Translating Connect Report Styles into Data Provider Functionality

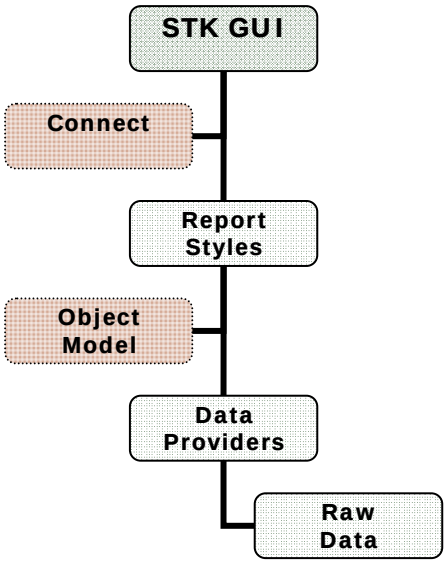
For use within the STK Object Model

Introduction

The Report related Connect commands that are traditionally used in STK Connect based applications to retrieve information about the state of an object are not directly available to users of the STK Object Model.

Instead, the Object Model provides direct access to the data provider tools that are exposed by each object in STK, which are the foundation of the report styles in Connect and the GUI.

This whitepaper outlines a technique used to translate a Report into it's corresponding set of Data Providers, and demonstrates how to programmatically retrieve the data contained within through the Object Model.



<u>Note:</u> Users are still able to utilize Connect commands within the STK Object Model by using the ExecuteCommand() method on the IAgStkObjectRoot interface.

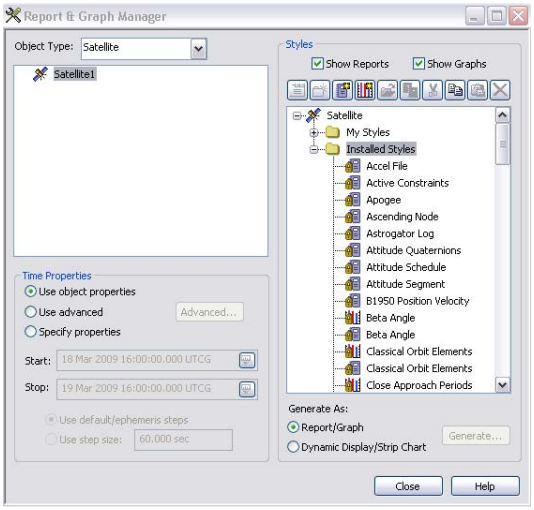
Report Styles

To browse the various Report Styles available for a particular object, and their corresponding Data Providers, right click an object in the STK *Object Browser*, navigate to the *Graph and Report Manager* menu. In the manager dialog expand the *Installed* Styles folder.

This will bring up a list of the various report styles available for the particular object.

These report styles are the same reports that are used by the Report related Connect commands to retrieve data from the underlying Data Providers.

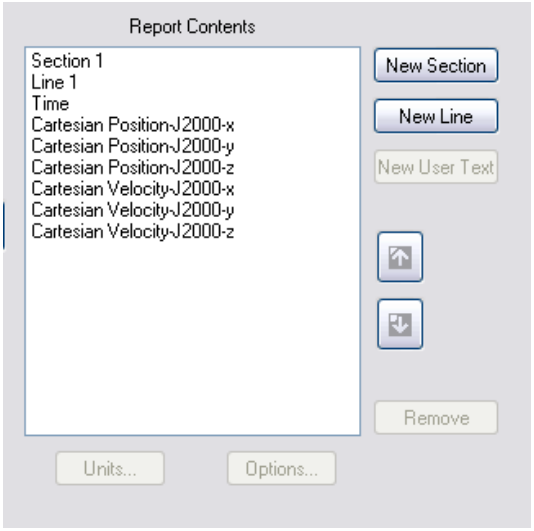
To see the Data Providers being used by a particular report style, click a report style, and click the *Properties* button to bring up the *Report Style* window.



Report Contents

In the *Report Contents* section of the *Report Style* window, we see the various Data Providers which are used to derive the particular Report. These Data Providers provide the actual data content to the report.

<u>Note:</u> The Section, Line, and Time elements are not actual data providers, and are used to provide formatting for the Report Style.

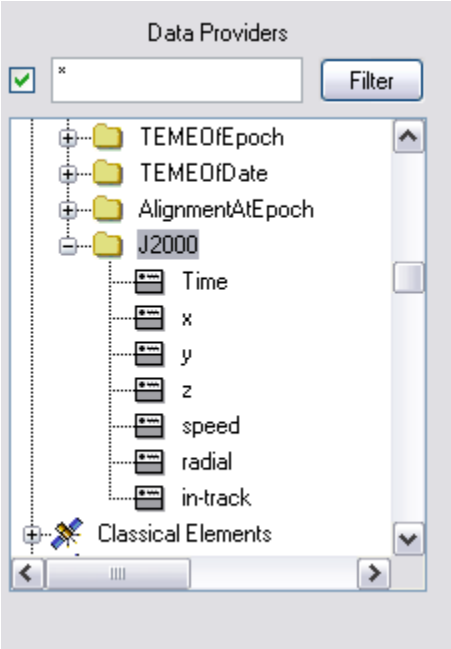


Data Providers

In the *Data Providers* section of the *Report Style* Window, we see all of the Data Providers available for a particular object.

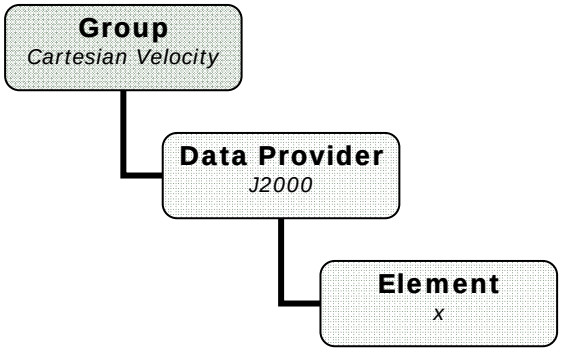
If we expand a particular *Group*, we see the nested *Data Providers* associated with that *Group*.

If we expand further by expanding a particular *Data Provider*, we see all of the data *Elements* that are associated with it.



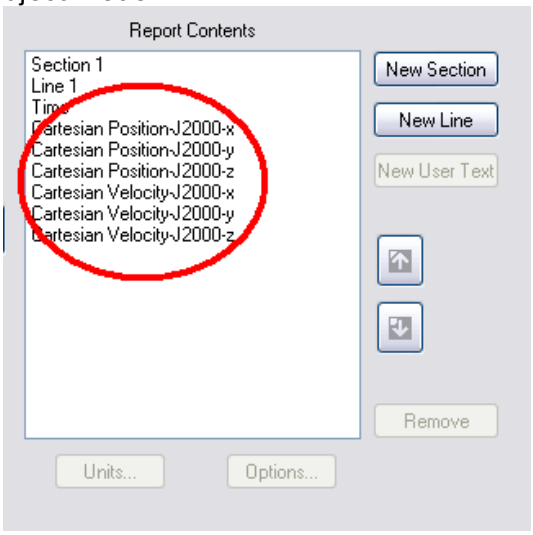
Groups, Data Providers, and *Elements* are the organizing principles of the Data Provider Functionality provided by the STK Object Model.

We will use these concepts in the sections below to describe programmatically translating Report Styles into their Data Provider equivalents in the Object Model.



Object Model

Now that we have explored the concepts of Report Styles and Data Providers, we will continue to use the example report used above, *J2000 Position Velocity*, to demonstrate retrieving it's data through the Object Model.



First, let's recall what Data Providers the report was constructed from.

In the *Report Contents* window above, we saw that the *J2000 Position Velocity* Report is made up of specific elements of the J2000 Data Provider from two Groups: *Cartesian Velocity* and *Cartesian Position*.

Getting Connected

Now let's assume we have a running instance of STK, with a scenario that contains a satellite called "Satellite1". The scenario's start date should be set to "1 Feb 2009 00:00:00.00". The scenario's stop date should be set to "2 Feb 2009, 00:00:00.00". Make sure the satellites associated times use the scenario's times as well. We issue the following code to connect to STK, and establish a reference to our Satellite.

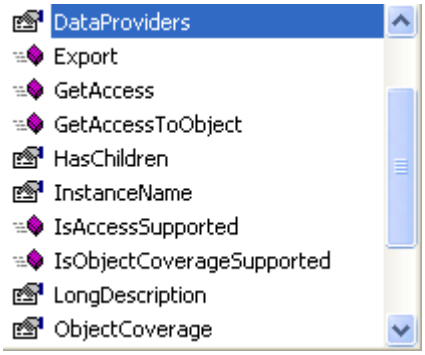
```
Dim Root As STKObjects.AgStkObjectRoot
Dim App As Object
App = GetObject(, "STK9.Application")
Root = App.Personality2

Dim Satellite As STKObjects.IAgStkObject
Satellite = Root.CurrentScenario.Children("Satellite1")
```

DataProviders Interface

We now have a reference to our Satellite, and are ready to use the Satellite's *DataProviders* Collection.

The collection contains all of the Data Providers we saw listed above in the *Data Providers* section of the *Report Style* window.

Note: We are also able to programmatically iterate through this collection, or generate an XML overview of it using the GetSchema() method of the interface.



Setting Up

To retrieve the data for the *J2000 Position Velocity* report, we need to setup its specific Data Providers for use in the Object Model. We use the various IAgDataProvider interfaces to do this:

```
Dim CartVel, CartPos As STKObjects.IAgDataProviderGroup
Dim CartVelJ2000, CartPosJ2000 As STKObjects.IAgDataProvider

CartVel = Satellite.DataProviders("Cartesian Velocity")
CartPos = Satellite.DataProviders("Cartesian Position")
CartVelJ2000 = CartVel.Group.Item("J2000")
CartPosJ2000 = CartPos.Group.Item("J2000")
```

We now have the basic interfaces set up to compute information from the Data Providers that our report is using. We are almost ready to retrieve the information, but first we need to cast our interfaces to a certain interface in order to provide the DataProvider with inputs so it can compute the proper data.

Data Provider PreData Inputs

Some Data Providers require input data before the calculation can provide data results. This data is known as "PreData".

There are two ways to ascertain if PreData is required for a particular Data Provider:

1. One can refer to the Data Provider documentation which will provide the format of the PreData if any is required.

2. Retrieve the Data Provider Schema and parse it for PreData tags.

```
Dim Satellite2 As STKObjects.IAgStkObject
Satellite2 = Root.CurrentScenario.Children("Satellite2")
 Dim schema As String = Satellite2.DataProviders.GetSchema()
```

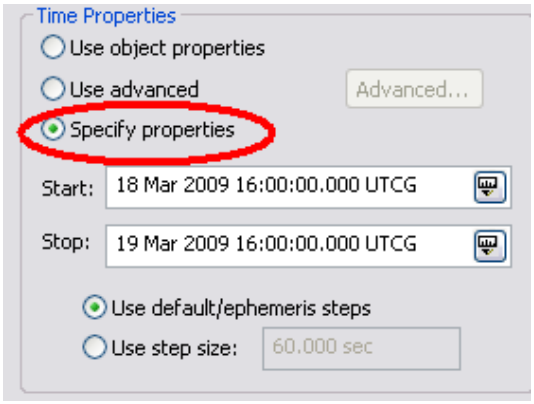
Once the format of the PreData is known one can set the PreData property on the Data Provider interface. This PreData property must be set before issuing the data provider's calculation method.

The following example demonstrates setting the Object Path as the PreData for the "RIC Coordinates" Data Provider and then calls the Data Provider's computation execution method.

```
Dim Satellite2 As STKObjects.IAgStkObject
Satellite2 = Root.CurrentScenario.Children("Satellite2")

Dim oProvider As STKObjects.IAgDataPrvTimeVar
oProvider = Satellite2.DataProviders("RIC Coordinates")

oProvider.PreData = "Satellite/Satellite1"

Dim oResult As STKObjects.IAgDrResult
oResult = oProvider.Exec(0, 90000, 1000)
```

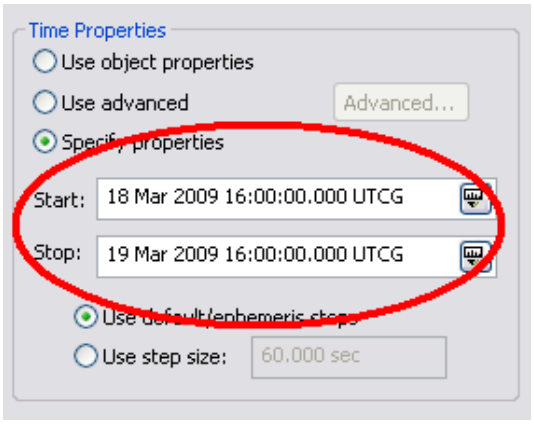
Data Provider Time Inputs

Let's look at the *Time Period* section of the *Report Window* in the GUI. Highlight *J2000 Position Velocity*, and click the *Specify properties* radio button.



Note: See the Report Styles section of this document for instructions on Navigating to this window.

The *J2000 Position Velocity* Report uses a time period to provide the underlying Data Providers information about what data to compute. We need to provide the same information to the Object Model Data Providers.



Retrieving the Data

Note: There are three different ways to compute the data, depending on the data provider type. The first method requires a time interval and step size; the second requires only a time interval; the third is independent of time. See Appendix A for samples of the different types.

We provide input information to the Data Providers by casting our Data Provider interfaces to the proper execution interface. In the case of the Cartesian Velocity and Position Data Providers, we cast to the *IAgDataPrvTimeVar* interface:

```
Dim VelTimeVariable, PosTimeVariable As STKObjects.IAgDataPrvTimeVar

VelTimeVariable = CartVelJ2000
PosTimeVariable = CartPosJ2000
```

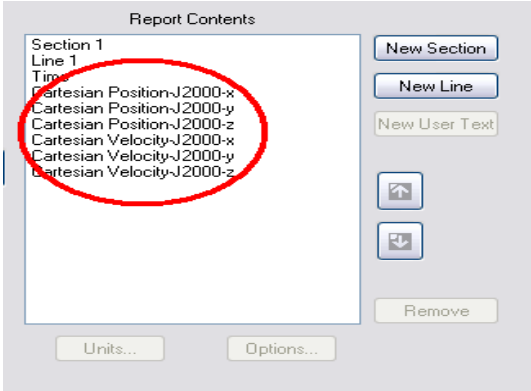
We are now ready to retrieve the information from our Data Providers. The data is always returned as an *IAgDrResult interface*. We provide the Exec method of our TimeVar interfaces with the Data Provider Inputs (Start Time, Stop Time, and Step size):

```
Dim VelResult, PosResult As STKObjects.IAgDrResult

VelResult = VelTimeVariable.Exec("18 Mar 2009 16:00:00.00", _
                                 "19 Mar 2009 16:00:00.00", _
                                 60)
PosResult = PosTimeVariable.Exec("18 Mar 2009 16:00:00.00", _
                                 "19 Mar 2009 16:00:00.00", _
                                 60)
```

VelResult and PosResult now contain the data from the *J2000 Cartesian Velocity* and *Cartesian Position* Data Providers. Still, we have gotten more data than the original Report contained.

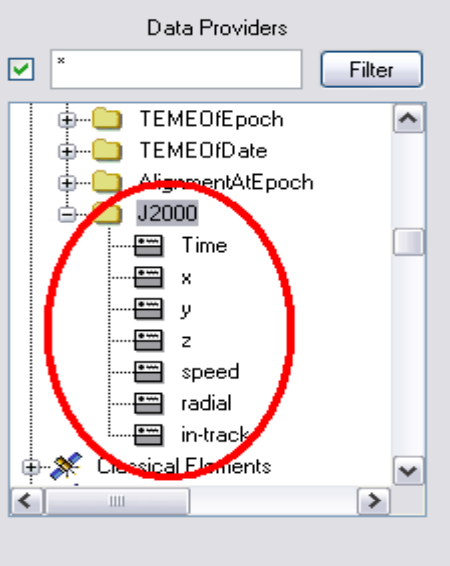
Retrieving Specific Elements

Recall that our original *Cartesian Position Velocity* Report contained only 4 elements of the Cartesian Velocity J2000 Group – *x, y, z,* and *speed*.



Similarly, the Cartesian Position J2000 Data Provider contained within our Report Style only contains 3 elements – *x, y,* and *z.*

When we executed the J2000 Data Provider of Cartesian Velocity, we actually retrieved 7 elements instead of the 4 specifically contained in the report, adding the *Time*, *radial*, and *in-track* elements to our DataProvider Result.



To be precise as possible in our translation, we want our IAgDRResult to contain only the elements which were contained in the original report. To do this, we make use of the *ExecElements()* method.

First, we specify in an array what elements we want to retrieve from the Data Provider. We then pass the array into the *ExecElements()* method:

```
Dim VelResult, PosResult As STKObjects.IAgDrResult

Dim VelElems = New Object(3) {"x", "y", "z", "speed"}
Dim PosElems = New Object(2) {"x", "y", "z"}

VelResult = VelTimeVariable.ExecElements("18 Mar 2009 16:00:00.00", _
                                         "19 Mar 2009 16:00:00.00", _
                                         60, _
                                         VelElems)
PosResult = PosTimeVariable.ExecElements("18 Mar 2009 16:00:00.00", _
                                         "19 Mar 2009 16:00:00.00", _
                                         60, _
                                         PosElems)
```

We now have the original data from the *J2000 Position Velocity* Report stored in our IAgDrResults. We are ready to traverse the data.

Traversing Result Data

If we look at what our original report looked like, the data in the report consisted of time intervals with various elements.

```
Satellite-Satellite1


      Time (UTCG)          vx (km/sec)    vy (km/sec)    vz (km/sec)    speed (km/sec)       x (km)
-----------------------    -----------    -----------    -----------    --------------    -----------
1 Feb 2009 00:00:00.000       0.000000       6.789530       3.686414          7.725760     6678.13700
1 Feb 2009 00:01:00.000      -0.535833       6.773181       3.677537          7.725760     6662.05557
1 Feb 2009 00:02:00.000      -1.069085       6.724210       3.650948          7.725760     6613.88865
1 Feb 2009 00:03:00.000      -1.597189       6.642855       3.606776          7.725760     6533.86834
1 Feb 2009 00:04:00.000      -2.117600       6.529507       3.545233          7.725760     6422.37990
1 Feb 2009 00:05:00.000      -2.627813       6.384712       3.466616          7.725760     6279.96033
1 Feb 2009 00:06:00.000      -3.125370       6.209167       3.371303          7.725760     6107.29553
```

Similarly, we need to cast our Result to the appropriate interface to make use of our data. In the case of our *J2000 Cartesian Velocity* and *Position* DataProviders, that interface is the *IAgDrIntervalCollection*. Since each data provider result shares the same result type, we can consolidate the data traversal into one method, which takes an *IAgDrResult* interface:

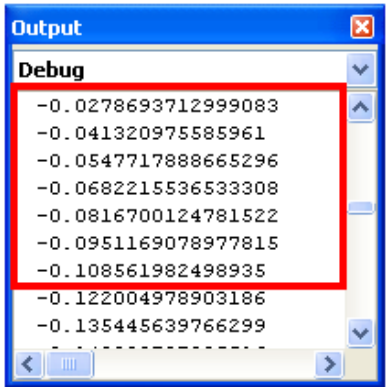
```
    Sub WriteIntervalData(ByVal Result As STKObjects.IAgDrResult)
        Dim IntervalsList As STKObjects.IAgDrIntervalCollection
        Dim Interval As STKObjects.IAgDrInterval
        Dim DataSet As STKObjects.IAgDrDataSet
        Dim Values As Object
        Dim Value As Object

        IntervalsList = Result.Intervals

        'Iterate through the Intervals
        For Each Interval In IntervalsList
            Console.WriteLine(Interval.StartTime)
            Console.WriteLine(Interval.StopTime)
            'Iterate through the DataSets stored in the Interval
            For Each DataSet In Interval.DataSets
                Console.WriteLine(DataSet.Count)
                Console.WriteLine(DataSet.ElementName)
                Console.WriteLine(DataSet.ElementType)
                Console.WriteLine(DataSet.UnitType)
                'Get the values stored in the DataSet
                Values = DataSet.GetValues()
                'Iterate through the array of values
                For Each Value In Values
                    Console.WriteLine(CStr(Value))
                Next
            Next
        Next
    End Sub
```

Note: The type of data returned by the DataProvider can be determined using the Category property of the IAgDrResult interface, which returns an enumeration describing the interface. The Value property is then cast to one of three interfaces, depending on the Category enumeration: IAgDrIntervalCollection, IAgDrSubSectionCollection, or IAgDrTextMessage. See Appendix A for samples of traversing the different types.

Finishing Up

Finally, we call our method with our *IAgDrResults*, and the data from the *J2000 Position Velocity* Report will be traversed and output*:*

```
WriteIntervalData(PosResult)
WriteIntervalData(VelResult)
```



Note: The vx column from the original report is shown in km/sec. Notice that the unit type and precision of the output data is left up to the user.

As mentioned above, it is up to the user to decide in what unit the data is returned. Issuing the following command before calling *WriteIntervalData()* will change the data that is output to be displayed in meters per second, rather then kilometers.

```
Root.UnitPreferences.SetCurrentUnit("DistanceUnit", "m")
```

Appendix A) Sample Code

Document Code (IAgDataPrvTimeVar)

```
Imports AGI
Sub Main()
    Dim Root As STKObjects.AgStkObjectRoot
    Dim App As Object
    App = GetObject(, "STK9.Application")
    Root = App.Personality2

    Dim Satellite As STKObjects.IAgStkObject
    Satellite = Root.CurrentScenario.Children("Satellite1")

    Dim CartVel, CartPos As STKObjects.IAgDataProviderGroup
    Dim CartVelJ2000, CartPosJ2000 As STKObjects.IAgDataProvider

    CartVel = Satellite.DataProviders("Cartesian Velocity")
    CartPos = Satellite.DataProviders("Cartesian Position")
    CartVelJ2000 = CartVel.Group("J2000")
    CartPosJ2000 = CartPos.Group("J2000")

    Dim VelTimeVariable, PosTimeVariable As STKObjects.IAgDataPrvTimeVar

    VelTimeVariable = CartVelJ2000
    PosTimeVariable = CartPosJ2000

    Dim VelResult, PosResult As STKObjects.IAgDrResult

    Dim VelElems = New Object(3) {"x", "y", "z", "speed"}
    Dim PosElems = New Object(2) {"x", "y", "z"}

    VelResult = VelTimeVariable.ExecElements("18 Mar 2009 16:00:00.00", _
                                             "19 Mar 2009 16:00:00.00", _
                                             60, _
                                             VelElems)
    PosResult = PosTimeVariable.ExecElements("18 Mar 2009 16:00:00.00", _
                                             "19 Mar 2009 16:00:00.00", _
                                             60, _
                                             PosElems)

    WriteIntervalData(PosResult)
    WriteIntervalData(VelResult)
End Sub
```

Computing Access (IAgDataPrvInterval)

```
Imports AGI
Sub Main()
    Dim Root As STKObjects.AgStkObjectRoot
    Dim App As Object
    App = GetObject(, "STK9.Application")
    Root = App.Personality2

    Dim Satellite As STKObjects.IAgStkObject
    Satellite = Root.CurrentScenario.Children("Satellite1")
    Dim Facility As STKObjects.IAgStkObject
    Facility = Root.CurrentScenario.Children("Facility1")

    Dim SatelliteAccess As STKObjects.IAgStkAccess
    SatelliteAccess = Satellite.GetAccessToObject(Facility)
```

```
        SatelliteAccess.ComputeAccess()

        Dim AccessProvider As STKObjects.IAgDataProvider
        AccessProvider = SatelliteAccess.DataProviders("Access Data")

        Dim AccessIntervalVar As STKObjects.IAgDataPrvInterval
        AccessIntervalVar = AccessProvider

        Dim AccessResult As STKObjects.IAgDrResult
        AccessResult = AccessIntervalVar.Exec("18 Mar 2009 16:00:00.00", _
                                              "19 Mar 2009 16:00:00.00")

        WriteIntervalData(AccessResult)
    End Sub
```

Facility Position (IAgDataPrvFixed)

```
    Imports AGI
    Sub Main()
        Dim Root As STKObjects.AgStkObjectRoot
        Dim App As Object
        App = GetObject(, "STK9.Application")
        Root = App.Personality2

        Dim Facility As STKObjects.IAgStkObject
        Facility = Root.CurrentScenario.Children(“Facility1”)

        Dim FacCartPos As STKObjects.IAgDataProvider
        FacCartPos = Facility.DataProviders("Cartesian Position")

        Dim PosFixedVar As STKObjects.IAgDataPrvFixed
        PosFixedVar = FacCartPos

        Dim FacPosResult As STKObjects.IAgDrResult
        FacPosResult = PosFixedVar.Exec()

        WriteIntervalData(FacPosResult)
    End Sub
```

Writing Interval Data

```
    Imports AGI
    Sub WriteIntervalData(ByVal Result As STKObjects.IAgDrResult)
        Dim IntervalsList As STKObjects.IAgDrIntervalCollection
        Dim Interval As STKObjects.IAgDrInterval
        Dim DataSet As STKObjects.IAgDrDataSet
        Dim Values As Object
        Dim Value As Object

        IntervalsList = Result.Intervals

        'Iterate through the Intervals
        For Each Interval In IntervalsList
            Console.WriteLine(Interval.StartTime)
            Console.WriteLine(Interval.StopTime)
            'Iterate through the DataSets stored in the Interval
            For Each DataSet In Interval.DataSets
                Console.WriteLine(DataSet.Count)
                Console.WriteLine(DataSet.ElementName)
                Console.WriteLine(DataSet.ElementType)
                Console.WriteLine(DataSet.UnitType)
                'Get the values stored in the DataSet
                Values = DataSet.GetValues()
```

```
                'Iterate through the array of values
                For Each Value In Values
                    Console.WriteLine(CStr(Value))
                Next
            Next
        Next
    End Sub
```

Writing SubSection Data

```
    Imports AGI
    Sub WriteSubSectionData(ByVal Result As STKObjects.IAgDrResult)
        Dim Sections As STKObjects.IAgDrSubSectionCollection
        Dim Section As STKObjects.IAgDrSubSection
        Dim Interval As STKObjects.IAgDrInterval
        Dim DataSet As STKObjects.IAgDrDataSet
        Dim Values As Object
        Dim Value As Object

        Sections = Result.Sections

        For Each Section In Sections
            Console.WriteLine(Section.Title)
            'Iterate through the intervals stored in the Section
            For Each Interval In Section.Intervals
                Console.WriteLine(Interval.StartTime)
                Console.WriteLine(Interval.StopTime)
                'Iterate through the DataSets stored in the Interval
                For Each DataSet In Interval.DataSets
                    Console.WriteLine(DataSet.Count)
                    Console.WriteLine(DataSet.ElementName)
                    Console.WriteLine(DataSet.ElementType)
                    Console.WriteLine(DataSet.UnitType)
                    'Get the values stored in the DataSet
                    Values = DataSet.GetValues()
                    'Iterate through the array of values
                    For Each Value In Values
                        Console.WriteLine(CStr(Value))
                    Next
                Next
            Next
            WriteIntervalData(Section.Intervals)
        Next
    End Sub
```

Writing TextMessage Data

```
    Imports AGI
    Sub WriteTextMessageData(ByVal Result As STKObjects.IAgDrResult)
        Dim MessageContainer As STKObjects.IAgDrTextMessage
        Dim Message As String

        MessageContainer = Result.Message

        'Iterate through each string in Message Container
        For Each Message In MessageContainer
            Console.WriteLine(Message)
        Next
    End Sub
```

Appendix B) Object Model Data Provider Diagrams