itemData[7, nPoint]
    sConditionName = itemData[8, nPoint]
    sSubConditionName = itemData[9, nPoint]
    lChangeMask = itemData[10, nPoint]
    lNewState = itemData[11, nPoint]
    lConditionQuality = itemData[12, nPoint]
    bAckRequired = itemData[13, nPoint]
    dActiveTime = itemData[14, nPoint]
    sActorID = itemData[15, nPoint]
    vEventAttributesArray = itemData[16,nPoint]

    # Here we Output the Data
    print(vOPCAEPumpHistID, sSource, sMessage, dTimeStamp, lSequenceNo,
lEventCategory, lSeverity)
```

Chapter 28 Reading Exaquantum data using OPC UA

The sections in this chapter outline what OPC UA is and how to access the OPC UA API using an application called UaExpert to read live and historical data.

28.1. What is OPC UA?

The original OPC standard, known as OPC Classic, was introduced in the mid-1990s, as a standard for interconnection of process control devices. It is based on the Microsoft Windows COM and DCOM communication technology. The OPC Classic standard has a number of shortcomings, notably:

- Use of DCOM – can be difficult to configure (especially when using Firewalls).
- DCOM has a non-configurable timeout of approximately 6-7 minutes.
- Restricted to Windows platforms mainly.
- Has security vulnerabilities.

OPC UA was introduced, as a standard, in the early-2000s, to address these issues. Some important benefits, when compared to OPC Classic, are:

- Firewall friendly.
- Improved data transfer security.
- Improved data integrity.
- Uses a single port, without needing a callback.

28.2. Connecting to the Exaquantum OPC UA API

To connect an OPC UA Client with an Exaquantum server using the Exaquantum OPC UA API, the connection parameters detailed in table 28-1 are required.

Table 28-1

Parameter	**Definition**	**Format**
URL	URL of the Exaquantum Server	opc.tcp://ExaquantumServerName
Port	Port of the Exaquantum Server for OPC UA communication	:34488 (this is the default port)

28.3. Use of UaExpert as an OPC UA Client

UaExpert is an OPC UA Client designed as a general-purpose client that supports OPC UA features such as Data Access, Alarms and Conditions, Historical Access and calling UA Methods. It is cross platform with the ability to extend functionality by plugins and is available for 64-bit Windows and Linux systems.

UA Expert can be downloaded from Unified Automations official website at the link below:

https://www.unified-automation.com/products/development-tools/uaexpert.html

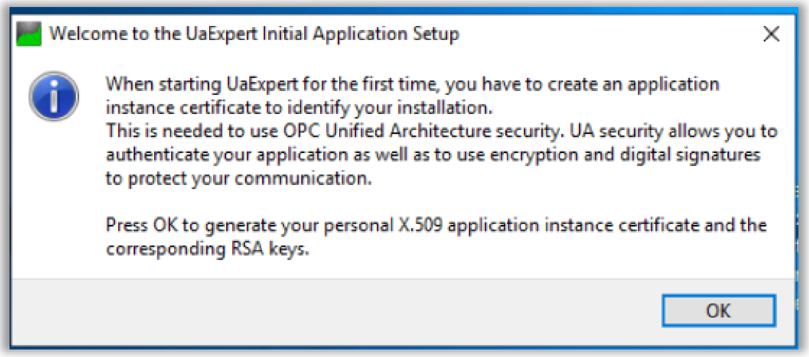
The rest of this chapter details how to use UaExpert to carry out the following:

- Run UaExpert for the first time
- Connect UaExpert to an Exaquantum Server
- Read Live Data
- Read Historical Data

28.3.1. Using UaExpert for the first time

Connection and reading of Exaquantum data with UaExpert must be performed once UaExpert is installed on the client machine.

1. Start UaExpert, when starting UaExpert for the first time, an information message is displayed:

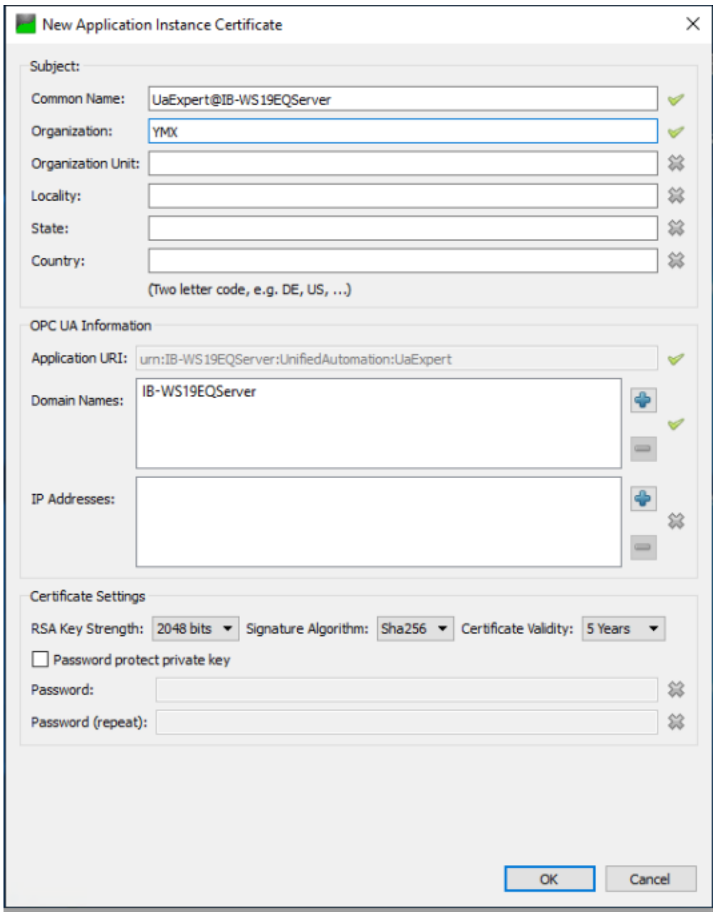


When starting UaExpert for the first time, you have to create an application instance certificate to identify your installation.
This is needed to use OPC Unified Architecture security. UA security allows you to authenticate your application as well as to use encryption and digital signatures to protect your communication.

Press OK to generate your personal X.509 application instance certificate and the corresponding RSA keys.

Figure 28-1 Welcome message to UaExpert

2. A new window will appear, where the only required subject is the “Organization”:



New Application Instance Certificate

Subject:

Common Name: UaExpert@IB-WS19EQServer

Organization: YMX

Organization Unit:

Locality:

State:

Country:

(Two letter code, e.g. DE, US, ...)

OPC UA Information

Application URI: urn:IB-WS19EQServer:UnifiedAutomation:UaExpert

Domain Names: IB-WS19EQServer

IP Addresses:

Certificate Settings

RSA Key Strength: 2048 bits Signature Algorithm: Sha256 Certificate Validity: 5 Years

☐ Password protect private key

Password:

Password (repeat):

Figure 28-2 "New Application Instance Certificate” Window

28.3.2. Connection to the Exaquantum PIMS Server

Next, the “NewProject” window will load. Follow next steps:

1. Click on the blue plus icon “Add Server”.

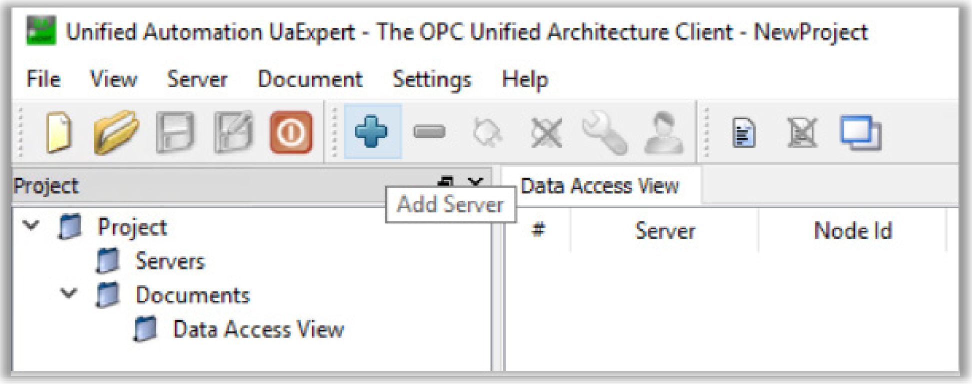
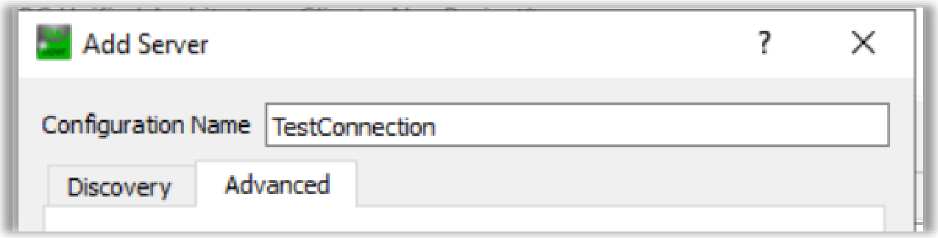


Figure 28-3 "New Project" Window

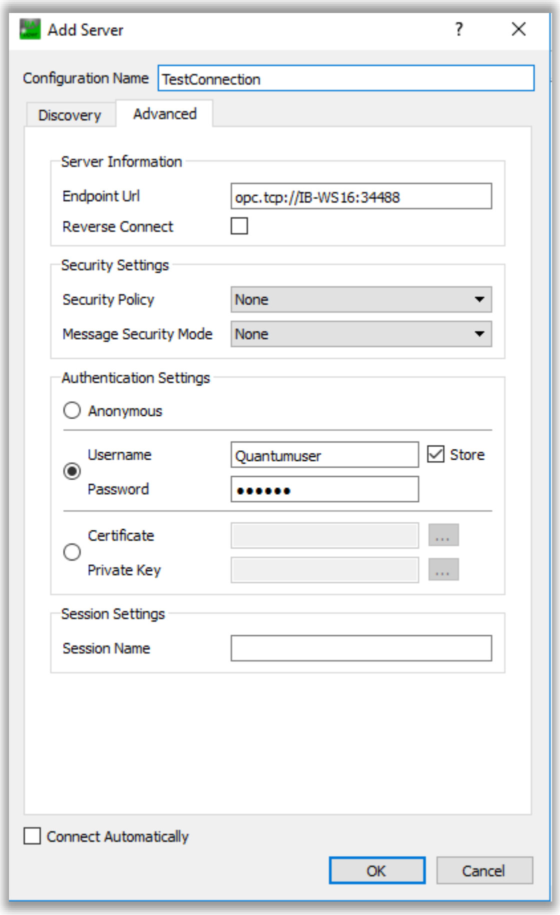
2. Fill the “Configuration Name” field with a descriptive name, i.e.: “TestConnection”.



Configuration Name: TestConnection

Figure 28-4 "Advanced" tab in "Add Server" Window

3. Select “Advanced” tab. Enter the “Endpoint Url” as indicated in Section 28.2, then enter a user and its password. Tick the “Store” checkpoint, then finally click the OK button.



Configuration Name: TestConnection

Endpoint Url: opc.tcp://IB-WS16:34488

Reverse Connect: ☐

Security Policy: None

Message Security Mode: None

○ Anonymous

● Username: Quantumuser ☑ Store

Password: ••••••

Session Name:

☐ Connect Automatically

Figure 28-5 Server Connection Configuration in "Add Server" Window

4. A plug icon as shown below, will appear; right-click on it and select “Connect”.

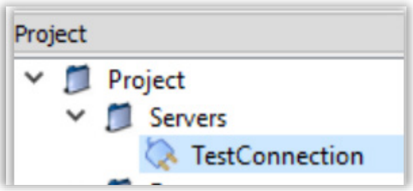
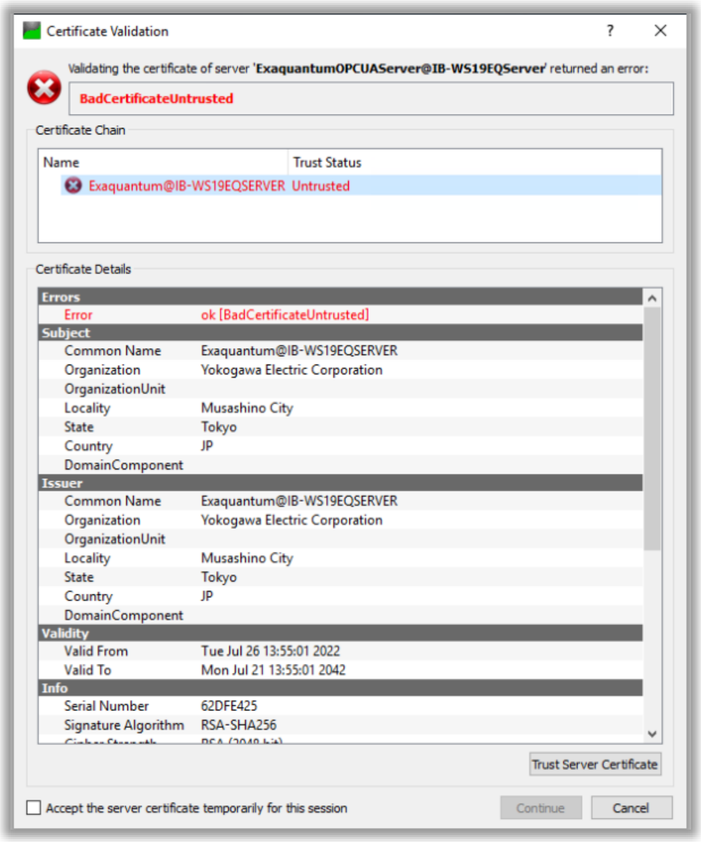


Figure 28-6 Plug Icon showing connection status

Note: that the first time that a connection to a Server is tried, the “Certificate Validation” window will appear. This is done as part of the Application authentication. Click on “Trust Server Certificate” button at the bottom and then “Continue”:


Certificate Validation

Validating the certificate of server '**ExaquantumOPCUAServer@IB-WS19EQServer**' returned an error:

BadCertificateUntrusted

Certificate Chain

Name	Trust Status
Exaquantum@IB-WS19EQSERVER	Untrusted

Certificate Details

Errors	
Error	ok [BadCertificateUntrusted]
Subject	
Common Name	Exaquantum@IB-WS19EQSERVER
Organization	Yokogawa Electric Corporation
OrganizationUnit	
Locality	Musashino City
State	Tokyo
Country	JP
DomainComponent	
Issuer	
Common Name	Exaquantum@IB-WS19EQSERVER
Organization	Yokogawa Electric Corporation
OrganizationUnit	
Locality	Musashino City
State	Tokyo
Country	JP
DomainComponent	
Validity	
Valid From	Tue Jul 26 13:55:01 2022
Valid To	Mon Jul 21 13:55:01 2042
Info	
Serial Number	62DFE425
Signature Algorithm	RSA-SHA256

Figure 28-7 “Certificate Validation” Window

5. If the User authentication is successful, then the plug icon will appear as shown below to show the connection is done:

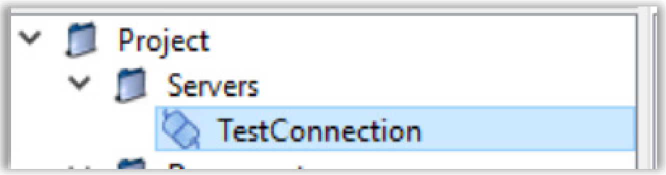


Figure 28-8 Plug Icon Connected

28.3.3. How to read Live Data

UaExpert's Data Access View can be used to display live data from one or more Exaquantum tags.

In order to use Data Access View, the next steps need to be followed:

1. Click on "Add Document" icon in the Document Toolbar:

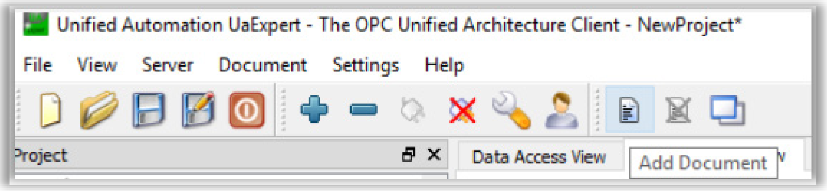
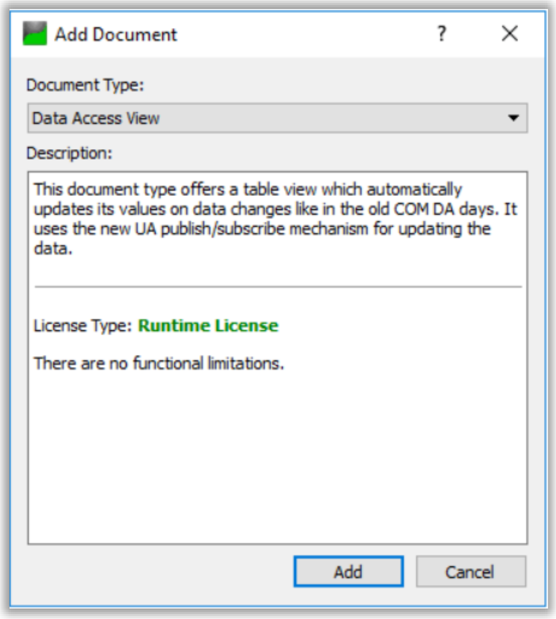

Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File
View
Server
Document
Settings
Help
Project
Data Access View
Add Document

Figure 28-9 "Add Document" Icon

2. Select "Data Access View" as "Document Type", and press "Add" button:



Document Type: Data Access View

Description:

This document type offers a table view which automatically updates its values on data changes like in the old COM DA days. It uses the new UA publish/subscribe mechanism for updating the data.

License Type: Runtime License

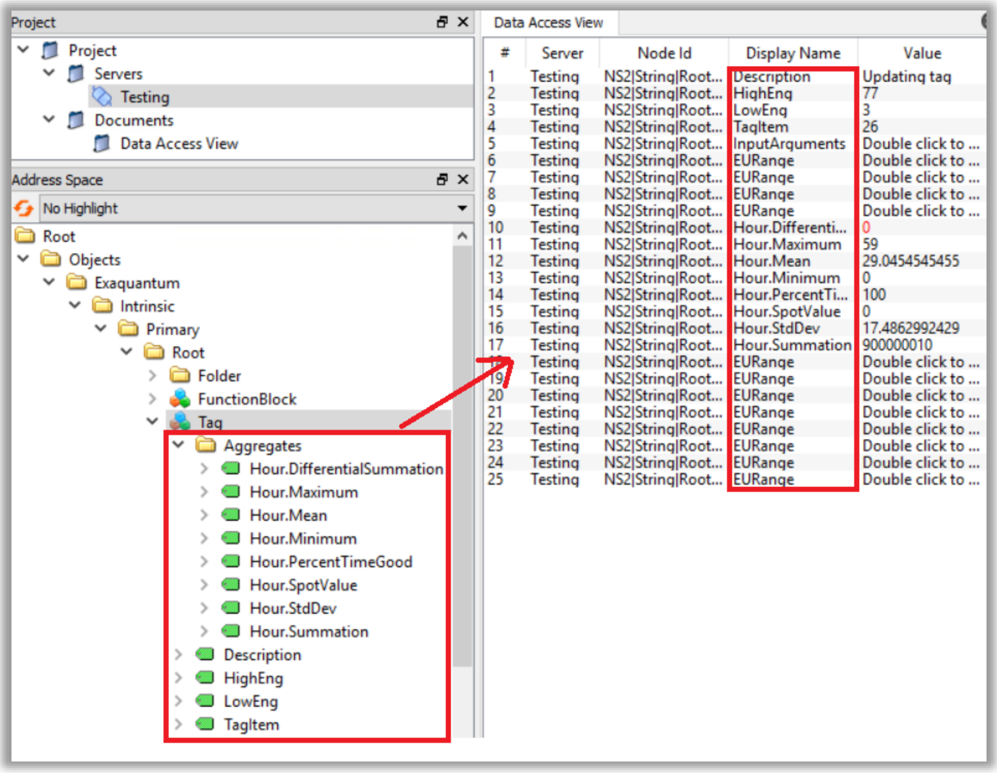
There are no functional limitations.

Figure 28-10 "Add Document" Window

Now, go to the "Address Space Window" (left side of the UaExpert's main window) and, (assuming the use of an Intrinsic namespace), browse via the "Intrinsic" folder down to "Root", and follow next steps:

1. Drag over the node which live data is going to be shown, from the "Address Space Window" into the "Data Access View" window.

2. The value of the node that was dragged into the "Data Access View", and the values of all its children, will be displayed:



#	Server	Node Id	Display Name	Value
1	Testing	NS2|String|Root...	Description	Updating tag
2	Testing	NS2|String|Root...	HighEng	77
3	Testing	NS2|String|Root...	LowEng	3
4	Testing	NS2|String|Root...	TagItem	26
5	Testing	NS2|String|Root...	InputArguments	Double click to ...
6	Testing	NS2|String|Root...	EURange	Double click to ...
7	Testing	NS2|String|Root...	EURange	Double click to ...
8	Testing	NS2|String|Root...	EURange	Double click to ...
9	Testing	NS2|String|Root...	EURange	Double click to ...
10	Testing	NS2|String|Root...	Hour.Differenti...	0
11	Testing	NS2|String|Root...	Hour.Maximum	59
12	Testing	NS2|String|Root...	Hour.Mean	29.0454545455
13	Testing	NS2|String|Root...	Hour.Minimum	0
14	Testing	NS2|String|Root...	Hour.PercentTi...	100
15	Testing	NS2|String|Root...	Hour.SpotValue	0
16	Testing	NS2|String|Root...	Hour.StdDev	17.4862992429
17	Testing	NS2|String|Root...	Hour.Summation	900000010
18	Testing	NS2|String|Root...	EURange	Double click to ...
19	Testing	NS2|String|Root...	EURange	Double click to ...
20	Testing	NS2|String|Root...	EURange	Double click to ...
21	Testing	NS2|String|Root...	EURange	Double click to ...
22	Testing	NS2|String|Root...	EURange	Double click to ...
23	Testing	NS2|String|Root...	EURange	Double click to ...
24	Testing	NS2|String|Root...	EURange	Double click to ...
25	Testing	NS2|String|Root...	EURange	Double click to ...

Figure 28-11 "Data Access View" Window

28.3.4. How to Read Historical Data

UaExpert's History Trend View can be used to display historical data from one or more Exaquantum tags into a graphical trend view. Note that UaExpert does not display Exaquantum's processed data (ad-hoc aggregations).

In order to use History Trend View, next steps need to be followed:

1. Click on "Add Document" icon in the Document Toolbar:

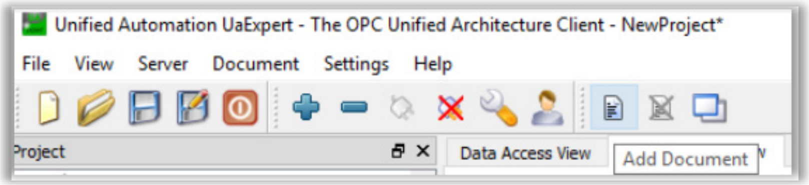
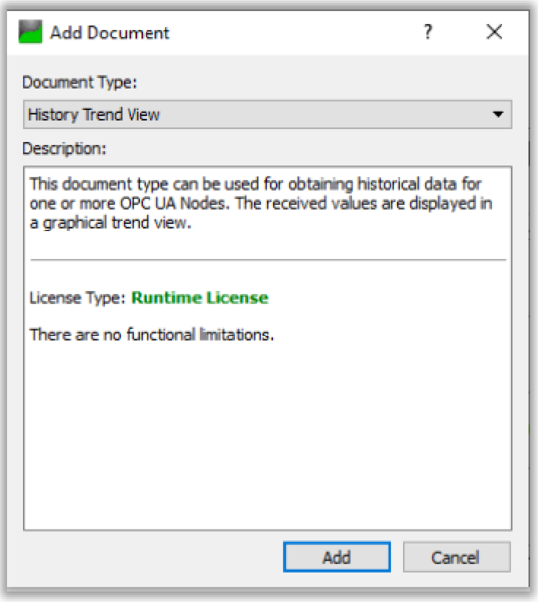

Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File
View
Server
Document
Settings
Help
Project
Data Access View
Add Document

Figure 28-12 "Add Document" Icon

2. Select “History Trend View” as “Document Type”, and press “Add” button:



Add Document

Document Type: History Trend View

Description:

This document type can be used for obtaining historical data for one or more OPC UA Nodes. The received values are displayed in a graphical trend view.

License Type: Runtime License

There are no functional limitations.

Figure 28-13 “Add Document” Window

Now, go to the “Address Space Window” (left side of the UaExpert’s main window) and, (assuming the use of an Intrinsic namespace), browse via the “Intrinsic” folder down to “Root”, and follow next steps:

1. Drag over the “TagItem” node which history data is going to be shown, from the “Address Space Window” into the “Configuration” window of the “History Trend View”:

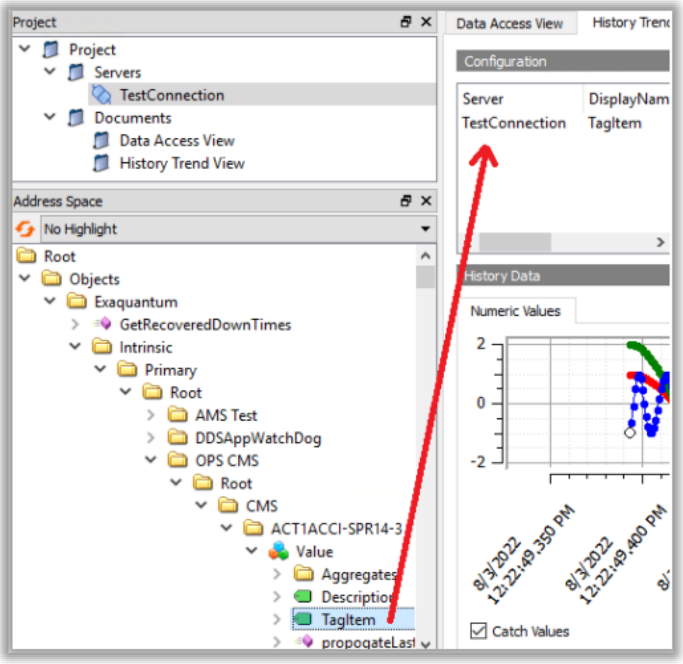
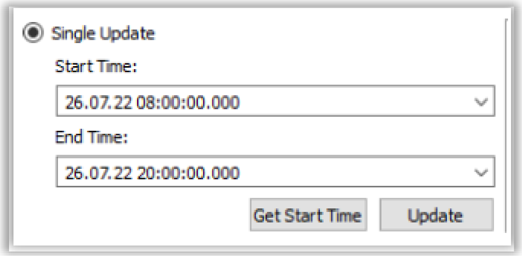


Figure 28-14 “Configuration” Window

2. Establish the time range to display:



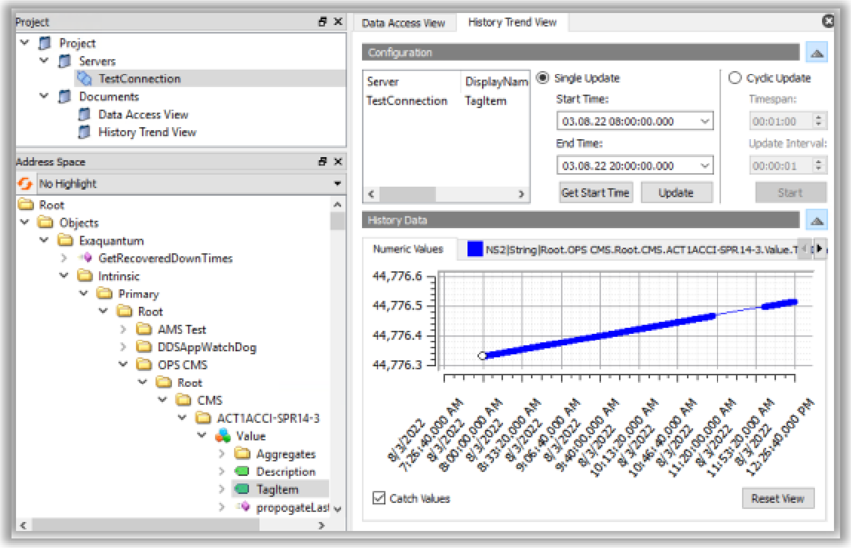
[✓ Single Update]

Start Time: 26.07.22 08:00:00.000

End Time: 26.07.22 20:00:00.000

Figure 28-15 Times setup

3. Press the “Update” button to obtain the updated graphic:



Server	DisplayNam
TestConnection	TagItem

[✓ Single Update]

Start Time: 03.08.22 08:00:00.000

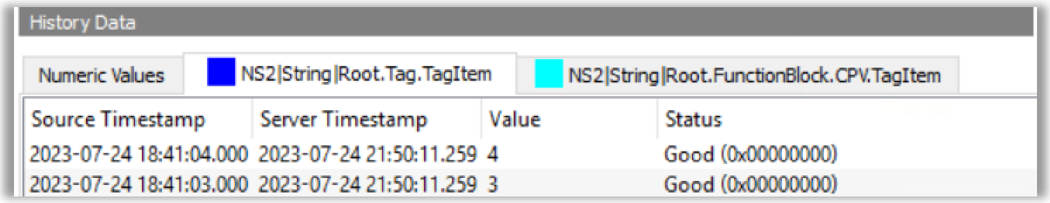
End Time: 03.08.22 20:00:00.000

Timespan: 00:01:00

Update Interval: 00:00:01

Figure 28-16 Graphic with the selected tag values

4. Have separate numbered headings for graphical view and tabular view:

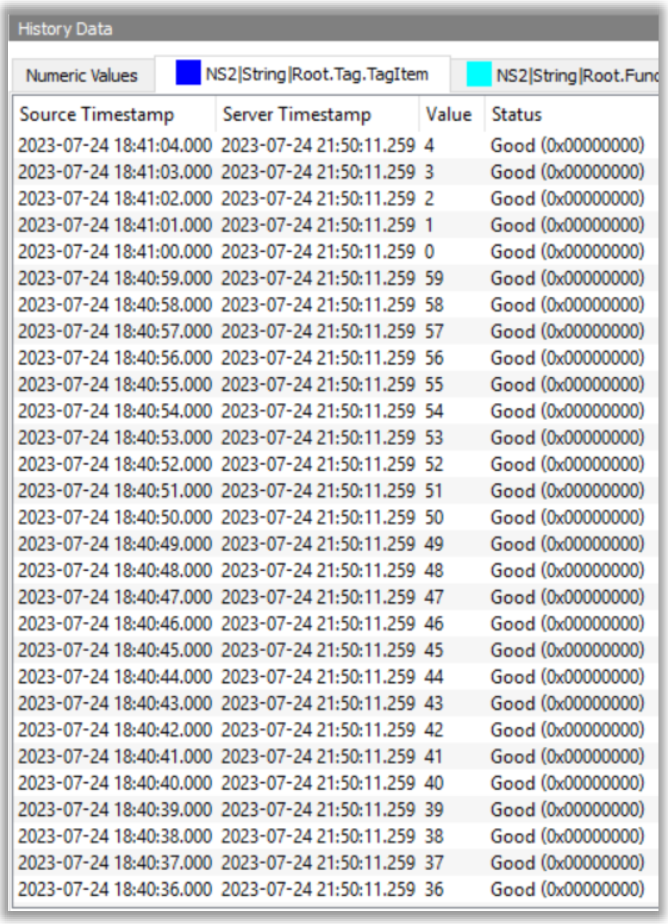


History Data

Numeric Values | 🟦 NS2|String|Root.Tag.TagItem | 🟦 NS2|String|Root.FunctionBlock.CPV.TagItem

Source Timestamp	Server Timestamp	Value	Status
2023-07-24 18:41:04.000	2023-07-24 21:50:11.259	4	Good (0x00000000)
2023-07-24 18:41:03.000	2023-07-24 21:50:11.259	3	Good (0x00000000)

Figure 28-17 Selecting the tabular view of a tag


History Data

Numeric Values | 🟦 NS2|String|Root.Tag.TagItem | 🟦 NS2|String|Root.Func

Source Timestamp	Server Timestamp	Value	Status
2023-07-24 18:41:04.000	2023-07-24 21:50:11.259	4	Good (0x00000000)
2023-07-24 18:41:03.000	2023-07-24 21:50:11.259	3	Good (0x00000000)
2023-07-24 18:41:02.000	2023-07-24 21:50:11.259	2	Good (0x00000000)
2023-07-24 18:41:01.000	2023-07-24 21:50:11.259	1	Good (0x00000000)
2023-07-24 18:41:00.000	2023-07-24 21:50:11.259	0	Good (0x00000000)
2023-07-24 18:40:59.000	2023-07-24 21:50:11.259	59	Good (0x00000000)
2023-07-24 18:40:58.000	2023-07-24 21:50:11.259	58	Good (0x00000000)
2023-07-24 18:40:57.000	2023-07-24 21:50:11.259	57	Good (0x00000000)
2023-07-24 18:40:56.000	2023-07-24 21:50:11.259	56	Good (0x00000000)
2023-07-24 18:40:55.000	2023-07-24 21:50:11.259	55	Good (0x00000000)
2023-07-24 18:40:54.000	2023-07-24 21:50:11.259	54	Good (0x00000000)
2023-07-24 18:40:53.000	2023-07-24 21:50:11.259	53	Good (0x00000000)
2023-07-24 18:40:52.000	2023-07-24 21:50:11.259	52	Good (0x00000000)
2023-07-24 18:40:51.000	2023-07-24 21:50:11.259	51	Good (0x00000000)
2023-07-24 18:40:50.000	2023-07-24 21:50:11.259	50	Good (0x00000000)
2023-07-24 18:40:49.000	2023-07-24 21:50:11.259	49	Good (0x00000000)
2023-07-24 18:40:48.000	2023-07-24 21:50:11.259	48	Good (0x00000000)
2023-07-24 18:40:47.000	2023-07-24 21:50:11.259	47	Good (0x00000000)
2023-07-24 18:40:46.000	2023-07-24 21:50:11.259	46	Good (0x00000000)
2023-07-24 18:40:45.000	2023-07-24 21:50:11.259	45	Good (0x00000000)
2023-07-24 18:40:44.000	2023-07-24 21:50:11.259	44	Good (0x00000000)
2023-07-24 18:40:43.000	2023-07-24 21:50:11.259	43	Good (0x00000000)
2023-07-24 18:40:42.000	2023-07-24 21:50:11.259	42	Good (0x00000000)
2023-07-24 18:40:41.000	2023-07-24 21:50:11.259	41	Good (0x00000000)
2023-07-24 18:40:40.000	2023-07-24 21:50:11.259	40	Good (0x00000000)
2023-07-24 18:40:39.000	2023-07-24 21:50:11.259	39	Good (0x00000000)
2023-07-24 18:40:38.000	2023-07-24 21:50:11.259	38	Good (0x00000000)
2023-07-24 18:40:37.000	2023-07-24 21:50:11.259	37	Good (0x00000000)
2023-07-24 18:40:36.000	2023-07-24 21:50:11.259	36	Good (0x00000000)

Figure 28-18 Tabular view of a tag

Chapter 29 Reading Exaquantum data with C# using the OPC UA API

The following Unified Automation Client SDK NuGet .NET packages must be included in the project:

- UnifiedAutomation.UaBase.dll
- UnifiedAutomation.UaBase.Windows.dll
- UnifiedAutomation.UaClient.dll

A basic setup tutorial is available from the Unified Automation website: https://documentation.unified-automation.com/uasdknet/3.1.3/html/L2ClientTutEmptyProject.html

The files for the .dll packages above, are included within the 'PreBuilt' folder of the OPC UA sample code project.

This section outlines examples of how to use the COM API for the synchronous read of:

- Historical Data
- Live Data

OPC UA is capable of Live and Historical data reads, in a Synchronous manner.

This process is defined in the following steps:

Table 29-1

Step	Method Description	Section
1	Import Libraries to Project	29.1.1
2	Create the Session	29.1.2
3	Add License	29.1.3
4	Accept Untrusted/Invalid Certificate	29.1.4
5	Set Untrusted Certificate	29.1.5
6	Start Application	29.1.6
7	Set UserIdentity	29.1.7
8	Load the Client Configuration	29.1.8
9	Create NodeId Collection	29.1.9
10	Retrieve Node List to Read	29.1.10
11	Read Live Values From Nodes	29.1.11
12	Create History Value IDs to Read From	29.1.12
13	Add Nodes to Read History Data From	29.1.13
14	Read History Values From Nodes	29.1.14

29.1. Common Activities

29.1.1. Importing Libraries to Project

```
using System;
using System.Collections.Generic;
using UnifiedAutomation.UaBase;
using UnifiedAutomation.UaClient;
using UnifiedAutomation.UaSchema;
```

29.1.2. Create the Session

```
Session m_session;
public Client(ConsoleClient.Configuration.IclientConfiguration settings)
        {Settings = settings; State = ClientState.Disconnected;}}
```

29.1.3. Add License

```
ApplicationLicenseManager.AddProcessLicenses(PlatformUtils.GetAssembly(ty
peof(Program)), "License.lic");
```

29.1.4. Accept Untrusted/Invalid Certificate

```
//Method to accept untrusted/invalid certificates
static void Application_UntrustedCertificate(object sender,
UntrustedCertificateEventArgs e)
      {List<StatusCode> validationErrors = e.ValidationError.Flattened;
      if (validationErrors.Count == 1)
            {e.Accept = true;
            e.Persist = true;}
      else
            {e.AcceptAll = true;
            e.Persist = true;}}
```

29.1.5. Set Untrusted Certificate

```
{ApplicationInstanceBase.Default.UntrustedCertificate += new
UntrustedCertificateEventHandler(Application_UntrustedCertificate);}
```

29.1.6. Start Application

```
//Starts the application
ApplicationInstanceBase.Default.Start(Program.Run, client);
```

29.1.7. Set UserIdentity

```
ClientState SetUsername()
      {if (Session.UserIdentity == null)
            {//Creates new UserIdentity to set the UserName and Password
            Session.UserIdentity = new UserIdentity();}
      Session.UserIdentity.IdentityType = UserIdentityType.UserName;
      Session.UserIdentity.UserName = Settings.Connection.UserName;
      Session.UserIdentity.Password = Settings.Connection.Password;
      return ClientState.Disconnected;}
```

29.1.8. Load the Client Configuration

```
public static IclientConfiguration LoadDefaults()
      { ClientConfigurationInMemory ret = new
      ConsoleClient.Configuration.ClientConfigurationInMemory()
      {ConfiguredNamespaces = new NamespaceTable()
      {"http://ymx.org/InformationModel/"}, Connection = new ConnectionData()
      {DiscoveryUrl = "opc.tcp://[server name]:[port number]",
            UserName = "<User Account with Exaquantum Access Permissions>",
      Password = "<Password for User Account in line above>",},
            ClientSettings = new ClientSettings(),
            CreateNodeIdCollection()}}
      return ret}
```

29.1.9. Create NodeId Collection

```
      //Creates list to assign the tags that will be read from
      ReadVariableIds = new List<NodeId>()
            {VariableIds.Server_ServerStatus_CurrentTime,
             new NodeId("<Tag Path>", 1),
             new NodeId("<Tag Path>", 1),},
             RegisteredNodes = null,
           Method = new Configuration.MethodDescription()
               {ObjectId = new NodeId("Demo.Method", 1),
                MethodId = new NodeId("Demo.Method.Multiply", 1),
                InputArguments = new List<Variant>()
                   {new Variant(10.0),
                    new Variant(20.0)}}}
```

29.1.10. Retrieve Node List to Read

```
      Ilist<ReadValueId> nodesToRead = NodesToReadFromConfiguration();
      var results = Session.Read(nodesToRead);
```

29.1.11. Read Live Values from Nodes

```
//Method to retrieve nodes to read values from
Ilist<ReadValueId> NodesToReadFromConfiguration()
      {Ilist<NodeId> configuredVariables;
      //Checks that there are nodes registered
      if (RegisteredNodes != null)
            {configuredVariables = RegisteredNodes;}
      else
            {configuredVariables = Settings.ReadVariableIds;}
      //List of values to read from the nodes
      List<ReadValueId> readValueIds = new List<ReadValueId>();
      foreach (NodeId nodeId in configuredVariables)
            {readValueIds.Add(new ReadValueId()
             {AttributeId = Attributes.Value, NodeId = nodeId});}
      return readValueIds;}
```

29.1.12. Create History Value IDs to Read From

```
      m_historyReadValueIdsWithContinuationPoint = new HistoryReadValueIdCollection();
      HistoryReadValueIdCollection nodesToRead = new HistoryReadValueIdCollection();
```

29.1.13. Add Nodes to Read History Data From

```
      //Add each node to read
     foreach (NodeId node in Settings.HistoryVariableIds)
     {nodesToRead.Add(new HistoryReadValueId() { NodeId = node });
```

29.1.14. Read History Values From Nodes

```
//Set details to read
//This sample reads the last 10 Historical values from the current time
     ReadRawModifiedDetails details = new ReadRawModifiedDetails()
      {StartTime = new DateTime(2017, 1, 1, 0, 0, 0, 0, DateTimeKind.Local),
     EndTime = DateTime.UtcNow,
     NumValuesPerNode = 10,
     IsReadModified = false,
     ReturnBounds = false,};

     //Creates list of HDA read results
     List<HistoryDataReadResult> results = Session.HistoryReadRaw(
      nodesToRead,
      details,
      TimestampsToReturn.Source,
new RequestSettings() { OperationTimeout = 10000 });}
```

This page intentionally left blank

Chapter 30 Reading Exaquantum data with Python using OPC UA

This section outlines examples of how to use the COM API for reading:

- Synchronous History Data
- Synchronous Live Data

The latest, or most recent, value of an Item can be read in a one-off, synchronous way.

This process is defined in the following steps:

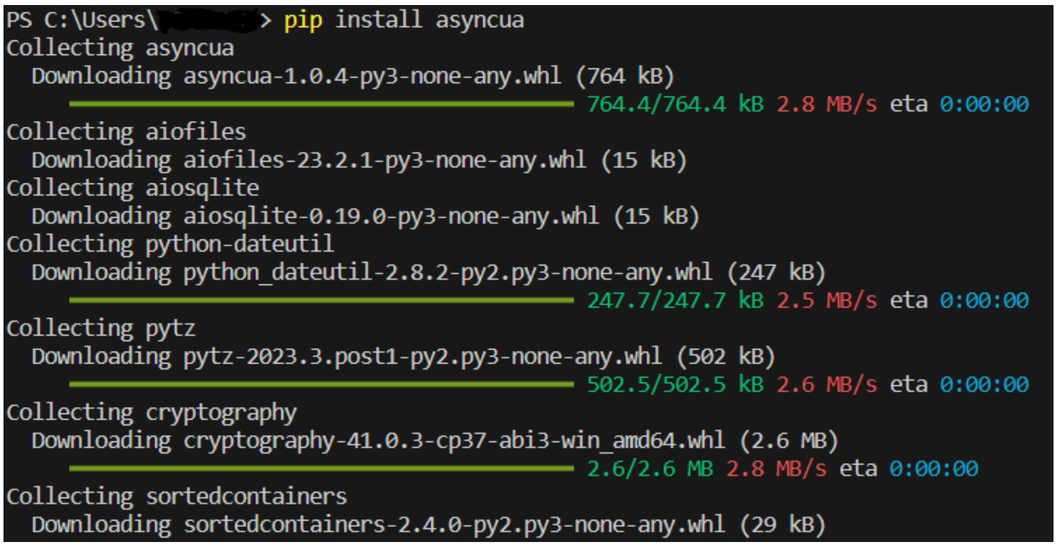
Table 30-1

Step	Section Description	Section Number
1	Import Libraries into Project	30.1.1
2	Connect to the OPC UA URL	30.1.2
3	Enter User Credentials	30.1.3
4	Obtain Namespace ID	30.1.4
5	Retrieve Node to Read Values From	30.1.5
6	Reading the Latest Item Value	30.2
7	Define Start and End Time	30.3.1
8	Read the History Values	30.3.2

The opcua-asyncio package on python3 must be installed on the Exaquantum Client. opcua-asyncio allows connection to the OPC UA server.

To install opcua-asyncio, execute ‘pip install asyncua’ in a PowerShell command prompt.

Python3 is installed via “Download Python | Python.org”



```
PS C:\Users\[redacted]> pip install asyncua
Collecting asyncua
  Downloading asyncua-1.0.4-py3-none-any.whl (764 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 764.4/764.4 kB 2.8 MB/s eta 0:00:00
Collecting aiofiles
  Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Collecting aiosqlite
  Downloading aiosqlite-0.19.0-py3-none-any.whl (15 kB)
Collecting python-dateutil
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.7/247.7 kB 2.5 MB/s eta 0:00:00
Collecting pytz
  Downloading pytz-2023.3.post1-py2.py3-none-any.whl (502 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 502.5/502.5 kB 2.6 MB/s eta 0:00:00
Collecting cryptography
  Downloading cryptography-41.0.3-cp37-abi3-win_amd64.whl (2.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 2.8 MB/s eta 0:00:00
Collecting sortedcontainers
  Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
```

Figure 30-1 Installing opcua-asyncio with PowerShell

30.1. Common Activities

30.1.1. Import Libraries into Project

```
import asyncio
from asyncua import Client, ua
//For HDA Reads
from datetime import datetime, timedelta
```

30.1.2. Connect to the OPC UA URL

```
client = Client(url="opc.tcp://<Server Name>:<Port number>")
```

30.1.3. Enter user Credentials for Server

```
client.set_user("<User Account with Exaquantum Access Permissions>")
client.set_password("<Password for User Account in line above>")
```

30.1.4. Obtain Namespace ID

```
nsidx = await client.get_namespace_index('http://ymx.org/InformationModel/')
```

30.1.5. Retrieve Node to Read Values From

```
var = client.get_node(ua.NodeId('Root.c1.TagItem', nsidx))
```

30.2. Reading the Latest Item Value

This type of read is used when a one-off request for the latest value is required.

This example reads the current Tag Value, Timestamp, and Quality.

```
data = await var.read_data_value()
```

```
        print('DATA VALUE: ',data.Value.Value, ' AT ', data.SourceTimestamp,
    'QUALITY:', data.StatusCode.value)
        return(data.Value.Value, data.SourceTimestamp, data.StatusCode.value)
```

30.3. Reading the History of Item Values

This type of read is used when a client can be suspended while the read request is being satisfied.

This example reads the values for an item from History for the last 10 minutes:

30.3.1. Define Start and End Time

```
end = datetime.now() - timedelta(minutes = 10)
start =datetime.now()
```

30.3.2. Read the History Values

```
    data = await var.read_raw_history(starttime=end, endtime=start)
    # Arrays to Store Data
    valList=[]
    timeList=[]
    qualityList = []
    for i in range(len(data)):
        itemValue = (data[i].Value.Value)
        valList.append(itemValue)
        itemTimeStamp = (data[i].SourceTimestamp)
        timeList.append(itemTimeStamp)
        itemQuality = (data[i].StatusCode.value)
        qualityList.append(itemQuality)

        print('DATA VALUE: ',itemValue, ' AT ', itemTimeStamp, 'QUALITY:',
itemQuality)
 return(valList, timeList)
```

This page intentionally left blank