ODTK Plugin Programming Guide

ODTK has the capability to call user defined code at the following plugin points:

1. **Scenario.Measurements.Files -** Custom readers associated with tracking data files.
2. **Satellite.ForceModel.Plugin -** Additional force modeling for the HPOP propagator.
3. **Satellite.ForceModel.SolarPressure.Model=ReflectionPlugin -** Define custom states for LightReflection modeling (SolarPressure, Albedo, Thermal radiation, etc…)
4. **Satellite.ForceModel.Drag.Model=DragModelPlugin -** Define custom states for drag modeling (CD, CL, etc…)
5. **GPSReceiver.MeasurementProcessing.SatelliteSelection.Method=Plugin -** GPS Satellite Selection algorithm specific to the GPS receiver being modeled by ODTK.
6. **{Simulator,Filter}.Events -** Custom scripts can be invoked for OnStart, OnHalt, On-Complete, OnInternalError, OnNoMoreMeas events.
7. **Satellite.FilterEvents -** Custom scripts can be called when an incoming measurement triggers MeasurementRejectThreshold or MeasurementAcceptTimer events.

All interfaces are implemented using COM and plugins can be programmed using a variety of scripting and compiled languages that support COM.

ODTK comes with plugin point examples written in **C++, C#, VBScript, JScript,** and **Perl.**

Examples shipped with ODTK are found at **C:\Program Files\AGI\ODTK 6\CodeSamples**. To test run the plugins, load a solution file into VS.NET and select "Build All". Then start ODTK and load the file TestScenario.sco. Running simulator will call the plugins and you can see their output in the message viewer.

In addition to this guide refer to the help section "Integrating with ODTK: ODTK Plugins" for details on the plugin interfaces, methods, and properties.

1 Plugin Registration

After a plugin is written in the language of your choice it must be registered:

1. Register plugins into the Microsoft Windows registry by using the Windows commands **regasm** or **regsvr32** (see help: ODTK Plugins: Plugin Components)
2. To register plugins with ODTK:
    a. Add Measurement Providers to the *Tools >> Options >> Plugin* menu and associate them with the file name extension of your tracking data files.

b. Register HPOP force model, SRP Light Reflection, and GPS Satellite Selection plugins by placing an XML registration file, similar to the one below, into either of the following folders:

C:\Program Files\AGI\ODTK 6\Plugins or

C:\Documents and Settings\<user>\My Documents\ODTK 6\Plugins

```
<?xml version = "1.0"?>
<AGIRegistry version = "1.0">
 <CategoryRegistry>

  <Category Name = "LightReflection">
    <Plugin DisplayName="SRP.Spherical.Perl"
                ProgID="AGI.SRP.LightReflection.Spherical.Perl.Example"/>
    <Plugin DisplayName="SRP.Spherical.CSharp"
                ProgID="AGI.SRP.LightReflection.Spherical.CSharp.Example"/>
    <Plugin DisplayName="SRP.NPlate.2P.Perl"

ProgID="AGI.SRP.LightReflection.NPlate.2Parameter.Perl.Example"/>
  </Category>

  <Category Name = "HPOP Plugins">
    <Plugin DisplayName="TDRS.SRP.Perl"
ProgID="AGI.HPOP.ForceModels.TDRS.SRP.Perl.Example"/>
    <Plugin DisplayName="TDRS.SRP.CSharp"
ProgID="AGI.HPOP.ForceModels.TDRS.SRP.CSharp.Example "/>
    <Plugin DisplayName="TDRS.SRP.VB.NET"
ProgID="AGI.HPOP.ForceModels.TDRS.SRP.VB.NET.Example "/>
  </Category>

  <Category Name = "GPSSatSelect">
    <Plugin DisplayName="CPP.SimplePDOP"
ProgID="AGI.GPS.SatSelect.Simple.PDOP"/>
    <Plugin DisplayName="CSharp.Example"
ProgID="AGI.GPS.SatSelect.CSharp.Example"/>
    <Plugin DisplayName="JScript.Example"
ProgID="AGI.GPS.SatSelect.JScript.Example"/>
    <Plugin DisplayName="VBScript.Example"
ProgID="AGI.GPS.SatSelect.VBScript.Example"/>
  </Category>

  <Category Name = "DragModel">
    <Plugin DisplayName = "Drag.Spherical.Perl"   ProgID =
"AGI.Drag.Spherical.Perl.Example"/>
    <Plugin DisplayName = "Drag.NPlate_Cube.Perl" ProgID =
"AGI.Drag.NPlate_Cube.Perl.Example"/>
    <Plugin DisplayName = "Drag.Lift.Perl"       ProgID =
"AGI.Drag.Lift.Perl.Example"/>
  </Category>
 </CategoryRegistry>
</AGIRegistry>
```

2 Important environment settings

ODTK interacts with plugins using COM interfaces defined in several DLLs delivered with the application. The following steps will point your Visual Studio 2005 project to those DLLs:

For C# and VB.NET: go to Project >> Add Reference… >> COM >> Browse… >> select:

C:\Program Files\AGI\ODTK 6\bin {AgAttrAutomation.dll, AgUtPlugin.dll, AgAsGPS.dll}

For C++ projects add the path to the ODTK bin directory to the following places:

1. Project >> Properties >> C/C++ >> General >> “Additional Include Directories"

2. Project >> Properties >> MIDL >> General >> “Additional Include Directories"

Make sure .NET projects have Configuration Properties >> Register for COM Interop >> True and C++ projects perform COM registration in Build Events >> Post-Build Event >> regsvr32.

3 C++ Measurement Data Provider

The following steps will take you through the creation of the plugin in Visual Studio 2005

1. File >> New Project >> Visual C++ Projects >> ATL Project >> Name: **MyMeasProvider**

    In ATL Project Wizard >> Application Settings >> Server Type: Dynamic-link Library

2. Project >> Add Class >> ATL Simple Object

    In ATL Simple Object Wizard >> Names >> Short Name: **SimpleObsReader**

3. View >> Class View >> Right-Click **CSimpleObsReader** >> Add >> Implement Interface

    In Implement Interface Wizard choose:

    - Implement Interface from: File: AgMach10.dll
    - Available type libraries: IAgProvideTrackingDataLib
    - Interface: IAgProvideTrackingData

4. Insert code snippets listed below into the .h and .cpp files generated by Visual Studio wizard.

4. Build >> Build Solution. Build will automatically register MyMeasProvider.dll COM objects.

5. In ODTK go to Tools >> Options >> Plugins >> Add: **MyMeasProvider.SimpleObsReader** and associate it with the extension myObs. Using any .myObs file as input will invoke the reader.

Insert this code into SimpleObsReader.h CSimpleObsReader class definition:

```
        double                  m_Min;
        std::string             m_FileName;
        IAgODGenericObsPtr      m_pObs;
        IAgODObsSetPtr          m_pObsSet;

        STDMETHOD(Reset)();

        STDMETHOD(OpenFile)(BSTR name, VARIANT_BOOL newFile);

        STDMETHOD(GetObsSet)(IAgODObsSetCollection * pColl, long * pNum);

        STDMETHOD(get_FileName)(BSTR * pVal)
        {
                *pVal = CComBSTR(m_FileName.c_str()).Detach();
                return S_OK;
        }
        STDMETHOD(get_SupportsSave)(VARIANT_BOOL * pVal)
        {
                *pVal = VARIANT_FALSE;
                return S_OK;
        }
```

Insert this code into SimpleObsReader.cpp:

```
STDMETHODIMP CSimpleObsReader::OpenFile(BSTR name, VARIANT_BOOL newFile)
{
        USES_CONVERSION;

        // Open data file, initialize counters, etc...
        m_Min = 0;
        m_FileName = W2A(name);

        if( /* file open error */ 0 )
        {
                return E_FAIL;
        }

        // Create objects to communicate with ODTK
        m_pObsSet.CreateInstance(CLSID_CAgODObsSet);
        m_pObs.CreateInstance(CLSID_CAgODGenericObs);

        return S_OK;
}
```

```
STDMETHODIMP CSimpleObsReader::Reset()
{
        // Rewind data file, release memory, etc...

        m_Min = 0;

        return S_OK;
}

STDMETHODIMP CSimpleObsReader::GetObsSet(
/*[in]*/         IAgODObsSetCollection *pObsSetColl,
/*[out,retval]*/ LONG *pNumObsSet )
{
        long jdn;
        double mam;
        IAgPropQtyPtr  pQty;
        IAgPropDatePtr pDate;

        (*pNumObsSet)=0;

        if( m_Min >= 300 ) return S_OK; // Only have 5 hours of "data"

        // Each ObsSet contains several measurements at the same time

        m_pObsSet->Clear();
        m_pObsSet->get_Date(&pDate);

        pDate->put_Unit(CComBSTR("UTCG"));
        pDate->put_Value(CComBSTR("1 Jan 2006 08:00:00"));

        m_pObsSet->get_JulianDay(&jdn);
        m_pObsSet->get_MinAfterMidnight(&mam);
        m_pObsSet->put_MinAfterMidnight(mam + m_Min);

        m_pObs->put_JulianDay(jdn);
        m_pObs->put_MinAfterMidnight(mam + m_Min);
        m_pObs->get_Value(&pQty);

        // Setting MeasType will set correct Dimension and internal Unit

        m_pObs->put_MeasureType(eMTRange);
        pQty->put_Unit(CComBSTR("km"));
        pQty->put_Value(1000.0 + m_Min);
        m_pObsSet->Add(m_pObs);

        m_pObs->put_MeasureType(eMTAzimuth);
        pQty->put_Unit(CComBSTR("deg"));
        pQty->put_Value(30.0 + m_Min);
        m_pObsSet->Add(m_pObs);

        m_pObs->put_MeasureType(eMTElevation);
        pQty->put_Unit(CComBSTR("deg"));
        pQty->put_Value(30.0 + m_Min/100.0);
        m_pObsSet->Add(m_pObs);

        pObsSetColl->Add(m_pObsSet); // Push ObsSet into ODTK

        (*pNumObsSet)++;

        m_Min += 1.0;                // Advance timer ...
```

```
        return S_OK;
}
```

4 C++ Plugin for GPS SatSelection

The following steps will take you through creation of the GPS SatSelection plugin in VS 2005

1. File >> New Project >> Visual C++ Projects >> ATL Project >> Name: **MyGPSSatSelect**

    In ATL Project Wizard >> Application Settings >> Server Type: Dynamic-link Library

2. Project >> Add Class >> ATL Simple Object

    In ATL Simple Object Wizard >> Names >> Short Name: **FancyGPSReceiver**

3. View >> Class View >> Right-Click **CFancyGPSReceiver** >> Add >> Implement Interface

    C:\Program Files\AGI\ODTK 6\bin\AgUtPlugin.dll > IAgUtPluginConfig

        C:\Program Files\AGI\ODTK 6\bin\AgAsGPS.dll > IAgAsGPSSatSelect-PluginEngine

4. Add the following code to the stdafx.h file (right after #pragma once)

```
#include <atlbase.h>
#include <atlcom.h>
#import "AgAttrAutomation.dll" raw_interfaces_only, no_namespace
```

5. Write some code for `CFancyGPSReceiver::Init(), Evaluate(), and Free()` using **C:\Program Files\AGI\ODTK 6\CodeSamples** as a reference or replace `return E_NOTIMPL;` with `return S_OK;`.

6. Build >> Build Solution. Build will automatically register MyGPSSatSelect.dll COM objects.

7. Create My Documents\ODTK 6\Plugins\MyFancySatSelect.xml with the following content:

```
<?xml version = "1.0"?>
<AGIRegistry version = "1.0">
  <CategoryRegistry>

     <Category Name = "GPSSatSelect">
       <Plugin DisplayName = "MyFancyGPSReceiver" ProgID =
"MyGPSSatSelect.FancyGPSReceiver"/>
     </Category>

  </CategoryRegistry>
</CategoryRegistry>
```

8. Start ODTK and create a new Scenario with a GPS Constellation, a Simulator, and a Satellite with a GPS Receiver subobject. Change GPSReceiver.MeasurementProcessing.SatelliteSelection.Method to Plugin and pick “MyFancyGPSReceiver” from the list. Run Simulator to see your plugin being called.

5 VBScript, JScript and Perl Plugins

A good method for getting started is to grab one of the examples that came with ODTK from **C:\Program Files\AGI\ODTK 6\CodeSamples\Extend\ODTK** and customize it with your own code.

6 SRP Light Reflection and Drag Model Plugins

The Light Reflection Plugin for SRP modeling and the Drag Model Plugin for drag modeling provide two major advantages over the HPOP force model plugin:

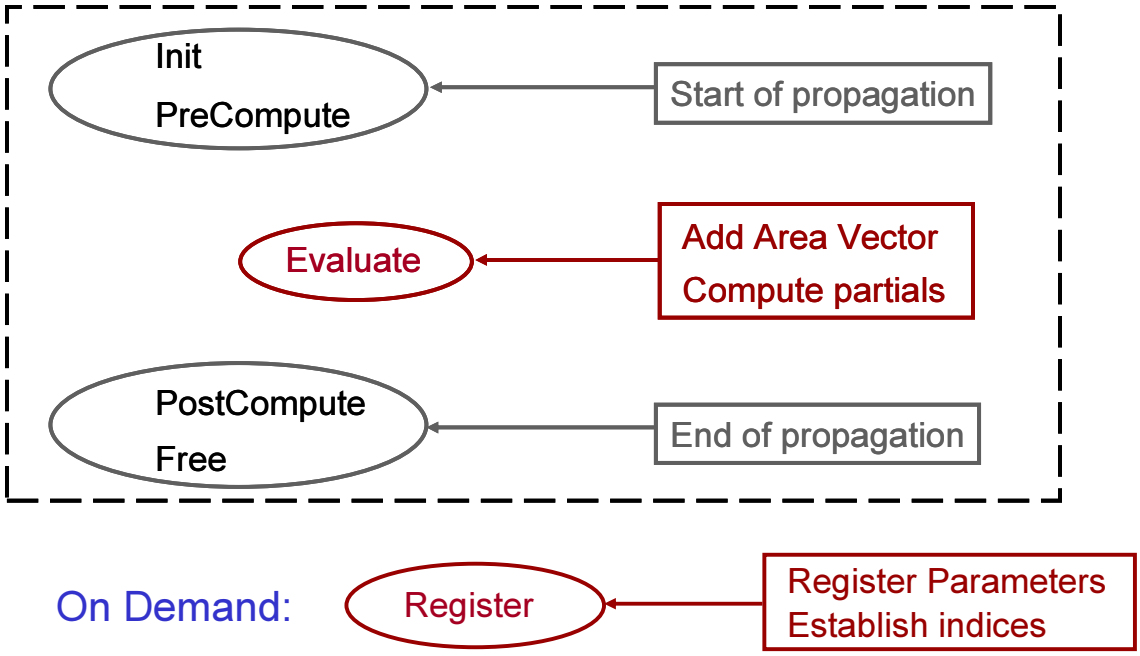
1. The user can define and estimate model parameters (where the model parameters are modeled as Gauss Markov parameters).

2. The user can model the light reflection or drag in the body frame and delegate ODTK to do the conversion to the inertial (or fixed, or RIC, or NTC) frame.

The Light Reflection interface methods and properties are defined in the AgAsHpopPlugin project. The IAgAsLightReflectionPlugin interface object defines interfaces that the plugin component must satisfy. Control parameters may be defined using the AgAttrAutomation project.

The Drag Model interface method and properties are also defined in the AgAsHpopPlugin project. The IAgAsDragModelPlugin interface object defines interfaces that the plugin component must satisfy. Control parameters may be defined using the AgAttrAutomation project.

An overview of the drag and light reflection model plugin components:



The required output of the plugin component is:

1. Area vector in satellite (body | inertial | fixed | RIC | NTC) frame.
2. Partials of area vector with respect to model parameters.
3. Partials of area vector with respect to satellite position and velocity.

6.1 SRP Light Reflection Plugin (Perl Example)

1. Start in directory <InstallDir>\CodeSamples\Extend\ODTK\SRP.LightReflection\WSC.
2. Copy SRP.Reflection.Spherical.Perl.pl to SRP.Reflection.UniqueShape.Perl.pl
3. Copy SRP.Reflection.Spherical.Perl.wsc to SRP.Reflection.UniqueShape.Perl.wsc
4. In the .wsc file update the GUID with a new value (Tools >> Create GUID in VS 2008)
5. In the .wsc file set the progid to "`Reflection.UniqueShape.Perl`" and update the description accordingly.
6. At the end of the .wsc file enter the correct name of the file that will host the code for your plugin:
<script language="PerlScript" src="SRP.Reflection.UniqueShape.Perl.pl"/>
7. Modify the plugin component code to incorporate the new SRP model.
8. Register new plugin by running `regsvr32 SRP.Reflection.UniqueShape.Perl.wsc`

9. Create the new .xml file or add the following line to the LightReflection category of an existing .xml file:

```
<Plugin DisplayName = "SRP.Reflection.W84" ProgID = "Reflec-
tion.UniqueShape.Perl"/>
```

10. Launch ODTK and set Satellite.ForceModel.SolarPressure.Model to ReflectionPlugin and pick “SRP.Reflection.W84” from the dropdown list of plugins.

6.2 Drag Model Plugin (Perl Example)

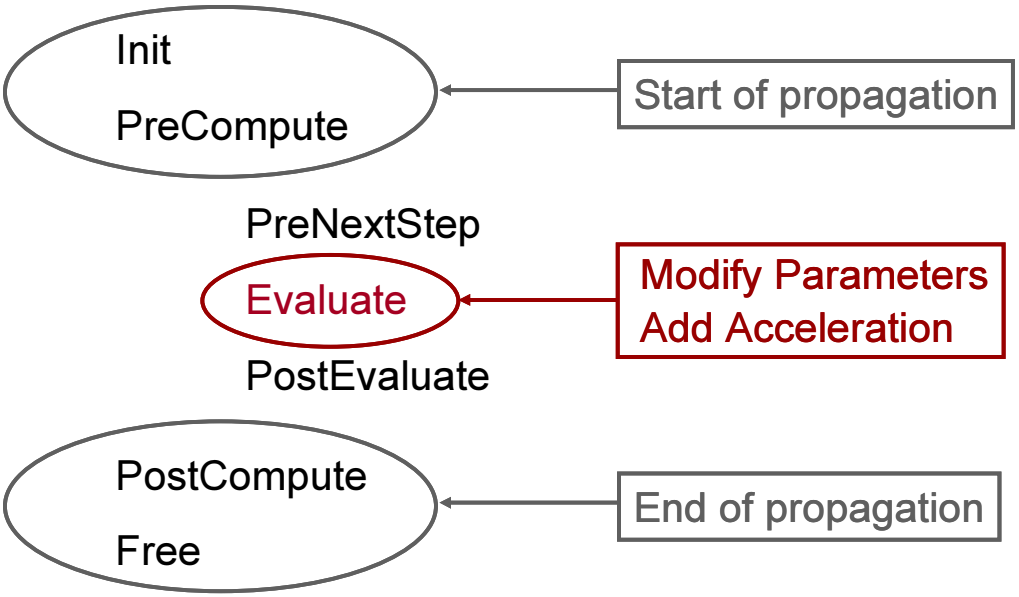
1. Use the same technique as in the SRP LightReflection example above.
2. Drag model files are in <InstallDir>\CodeSamples\Extend\ODTK\DragModels\WSC.
3. The xml plugin category is DragModel
4. The satellite force model plugin point is Satellite.ForceModel.Drag.Model.

7 HPOP Force Model Plugin

The HPOP Force Model Plugin allows the user to define a new custom acceleration model, or to modify the current acceleration computations for the existing SRP or Drag acceleration models (with the restriction that the user is constrained by the current model as to which parameters can be estimated).

The HPOP force model interface method and properties are defined in the AgAsHpopPlugin project. The IAgAsHPOPPlugin interface object defines interfaces that the plugin component must satisfy. Control parameters may be defined using the AgAttrAutomation project.

An overview of the HPOP force model plugin component is as follows, where the required plugin component output is the computed inertial acceleration vector.



7.1 HPOP Force Model Plugin (Perl Example)

1. Use the same technique as in the SRP LightReflection example above.
2. The category name is “HPOP Plugins” and the plugin point is Satellite.ForceModel.Plugin

7.2 Custom Scripts for Event Handling

Event handlers can be implemented in VBScript, Jscript, and Perl. The file name extension is determined by the scripting language used. The file name is determined by the function called to handle the event. Error messages and the object generating the event are passed to the called function.

For example, if `Simulator.Events.OnComplete` is set to be handled by `C:\Test\BeepOnce.js` then the Jscript code should look like this:

```
function BeepOnce(msg,simObj)
{
	var odtk = simObj.Parent.Parent.Application;
	odtk.WriteMessage("BeepOnce for " + simObj.name + " Msg was:" +
msg,"info");
}
```

More examples can be found at **C:\Program Files\AGI\ODTK 6\ODTK\AppData\Scripts**.