CONFIDENTIAL INFORMATION

ATLAS 10 COM API MIGRATION GUIDE
ATLAS PLATFORM
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Reviewer</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>22/11/2017</td>
<td>Gonzalo Abella</td>
<td>Chris Johnson</td>
<td>Initial Draft.</td>
</tr>
<tr>
<td>1.1</td>
<td>18/12/2017</td>
<td>Matthew Bristow</td>
<td>Chris Johnson</td>
<td>Reskin, and additional Known Issues.</td>
</tr>
<tr>
<td>1.2</td>
<td>05/01/2017</td>
<td>Matthew Bristow</td>
<td>Chris Johnson</td>
<td>Merged with master branch.</td>
</tr>
<tr>
<td>1.3</td>
<td>11/05/2018</td>
<td>Matthew Bristow</td>
<td>Steven Morgan</td>
<td>Added Excel example. Added MATLAB example. Updated C# example.</td>
</tr>
<tr>
<td>1.4</td>
<td>21/05/2018</td>
<td>Steven Morgan</td>
<td>Matthew Bristow</td>
<td>Type library version 10.3 Added description of Sets, Pages, and Set Association APIs.</td>
</tr>
<tr>
<td>1.4.1</td>
<td>17/07/2018</td>
<td>Matthew Bristow</td>
<td>Steven Morgan</td>
<td>Fixed error in VBA Example Fixed spelling in C# Example</td>
</tr>
</tbody>
</table>
## CONTENTS

1. Important Notes ........................................................................................................ 3
2. Application Version .................................................................................................... 3
3. Application Licensing ............................................................................................... 3
4. OCS Software Dependencies ...................................................................................... 3
5. Prerequisites ................................................................................................................ 3
6. Introduction .................................................................................................................. 4
   6.1 Atlas 10 [COM] Automation API .......................................................................... 4
   6.2 Atlas 10 [WCF] Automation API ......................................................................... 4
7. ATLAS 10 COM ActiveX Object Model ..................................................................... 4
8. Migration ATLAS 9 to ATLAS 10 ............................................................................. 5
   8.1 How to use the ATLAS 10 COM API with Excel .................................................. 5
   8.2 How to use the ATLAS 10 COM API with Visual Studio ....................................... 6
9. Known Issues .............................................................................................................. 7
10. ATLAS 10 Specific APIs ........................................................................................... 8
    10.1 Sets, Pages, and Set Association ....................................................................... 8
11. Examples .................................................................................................................... 11
    11.1 VBA .................................................................................................................. 11
    11.2 C# Console Application .................................................................................... 12
    11.3 MATLAB .......................................................................................................... 13
1 Important Notes
This release is an update release of the ATLAS Platform featuring the Automation API.

Feedback/Support
If there are issues, please contact your Track Support Engineer for further assistance. You can also submit bugs and suggestions for future releases through the ATLAS 10 Zendesk Portal or email ATLAS 10 Support.

2 Application Version
ATLAS 10.2.18150.4

3 Application Licensing
ATLAS 10 Evaluation

4 OCS Software Dependencies
SQLRace currently supported Version 2.1.18123.1
Recommended SQLRace Database Version 1.50

5 Prerequisites
Microsoft .NET Framework 3.5
Microsoft .NET Framework 4.6.2
6 Introduction
To aid teams with their transition to Atlas 10 and SQL Race, MAT have provided a temporary Automation API (COM) that closely matches the legacy ATLAS 9 Automation API. This is not to be confused with the new ATLAS 10 Automation API. For clarity, there will be two ATLAS 10 automation APIs.

6.1 Atlas 10 [COM] Automation API
This API has a strict one season life span. After which the API will be retired. This API matches the ATLAS 9 API as closely as possible so that users only need to make minimal changes within their scripts.

6.2 Atlas 10 [WCF] Automation API
This is a new Automation API designed from the bottom up to work with Atlas 10.

7 ATLAS 10 COM ActiveX Object Model
For a detailed view of the ActiveX objects, properties, methods and events please see the accompanied Excel document titled “ATLAS 10 COM API Specification”.

Figure 1, ATLAS 10 ActiveX Object Model
- Blue boxes show the areas that have been implemented in the ATLAS 10 COM API.
- Grey boxes will not be implemented.
8 Migration ATLAS 9 to ATLAS 10

8.1 How to use the ATLAS 10 COM API with Excel

1. Add a reference to the ATLAS 10.3 Object Library.
   a. Click the Tools tab.
   b. Click References.
   c. Select Atlas 10.3 Object Library from the Available References list.
   d. Click OK.

2. Excel will automatically use the correct version for 32 or 64-bit versions.
3. Use as you did with ATLAS 9 but two objects have changed.
   a. Use the ATLAS10 object instead of ATLAS.
   b. Use ParameterDataAccessObject instead of ParameterDataAccess_Object.
8.2 How to use the ATLAS 10 COM API with Visual Studio

1. Add a reference of the ATLAS 10.3 Object Library.
   a. Expand your project.
   b. Right click on References and Select Add Reference.
   c. Click on the COM option, on the left hand side.
   d. Select and Tick the Atlas 10.3 Object Library from the list.
   e. Click OK.

2. For 32-bit compatibility make sure your project has “Prefer 32-bit” checked.

3. Use as you did with ATLAS 9 but use the ATLAS10 object instead of ATLAS.
9 Known Issues

- Layers are only accessible once a session has been loaded. Therefore, if the session cannot be loaded it is recommended to wrap Layers.Layer[] in a try/catch.
- Possible mismatches in lap times of 1 microsecond due to rounding of double types.
- When loading sessions, the layer's index is ignored. Sessions will be added in to the layers sequentially.
- Parameters.Remove does not remove in-memory channels.
- SessionLoaded is asynchronous therefore consecutive calls may fail until the session is fully loaded. Wait until OnSessionLoaded event is invoked is recommended.
- Some Parameter properties do not exist in the interface returned. To access them it is recommended to cast the returned object to the ATLAS10 object e.g. ATLAS10.Parameter.
- Some objects which represent state of the application such as Displays or Layers might not reflect manual changes. It is recommended to create new instances if changes are made to the application this way.
- OnTelemetryStarted event is invoke after processing the PGVs, therefore there is a delay between the start recording action and the event invocation.
- CloseSession uses Layer number, not Index.
10) ATLAS 10 Specific APIs

10.1 Sets, Pages, and Set Association
Note: the required type library version is 10.3 and thus references may need to be updated.

The Sets, Pages, and Set Association APIs are as follows:

10.1.1 IWorkbook5
Sets getter property
Get the sets collection (ISets)

Pages getter property
Get the pages collection (IPages)

Associate(set) method
Associate a set (ISet) to all pages and displays (pass null for no association)
Errors:
COMException: NoSetFound

10.1.2 IDisplay5
AssociatedSet getter/setter property
Get and associate a set (ISet) to a display (assign null for no association)
Errors:
COMException: NoDisplayFound, SetAssociationLocked, NoSetFound

IsLockedToCurrentAssociation getter/setter property
Get and change lock display to current set association
Errors:
COMException: NoDisplayFound

10.1.3 ISets
NewEnum getter property
Enumerator for use by foreach

[string] getter property
Get a set (ISet) by name (returns null if not found)
Errors:
NullReferenceException (when assigned null)

Index[int] getter property
Get a set (ISet) by index
Errors:
ArgumentNullException (when index is out of range)

Count getter property
Get the set (ISet) count

ActiveSet getter/setter property
Get and change the active set (ISet)
The first set defaults to active
Any methods that modify a set, e.g. LoadSession, now act on the active set
Errors:
NullReferenceException (when assigned null)
COMException: NoSetFound

Add(name) method
Adds a set with the given name and returns the set (ISet)
The active set is not changed, it must be explicitly changed via the ActiveSet property
Errors:
NullReferenceException (when passed null)
COMException: InvalidName, MaximumSetsReached (adding more than 15 sets)

Remove(name) method
Remove the set (ISet) with the given name
Errors:
NullReferenceException (when passed null)
COMException: NoSetFound, MinimumSetsReached (removing last Set)

Refresh method
Synchronises (add and remove sets) and update the properties of existing sets
If the active set has been manually removed, the active set defaults to the first set

10.1.4 ISet
Name getter/setter property
Get the set name and rename the set
Errors:
NullReferenceException (when assigned null)
COMException: InvalidName, NoSetFound

SessionCount getter property
Get the number of composite sessions loaded into this set
Errors:
COMException: NoSetFound

10.1.5 IPages
NewEnum getter property
Enumerate for use by foreach
[string] getter property
Get a page (IPage) by title (returns null if not found)
Errors:
NullReferenceException (when assigned null)

Index[int] getter property
Get a page (IPage) by index
Errors:
ArgumentException (when index is out of range)
Count getter property
Get the page (IPage) count

ActivePage getter/setter property
Get and activate (change tab of) page (IPage)
Displays are added to the active page
Errors:
NullReferenceException (when assigned null)
COMException: NoPageFound

Add(title) method
Adds a page with the given title and returns the page (IPage)
Errors:
NullReferenceException (when passed null)
COMException: InvalidTitle

Duplicate(title) method
Duplicates the page with the given title and returns the page (IPage)
Errors:
NullReferenceException (when passed null)
ArgumentException
COMException: NoPageFound

Remove(title) method
Remove the page (IPage) with the given title
Errors:
NullReferenceException (when passed null)
COMException: NoPageFound, MinimumPagesReached (removing last page)

Refresh method
Synchronises (add and remove pages) and update the properties of existing pages
10..1.6 IPage

**Title getter/setter property**
Get the page title and rename the page
Errors:
- NullReferenceException (when assigned null)
- COMException: InvalidTitle, NoPageFound

**TabColor getter/setter property**
Get and change the page tab colour
Errors:
- COMException: NoPageFound

**AssociatedSet getter/setter property**
Get and associate a set (ISet) to the page and displays (assign null for no association)
Errors:
- COMException: NoPageFound, SetAssociationLocked, NoSetFound

**IsLockedToCurrentAssociation getter/setter property**
Get and change lock page to current set association
Errors:
- COMException: NoPageFound

10..1.7 Notes

COMException of NoPageFound/NoSetFound may be returned when a page/set is deleted manually by the user (to prevent the issue ensure the Refresh method is called).

The following existing A9 APIs are not supported:

- IWorkbook2
  - ClosePage, OpenPage
- IWorkbook3
  - PageCount
- IWorkbook4
  - SavePage
11 Examples

11.1 VBA

Sub GetData()

    ' Create a new instance of the ATLAS 10 Application object
    Dim objATLAS As New ATLAS10.Application

    ' Get the current workbook
    Dim objWorkbook As ATLAS10.Workbook
    Set objWorkbook = objATLAS.Workbook

    ' Get Layer 1
    Dim objLayer As ATLAS10.Layer
    Set objLayer = objWorkbook.Layers(0)

        ' If there is a session loaded into the layer
        If objLayer.IsSessionLoaded Then

            ' Then get the session in that layer
            Dim objSession As ATLAS10.Session
            Set objSession = objLayer.Session

            ' Get the lap distance parameter
            Dim objParam As ATLAS10.Parameter
            Set objParam = objSession.Parameters("vCar:Chassis")

            ' Create a Parameter Data Access Object (PDA) for this parameter
            Dim objPDA As ATLAS10.ParameterDataAccessObject
            Set objPDA = objSession.CreateParamDataAccess(objParam)

            ' Get the fastest lap
            Dim objFastestLap As ATLAS10.Lap
            Set objFastestLap = objSession.Laps.FastestLap

            ' Position the PDA at the start of the fastest lap
            objPDA.Goto (objFastestLap.StartTime)

            ' Use mean sub sampling
            objPDA.SampleMode = SampleModeMean

            ' Get samples at 10 Hz
            objPDA.SampleTime = 10000000

            ' Determine the number of samples required at 10 Hz
            Dim nSamples As Long
            nSamples = objFastestLap.LapTime / 10000000

            ' Retrieve the data
            Dim varData As Variant
            Dim varStatus As Variant
            Call objPDA.GetNextData(nSamples, varData, varStatus)

            ' varData contains the array of data values
            ' varStatus contains the array of the data status values

        End If

End Sub
11.2 C# Console Application

```csharp
using ATLAS10;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a new instance of the ATLAS Application object
            var app = new Application();

            // Get the current workbook
            var workbook = app.Workbook;

            // Get Layer 1
            var layer = workbook.Layers[0];

            // If there is a session loaded into the layer
            if (layer.IsSessionLoaded)
            {
                // Then get the session in that layer
                var session = layer.Session;

                // Get the lap distance parameter
                var parameter = session.Parameters["vCar:Chassis"];

                // Create a Parameter Data Access
                var parameterDataAccess = session.CreateParamDataAccess(parameter);

                // Get the fastest lap
                var fastestLap = session.Laps.FastestLap;

                // Position the PDA at the start of the fastest lap
                parameterDataAccess.Goto(fastestLap.StartTime);

                // Use mean sub sampling
                parameterDataAccess.SampleMode = ESampleMode.SampleModeMean;

                // Get samples at 10 Hz
                parameterDataAccess.SampleTime = 10000000;

                // Determine the number of samples required at 10 Hz
                var nSamples = fastestLap.LapTime / 10000000;

                // Retrieve the data
                object varData;
                object varStatus;
                parameterDataAccess.GetNextData((int)nSamples,
                                                out varData,
                                                out varStatus);

                // varData contains the array of data values
                // varStatus contains the array of the data status values
            }
        }
    }
}```
function [Data, Status, Time] = A10(nLayer, strParamID, dfTime, nSamples)

% Get hold of the layer
hATLAS = actxserver('ATLAS10.Application');
hWorkbook = get(hATLAS, 'Workbook');
hLayers = get(hWorkbook, 'Layers');
hLayer = get(hLayers, 'Item', nLayer);

% Determine if there is a session in the layer
bSession = get(hLayer, 'IsSessionLoaded');
if ( bSession )
    % Get the parameter and PDA
    hSession = get(hLayer, 'Session');
    hParams = get(hSession, 'Parameters');
    hParam = get(hParams, 'Item', strParamID);
    hPDA = invoke(hSession, 'CreateParamDataAccess', hParam);
    % Get the data
    invoke(hPDA, 'Goto', dfTime)
    SampleData = invoke(hPDA,'GetNextSamplesMATLAB', nSamples);
    % Extract the data
    DataCells = SampleData(1, 1:nSamples);
    StatusCells = SampleData(2, 1:nSamples);
    TimeCells = SampleData(3, 1:nSamples);
    % Convert the cell arrays to double arrays
    Data = cat(1, DataCells{:});
    Status = cat(1, StatusCells{:});
    Time = cat(1, TimeCells{:});
    % Release the ActiveX objects
    release(hPDA);
    release(hParam);
    release(hParams);
    release(hSession);
else
    % Empty output variables
    Data = [];
    Status = [];
    Time = [];
end
release(hLayer);
release(hLayers);
release(hWorkbook);
release(hATLAS);