Syncfusion
Deliver innovation with ease

Essential Studio 2013 Volume 4 - v.11.4.0.26

Essential OlapCommon

Contents

1 Business Intelligence (BI)

1.1 What is BI?

Business Intelligence (BI) simplifies information to enable all decision makers of an organization to access information easily. This helps the decision makers at all level to understand, analyze, collaborate, and act on information anytime, anywhere.

Here is how Wikipedia defines BI.

1.2 Why to use BI?

It is hard to over emphasize the importance of Business Intelligence (BI) in today's world. It is impossible to make strategic business decisions without analyzing the past business performance. Businesses are increasingly investing heavily in tools and services that let decision makers visually analyze data in myriad ways.

Several products and solutions have emerged to cater to the Business Intelligence market; Lessons have been learnt, and currently Online Analytical Processing (OLAP) is the de facto standard for persisting and accessing BI data. Applications built using OLAP include sales reports, executive reports, forecasting, and so on.

1.3 What's new in BI?

While BI has been an expensive affair in the past, requiring several thousands of dollars in investment for integrating a solution into an enterprise, recent technological developments have considerably reduced the cost to implement and own an OLAP-based BI solution.

Microsoft's SQL Server Analysis Services (SSAS) is one such solution, which is a set of OLAP services provided as part of Microsoft SQL Server. The SSAS is available for almost a decade, is a mature, very cost-effective solution for maintaining multidimensional (also called cube) data. With an efficient OLAP storage mechanism in place, you only need another efficient OLAP visualization tool set to complete your BI needs. This is where Syncfusion OLAP controls come into place.

1.3.1 Multi-dimensional Data

Multi-dimensional structure is defined as "a variation of the relational model that uses multidimensional structures to organize data and express the relationships between data". The structure is broken into cubes which are able to store and access data within the confines of each cube. Each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions. Even when data is manipulated it is still easy to access as well as be a compact type of database. The data still remains interrelated. Multidimensional structure is quite popular for analytical databases that use OLAP applications. Analytical databases use these databases because of their ability to deliver answers quickly to complex business queries. Data can be seen through different ways, which gives a broader picture of a problem unlike other models.

Multi-dimensional database products were commercially popularized as **Online Analytical Processing** (OLAP) systems to help analysts do decision support on large historical data. They expose a multidimensional view of the data with categorical attributes like Products and Stores forming the dimensions and numeric attributes like Sales and Revenue forming the measures or cells of the multidimensional cube. Dimensions are usually associated with the hierarchies that specify aggregation levels. For instance, store-name ->city ->state is a hierarchy on the Store dimension. The measure attributes are aggregated to various levels of detail about the combination of dimension attributes using functions like sum, average, count and variance. OLAP products provide convenient tools for exploring the data cubes through navigational operators like select, drill-down, roll-up and pivot conforming to the multidimensional view of data. An analyst can interactively invoke sequences of these simple operations to visualize the measures along various combinations of dimensions and at various levels of aggregation. Our OLAP Architecture provides you the easiest way to the extreme analysis of data.

2 Online Analytical Processing (OLAP)

Online Analytical Processing (OLAP) is a service that performs a real time, multi-dimensional analysis of complex data stored in a database. The OLAP typically consists of a server component that uses specialized algorithms, indexing tools to efficiently process data mining tasks and an OLAP client that efficiently lets you visualize the OLAP data pertaining to your business needs.

Here is how Wikipedia defines OLAP.

2.1 ADOMD.NET

The Syncfusion OLAP controls use the ADOMD.NET, which is a Microsoft .NET framework provider for retrieving data from OLAP servers in .NET platform. While ADOMD.NET primarily retrieves OLAP data from SQL Server Analysis Services (Microsoft's OLAP Server), it's adherence to industry standards like XML/A allows you to access any OLAP server (SAP, SAS, Hyperion, etc.) through it and hence provides you the ability to visualize using Syncfusion OLAP control, OLAP data from myriads of data sources including Microsoft's SSAS.

2.2 ADOMD.NET assembly and setup files information

The following assembly is required to run Syncfusion' OLAP samples:

- Microsoft.AnalysisServices.AdomdClient.dll

Install the following setup files for the above assembly:

- SQLServer2005_ADOMD.msi and SQLServer2005_ASOLEDB9.msi

Or

- SQLSERVER2008_ASADOMD10.msi and SQLSERVER2008_ASOLEDB10.msi

These setup files can be downloaded at Microsoft download center.

 ***Note: By default the below setup files will be installed while installing Syncfusion' Essential studio setup for BI edition:***

SQLSERVER2008_ASADOMD10.msi and SQLSERVER2008_ASOLEDB10.msi

3 Syncfusion OLAP Architecture

Syncfusion OLAP architecture allows you to build a full life cycle reporting solution for your enterprise. Here are the important pieces of the architecture:

- **OLAP Access Layer** - Built on top of ADOMD.NET and provides a high level object model to let you easily define reports.
- **OLAP Controls** - Chart, Grid, Gauge, Client for ASP.NET (excluding Gauge), WPF, Silverlight, ASP.NET MVC (Grid only).
- **OLAP Report Builder** – RAD (Rapid Application Development) tool lets you select the dimensions you are interested in visualizing and also lets you define the appearance for the Chart and Grid.

The following screenshot shows how the Syncfusion OLAP components allow you to build a full life cycle reporting solution for your enterprise.

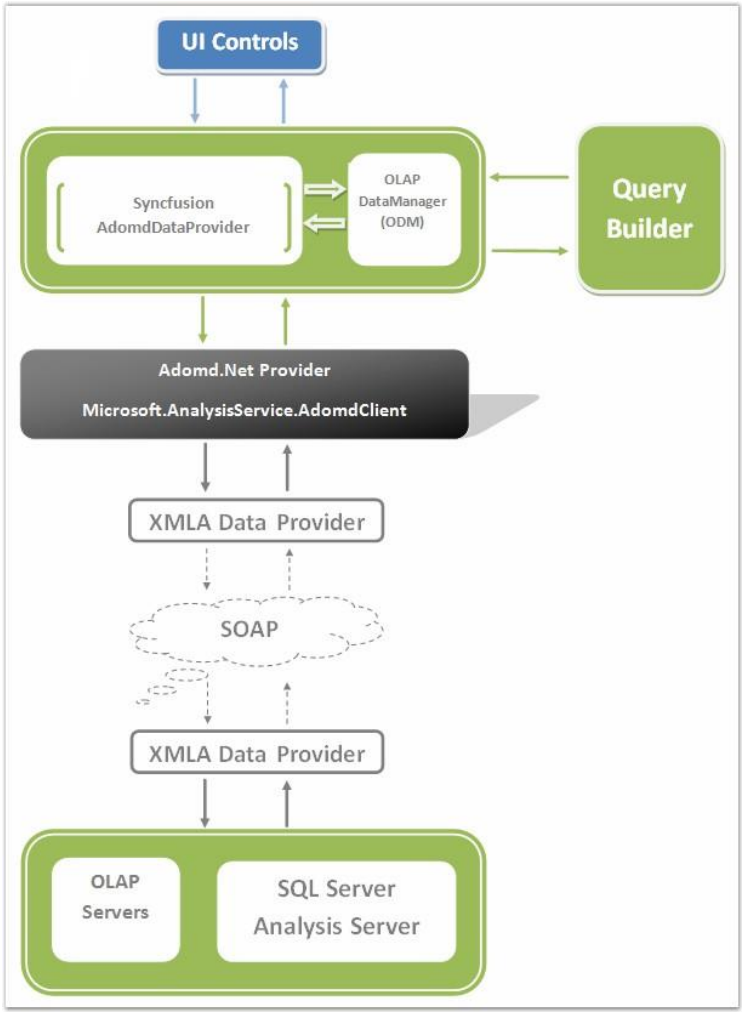

Figure 1: Syncfusion OLAP Architecture

3.1 OLAP Base

The OLAP Base is a class library that contains several namespaces and classes to perform data processing operations required by OLAP controls. OLAP base is the processing unit of all Syncfusion OLAP controls. From establishing the connection and retrieving the data, processing it and providing the formatted input for each control, everything is taken care by the OLAP base.

Syncfusion OLAP controls communicate to the OLAP cube through the OlapDataManager class available in Syncfusion.Olap.Base namespace.

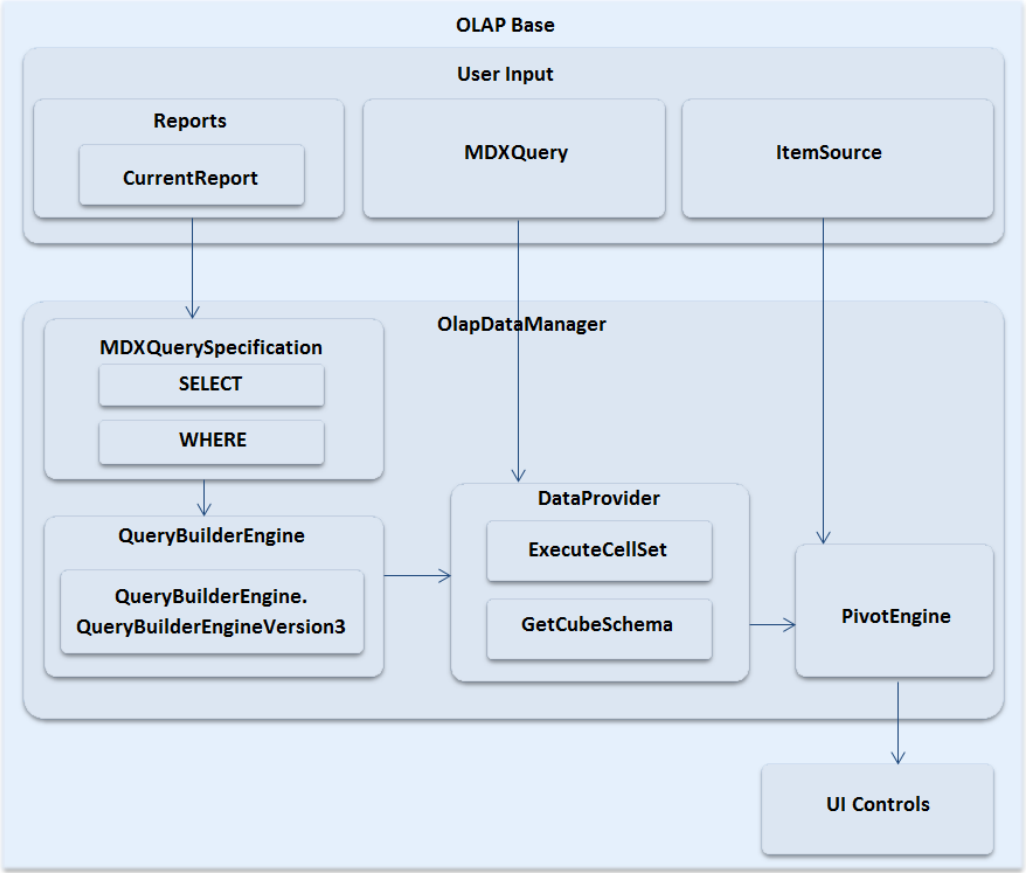

Figure 2: OLAP Base Architecture

Note: This class library was organized under Syncfusion.Olap.Base assembly.

3.2 OLAP Silverlight Base

OLAP Silverlight Base is a class library, which contains several namespaces and classes to perform data processing operation required by OLAP Silverlight controls. The OlapDataManager retrieves OLAP data and binds the result to an OLAP Control.

Note: This class library was organized under Syncfusion.OlapSilverlight.Base assembly.

3.3 OLAP Silverlight Base Wrapper

The OLAP Silverlight Base Wrapper is a class library, which contains several namespaces and classes. This library helps to perform data conversion between OLAP Silverlight Base and OLAP Base. Data Conversion process is used to achieve the following features:

1. **Establishing the connection, retrieving data** by converting the OLAP Silverlight Base information to OLAP Base information.
2. **Send retrieved data to OLAP Silverlight Base** by converting the OLAP Base data to OLAP Silverlight Base data.

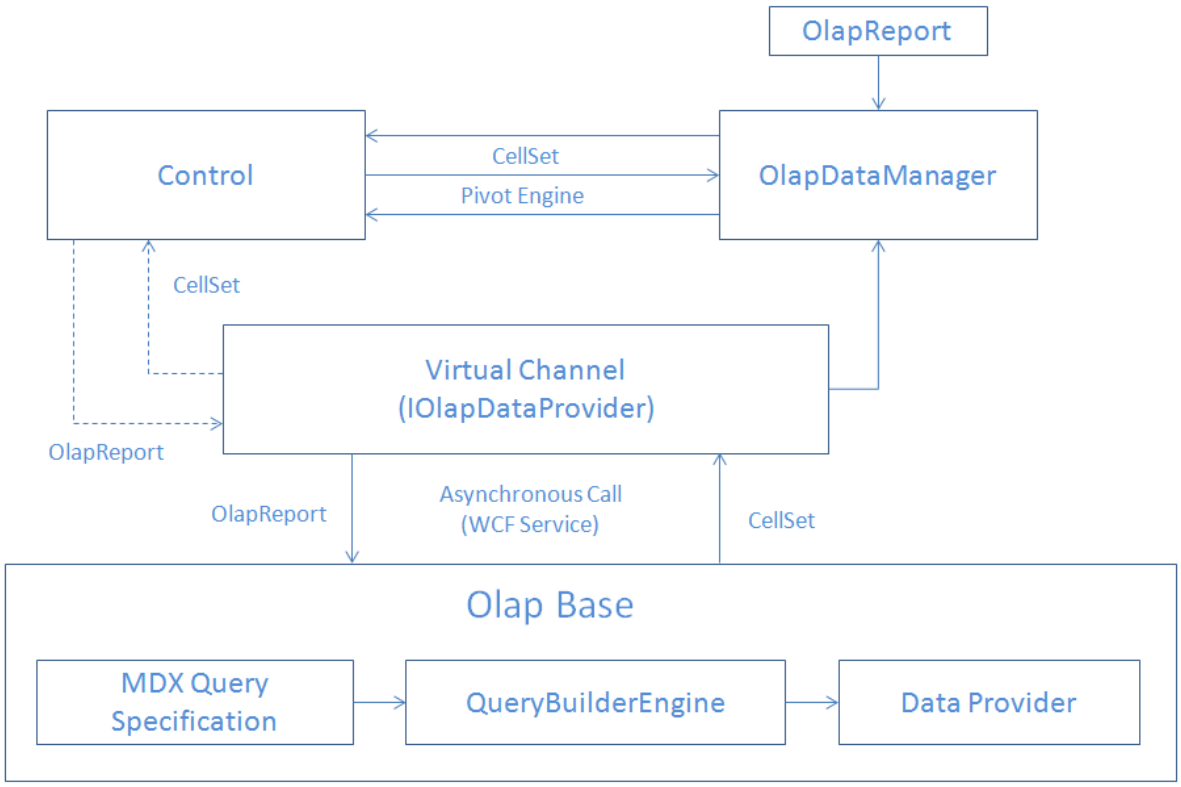

Figure 3: Dataflow in Silverlight

Note: This class library was organized under Syncfusion.OlapSilverlight.BaseWrapper assembly.

3.3.1 WCF Service

In order to query the database, OLAP Silverlight controls use WCF Service. The DataProvider property of OlapDataManager performs the service call operations.

Creating and connecting a WCF Service.

4 Concepts

4.1 OlapDataProvider

The database connectivity related works are all taken care of by this part of the base. Here we are using **Microsoft.AnalysisService.AdomdClient** data provider. Establishing the connection, checking the state of the connection and closing the connection are basic operations provided by the general data provider, but we need some information beyond this in order to provide the input for our controls.

This part of the base will get the connection information and establish a connection with the specified data source and retrieve the information from the data base and store it in its classes. This part of the base will have the required logic to retrieve the information from the data base and store it in the **object of class** in **Data** namespace.

All the information about the connected cube will intersect and be stored in **object of classes** in **Data** namespace, which are equivalent to the classes in the **AdomdClient**. This information is required to provide the input for OLAP controls.

Important class in DataProvider namespace:

- AdomdDataProvider

4.1.1 AdomdDataProvider

The important properties and methods in AdomdDataProvider class are tabulated below:

Table 1: Properties

Property Name	Description	Type	Value it Accepts	Reference Link
CatalogName	To get the connected database name	string	-	-
ConnectionString	To set or get the connection string	string	-	-
CurrentCellSet	To get the currently executed **CellSet**	CellSet	-	-
GetCubes	To get the information about the cubes in the connected data source	CubeInfoCollection	-	-

Table 2: Methods

Method Name	Description	Parameters	Return Type	Reference Link
ExecuteCellSet	Four arguments should be given to invoke this method. The arguments are MDX query as string, drill down state of result set as bool, query append info as bool and finally get the **OlapReport**. This method will generate the **CellSet** for the given query or **OlapReport**.	Mdx Query as string, drill down state as bool, Property append status as bool and Current OlapReort of OlapDataManager	CellSet	-
GetCubeSchema	This method will get the cube name and intersect the cube to get all the information about the cube and return an object of type **CubeSchema**, which will contain all the information. **CubeSchema** is a class in Data namespace.	Cube Name as string	CubeSchema	-
GetChildMembers	This method will get the member element and the expander state and return the child members of the given member as MemberCollection.	Parent member as Member and drill down state as bool	MemberCollection	-
GetLevelMembers	This method will get the level element as argument and return the member elements of that level as MemberCollection.	Level	MemberCollection	-
GetParentMember	This method is used to find the parent member of a member element. By passing a member element as an argument, this element will return its parent	Member	Member	-

Method Name	Description	Parameters	Return Type	Reference Link
	member.			
ValidateConnectionString	This method will validate the specified connection string in the data manager or in the data provider and return the validated status.	-	bool	-

4.2 OlapDataManager

OlapDataManager is the most important class in the whole OLAP Base. All the information transfers from the control to OLAP base will happen through this class and this will retain the current state of the base objects. The connection is established in the Data provider of the OLAP Base, but the information required in establishing the connection is given to the data provider through the **OlapDataManager**.

Table 3: Constructors

Constructors	Description	Parameters	Return Type	Reference Link
OlapDataManager()	Default constructor	-	Void	-
OlapDataManager(string)	Accepts the connection string as argument and passes it to the Data Provider to establish the connection with data source.	String	Void	-
OlapDataManager(AdomdDataProvider)	Accepts the Data Provider as argument and processes the cube that is connected with the given data provider.	AdomdDataProvider	Void	-

Establishing connection with the SSAS server

The following code snippet describes establishing connection with the server:

```
[C#]

OlapDataManager olapDataManager = new OlapDataManager("DataSource=local
```

```
host; Initial Catalog=Adventure Works DW");
```

[VB]

```
OlapDataManager olapDataManager = New
OlapDataManager("DataSource=localhost;
Initial Catalog=Adventure Works DW")
```

Or

Establishing Connection with the SSAS server through Data Provider

The following code snippet describes establishing connection with the server:

[C#]

```
AdomdDataProvider dataProvider = new
AdomdDataProvider("DataSource=localhost;
Initial Catalog=Adventure Works DW");
OlapDataManager olapDataManager = new OlapDataManager(dataProvider);
```

[VB]

```
Dim dataProvider As AdomdDataProvider = New
AdomdDataProvider("DataSource=localhost;
Initial Catalog=Adventure Works DW")
Dim olapDataManager As OlapDataManager = New
OlapDataManager(dataProvider)
```

Establishing connection with the offline cube

The following code snippet describes establishing connection with the offline cube:

[C#]

```
OlapDataManager olapDataManager = new OlapDataManager(@"Data Source =
C:\ Common\Data\OfflineCube\Adventure_Works_Ext.cub; Provider =
MSOLAP;");
```

[VB]

```
OlapDataManager olapDataManager = New OlapDataManager("Data Source =
C:\\ Common\\Data\\OfflineCube\\Adventure_Works_Ext.cub; Provider =
MSOLAP;")
```

Establishing connection with XMLA Server:

XML for Analysis (XMLA) is a standard that allows the client applications to transfer multi-dimensional or OLAP data sources, which is available in the Mondrian Server. The back and forth communication is done using the web standards – HTTP, SOAP, and XML. The query language used is MDX, is most widely supported for reporting from multi-dimensional data stores.

Use Case Scenarios

XMLA provides the most efficient way to access an OLAP database over the Internet.

Connecting to Mondrian Server

The following code illustrates how to connect to the Mondrian server:

```
[C#]


// Connecting to Mondrian Server
OlapDataManager DataManager = new OlapDataManager("Data Source = http://localhost:8080/mondrian/xmla; Initial Catalog = FoodMart;");
DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

```
[VB]


' Connecting to Mondrian Server
Dim DataManager As OlapDataManager = New OlapDataManager("Datasource = http://bi.syncfusion.com:8080/mondrian/xmla; Initial Catalog=FoodMart;")
DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Providers.Mondrian
```

Click here for more information about Mondrian XMLA configurations.

Connecting to Active Pivot Server

The following code illustrates how to connect to Active Pivot server:


```
[C#]


// Connecting to Active Pivot Server
OlapDataManager DataManager = new
OlapDataManager(@"Data Source=http://localhost:8081/var_s/xmla;
Initial Catalog=VaRCubes;
User ID=; Password=; Transport Compression=None;");
DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.ActivePivot;
```

```
[VB]


' Connecting to Active Pivot Server
Dim DataManager As OlapDataManager = New OlapDataManager("Data
Source=http://localhost:8081/var_s/xmla;  Initial Catalog=VaRCubes;
User ID=; Password=; Transport Compression=None;")
DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.ActivePivot
```

Click here for more information on Active Pivot server.

4.2.1 Properties and Methods

Properties

The properties and their descriptions are tabulated below:

Table 4: Properties

Property Name	Description	Type	Value it Accepts	Reference Link
ConnectionString	Used to pass the connection string to establish the connection. The user can also get the connection string using this property.	string	String	-
CurrentCellSet	The user can get the **CellSet** of their input from	CellSet	-	-

Property Name	Description	Type	Value it Accepts	Reference Link
	here.			
CurrentCubeName	Used to set or get the current cube name. When set, the whole data procession will change to the specified cube name.	string	String	-
CurrentCubeSchema	The user can get the **CubeSchema** of the currently connected cube.	CubeSchema	-	-
CurrentReport	Used to load an **OlapReport** or get the currently loaded report.	OlapReport	OlapReport	OlapReport
DataProvider	Used to set a data provider and get the existing data provider.	IDataProvider	IDataProvider	DataProvider
IsCurrentReportModified	Used to get or set the modified status of the currently loaded report.	bool	True/False	-
Itemsource	Used to bind the Non-OLAP data to **OlapDataManager.**	Object	Enumerable collection/ ITyped List	-
MdxQuery	Used to pass the MDX query as input.	string	String	MdxQuery
PivotEngine	Used to get the **PivotEngine** for the given input.	PivotEngine	PivotEngine	-
QuerySpecification	Used to get the **MDXQuerySpecification** for the given **OlapReport**.	MDXQuerySpecification	-	QuerySpecification
ReportPath	Used to get or set the report path to store the report as an XML file.	string	-	-
Reports	Contains the report collection of the	OlapReportCollec	OlapReportColl	-

Property Name	Description	Type	Value it Accepts	Reference Link
	OlapDataManager.	tion	ection	

Table 5: Methods

Method Name	Description	Parameters	Return Type	Reference Link
AddReport	Used to add a new **OlapReport** to the report collection of **OlapDataManager**.	OlapReport	Void	-
CloneOlapDataManager	Create a clone from the current state of **OlapDataManager** and return the new copy of **OlapDataManager**. The **IDataProvide**r will be same for both the managers.		OlapDataManager	-
ExecuteCellSet	This is an overloaded method that can be invoked by calling or by passing the **MDXQuerySpecification** or by passing the MDX query as string. This method will invoke the data processing and will return the processed **CellSet**, which will be further formatted.		CellSet	ExecuteCellSet
ExecuteOlapTable	Used to create pivot engine from the **CellSet**.	-	PivotEngine	
GetMDXQuery	This method will generate the MDX query from the **MDXQuerySpecification** and		string	-

Method Name	Description	Parameters	Return Type	Reference Link
	return the query as a string.			
GetReport	This method will return the **OlapReport** collection by giving the Xml report file name as an input.	string	OlapReportCollection	-
GetReportAsStream	This method will return the current report collection along with current state of **OlapDataManager** as a Stream.	-	Stream	GetReportAsStream
LoadOlapDataManager	Will accept the **OlapReport** as argument and Load the **OlapDataManager** with the given **OlapReport**.	OlapReport	Void	LoadOlapDataManager
LoadReport	Used to get the report name as an argument and pick the specified report from the report collection to load the **OlapDataManager** with that report.	string	Void	-
LoadReportDefinitionFile	Used to get the XML report file as input and load the **OlapDataManager** with the report in that file.	string	Void	LoadReportDefinitionFile
LoadReportDefinitionFromStream	Used to get a stream as input and read the report from that stream and load the **OlapDataManager** with that report.	Stream	Void	LoadReportDefinitionFromStream
NotifyElementModified	Used to trigger the **ElementModified** event.	-	Void	-

Method Name	Description	Parameters	Return Type	Reference Link
NotifyReportChanged	Used to trigger the **ReportChanged** event.	-	Void	-
RemoveReport	Used to get the report name as input and remove that report from report collection.	string	Void	RemoveReport
RenameReport	Used to rename the current report of **OlapDataManager**.	Index as **int** and new Report Name as **string**	void	RenameReport
SaveReport	Used to get the file name and store the current report collection along the current state of the **OlapDataManager** as an Xml file.	string	Void	SaveReport
SetCurrentReport	Used to set an **OlapReport** as the current report of the **OlapDataManager**. This method is the initial method that will start data processing.	OlapReport	Void	SetCurrentReport

4.2.2 UseWhereClauseForSlicing

The UseWhereClauseForSlicing property facilitates the user to decide whether the MDX query parser engine should consider 'Where' or 'Select' clause for slicing data.

Use Case Scenarios

While slicing dimensions with a specific range of measures using 'Select' clause in MDX query, an exception is thrown. This can be resolved by using the 'Where' clause for slicing.

Example: Slicing the Date dimension from months of 2002 to months of 2003 will throw an exception when 'Select' clause is used.

Properties

Table 6: Property Table

Property	Description	Type	Data Type	Reference links

UseWhereClause ForSlicing	Enables the user to decide whether the MDX Query Parser Engine should consider the 'Where' or 'Select' clause for slicing operation	Server side	Boolean	----

Adding UseWhereClauseForSlicing property to an Application:
Refer to 5.22

4.2.3 Drill Through

This feature enables the user to drill through any value and see the data which formed the value.

The following code snippet illustrates how to drill through using MDX Query in OlapDataManager:

```
[C#]
string query = @"drillthrough Select { [Customer].[Customer Geography].
[Country].&[Australia] } on 0, { [Date].[Fiscal].[Fiscal Year].&[2002]
} on 1 from [Adventure Works] Where [Internet Sales Amount]";
// executing the drill-through operation.
olapDataManager.Execute(query);
```

```
[VB]
Dim query As String = "drillthrough Select { [Customer].[Customer
Geography].[Country].&[Australia] } on 0, { [Date].[Fiscal].[Fiscal
Year].&[2002] } on 1 from [Adventure Works] Where [Internet Sales
Amount]"
'Executing the drill-through operation.
olapDataManager.Execute(query)
```

4.3 OlapReport

OlapReport is an object that contains information about the cube element that has to be included for processing along its axis position and filter and sorting constraints. **OlapReport** has categorized the elements based on their characteristics as below:

- Dimension Element
  - Hierarchy Element
  - Level Element
  - Member Elements
- Measure Element
- KPI Element
- NamedSet Element

- Sort Element
- Calculated Member
- Subset Element
- Summary Element

These elements are to get the cube element information from the user. You can create an instance of this element and add the required information about the element to pick that specific element from the data base for processing.

These elements come under the following four elements group, which are based on the axis position and filtering constraints.

- Categorical Elements – contains the elements that has to come under categorical axis
- Series Elements – contains elements that has to come under series axis
- Slicer Elements – contains elements that have to come under slicer elements.
- Filter Elements – contains elements that are subjected to filter.

All the elements are internally maintained as Item.

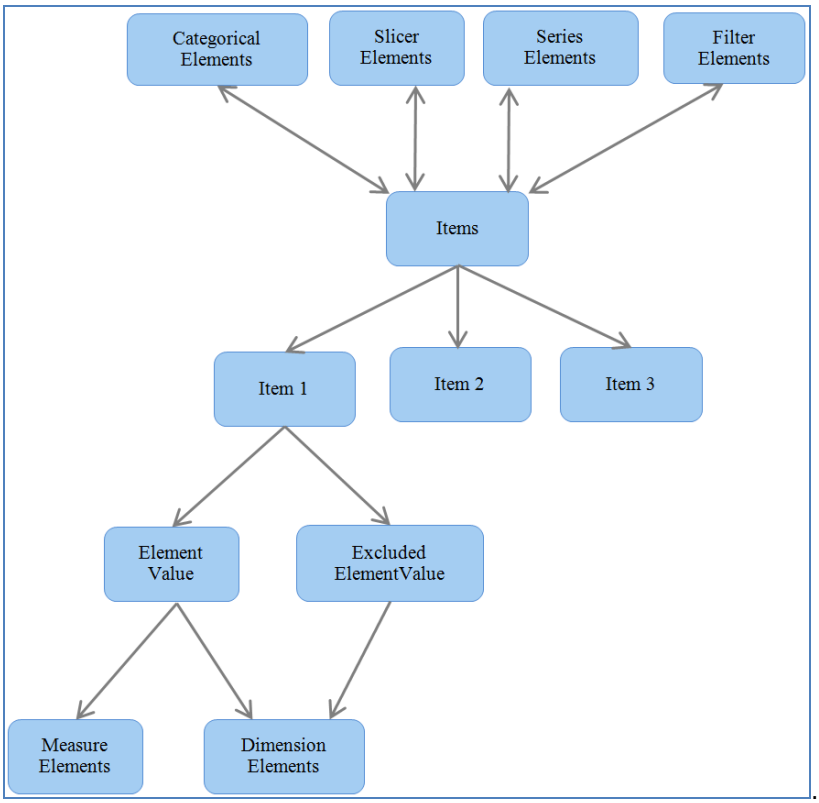

Figure 4: Architecture of Items

4.3.1 Properties and Methods

The OlapReport properties and its descriptions are tabulated:

Table 7: Properties

Name	Description	Type	Value it accepts	Reference Link
CalculatedMembers	Used to set and get the calculated members of the **OlapReport**.	Items	Items	CalculatedMembers
CategoricalElements	Contains the element that are said to be in categorical axis. We can add an element and get the collection of elements that comes under the categorical axis.	Items	Items	-
CurrentCubeName	Used to set or get the current cube name of the Report.	string	String	-
FilterElements	Contains elements that are subjected to Filter constraints and a filter expression along the measure on which the filter expression is built.	Items	Items	FilterElements
Name	Used to set or get the report name.	string	String	-
SeriesElements	Contains elements that are said to be in series axis. We can add an element and get the collection of elements that comes under the categorical axis.	Items	Items	-
ShowEmptyColumnData	Used to show/hide the empty column in the result set.	bool	True/False	-
ShowemptyRowData	Used to show/hide the empty row in the result set.	bool	True/False	-
ShowExpanders	Used to show/hide the expander buttons in the Olap contorls.	bool	True/False	ShowExpanders
SliceElements	Contains the element that are said to be in slicer axis. We can add an element and get the	Items	Items	-

Name	Description	Type	Value it accepts	Reference Link
	collection of elements that comes under the categorical axis.			
TogglePivot	Used to swap the elements in the column axis and row axis.	bool	True/False	-
Tag	Holds the backup information of the OLAP Report.	CLR.	String	Tag property in OlapReport

4.3.2 Dimension Element

A simple dimension object is composed of basic information such as Name, Hierarchy, Level and Members. We can create the dimension element by specifying its name and providing the hierarchy and level name.

The dimension element will contain the hierarchical details and information about each included level elements in that hierarchy. A hierarchy can have any number of level elements and the level elements can have any number of members and the member elements can have any number of child members.

Hierarchy Element

Each element of a dimension could be summarized using a hierarchy. The hierarchy is a series of parent-child relationship, where a parent member represents the consolidation of members which are its children. Parent members can be further aggregated as the children of another parent.

For example, May 2005 could be summarized into Second Quarter 2005 which in turn would be summarized in the year 2005.

Level Element

Level element is the child of hierarchy element which contains a set of members, each of which has the same rank within a hierarchy.

Member Element

Member element represents a member of a level in a cube, the children of a member of level.

Member Properties

Member properties cover the basic information about each member in each tuple. This basic information includes the member name, parent level, the number of children, and so on. Member properties are available for all members at a given level.

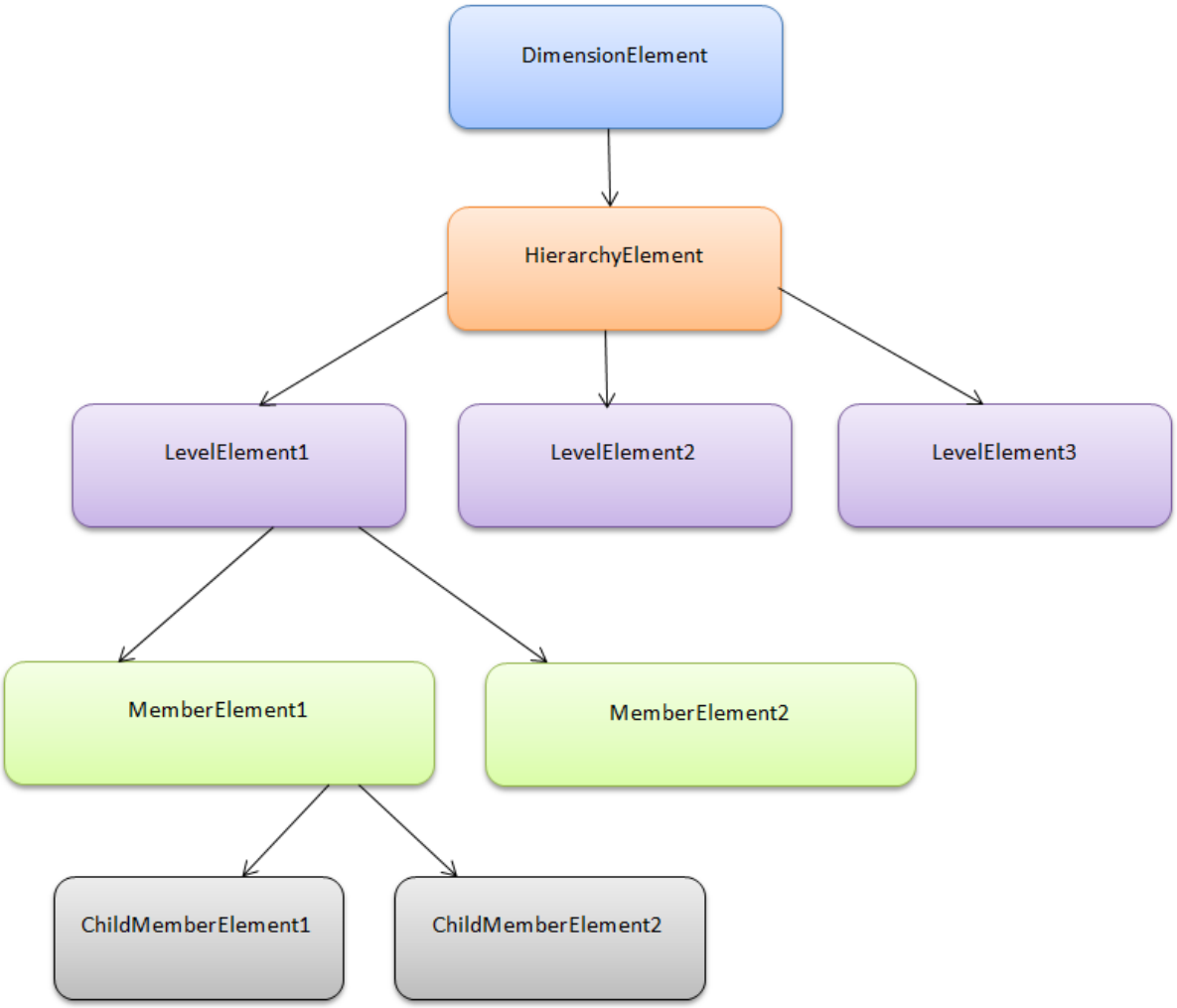

Figure 5: Hierarchical structure of a dimension

The following code will illustrate the creation of different types of dimensions:

Creating Simple Dimension Element

```
[C#]

DimensionElement dimensionElementColumn = new DimensionElement();
// Specifying the Name for Column Dimension Element
dimensionElementColumn.Name = "Customer";
// Specifying the Hierarchy and Level Element Name
dimensionElementColumn.AddLevel("Customer Geography", "Country");
```



```
[VB]

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Name for Row Dimension Element
dimensionElementRow.Name = "Date"
' Specifying the Hierarchy and Level Element Name
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")
```

Creating Sliced Dimension

```
[C#]

DimensionElement dimensionElementSlicer = new DimensionElement();
// Specifying the dimension Name
dimensionElementSlicer.Name = "Product";
// Adding the Level Name along with the Hierarchy Name
dimensionElementSlicer.AddLevel("Product Categories", "Category");
// Adding the Member Element
dimensionElementSlicer.Hierarchy.LevelElements["Category"].Add("Bikes")
;
dimensionElementSlicer.Hierarchy.LevelElements["Category"].IncludeAvail
ableMembers = true;
```

```
[VB]


Dim dimensionElementSlicer As DimensionElement = New DimensionElement()
' Specifying the dimension Name
dimensionElementSlicer.Name = "Product"
' Adding the Level Name along with the Hierarchy Name
dimensionElementSlicer.AddLevel("Product Categories", "Category")
' Adding the Member Element
dimensionElementSlicer.Hierarchy.LevelElements("Category").Add("Bikes")
dimensionElementSlicer.Hierarchy.LevelElements("Category").IncludeAvail
ableMembers = True
```

4.3.3 Measure Element

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. In addition, measures are the central values of a cube that are analyzed. That is, measures are the numeric data of primary interest to end users browsing a cube. The measures you select depend on the types of information end users request. Some common measures are sales, cost, expenditures, and production count.

We can create a measure element just by specifying its name. The following code will illustrate the creation of a measure element:

```
[C#]


MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Measure Elements
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
Sales Amount" });
```

```
[VB]


Dim measureElementColumn As MeasureElements = New MeasureElements()
'Specifying the Measure Elements
measureElementColumn.Elements.Add(New MeasureElement With {.Name =
"Internet Sales Amount"})
```


4.3.4 Key Performance Indicator (KPI) Element

Key Performance Indicator(KPI) is a collection of calculations that are associated with a measure group in a cube that are used to evaluate business success. Typically, these calculations are a combination of Multidimensional Expressions (MDX) or calculated members. The KPIs also have an additional metadata that provides information about how client applications should display the results of the KPI's calculations.

The different types of KPI Indicators are:

- KPI Goal
- KPI Status
- KPI Trend
- KPI Value

We can create a KPI element by specifying its name and giving details of the indicator that are included in the element.

```
[C#]
KpiElements kpiElement = new KpiElements();
// Specifying the KPI Element name and configuring its Indicators
kpiElement.Elements.Add(new KpiElement { Name = "Internet Revenue",
ShowKPIGoal = true, ShowKPIStatus = true, ShowKPIValue = true,
ShowKPITrend = true });
```

```
[VB]
Dim kpiElement As KpiElements = New KpiElements()
' Specifying the KPI Element name and configuring its Indicators
kpiElement.Elements.Add(New KpiElement With {.Name =
"Internet Revenue", .ShowKPIGoal = True, .ShowKPIStatus = True,
 .ShowKPIValue = True, .ShowKPITrend = True})
```

4.3.5 NamedSet Element

A named set is a collection of tuples and members, which can be defined and saved as a part of the cube definition. Named set records reside inside the sets folder, which is under a dimension element. These elements can be dragged to Categories/Series/Slicer axis of Axes Element Builder. To help make working with a lengthy, complex, or commonly used expression easier, Multidimensional Expressions (MDX) lets you to define a named set.

The following code will describe the creation of a Named set Element:

```
[C#]
NamedSetElement dimensionElementRow = new NamedSetElement();
// Specifying the dimension name
dimensionElementRow.Name = "Negative Margin Products";
```

```
[VB]
Dim dimensionElementRow As NamedSetElement = New NamedSetElement()
' Specifying the dimension name
dimensionElementRow.Name = "Negative Margin Products"
```

4.3.6 Sort Element

The result set can be sorted by using the **SortElement**. We can create Sort elements and add it to the **OlapReport**. There are four types of sorting orders. They are:

- ASC – Sort the elements in ascending order.
- BASC – Sort the elements in ascending order by breaking the Hierarchy.
- DESC – Sort the elements in Descending order.
- BDESC – Sort the elements in Descending order by breaking the Hierarchy.

The following report will illustrate the creation of Sort element**:**

```
[C#]
SortElement sortElement = new SortElement(AxisPosition.Categorical,
SortOrder.BDESC, true);
sortElement.Element.UniqueName = "[Measures].[Internet Sales Amount]";
```

```
[VB]
Dim sortElement As SortElement = New
SortElement(AxisPosition.Categorical, SortOrder.BDESC, True)
sortElement.Element.UniqueName = "[Measures].[Internet Sales Amount]"
```

4.3.7 Calculated Member

Calculated Members are the customized measures or dimension members created with the cube. The values are calculated at run-time. It is a user defined Element. The two types of calculated members are as follows:

1. **Calculated Measure** – Calculated member created from a measure element
2. **Calculated Dimension** – Calculated member created from a dimension

Calculated Member to be defined in **OlapReport** requires the following definitions:

- Name – Name of the calculated member
- Expression – Expression to form the calculated member
- Measure/Dimension Element – You should add a measure or dimension element from which the calculated member should be created.

The following steps will explain how to create and add a calculated member in an OlapReport

1. Create the dimension measure or dimension element from which the calculated member has to be created.
2. Create the calculated member by giving the name and expression.
3. Add the element created in Step 1 to the calculated member.
4. Once the calculated member is defined, add that member to the calculated member collection in an OlapReport.
5. Add the newly created calculated members in the categorical or series axis of an OlapReport.

The following code snippet will describe the creation and addition of a calculated member in OlapReport:

Calculated Measure

```
[C#]
MeasureElement measureElement = new MeasureElement();
measureElement.Name = "Order Quantity";
CalculatedMember calculatedMeasure = new CalculatedMember();
calculatedMeasure.Name = "Oder on Discount";
calculatedMeasure1.Expression = "[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)";
calculatedMeasure1.AddElement(measureElement);
olapReport.CalculatedMembers.Add(calculatedMeasure);
```

```
[VB]
Private measureElement As MeasureElement = New MeasureElement()
measureElement.Name = " Order Quantity "
Dim calculatedMeasure As CalculatedMember = New CalculatedMember()
calculatedMeasure.Name = " Order on Discount "
calculatedMeasure1.Expression = "[Measures].[Order Quantity] +
([Measures].[Order Quantity] *0.10)"
calculatedMeasure1.AddElement(measureElement)
olapReport.CalculatedMembers.Add(calculatedMeasure)
```

Calculated Dimension

```
[C#]
DimensionElement internalDimension = new DimensionElement();
internalDimension.Name = "Product";
internalDimension.AddLevel("Product Categories", "Category");
// Calculated Dimension
CalculatedMember calculateDimension = new CalculatedMember();
calculateDimension.Name = "Bikes & Components";
calculateDimension.Expression = "([Product].[Product Categories].
[Category].[Bikes] + [Product].[Product Categories].[Category].
[Components] )";
calculateDimension.AddElement(internalDimension);
olapReport.CalculatedMembers.Add(calculateDimension)
```

```
[VB]
Dim internalDimension As DimensionElement = New DimensionElement()
internalDimension.Name = "Product"
internalDimension.AddLevel("Product Categories", "Category")
' Calculated Dimension
Dim calculateDimension As CalculatedMember = New CalculatedMember()
calculateDimension.Expression = "Bikes & Components"
calculateDimension.Expression = "([Product].[Product Categories].
[Category].[Bikes] + [Product].[Product Categories].[Category].
[Components] )"
calculateDimension.AddElement(internalDimension)
olapReport.CalculatedMembers.Add(calculateDimension)
```

4.3.8 Subset Element

Subset Elements are used to filter the result set by their count. It will just filter the number records and number of fields in the result set.

The following codes will illustrate the creation of a Subset Element:

```
[C#]
SubsetElement subSetElementColumn = new SubsetElement(5);
subSetElementColumn.Name = "Top 5 Elements";
```

```
SubsetElement subSetElementRow = new SubsetElement(3);
subSetElementRow.Name = "Top 3 Elements";

olapReport.CategoricalElements.SubSetElement = subSetElementColumn;

olapReport.SeriesElements.SubSetElement = subSetElementRow;
```

[VB]

```
Dim subSetElementColumn As SubsetElement = New SubsetElement(5)

subSetElementColumn.Name = "Top 5 Elements"

Dim subSetElementRow As SubsetElement = New SubsetElement(3)

subSetElementRow.Name = "Top 3 Elements"

olapReport.CategoricalElements.SubSetElement = subSetElementColumn

olapReport.SeriesElements.SubSetElement = subSetElementRow
```

4.3.9 Summary Elements

Summary elements come when the base deals with Non-OLAP data. This is similar to the measure element in the OLAP data. This element will be considered as the measure and this value will be displayed in the value cell.

The following codes will describe the creation of the Summary elements:

[C#]

```
 // Specifying the Summary Elements
SummaryElements summaries = new SummaryElements();
summaries.Add(new SummaryInfo { Column = "Quantity", Key = "Quantity",
Type = SummaryType.Sum });
summaries.Add(new SummaryInfo { Column = "Amount", Key = "Amount", Type
 = SummaryType.Sum, FormatString = "{0:c}" })
```

[VB]

```
' Specifying the Summary Elements

Dim summaries As SummaryElements = New SummaryElements()

summaries.Add(New SummaryInfo With {.Column = "Quantity", .Key = "Quantity",
.Type = SummaryType.Sum})

summaries.Add(New SummaryInfo With {.Column = "Amount", .Key = "Amount",
.Type = SummaryType.Sum, .FormatString = "{0:c}"})
```

4.3.10 Filtering slicer elements by range

This feature enables you to specify a range for filter elements in the slicer field. You have to specify the start and end value to set the range. Multiple ranges can be added for the filter elements in slicer field.

Use Case Scenarios

It is not required to enter each member manually. This makes filtering easy.

Class

Table 8: Class Table

Name	Description
SlicerRangeFiltersInfo	Used to filter values from one range to another. Unique name of the member element for start and end value need to be specified. The name of the member element can also be specified for start and end value when custumer builds the unique name*.

* Name of the member element can be specified only when name is formed with dimension name, hierarchy name and level name.

Constructor

Table 9: Constructor Table

Syntax	Description	Parameter
SlicerRangeFiltersInfo(string startValueUniqueName, string endValueUniqueName)	Initializes SlicerRangeFiltersInfo with unique name as star and end values.	Unique name for start and end value.
SlicerRangeFiltersInfo(string dimensionName, string hierarchyName, string levelName, string startValueName, string endValueName	Initializes SlicerRangeFiltersInfo with name of dimension, hierarchy, level, star value and end value.	Name for dimension, hierarchy, level, start value and end value.

Properties

Following table consists of SlicerRangeFiltersInfo class's property:

Table 10: Properties Table

Property	Description	Type	Data Type	Reference links

DimensionName	Specify the dimension name	None	String	NA
HierarchyName	Specify the hierarchy name	None	String	NA
LevelName	Specify the level name	None	String	NA
StartValue	Specify the unique name or name of the member element*	None	String	NA
EndValue	Specify the unique name or name of the member element*	None	String	NA

* Name of the member element can be specified only when the name is formed with dimension name, hierarchy name and level name.

Adding a range for the filter elements in slicer field:

There are two methods to add range for the filter elements in a slicer field.

In the first method, you can specify the unique name for start and end value. The following code illustrates this:

[C#]

```
olapReport.SlicerRangeFilters.Add(new SlicerRangeFiltersInfo("[TimeFlat].[201010100031]", "[TimeFlat].[201010100037]"));
```

[VB]

```
olapReport.SlicerRangeFilters.Add(New SlicerRangeFiltersInfo("[TimeFlat].[201010100031]", "[TimeFlat].[201010100037]"))
```

In the second method, you can specify the member name along with dimension name, hierarchy name and level name. Entering the unique name for start and end value is not mandatory. The following code illustrates this:

[C#]

```
olapReport.SlicerRangeFilters.Add(new SlicerRangeFiltersInfo { DimensionName = "TimeFlat", HierarchyName = "TimeFlat", LevelName = "TimeId", StartValue = "201010100031", EndValue = "201010100037" });
```

```
[VB]
olapReport.SlicerRangeFilters.Add(New SlicerRangeFiltersInfo With {.Dimensio
nName = "TimeFlat", .HierarchyName = "TimeFlat", .LevelName = "TimeId", .Sta
rtValue = "201010100031", .EndValue = "201010100037"})
```

	Number of failed connections						
	Austria	Brazil	England	Germany	Poland	United States	Total
#null	2	0	1	1			4
Linux		0	1		1		2
Mac Os	1	1	1	1			4
Solaris					0	1	1
Windows		1	0	1	0		2
Total	3	2	3	3	1	1	14

Figure 6: Before applying range for filtering

	Number of failed connections						
	Austria	Brazil	England	Germany	Poland	United States	Total
#null	0		1				1
Linux		0	1				1
Mac Os			1				1
Solaris						1	1
Windows		0		0			0
Total	0	0	3	0		1	5

Figure 7: After applying range for filtering

4.3.11 Creating the OlapReport

To create a report:

1. Instantiate a new object for **OlapReport**.
2. Create the required elements like dimension element, measure elements.
3. Add the created element in the desired axis (Column or Categorical, Row or Series, Filter or Slicer) elements.
4. Then bind the created report to the **OlapDataManager** using the **SetCurrentReport()** method or assign the report to **OlapDataManager's** current report property.

4.3.11.1 Sample Reports for OLAP data

This section gives you the code snippet to generate different types of report for OlapDataManager.

4.3.11.1.1 Simple Report

[C#]

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
 Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

/// Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.HierarchyName = "Customer Geography"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Int
```

```
ernet Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

''' Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
'''Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn)
'''Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.2 Report with slicing operation

[C#]

```
OlapReport olapReport = new OlapReport();

olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();

//Specifying the dimension Name
dimensionElementColumn.Name = "Customer";

//Adding the Level Name along with the Hierarchy Name
dimensionElementColumn.AddLevel("Customer Geography", "Country");

DimensionElement dimensionElementRow = new DimensionElement();

//Specifying the dimension Name
dimensionElementRow.Name = "Date";

//Adding the Level Name along with the Hierarchy Name
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

DimensionElement dimensionElementSlicer = new DimensionElement();

dimensionElementSlicer.Name = "Sales Channel";
dimensionElementSlicer.AddLevel("Sales Channel", "Sales Channel");

MeasureElements measureElementRow = new MeasureElements();
measureElementRow.Elements.Add(new MeasureElement { Name = "Internet Sa
les Amount" });
```

```
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.SeriesElements.Add(dimensionElementRow);
olapReport.SlicerElements.Add(dimensionElementSlicer);
olapReport.SeriesElements.Add(measureElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()

'Specifying the dimension Name
dimensionElementColumn.Name = "Customer"

'Adding the Level Name along with the Hierarchy Name
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim dimensionElementRow As DimensionElement = New DimensionElement()

'Specifying the dimension Name
dimensionElementRow.Name = "Date"

'Adding the Level Name along with the Hierarchy Name
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

Dim dimensionElementSlicer As DimensionElement = New DimensionElement()
dimensionElementSlicer.Name = "Sales Channel"
dimensionElementSlicer.AddLevel("Sales Channel", "Sales Channel")

Dim measureElementRow As MeasureElements = New MeasureElements()
measureElementRow.Elements.Add(New MeasureElement { Name = "Internet Sales
Amount" });
```



```
measureElementRow.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})


olapReport.CategoricalElements.Add(dimensionElementColumn)

olapReport.SeriesElements.Add(dimensionElementRow)

olapReport.SlicerElements.Add(dimensionElementSlicer)

olapReport.SeriesElements.Add(measureElementRow)
```

4.3.11.1.3 Report with dicing operation

[C#]

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
//Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
 Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.HierarchyName = "Fiscal";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

//Specifying Excluded column elements
DimensionElement excludedColumnElement = new DimensionElement();
excludedColumnElement.Name = "Customer";
excludedColumnElement.HierarchyName = "Customer Geography";
excludedColumnElement.AddLevel("Customer Geography", "Country");
excludedColumnElement.Hierarchy.LevelElements["Country"].Add("Canada");
excludedColumnElement.Hierarchy.LevelElements["Country"].Add("France");
excludedColumnElement.Hierarchy.LevelElements["Country"].Add("United Ki
ngdom");
excludedColumnElement.Hierarchy.LevelElements["Country"].Add("United St
ates");
```

```
//Spefifying the Excluded row elements
DimensionElement excludedRowElement = new DimensionElement();
excludedRowElement.Name = "Date";
excludedRowElement.AddLevel("Fiscal", "Fiscal Year");
excludedRowElement.AddLevel("Fiscal", "Month");
excludedRowElement.AddLevel("Fiscal", "Fiscal Quarter");
excludedRowElement.AddLevel("Fiscal", "Fiscal Semester");
excludedRowElement.Hierarchy.LevelElements["Fiscal Year"].Add("FY 2002"
);
excludedRowElement.Hierarchy.LevelElements["Fiscal Year"].Add("FY 2004"
);
excludedRowElement.Hierarchy.LevelElements["Fiscal Year"].Add("FY 2005"
);
excludedRowElement.Hierarchy.LevelElements["Fiscal Semester"].Add("H2 F
Y 2003");
excludedRowElement.Hierarchy.LevelElements["Month"].Add("August 2002");
excludedRowElement.Hierarchy.LevelElements["Month"].Add("September 2002
");
excludedRowElement.Hierarchy.LevelElements["Fiscal Quarter"].Add("Q2 FY
 2003");
excludedRowElement.Hierarchy.LevelElements["Fiscal Quarter"].Add("Q2 FY
 2003");

///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn,excludedColum
nElement);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow,excludedRowElement);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()

olapReport.Name = "Customer Report"

olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()

'Specifying the Name for the Dimension Element

dimensionElementColumn.Name = "Customer"

'Specifying the Hierarchy Name

dimensionElementColumn.HierarchyName = "Customer Geography"

dimensionElementColumn.AddLevel("Customer Geography", "Country")
```



```
Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement { Name = "Internet Sales Amount" });
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.HierarchyName = "Fiscal"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

'Specifying Excluded column elements
Dim excludedColumnElement As DimensionElement = New DimensionElement()
excludedColumnElement.Name = "Customer"
excludedColumnElement.HierarchyName = "Customer Geography"
excludedColumnElement.AddLevel("Customer Geography", "Country")
excludedColumnElement.Hierarchy.LevelElements("Country").Add("Canada")
excludedColumnElement.Hierarchy.LevelElements("Country").Add("France")
excludedColumnElement.Hierarchy.LevelElements("Country").Add("United Kingdom")
excludedColumnElement.Hierarchy.LevelElements("Country").Add("United States")



'Spefifying the Excluded row elements
Dim excludedRowElement As DimensionElement = New DimensionElement()
excludedRowElement.Name = "Date"
excludedRowElement.AddLevel("Fiscal", "Fiscal Year")
excludedRowElement.AddLevel("Fiscal", "Month")
excludedRowElement.AddLevel("Fiscal", "Fiscal Quarter")
excludedRowElement.AddLevel("Fiscal", "Fiscal Semester")
excludedRowElement.Hierarchy.LevelElements("Fiscal Year").Add("FY 2002")
excludedRowElement.Hierarchy.LevelElements("Fiscal Year").Add("FY
```



```
2004")

excludedRowElement.Hierarchy.LevelElements("Fiscal Year").Add("FY
2005")

excludedRowElement.Hierarchy.LevelElements("Fiscal Semester").Add("H2
FY 2003")

excludedRowElement.Hierarchy.LevelElements("Month").Add("August 2002")

excludedRowElement.Hierarchy.LevelElements("Month").Add("September
2002")

excludedRowElement.Hierarchy.LevelElements("Fiscal Quarter").Add("Q2 FY
2003")

excludedRowElement.Hierarchy.LevelElements("Fiscal Quarter").Add("Q2 FY
2003")


'''Adding Column Members

olapReport.CategoricalElements.Add(dimensionElementColumn,excludedColum
nElement)

'''Adding Measure Element

olapReport.CategoricalElements.Add(measureElementColumn)

'''Adding Row Members

olapReport.SeriesElements.Add(dimensionElementRow,excludedRowElement)
```

4.3.11.1.4 Ordered Report

```
[C#]


OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();

//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
//Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

//Creating Measure Elements
MeasureElements measureElementColumn = new MeasureElements();
```

```
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
 Sales Amount" });
DimensionElement dimensionElementRow = new DimensionElement();

//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

SortElement sortElement = new SortElement(AxisPosition.Categorical, Sor
tOrder.BDESC, true);
sortElement.Element.UniqueName = "[Measures].[Internet Sales Amount]";

///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
//Adding Sort Element
olapReport.CategoricalElements.Add(sortElement);
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()

olapReport.Name = "Customer Report"

olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()


'Specifying the Name for the Dimension Element

dimensionElementColumn.Name = "Customer"

'Specifying the Hierarchy Name

dimensionElementColumn.HierarchyName = "Customer Geography"

dimensionElementColumn.AddLevel("Customer Geography", "Country")


'Creating Measure Elements

Dim measureElementColumn As MeasureElements = New MeasureElements()

measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
```

```
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

Dim sortElement As SortElement = New SortElement(AxisPosition.Categorical,
SortOrder.BDESC, True)
sortElement.Element.UniqueName = "[Measures].[Internet Sales Amount]"

'''Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
'''Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn)
'Adding Sort Element
olapReport.CategoricalElements.Add(sortElement)
'''Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.5 Report with Filter

```
[C#]

OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

//Creating Measure Element
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name =
"Internet Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
```



```
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

FilterElement filterElement = new FilterElement(AxisPosition.Categorica
l);
filterElement.Elements.Add(measureElementColumn);
filterElement.Elements.Add(dimensionElementColumn);
filterElement.FilterCase = FilterCase.GreaterThan;
filterElement.FilterValue.Add(new MeasureElement { Name =


"Internet Sales Amount", Visible = true });
filterElement.FilterValue.Add(new FilterValue { Filter Value = 2700000.
00 });
filterElement.IsFilterCondition = true;
/// Adding Column Members
olapReport.CategoricalElements.Add(new Item { ElementValue =

dimensionElementColumn });
olapReport.CategoricalElements.IsFilterOrSortOn = true;
olapReport.FilterElements.Add(new Item { ElementValue = filterElement }
);

olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()

olapReport.Name = "Customer Report"

olapReport.CurrentCubeName = "Adventure Works"


Dim dimensionElementColumn As DimensionElement = New DimensionElement()

'Specifying the Name for the Dimension Element

dimensionElementColumn.Name = "Customer"

dimensionElementColumn.AddLevel("Customer Geography", "Country")


'Creating Measure Element

Dim measureElementColumn As MeasureElements = New MeasureElements()

measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})


Dim dimensionElementRow As DimensionElement = New DimensionElement()
```



```
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")


Dim filterElement As FilterElement = New
FilterElement(AxisPosition.Categorical)
filterElement.Elements.Add(measureElementColumn)
filterElement.Elements.Add(dimensionElementColumn)
filterElement.FilterCase = FilterCase.GreaterThan


filterElement.FilterValue.Add(New MeasureElement With {.Name = "Internet
Sales Amount", .Visible = True})


filterElement.FilterValue.Add(New FilterValue With {.Filter Value =
2700000.00})
filterElement.IsFilterCondition = True
olapReport.CategoricalElements.Add(New Item With {.ElementValue =
dimensionElementColumn})
olapReport.CategoricalElements.IsFilterOrSortOn = True
olapReport.FilterElements.Add(New Item With {.ElementValue = filterElement})
```

4.3.11.1.6 Report with subset

```
[C#]


OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
//Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
```

```
 Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

//Specifying the SubsetElement
//Specify the start index and end index to retrieve the records.
SubsetElement subSetElementColumn = new SubsetElement(5);
subSetElementColumn.Name = "Top 5 Elements";

SubsetElement subSetElementRow = new SubsetElement(3);
subSetElementRow.Name = "Top 3 Elements";

///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.CategoricalElements.SubSetElement = subSetElementColumn;
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
olapReport.SeriesElements.SubSetElement = subSetElementRow;
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

'Creating Measure Element
Dim olapReport As OlapReport = New OlapReport()
olapReport.CurrentCubeName = "Adventure Works"
Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
```



```
dimensionElementColumn.Name = "Customer"
'Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

'Specifying the SubsetElement
'Specify the start index and end index to retrieve the records.
Dim subSetElementColumn As SubsetElement = New SubsetElement(5)
subSetElementColumn.Name = "Top 5 Elements"

Dim subSetElementRow As SubsetElement = New SubsetElement(3)
subSetElementRow.Name = "Top 3 Elements"

'''Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
'''Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn)
olapReport.CategoricalElements.SubSetElement = subSetElementColumn
'''Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
olapReport.SeriesElements.SubSetElement = subSetElementRow
```

4.3.11.1.7 Drill down report

[C#]

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
//Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name =
                                  "Internet Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.HierarchyName = "Fiscal";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].Add("FY 2002");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
                            MemberElements[0].ShowChildMembers = true;
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
                            MemberElements[0].Add("H1 FY 2002");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
      MemberElements[0].ChildMemberElements[0].ShowChildMembers = true;
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
MemberElements[0].ChildMemberElements[0].Add("Q1 FY 2002");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
MemberElements[0].ChildMemberElements[0].
ChildMemberElements[0].ShowChildMembers = true;
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
MemberElements[0].ChildMemberElements[0].
ChildMemberElements[0].Add("July 2001");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].
MemberElements[0].ChildMemberElements[0].
ChildMemberElements[0].ChildMemberElements[0].ShowChildMembers = true;
```

[VB]


```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

'Creating Measure Element
Dim olapReport As OlapReport = New OlapReport()
olapReport.CurrentCubeName = "Adventure Works"
Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
'Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography"
Dim olapReport As OlapReport = New OlapReport()
olapReport.CurrentCubeName = "Adventure Works"
Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
'Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.HierarchyName = "Fiscal"
```

```
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").Add("FY 2002")

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ShowChildMembers = True

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).Add("H1 FY 2002")

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ChildMemberElements(0).ShowChildMembers = True

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ChildMemberElements(0).Add("Q1 FY 2002")

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ChildMemberElements(0).
ChildMemberElements(0).ShowChildMembers = True

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ChildMemberElements(0). ChildMemberElements(0).Add("July
2001")

dimensionElementRow.Hierarchy.LevelElements("Fiscal Year").
MemberElements(0).ChildMemberElements(0).
ChildMemberElements(0).ChildMemberElements(0).ShowChildMembers = True
```

4.3.11.1.8 Report with Top count Filter

```
[C#]


OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

//Creating Measure Element
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name =
                                                "Internet Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
```



```
//Filter the top 5 elements of "Internet Sales Amount".
TopCountElement topCountElement = new
TopCountElement(AxisPosition.Categorical, 5);
topCountElement.MeasureName = "Internet Sales Amount";

/// Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(topCountElement);
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

'Creating Measure Element
Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

'Filter the top 5 elements of "Internet Sales Amount".
```



```
Dim topCountElement As TopCountElement = New
TopCountElement(AxisPosition.Categorical, 5)

topCountElement.MeasureName = "Internet Sales Amount"


''' Adding Column Members

olapReport.CategoricalElements.Add(dimensionElementColumn)

'''Adding Measure Element

olapReport.CategoricalElements.Add(measureElementColumn)

'''Adding Measure Element

olapReport.CategoricalElements.Add(topCountElement)

'''Adding Row Members

olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.9 Report with Named set

```
[C#]


OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the dimension Name
dimensionElementColumn.Name = "Customer";
//Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
//Specifying the measure elements
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
 Sales Amount" });

//Specifying the NamedSet Element
NamedSetElement dimensionElementRow = new NamedSetElement();
//Specifying the dimension name
dimensionElementRow.Name = "Core Product Group";

///Adding the Column members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding the Measure elements
olapReport.CategoricalElements.Add(measureElementColumn);
///Adding the Row members
```



```
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the dimension Name
dimensionElementColumn.Name = "Customer"
'Specifying the Hierarchy Name
dimensionElementColumn.HierarchyName = "Customer Geography"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim measureElementColumn As MeasureElements = New MeasureElements()
'Specifying the measure elements
measureElementColumn.Elements.Add(New MeasureElement With {.Name =
                                        "Internet Sales Amount"})

'Specifying the NamedSet Element
Dim dimensionElementRow As NamedSetElement = New NamedSetElement()
'Specifying the dimension name
dimensionElementRow.Name = "Core Product Group"

'''Adding the Column members
olapReport.CategoricalElements.Add(dimensionElementColumn)
'''Adding the Measure elements
olapReport.CategoricalElements.Add(measureElementColumn)
'''Adding the Row members
olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.10 Report with calculated member

[C#]

```
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.HierarchyName = "Customer Geography";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

DimensionElement internalDimension = new DimensionElement();
internalDimension.Name = "Product";
internalDimension.AddLevel("Product Categories", "Category");

//// Calculated Measure
CalculatedMember calculatedMeasure1 = new CalculatedMember();
calculatedMeasure1.Name = "Oder on Discount";
calculatedMeasure1.Expression = "[Measures].[Order Quantity] + ([Measur
es].[Order Quantity] * 0.10)";
calculatedMeasure1.AddElement(new MeasureElement { Name = "Order Quanti
ty" });

//// Calculated Measure
CalculatedMember calculatedMeasure2 = new CalculatedMember();
calculatedMeasure2.Name = "Sales Range";
calculatedMeasure2.AddElement(new MeasureElement { Name = "Sales Amount
" });
calculatedMeasure2.Expression = "IIF([Measures].[Sales Amount]>200000,\
"High\",\"Low\")";

// Calculated Dimension
CalculatedMember calculateDimension = new CalculatedMember();
calculateDimension.Name = "Bikes & Components";
calculateDimension.Expression = "([Product].[Product Categories].[Categ
ory].[Bikes] + [Product].[Product Categories].[Category].[Components] )
";
calculateDimension.AddElement(internalDimension);

MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Sales Am
ount" });
measureElementColumn.Elements.Add(new MeasureElement { Name = "Order Qu
antity" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
```

```
//// Adding Calculated members in calculated member collection
olapReport.CalculatedMembers.Add(calculatedMeasure1);
olapReport.CalculatedMembers.Add(calculateDimension);
olapReport.CalculatedMembers.Add(calculatedMeasure2);

/// Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(calculateDimension);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.CategoricalElements.Add(calculatedMeasure1);
olapReport.CategoricalElements.Add(calculatedMeasure2);

///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim dimensionElementColumn As DimensionElement = New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.HierarchyName = "Customer Geography"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim internalDimension As DimensionElement = New DimensionElement()
internalDimension.Name = "Product"
internalDimension.AddLevel("Product Categories", "Category")

'// Calculated Measure
Dim calculatedMeasure1 As CalculatedMember = New CalculatedMember()
calculatedMeasure1.Name = "Oder on Discount"
calculatedMeasure1.Expression = "[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)"
calculatedMeasure1.AddElement(New MeasureElement With {.Name = "Order
Quantity"})
```



```
'// Calculated Measure
Dim calculatedMeasure2 As CalculatedMember = New CalculatedMember()
calculatedMeasure2.Name = "Sales Range"
calculatedMeasure2.AddElement(New MeasureElement With {.Name = "Sales
Amount"})
calculatedMeasure2.Expression = "IIF([Measures].[Sales
Amount]>200000,""High"",""Low"")"

' Calculated Dimension
Dim calculateDimension As CalculatedMember = New CalculatedMember()
calculateDimension.Name = "Bikes & Components"
calculateDimension.Expression = "([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )"
calculateDimension.AddElement(internalDimension)

Dim measureElementColumn As MeasureElements = New MeasureElements()
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Sales
Amount"})
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Order
Quantity"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")



'// Adding Calculated members in calculated member collection
olapReport.CalculatedMembers.Add(calculatedMeasure1)
olapReport.CalculatedMembers.Add(calculateDimension)
olapReport.CalculatedMembers.Add(calculatedMeasure2)

''' Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
olapReport.CategoricalElements.Add(calculateDimension)
```



```
'''Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn)
olapReport.CategoricalElements.Add(calculatedMeasure1)
olapReport.CategoricalElements.Add(calculatedMeasure2)

'''Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.11 Report with KPI Element

[C#]

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Products Sales Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
// Specifying the name for Dimension Element for Column
dimensionElementColumn.Name = "Product";
dimensionElementColumn.AddLevel("Product Categories", "Category");
dimensionElementColumn.Hierarchy.LevelElements["Category"].Add(this.pro
ductName);
dimensionElementColumn.Hierarchy.LevelElements["Category"].IncludeAvail
ableMembers = true;

MeasureElements measureElementColumn = new MeasureElements();
// Specifying the name for Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Gross Pr
ofit" });

DimensionElement dimensionElementRow = new DimensionElement();
// Specifying the Name for Row Dimension Element
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

KpiElements kpiElement = new KpiElements();
kpiElement.Elements.Add(new KpiElement { Name = "Revenue", ShowKPIStatu
s = true, ShowKPIGoal = false, ShowKPITrend = true, ShowKPIValue = true
 });

// Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(kpiElement);
```

```
// Adding Measure Elements
olapReport.CategoricalElements.Add(measureElementColumn);
// Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Products Sales Report"
olapReport.CurrentCubeName = "Adventure Works"

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
' Specifying the name for Dimension Element for Column
dimensionElementColumn.Name = "Product"
dimensionElementColumn.AddLevel("Product Categories", "Category")
dimensionElementColumn.Hierarchy.LevelElements("Category").Add(Me.produ
ctName)
dimensionElementColumn.Hierarchy.LevelElements("Category").IncludeAvail
ableMembers = True

Dim measureElementColumn As MeasureElements = New MeasureElements()
' Specifying the name for Measure Element
measureElementColumn.Elements.Add(New MeasureElement With {.Name =
"Gross Profit"})

Dim dimensionElementRow As DimensionElement = New DimensionElement()
' Specifying the Name for Row Dimension Element
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

Dim kpiElement As KpiElements = New KpiElements()
kpiElement.Elements.Add(New KpiElement With {.Name = "Revenue",
.ShowKPIStatus = True, .ShowKPIGoal = False, .ShowKPITrend = True,
.ShowKPIValue = True})
```



```
' Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
olapReport.CategoricalElements.Add(kpiElement)
' Adding Measure Elements
olapReport.CategoricalElements.Add(measureElementColumn)
' Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
```

4.3.11.1.12 Report with member properties

[C#]

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Employee Sales Report";
// Specifying the current cube name
olapReport.CurrentCubeName = "Adventure Works";

MeasureElements measureElementColumn = new MeasureElements();
// Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name =


"Sales Amount Quota" });

DimensionElement dimensionElementRow = new DimensionElement();
// Specifying the Dimension Name
dimensionElementRow.Name = "Employee";
// Specifying the Hierarchy and level name for the Dimension Element
dimensionElementRow.AddLevel("Employees", "Employee Level 02");
dimensionElementRow.Hierarchy.LevelElements["Employee Level 02"].
                                    IncludeAvailableMembers = true;


// Adding the Member properties to the Dimension Element
dimensionElementRow.MemberProperties.Add(new MemberProperty("Title",


"[Employee].[Employees].[Title]"));
dimensionElementRow.MemberProperties.Add(new MemberProperty("Phone",


"[Employee].[Employees].[Phone]"));
dimensionElementRow.MemberProperties.Add(new MemberProperty("Email Addr
ess",
```

```
"[Employee].[Employees].[Email Address]"));

DimensionElement dimensionElementColumn = new DimensionElement();
// Specifying the Dimension Name
dimensionElementColumn.Name = "Product";
// Specifying the Hierarchy and level name for the Dimension Element
dimensionElementColumn.AddLevel("Product Categories", "Category");
dimensionElementColumn.MemberProperties.Add(new MemberProperty("Dealer
Price", "[Product].[Product Categories].[Dealer Price]"));
dimensionElementColumn.MemberProperties.Add(new MemberProperty("Standar
d Cost", "[Product].[Product Categories].[Standard Cost]"));

// Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);

///Adding Column Members
olapReport.CategoricalElements.Add(measureElementColumn);
```

[VB]

```
Dim olapReport As OlapReport = New OlapReport()

olapReport.Name = "Employee Sales Report"

' Specifying the current cube name

olapReport.CurrentCubeName = "Adventure Works"


Dim measureElementColumn As MeasureElements = New MeasureElements()

' Specifying the Name for the Measure Element

measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Sales
Amount Quota"})


Dim dimensionElementRow As DimensionElement = New DimensionElement()

' Specifying the Dimension Name

dimensionElementRow.Name = "Employee"

' Specifying the Hierarchy and level name for the Dimension Element

dimensionElementRow.AddLevel("Employees", "Employee Level 02")

dimensionElementRow.Hierarchy.LevelElements("Employee Level 02").
```



```
IncludeAvailableMembers = True



' Adding the Member properties to the Dimension Element
dimensionElementRow.MemberProperties.Add(New MemberProperty("Title",
"[Employee].[Employees].[Title]"))
dimensionElementRow.MemberProperties.Add(New MemberProperty("Phone",
"[Employee].[Employees].[Phone]"))
dimensionElementRow.MemberProperties.Add(New MemberProperty("Email Address",
"[Employee].[Employees].[Email Address]"))

Dim dimensionElementColumn As DimensionElement = New DimensionElement()
' Specifying the Dimension Name
dimensionElementColumn.Name = "Product"
' Specifying the Hierarchy and level name for the Dimension Element
dimensionElementColumn.AddLevel("Product Categories", "Category")
dimensionElementColumn.MemberProperties.Add(New MemberProperty("Dealer
Price", "[Product].[Product Categories].[Dealer Price]"))
dimensionElementColumn.MemberProperties.Add(New MemberProperty("Standard
Cost", "[Product].[Product Categories].[Standard Cost]"))

' Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)

'''Adding Column Members
olapReport.CategoricalElements.Add(measureElementColumn)
```

4.3.11.2 Sample Report for Non-OLAP data

The following code snippet illustrates the sample OlapReport for an OlapDataManager:

```
[C#]


OlapReport olapReport = new OlapReport();
olapReport.Name = "Sales Report";

// Specifying the Row Dimension Element
DimensionElement dimensionElementRow = new DimensionElement();
```

```
dimensionElementRow.Name = "Geography";
dimensionElementRow.Hierarchy = new HierarchyElement() { Name = "Produc
t Hierarchy" };

dimensionElementRow.Hierarchy.LevelElements.Add(new LevelElement() { Na
me = "Product" });
dimensionElementRow.Hierarchy.LevelElements.Add(new LevelElement() { Na
me = "Date" });

// Specifying the Column Dimension Element
DimensionElement dimensionElementColumn = new DimensionElement();
dimensionElementColumn.Name = "Geography";
dimensionElementColumn.Hierarchy = new HierarchyElement() { Name = "Geo
graphy Hierarchy" };

dimensionElementColumn.Hierarchy.LevelElements.Add(

new LevelElement() { Name = "Country" });
dimensionElementColumn.Hierarchy.LevelElements.Add(

new LevelElement() { Name = "State" });

// Specifying the Summary Elements
SummaryElements summaries = new SummaryElements();
summaries.Add(new SummaryInfo { Column = "Quantity", Key = "Quantity",
Type = SummaryType.Sum });
summaries.Add(new SummaryInfo { Column = "Amount", Key = "Amount", Type
 = SummaryType.Sum, FormatString = "{0:c}" });

// Adding the Row Elements

olapReport.SeriesElements.Add(new Item { ElementValue = summaries });
```

```
[VB]


Dim olapReport As OlapReport = New OlapReport()
olapReport.Name = "Sales Report"


' Specifying the Row Dimension Element
Dim dimensionElementRow As DimensionElement = New DimensionElement()
dimensionElementRow.Name = "Geography"
dimensionElementRow.Hierarchy = New HierarchyElement() With {.Name = "P
roduct Hierarchy"}

dimensionElementRow.Hierarchy.LevelElements.Add(

New LevelElement() With {.Name = "Product"})
dimensionElementRow.Hierarchy.LevelElements.Add(

New LevelElement() With {.Name = "Time"})
```

```
' Specifying the Column Dimension Element
Dim dimensionElementColumn As DimensionElement = New DimensionElement()
dimensionElementColumn.Name = "Geography"
dimensionElementColumn.Hierarchy = New HierarchyElement() With {.Name =
 "Geography Hierarchy"}

dimensionElementColumn.Hierarchy.LevelElements.Add(New LevelElement() W
ith {.Name = "Country"})
dimensionElementColumn.Hierarchy.LevelElements.Add(New LevelElement() W
ith {.Name = "State"})

' Specifying the Summary Elements
Dim summaries As SummaryElements = New SummaryElements()
summaries.Add(New SummaryInfo With {.Column = "Quantity", .Key = "Quant
ity", .Type = SummaryType.Sum})
summaries.Add(New SummaryInfo With {.Column = "Amount", .Key = "Amount"
, .Type = SummaryType.Sum, .FormatString = "{0:c}"})

' Adding the Row Elements
olapReport.SeriesElements.Add(New Item With {.ElementValue = summaries}
)
```

4.3.12 Binding the OlapReport to OlapDataManager

Once the report is created, the report is set to the current report property of the OlapDataManager. Further data processing, query creation and cell set creation starts from the OlapDataManager.

Binding the OlapReport in OlapDataManager.

4.3.13 Paging

Paging enables the user to view large records by breaking them into smaller segments.

Paging feature can be achieved by setting the *EnablePaging* property to *true* in a report.

The following code snippet demonstrates how to enable paging in current report:

[C#]

```
olapDataManager.CurrentReport.EnablePaging = true;
```

[VB]

```
olapDataManager.CurrentReport.EnablePaging = True
```

The user can customize the page settings such as current page, page size (for both row and column).

The following code explains how to customize current page and page size settings:

```
[C#]
olapDataManager.CurrentReport.PagerOptions.CategorialCurrentPage = 1;
olapDataManager.CurrentReport.PagerOptions.SeriesCurrentPage = 2;
olapDataManager.CurrentReport.PagerOptions.CategorialPageSize = 50;
olapDataManager.CurrentReport.PagerOptions.SeriesPageSize = 50;
```

```
[VB]
olapDataManager.CurrentReport.PagerOptions.CategorialCurrentPage = 1
olapDataManager.CurrentReport.PagerOptions.SeriesCurrentPage = 2
olapDataManager.CurrentReport.PagerOptions.CategorialPageSize = 50
olapDataManager.CurrentReport.PagerOptions.SeriesPageSize = 50
```

4.3.14 Drill Position

Drill position enables the user to drill the current position of a selected member in the OlapReport. This will exclude the drilled data of the selected member in other positions by using MDX query.

The following code illustrates how to achieve drill position support in current report:

```
[C#]
olapDataManager.CurrentReport.DrillType = DrillType.DrillPosition;
```

```
[VB]
olapDataManager.CurrentReport.DrillType = DrillType.DrillPosition
```

4.3.15 Drill Replace

Drill Replace displays only the immediate child members and ancestors on drill-down and drill-up respectively.

The following code illustrates how to achieve drill replace support in a current report:

```
[C#]
olapDataManager.CurrentReport.DrillType = DrillType.DrillReplace;
```

```
[VB]
olapDataManager.CurrentReport.DrillType = DrillType.DrillReplace
```

4.3.16 MDX Query Parsing

4.3.16.1 MDX Query binding with drill up and drill down operations

This feature provides support for the drill up and drill down operations for BI components such as BI Grid, BI Chart, and BI Client in order to view the OLAP data through different levels using an MDX query instead of the OlapReport class. Previously, the drill up and drill down operations were supported by the OlapReport class only.

Use Case Scenarios

This feature is useful when the user wants to perform drill up or drill down operations with the OLAP data using an MDX query instead of the OlapReport class.

Properties

Table 11: Property Table

Property	Description	Type	Data Type	Reference links
AllowMdxToReportParse	Gets or sets a value indicating whether to parse the given MDX into the OlapReport class or not. The default value is true.	CLR	Boolean	N/A

Sample Links

OlapGrid [WPF]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<version>\WPF\OlapGrid.WPF\Samples\Defining Reports\MDX Query Demo

OlapChart [WPF]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<version>\WPF\OlapChart.WPF\Samples\Defining Reports\MDX Query Demo

OlapClient [WPF]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<Version>\WPF\OlapClient.WPF\Samples\Product Showcase\MDX Query Demo

OlapGrid [Silverlight]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<Version>\BI\Silverlight\OlapGrid.Silverlight\ReportDefinition\GridMDXQueryDemo

OlapChart [Silverlight]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<Version>\BI\Silverlight\OlapChart.Silverlight\DefiningReports\ChartMDXQueryDemo

OlapClient [Silverlight]

<InstalledDrive>:\Users\<UserName>\AppData\Local\Syncfusion\EssentialStudio\<Version>\BI\Silverlight\OlapClient.Silverlight\ProductShowcase\MDXQueryDemo

4.3.16.2 Adding MDX Query binding with drill up and drill down operations to an Application

The following code samples are used to enable and disable the MDX to OLAP parsing and processing of an MDX query into OLAP data.

[C#]

```
//Enable MDX to OLAP parsing.
        OlapDataManager olapDataManager = new OlapDataManager();
        olapDataManager.MdxQuery = "MDX Query Here";
        olapDataManager.ExecuteCellSet();

//Disable MDX to OLAP parsing.
        OlapDataManager olapDataManager = new OlapDataManager();
        olapDataManager.AllowMdxToOlapReportParse = false;
        olapDataManager.MdxQuery = "MDX Query Here";
        olapDataManager.ExecuteCellSet();
```

[VB]

```
'Enable MDX to OLAP parsing.
        Dim olapDataManager As New OlapDataManager()
        olapDataManager.MdxQuery = "MDX Query Here"
        olapDataManager.ExecuteCellSet()

'Enable MDX to OLAP parsing.
        Dim olapDataManager As New OlapDataManager()
        olapDataManager.AllowMdxToOlapReportParse = False
```



```
olapDataManager.MdxQuery = "MDX Query Here"
olapDataManager.ExecuteCellSet()
```

4.3.17 Virtual KPI Element

Key performance indicators can be virtually defined during run time. This feature enables users to create KPIs without storing them in SSAS (SQL Server Analysis Services). This feature is very useful when users want to define KPIs at run time and also minimize the time necessary to create KPIs.

4.3.17.1 Properties

Table 12: VirtualKpiElement Class Properties

Property	Description	Type	Data Type
Name	Gets or sets the name for VirtualKpiElement.	CLR	String
KpiValueExpression	Gets or sets the value which indicates the value expression for VirtualKpiElement.	CLR	String
KpiGoalExpression	Gets or sets the value which indicates the goal expression for VirtualKpiElement.	CLR	String
KpiStatusExpression	Gets or sets the value which indicates the status expression for VirtualKpiElement.	CLR	String
KpiTrendExpression	Gets or sets the value which indicates the trend expression for VirtualKpiElement.	CLR	String
KpiTrendGraphic	Gets or sets the name of the graphic used to represent the result of the trend expression.	CLR	String
KpiStatusGraphic	Gets or sets the name of the graphic used to represent the result of the status expression.	CLR	String

Table 13: OlapReport Class Properties

Property	Description	Type	Data Type
VirtualKpiElements	Gets or sets the collection of VirtualKpiElement.	CLR	Items

4.3.17.2 Adding Virtual KPI Element to the OlapReport

OLAP Report Definition with VirtualKpiElement

```
[C#]
        public OlapReport VirtualKPIReport()
        {
            OlapReport olapReport = new OlapReport("Virtual KPI Report");
            olapReport.CurrentCubeName = "Adventure Works";
            MeasureElements measureElements = new MeasureElements();
            measureElements.Add(new MeasureElement { Name = "Internet Sales
Amount" });
            olapReport.CategoricalElements.Add(measureElements);

            VirtualKpiElement virtualKPIElement = new VirtualKpiElement();
            virtualKPIElement.Name = "Sample KPI";
            virtualKPIElement.KpiValueExpression = ""; //Value expression
            virtualKPIElement.KpiGoalExpression = ""; //Goal expression
            virtualKPIElement.KpiStatusExpression = ""; //Status expression
            virtualKPIElement.KpiTrendExpression = ""; //Trend expression
            virtualKPIElement.StatusGraphic = ""; //Status graphic
            virtualKPIElement.TrendGraphic = ""; //Trend graphic
            olapReport.VirtualKpiElements.Add(virtualKPIElement);
            olapReport.CategoricalElements.Add(virtualKPIElement);

            DimensionElement internalDimension = new DimensionElement();
            internalDimension.Name = "Product";
            internalDimension.AddLevel("Product Categories", "Category");
            olapReport.SeriesElements.Add(internalDimension);
            return olapReport;
        }
```

[VB]

```
       Public Function VirtualKPIReport() As OlapReport
                  Dim olapReport As New OlapReport("Virtual KPI
Report")
                  olapReport.CurrentCubeName = "Adventure Works"
                  Dim measureElements As New MeasureElements()
                  measureElements.Add(New MeasureElement With {.Name =
"Internet Sales Amount"})
                  olapReport.CategoricalElements.Add(measureElements)

                  Dim virtualKPIElement As New VirtualKpiElement()
                  virtualKPIElement.Name = "Sample KPI"
                  virtualKPIElement.KpiValueExpression = "" 'Value
expression
                  virtualKPIElement.KpiGoalExpression = "" 'Goal
expression
                  virtualKPIElement.KpiStatusExpression = "" 'Status
expression
                  virtualKPIElement.KpiTrendExpression = "" 'Trend
expression
                  virtualKPIElement.StatusGraphic = "" 'Status graphic
                  virtualKPIElement.TrendGraphic = "" 'Trend graphic
                  olapReport.VirtualKpiElements.Add(virtualKPIElement)
                  olapReport.CategoricalElements.Add(virtualKPIElement)

                  Dim internalDimension As New DimensionElement()
                  internalDimension.Name = "Product"
                  internalDimension.AddLevel("Product Categories",
"Category")
                  olapReport.SeriesElements.Add(internalDimension)
                  Return olapReport
End Function
```


Sample of Value, Goal, Status, and Trend expressions

```
[Value Expression]
            [Measures].[Reseller Sales Amount]
```

```
[Goal Expression]
            [Measures].[Sales Amount Quota]
```

```
[Status Expression]
Case
    When [Measures].[Reseller Sales Amount] / [Measures].[Sales Amount
Quota] >  1
    Then 1
    When [Measures].[Reseller Sales Amount] / [Measures].[Sales Amount
Quota] <= 1
          And
          [Measures].[Reseller Sales Amount] / [Measures].[Sales Amount
Quota] >= .85
    Then 0
    Else -1
End
```

```
[Trend Expression]
Case
    When IsEmpty
          (
            ParallelPeriod
            (
              [Date].[Fiscal].[Fiscal Year],
```

```
            1,
            [Date].[Fiscal].CurrentMember
          )
        )
    Then 0
    When VBA!Abs
        (
          (
            [Measures].[Reseller Sales Amount]
            -
            (
              [Measures].[Reseller Sales Amount],
              ParallelPeriod
              (
                [Date].[Fiscal].[Fiscal Year],
                1,
                [Date].[Fiscal].CurrentMember
              )
            )
          )
          /
          (
            [Measures].[Reseller Sales Amount],
            ParallelPeriod
            (
              [Date].[Fiscal].[Fiscal Year],
              1,
              [Date].[Fiscal].CurrentMember
            )
          )
        ) <=.02
    Then 0
```

```
    When (
            [Measures].[Reseller Sales Amount]
            -
            (
              [Measures].[Reseller Sales Amount],
              ParallelPeriod
              (
                [Date].[Fiscal].[Fiscal Year],
                1,
                [Date].[Fiscal].CurrentMember
              )
            )
          )
          /
          (
            [Measures].[Reseller Sales Amount],
            ParallelPeriod
            (
              [Date].[Fiscal].[Fiscal Year],
              1,
              [Date].[Fiscal].CurrentMember
            )
          ) >.02
    Then 1
    Else -1
End
```

4.4 QueryBuilderEngine

This class will generate the query from the MDXQuerySpecification. The query generation starts from invoking the **GenerateQueryEx()** static method of **QueryBuilderEngineVersion3**, the inner class of **QueryBuilderEngine** class.

4.4.1 MDXQuerySpecification

MDXQuerySpecification is the base for query creation. The MDXQuerySpecification will categorize the element into three clauses namely:

- With
- Select and
- Where

The elements in the given report are iterated and stored according to the specification.

- The calculated member element in the given report will be added under the **With** clause.
- The categorical and series items in the given report's categorical elements and series element will come in the **Select** category.
- The **Slicer** and **Filter** items will comes under the **Where** category.

Based on the current report, the **TogglePivot** value and the axis position of each item will be assigned before it is stored in the appropriate clauses.

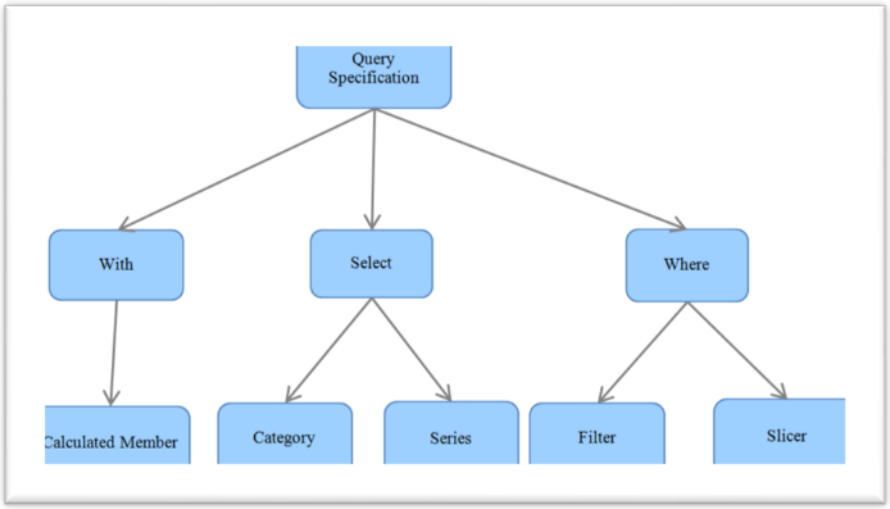

Figure 8: MDXQuerySpecification

Once the query specification is created, the **GenerateQueryEx()** method of QueryBuilderEngine was called by passing the created MDXQuerySpecification to generate the query.

4.4.2 Steps in Query Generation

To generate a query:

1. Get the query specification and iterate through the items in each category (With, Select and Where) of specification and separately store the column elements, row elements, filter elements and slice elements.
2. The filter, slicer and sort elements are moved to the appropriate axis based on their axis property.
3. Once the initial level categorizing of elements is completed, it starts creating a query string by writing using **With** or **Select**.
4. Now it starts writing the query by checking each and every property.

By invoking the **BuildAxisElements()** method, the building query for the column axis elements and row axis elements are done.

The helper methods for the **BuildAxisElements()** are:

- BuildAxisItems
- BuildDimensionElement
- If no level is specified, the **GetDefaultLevel()** method will be called else
- The **BuildHierarchyElement()** will be called
- BuildLevelElements
- If the level member count is greater than zero the **BuildMemberElement()** will be called else the **GetDefaultLevel()** method will be called.

The **BuildAxisElements()** will build the query for the column element when we pass the column items and it will generate the query for the row elements when the row items is passed as an argument. The **BuildAxisElements()** method will return a Boolean value which represents whether the KPI element is existing in the given item list or not. Based on that return, the value of the KPI Element axis was fixed.

Up to this the **Select** section of the query was built and now the **From** section. The **From** section will start by invoking the **BuildFiltereCondition**() method.

The Helper method for the **BuildFiltereCondition()** is given below:

1. **BuildFilteredElements()** - This method iterates through the elements and appends the parent details and child member details in the query based on the axis either in a row or in a column.
2. Then it comes to the **Where** section, by invoking the **BuildSlicerElements**() method.
3. Then the **BuildSlicerItem** is invoked. This method will check whether the given item is Dimension or Measure or KPI or NamedSet or Level and based on this the appropriate query part will appended with the query.

Finally, the Cell properties will append with the created query and the query string will be returned to the **OlapDataManager**. By using the newly generated query, the OlapDataManager will invoke the **ExecuteCellSet** of **DataProvide**, which will return the **CellSet**.

This **CellSet** will be used to create the **PivotEngine**, which will give the input to the controls.

4.5 OLAP Data Processing

The OlapDataManager accepts the input from the user and processes the data based on the input supplied and generates the formatted output which Syncfusion OLAP controls can understand. The OlapDataManager can process two types of data namely:

- OLAP data (Cube)
- Non-OLAP data (Enumerable collection, ITyped List)

4.5.1 Steps in processing OLAP Data

To process the OLAP data:

1. Get the input to establish a connection to the specified data source. The connection information can be given as a connection string that contains information about the data source.

   Example connection string: “DataSource = localhost; Initial Catalog = Adventure Works DW”;
2. Connect to a server or an off-line cube as a data source.
3. Once the connection is established, provide input for data processing. You can give two types of inputs to process the OLAP data. They are:
   - MDX Query - You can write a MDX Query for the database and give that query as an input.
   - OlapReport - You can create an OLAP report and give that report as an input to the OlapDataManager.
4. The output will not differ, based on the input type. The processing method will differ in OLAP base, based on the input type.
5. If MDX query as given as an input then the query will be executed on the connected data base.
6. If **OlapReport** is given as an input:
   a. MDX query specification will be created based on the **OlapReport**.
   b. From the MDX query specification, the MDX query will be generated with the help of **OlapQueryBuilderEngine**.
   c. The generated query will be executed on the connection database.
7. Once the query is executed either from the MDX query input or from the **OlapReport** input, the result is obtained.
8. This result set will be formatted to provide input for the controls. The formatted input should be passed to the controls.
9. The output will be displayed in the controls.

In Silverlight:

1. The OlapDataManager requires a OlapReport and a Virtual Channel in the form of *OlapDataManager*.
2. The *OlapDataManager* is a set to control and in the *DataBind* method, the control will make an asynchronous call with the help of a virtual channel provided in *OlapDataManager.*
3. Now with the help of WCF Service, the control communicates with Olap base to retrieve *CellSet*.
4. Now the Control passes the obtained *CellSet* to the *OlapDataManager* and in turn the *OlapDataManager* returns the *PivotEngine* to the control.
5. The output will reflect in the controls.

4.5.2 Steps in processing Non-OLAP data

To process the Non-OLAP (Such as IEnumerable, IList, DataTable, DataView, etc.) data and provide the specified formatted input to the controls. For the Non-OLAP data no need to establish any connection.

To process the Non-OLAP data:

1. Give the two inputs:

- Item Source
- OlapReport

2. Provide an Item source. The source can be:
    - IEnumerable Collection and
    - ITyped List
3. Once the Item source was bounded, give the **OlapReport**.
4. The given source will be formatted based on the given **OlapReport** and the result set will be passed to the controls.
5. The output will be displayed in the controls.

5 How To

5.1 Establish the connection for an SSAS Server

A valid string is required to establish connection for an OlapDataManager.

Here is the code snippet that demonstrates how to connect SSAS by using connection string:

```
[C#]
OlapDataManager dataManager = new OlapDataManager("DataSource=localhost
; Initial Catalog=Adventure Works DW");
```

```
[VB]
Dim dataManager As New OlapDataManager("DataSource=localhost; Initial
Catalog=Adventure Works DW")
```

Or

```
[C#]
Syncfusion.Olap.DataProvider.IDataProvider dataProvider = new Syncfusio
n.Olap.DataProvider.AdomdDataProvider("DataSource=localhost; Initial Ca
talog=Adventure Works DW");
OlapDataManager dataManager = new OlapDataManager(dataProvider);
```

```
[VB]
Dim dataProvider As Syncfusion.Olap.DataProvider.IDataProvider = New
Syncfusion.Olap.DataProvider.AdomdDataProvider("DataSource=localhost;
Initial Catalog=Adventure Works DW")
Dim dataManager As New OlapDataManager(dataProvider)
```

5.2 Establish the connection for a Cube file

A valid string is required to establish connection for an OlapDataManager.

Here is the code snippet that demonstrates how to connect cube file by using connection string:

```
[C#]
OlapDataManager dataManager = new OlapDataManager("DataSource=
AdventureWorks_Ext.cub; Provider=MSOLAP");
```

```
[VB]
Dim dataManager As New OlapDataManager("DataSource=
AdventureWorks_Ext.cub; Provider=MSOLAP")
```

Or

```
[C#]
```

```
Syncfusion.Olap.DataProvider.IDataProvider dataProvider = new Syncfusio
n.Olap.DataProvider.AdomdDataProvider("DataSource=
AdventureWorks_Ext.cub; Provider=MSOLAP");
OlapDataManager dataManager = new OlapDataManager(dataProvider);
```

```
[VB]
Dim dataProvider As Syncfusion.Olap.DataProvider.IDataProvider = New
Syncfusion.Olap.DataProvider.AdomdDataProvider("DataSource=
AdventureWorks_Ext.cub; Provider=MSOLAP")
Dim dataManager As New OlapDataManager(dataProvider)
```

5.3 Establish Role-based Connection

You can apply Role-based filtering to OLAP components by specifying the appropriate role name designed for specific set of users in the SSAS Cube. You must specify the role name in the "**Roles**" attribute of the connection string. The following code example illustrates this.

```
[C#]

OlapDataManager olapDataManager = new OlapDataManager(@"Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Roles=Role1; Initial
Catalog=Adventure Works DW 2008 SE;");
```

```
[VB]

Dim olapDataManager As OlapDataManager = New OlapDataManager("Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Roles=Role1; Initial
Catalog=Adventure Works DW 2008 SE;")
```

5.4 Connecting to Mondrian Server through XMLA

The following code illustrates how to connect to the Mondrian server:

```
[C#]

// Connecting to Mondrian Server

OlapDataManager DataManager = new OlapDataManager("Data Source =
http://localhost:8080/mondrian/xmla; Initial Catalog = FoodMart;");
//Where localhost is the machine name which has installed Mondrian
Services. For example http://bi.syncfusion.com:8080/mondrian/xmla

DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.Mondrian;
```



```
[VB]

' Connecting to Mondrian Server

Dim DataManager As OlapDataManager = New OlapDataManager("Datasource =
http://bi.syncfusion.com:8080/mondrian/xmla; Initial
Catalog=FoodMart;") 'Where localhost is the machine name which has
installed Mondrian Services. For example
http://bi.syncfusion.com:8080/mondrian/xmla

DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.Mondrian
```

Click here for more information about Mondrian XMLA configurations.

5.5 Connect ActivePivot Server through XMLA

The user can connect the Active Pivot server through XMLA (XML for Analysis) services using the OlapDataManager in our OLAP controls.

The following code illustrates how to connect to Active Pivot server:

```
[C#]

// Connecting to Active Pivot Server

OlapDataManager DataManager = new
OlapDataManager(@"Data Source=http://localhost:8081/var_s/xmla;
Initial Catalog=VaRCubes;
User ID=; Password=; Transport Compression=None;");
DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.ActivePivot;
```

```
[VB]

' Connecting to Active Pivot Server

Dim DataManager As OlapDataManager = New OlapDataManager("Data
Source=http://localhost:8081/var_s/xmla;  Initial Catalog=VaRCubes;
User ID=; Password=; Transport Compression=None;")

DataManager.DataProvider.ProviderName = Syncfusion.Olap.DataProvider.Pr
oviders.ActivePivot
```

Click here for more information about Active Pivot server.

5.6 Create a WCF Service for Silverlight OLAP control

The following steps will define the adding of WCF Service to the Web project and configuring it with IOlapDataProvider:

1. Add WCF Service (from the installed templates) in to the Web project.
2. Inherit the newly added WCF service with IOlapDataProvider and explicitly implement the IOlapDataProvider. The connection to the database is done with the help of WCF service. The service has to be created and instantiated as described below:

The WCF Service has to implement the IOlapDataProvider interface that requires the OlapDataProvider which can be instantiated by passing the connection string.

The code snippet explains how to create OLAP WCF service in Web project:

[C#]

```
[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibility
RequirementsMode.Allowed)]
    [ServiceBehavior(IncludeExceptionDetailInFaults = true)]
    public class OlapManager : IOlapDataProvider
    {
        #region Private variables

        private readonly OlapDataProvider _dataManager;

        #endregion

        #region Constructor
        /// <summary>
        /// Initializes a new instance of the <see cref="OlapManager"/>
 class.
        /// </summary>
        public OlapManager()
        {
            _dataManager = new OlapDataProvider("DataSource=localhost;
Initial Catalog=Adventure Works DW");
        }
        #endregion


#region IOlapDataProvider Members


       public CellSet ExecuteOlapReportWithLevelTypeAll(OlapReport report,
bool showLevelTypeAll)
        {
            CellSet cellSet =
_dataManager.ExecuteOlapReportWithLevelTypeAll(report, showLevelTypeAll);
            _dataManager.DataProvider.CloseConnection();
            return cellSet;
        }


        /// <summary>
        /// Gets the MDX query.
        /// </summary>
        /// <param name="report">The report.</param>
        /// <returns></returns>
```

```
        public string
GetMdxQuery(Syncfusion.OlapSilverlight.Reports.OlapReport report)
        {
            string mdxReturned = this._dataManager.GetMdxQuery(report);
            // Closing the Provider Connection
            this._dataManager.DataProvider.CloseConnection();
            return mdxReturned;
        }

        /// <summary>
        /// Executes the cell set.
        /// </summary>
        /// <param name="report">The report.</param>
        /// <returns></returns>
        public CellSet ExecuteOlapReport(OlapReport report)
        {
            CellSet cellSet = _dataManager.ExecuteOlapReport(report);
            _dataManager.DataProvider.CloseConnection();
            return cellSet;
        }

        /// <summary>
        /// Executes the MDX query.
        /// </summary>
        /// <param name="mdxQuery">The MDX query.</param>
        /// <returns></returns>
        public CellSet ExecuteMdxQuery(string mdxQuery)
        {
            CellSet cellSet = _dataManager.ExecuteMdxQuery(mdxQuery);
            _dataManager.DataProvider.CloseConnection();
            return cellSet;
        }

        /// <summary>
        /// Gets the cubes.
        /// </summary>
        /// <returns></returns>
        public CubeInfoCollection GetCubes()
        {
            CubeInfoCollection cubes = _dataManager.GetCubes();
            _dataManager.DataProvider.CloseConnection();
            return cubes;
        }

        /// <summary>
        /// Gets the cube schema.
        /// </summary>
        /// <param name="cubeName">Name of the cube.</param>
```



```
        /// <returns></returns>
        public CubeSchema GetCubeSchema(string cubeName)
        {
            CubeSchema cubeSchema = _dataManager.GetCubeSchema(cubeName);
            _dataManager.DataProvider.CloseConnection();
            return cubeSchema;
        }

        /// <summary>
        /// Gets the child members.
        /// </summary>
        /// <param name="memberUniqueName">Name of the member
unique.</param>
        /// <param name="cubeName">Name of the cube.</param>
        /// <returns></returns>
        public MemberCollection GetChildMembers(string memberUniqueName,
string cubeName)
        {
            MemberCollection childMembers =
_dataManager.GetChildMembers(memberUniqueName, cubeName);
            _dataManager.DataProvider.CloseConnection();
            return childMembers;
        }

        /// <summary>
        /// Gets the level members.
        /// </summary>
        /// <param name="levelUniqueName">Name of the level unique.</param>
        /// <param name="cubeName">Name of the cube.</param>
        /// <returns></returns>
        public MemberCollection GetLevelMembers(string levelUniqueName,
string cubeName)
        {
            MemberCollection levelMembers =
_dataManager.GetLevelMembers(levelUniqueName, cubeName);
            _dataManager.DataProvider.CloseConnection();
            return levelMembers;
        }

        /// <summary>
        /// Executes the specified MDX query.
        /// </summary>
        /// <param name="mdxQuery">The MDX query.</param>
        /// <returns></returns>
        public object Execute(string mdxQuery)
        {
            var resultSet = this._dataManager.Execute(mdxQuery);
            // Closing the Provider Connection
            _dataManager.DataProvider.CloseConnection();
```

```
            return resultSet;
        }

        /// <summary>
        /// Executes the olap report with total count.
        /// </summary>
        /// <param name="report">The report.</param>
        /// <returns></returns>
        public
Syncfusion.OlapSilverlight.Common.SerializableDictionary<string, object>
ExecuteOlapReportWithTotalCount(OlapReport report)
        {

Syncfusion.OlapSilverlight.Common.SerializableDictionary<string, object>
dict = this._dataManager.ExecuteOlapReportWithTotalCount(report);
            // Closing the Provider Connection
            this._dataManager.DataProvider.CloseConnection();
            return dict;
        }

        #endregion
```

[VB]

```
<AspNetCompatibilityRequirements(RequirementsMode :=
AspNetCompatibilityRequirementsMode.Allowed)> _
<ServiceBehavior(IncludeExceptionDetailInFaults := True)> _
Public Class OlapManager
        Implements IOlapDataProvider

#Region "Member"
       Private ReadOnly _dataManager As OlapDataProvider
#End Region


#Region "Constructor"
        Public Sub New()
                _dataManager = New OlapDataProvider("DataSource=localhost;
Initial Catalog=Adventure Works DW")
        End Sub
#End Region

#Region "IOlapDataProvider Members"

        Public Function ExecuteOlapReportWithLevelTypeAll(ByVal report As
OlapReport, ByVal showLevelTypeAll As Boolean) As CellSet
            Dim cellSet As CellSet =
_dataManager.ExecuteOlapReportWithLevelTypeAll(report, showLevelTypeAll)
            _dataManager.DataProvider.CloseConnection()
            Return cellSet
        End Function
```

```
        Public Function GetMdxQuery(ByVal report As
Syncfusion.OlapSilverlight.Reports.OlapReport) As String
            Dim mdxReturned As String = Me._dataManager.GetMdxQuery(report)
            Me._dataManager.DataProvider.CloseConnection()
            Return mdxReturned
        End Function
        Public Function ExecuteOlapReport(ByVal report As OlapReport) As
CellSet
            Dim cellSet As CellSet = _dataManager.ExecuteOlapReport(report)
            _dataManager.DataProvider.CloseConnection()
            Return cellSet
        End Function
        Public Function ExecuteMdxQuery(ByVal mdxQuery As String) As
CellSet
            Dim cellSet As CellSet = _dataManager.ExecuteMdxQuery(mdxQuery)
            _dataManager.DataProvider.CloseConnection()
            Return cellSet
        End Function
        Public Function GetCubes() As CubeInfoCollection
            Dim cubes As CubeInfoCollection = _dataManager.GetCubes()
            _dataManager.DataProvider.CloseConnection()
            Return cubes
        End Function
        Public Function GetCubeSchema(ByVal cubeName As String) As
CubeSchema
            Dim cubeSchema As CubeSchema =
_dataManager.GetCubeSchema(cubeName)
            _dataManager.DataProvider.CloseConnection()
            Return cubeSchema
        End Function
        Public Function GetChildMembers(ByVal memberUniqueName As String,
ByVal cubeName As String) As MemberCollection
            Dim childMembers As MemberCollection =
_dataManager.GetChildMembers(memberUniqueName, cubeName)
            _dataManager.DataProvider.CloseConnection()
            Return childMembers
        End Function
        Public Function GetLevelMembers(ByVal levelUniqueName As String,
ByVal cubeName As String) As MemberCollection
            Dim levelMembers As MemberCollection =
_dataManager.GetLevelMembers(levelUniqueName, cubeName)
            _dataManager.DataProvider.CloseConnection()
            Return levelMembers
        End Function
        Public Function Execute(ByVal mdxQuery As String) As Object
            Dim resultSet As var = Me._dataManager.Execute(mdxQuery)
            _dataManager.DataProvider.CloseConnection()
            Return resultSet
```



```
        End Function
        Public Function ExecuteOlapReportWithTotalCount(ByVal report As
OlapReport) As Syncfusion.OlapSilverlight.Common.SerializableDictionary(Of
String, Object)
            Dim dict As
Syncfusion.OlapSilverlight.Common.SerializableDictionary(Of String, Object)
= Me._dataManager.ExecuteOlapReportWithTotalCount(report)
            Me._dataManager.DataProvider.CloseConnection()
            Return dict
        End Function
#End Region
#End Class
```

3. Include the custom binding and the service endpoint address in the Web.Config file.

[Web.Config]

```
   <!--Binding-->
   <bindings>
     <customBinding>
       <binding name="binaryHttpBinding">
        <binaryMessageEncoding/>
        <httpTransport maxReceivedMessageSize="2147483647"/>
       </binding>
     </customBinding>
   </bindings>

   <!—Endpoint Address-->
   <services>
     <service name="SilverlightApplication1.Web.OlapManager" >
       <endpoint address="binary" binding="customBinding" bindingConfig
uration="binaryHttpBinding" contract="Syncfusion.OlapSilverlight.Manage
r.IOlapDataProvider">
       </endpoint>
     </service>
   </services>
```

5.7 Connect WCF Service in Silverlight application

The user can connect the above WCF service using Channel Factory by CustomBinding (Or BasicHttpBinding) and End Point Address values.

The below code snippet demonstrates how to connect the WCF service:

[C#]

```
Binding customBinding = new CustomBinding(new BinaryMessageEncodingBind
ingElement(), new HttpTransportBindingElement { MaxReceivedMessageSize
= 2147483647 });
EndpointAddress address = new EndpointAddress(new Uri(App.Current.Host.
Source.ToString() + "../../../../Services/OlapManager.svc/binary"));
ChannelFactory<IOlapDataProvider> clientChannel = new ChannelFactory<IO
```



```
lapDataProvider>(customBinding, address);
IOlapDataProvider _dataProvider = clientChannel.CreateChannel();

////Sets the data provider (WCF Service connection) in OlapDataManager
_olapDataManager.DataProvider = _dataProvider;
```

```
[VB]
Dim customBinding As Binding = New CustomBinding(New
BinaryMessageEncodingBindingElement(),New HttpTransportBindingElement()
With { _
       Key .MaxReceivedMessageSize = 2147483647 _
})
Dim address As New EndpointAddress(New
Uri(App.Current.Host.Source.ToString()
& "../../../../Services/OlapManager.svc/binary"))
Dim clientChannel As New ChannelFactory(Of
IOlapDataProvider)(customBinding, address)
Dim _dataProvider As IOlapDataProvider = clientChannel.CreateChannel()

'''Sets the data provider (WCF Service connection) in OlapDataManager
_olapDataManager.DataProvider = _dataProvider
```

5.8 Bind an OlapReport with OlapDataManager

Once the connection is established, you can create and bind the OlapReport to the manger by using any one of the following property and methods:

Property:

1. CurrentReport

Methods:

1. SetCurrentReport
2. LoadOlapDataManager
3. LoadReportDefinitionFile
4. LoadReportDefinitionFromStream

Methods for Silverlight:

1. SetCurrentReport
2. LoadReportFromStream

The following code snippet will illustrate the binding of OlapReport using these methods with a sample OlapReport:

Sample OlapReport

```
[C#]
```



```
OlapDataManager OlapDataManager = new OlapDataManager
(@"DataSource=localhost; Initial Catalog=Adventure Works DW");
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";

DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");

MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement
{ Name = "Internet Sales Amount" });

DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");

///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
Dim OlapDataManager As OlapDataManager = New OlapDataManager
("DataSource=localhost; Initial Catalog=Adventure Works DW")
Dim olapReport1 As OlapReport = New OlapReport()
olapReport.Name = "Customer Report"
olapReport1.CurrentCubeName = "Adventure Works"
Dim dimensionElementColumn As DimensionElement =
New DimensionElement()
'Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer"
dimensionElementColumn.AddLevel("Customer Geography", "Country")

Dim measureElementColumn As MeasureElements = New MeasureElements()
'Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Int
ernet Sales Amount"})
```

```
Dim dimensionElementRow As DimensionElement = New DimensionElement()
'Specifying the Dimension Name
dimensionElementRow.Name = "Date"
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year")

'Adding Column Members
olapReport1.CategoricalElements.Add(dimensionElementColumn)
'Adding Measure Element
olapReport1.CategoricalElements.Add(measureElementColumn)
'Adding Row Members
olapReport1.SeriesElements.Add(dimensionElementRow)
```

5.8.1 CurrentReport

Once the OlapReport is created by assigning the report to the **CurrentReport** property of the OlapDataManager, you can bind the OlapReport to the OlapDataManager.

The following code snippet explains the assigning of an OlapReport to CurrentReport property:

[C#]

```
OlapDataManager.CurrentReport = olapReport;
```

[VB]

```
OlapDataManager.CurrentReport = olapReport
```

5.8.2 SetCurrentReport

The following code snippet explains setting an OlapReport to CurrentReport of an OlapDataManager using SetCurrentReport method:

[C#]

```
OlapDataManager.SetCurrentReport(olapReport);
```

[VB]

```
OlapDataManager.SetCurrentReport(olapReport)
```

5.8.3 LoadOlapDataManager

The following code snippet explains loading an OlapReport to CurrentReport property of an OlapDataManager using LoadOlapDataManager method:

[C#]

```
OlapDataManager.LoadOlapDataManager(olapReport);
```

```
[VB]
OlapDataManager.LoadOlapDataManager(olapReport)
```

5.8.4 LoadReportDefinitionFile

You can bind the OlapReport as xml file to OlapDataManager using the LoadReportDefinitionFile method. This contains two steps as follows:

1. Loading the report definition file and
2. Loading a specific report in that file by giving its name

The following code snippet will illustrate the loading of the report definition file:

```
[C#]
olapDataManager.LoadReportDefinitionFile(@"C:\SampleReports\SalesAnalysis.xml");
olapDataManager.LoadReport("SalesOn2003");
```

```
[VB]
olapDataManager.LoadReportDefinitionFile("C:\SampleReports\SalesAnalysis.xml")
olapDataManager.LoadReport("SalesOn2003")
```

5.8.5 LoadReportDefinitionFromStream

You can load the OlapReport as a stream to the OlapDataManager using LoadReportDefinitionFromStream method. This contains two steps as follows:

1. Loading the report stream
2. Loading the specific report in that stream by giving its name

The following code snippet will illustrate the loading of the report definition from a stream:

```
[C#]
olapDataManager.LoadReportDefinitionFromStream(reportStream);
olapDataManager.LoadReport("SalesOn2003");
```

```
[VB]
olapDataManager.LoadReportDefinitionFromStream(reportStream)
olapDataManager.LoadReport("SalesOn2003")
```

Sequential Diagram

The following sequential diagram shows the workflow of OlapBase when user gives input as OlapReport.

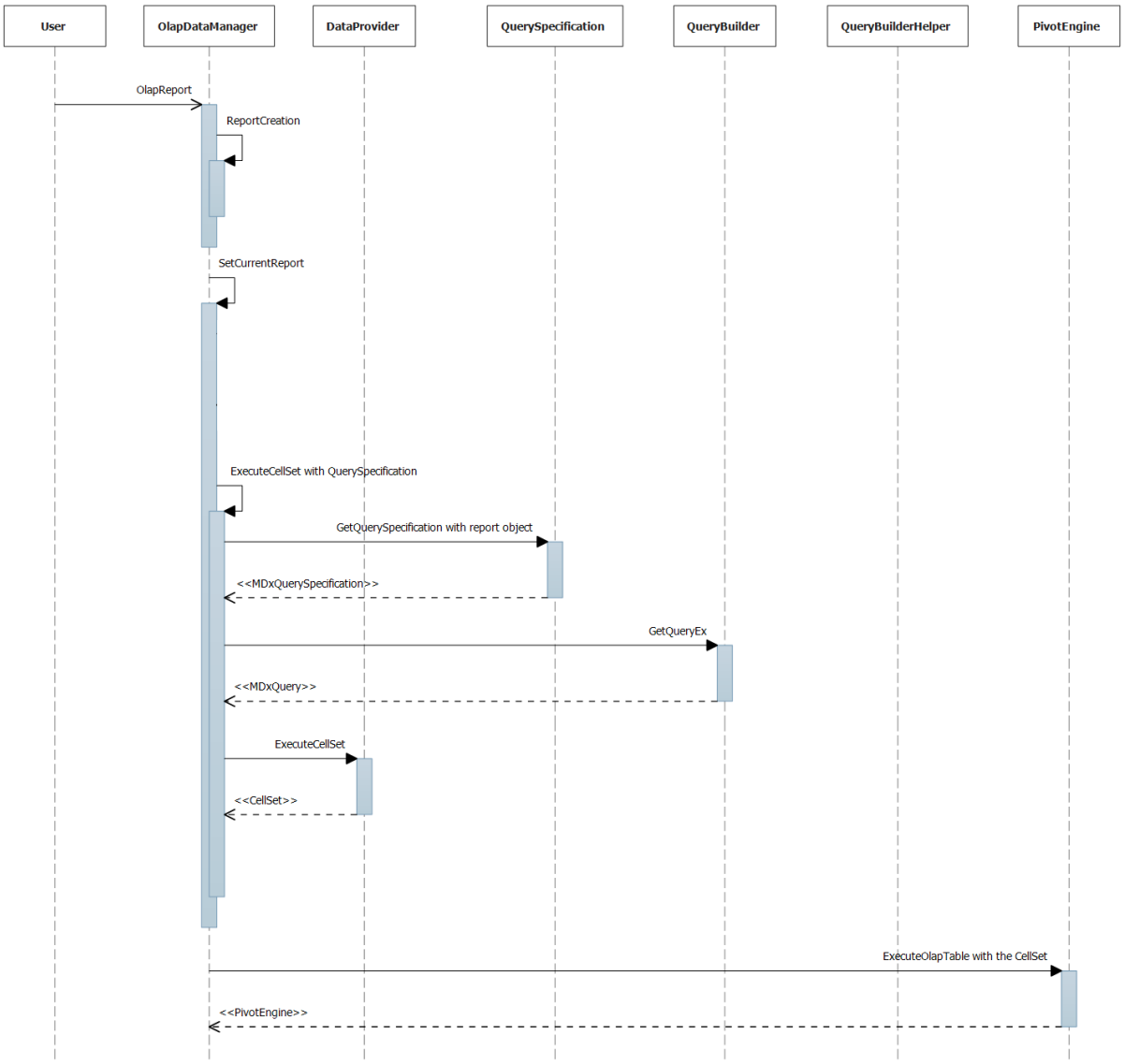

Figure 9: Olap Base sequential diagram

5.9 Bind the MDX query to OlapDataManager

MDX query is one of the inputs accepted by the OlapDataManager to process the data in the connected data source. There are two way to pass the MDX query to OlapDataManager:

1. Through **MdxQuery** property
2. Through ExecuteCellSet() method argument

OlapDataManager will accept the MDX query in the string format through any one of this and process the data based on the query. Once the connection is established you can pass the MDX query in string format.

The following code will illustrate the passing of the MXD query as input:

[C#]

```
OlapDataManager olapDataManager = new OlapDataManager("DataSource=local
host; Initial Catalog=Adventure Works DW");
string mdxQuery =
@"SELECT NON EMPTY ({{Hierarchize({DrilldownLevel({
[Customer].[Customer Geography].[All Customers]})})} *
{[MEASURES].[Internet Sales Amount]}}) dimension properties
 member_type ON COLUMNS, NON EMPTY ({{Hierarchize({
DrilldownLevel({[Date].[Fiscal].[All Periods]})})}} )
dimension properties member_type ON ROWS
FROM [Adventure Works]  CELL PROPERTIES
VALUE, FORMAT STRING, FORMATTED VALUE";
olapDataManager.MdxQuery = mdxQuery;
olapDataManager.ExecuteCellSet();
```

[VB]

```
Dim olapDataManager As OlapDataManager = New
OlapDataManager("DataSource=localhost; Initial Catalog=Adventure Works DW")
Dim mdxQuery As String = "SELECT NON EMPTY ({{Hierarchize({DrilldownLevel({
[Customer].[Customer Geography].[All Customers]})})} *
{[MEASURES].[Internet Sales Amount]}}) dimension properties
 member_type ON COLUMNS, NON EMPTY ({{Hierarchize({
DrilldownLevel({[Date].[Fiscal].[All Periods]})})}} )
dimension properties member_type ON ROWS
FROM [Adventure Works]  CELL PROPERTIES
VALUE, FORMAT_STRING, FORMATTED_VALUE"
olapDataManager.MdxQuery = mdxQuery
olapDataManager.ExecuteCellSet()
```

This will accept the MDX query as a string and assign it to the OlapDataManager'd mdxQuery property and invoke the data process.

[C#]

```
OlapDataManager olapDataManager = new OlapDataManager("DataSource=local
host; Initial Catalog=Adventure Works DW");
string mdxQuery =
@"SELECT NON EMPTY ({{Hierarchize({DrilldownLevel({
[Customer].[Customer Geography].[All Customers]})})} *
{[MEASURES].[Internet Sales Amount]}}) dimension properties
 member_type ON COLUMNS, NON EMPTY ({{Hierarchize({
DrilldownLevel({[Date].[Fiscal].[All Periods]})})}} )
dimension properties member_type ON ROWS
FROM [Adventure Works]  CELL PROPERTIES
VALUE, FORMAT_STRING, FORMATTED_VALUE";
olapDataManager.ExecuteCellSet(mdxQuery);
```

[VB]

```
Dim olapDataManager As OlapDataManager = New
OlapDataManager("DataSource=localhost; Initial Catalog=Adventure Works DW")
Dim mdxQuery As String = "SELECT NON EMPTY ({{Hierarchize({DrilldownLevel({
[Customer].[Customer Geography].[All Customers]})})} *
{[MEASURES].[Internet Sales Amount]}}) dimension properties
 member_type ON COLUMNS, NON EMPTY ({{Hierarchize({
DrilldownLevel({[Date].[Fiscal].[All Periods]})})}} )
dimension properties member_type ON ROWS
FROM [Adventure Works]  CELL PROPERTIES
VALUE, FORMAT_STRING, FORMATTED_VALUE"
olapDataManager.ExecuteCellSet(mdxQuery)
```

Sequential Diagram

The following sequential diagram is matching when user gives input as MDX query:

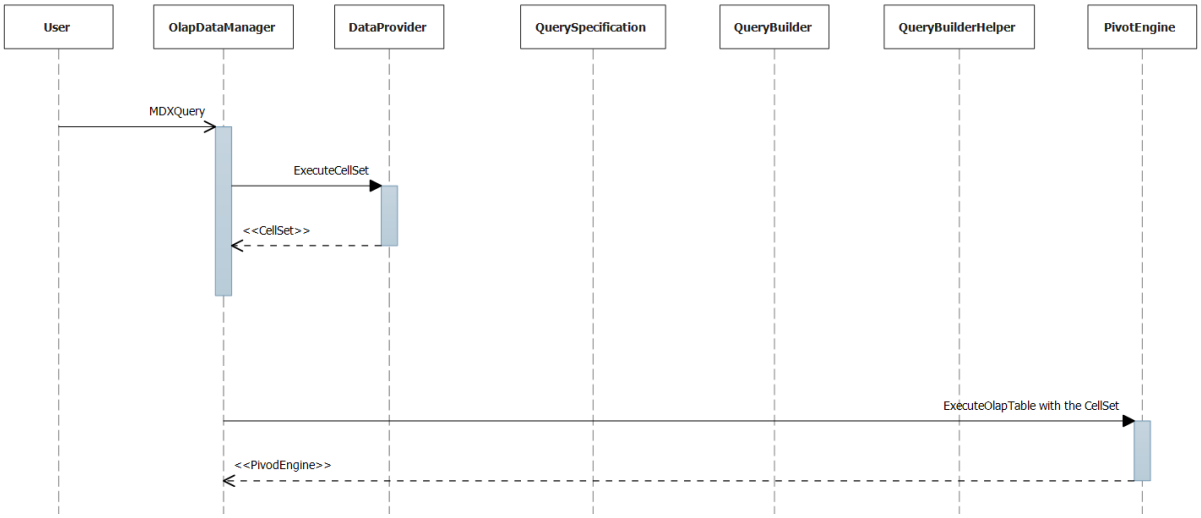

Figure 10: Olap base sequential diagram

5.10 Bind the Non-OLAP data to OlapDataManager

To bind the Non-OLAP data, you should bind an item source to the OlapDataManager's item source property and give the Non-OLAP data report to process the given item source. The item source can be an Enumerable collection or an ITyped List.

The following code will illustrate the binding of the Non-OLAP data**.** Here we have used a sample Enumerable collection "ProductSalesCollection" and a sample Olap report "salesReport":

```
[C#]
ProductSalesCollection productSales = new ProductSalesCollection();
olapDataManager.ItemSource = productSales;

olapDataManager.SetCurrentReport(salesReport);
```

```
[VB]


Dim productSales As ProductSalesCollection = New
ProductSalesCollection()
olapDataManager.ItemSource = productSales


olapDataManager.SetCurrentReport(salesReport)
```

Sequential Diagram

The following sequential diagram shows the workflow of OlapBase when the input is a Non-OLAP data:

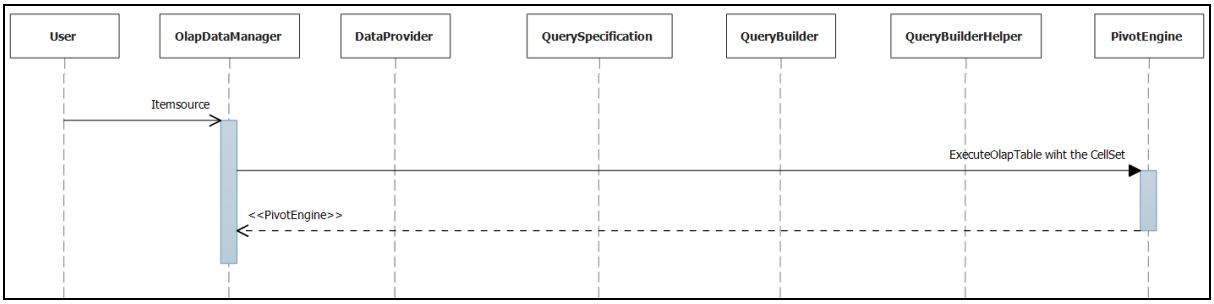

Figure 11: Olap base Sequential diagram

5.11 Save the report as xml file

The user can save the current report set of OlapDataManager as an xml file for the future needsby using the SaveReport method.

The following code snippet will illustrate the saving of the current report set as an xml file:

```
[C#]

olapDataManager.SaveReport(@"C:\SampleReport\RevenueAnalysis.xml");
```

```
[VB]

olapDataManager.SaveReport("C:\SampleReport\RevenueAnalysis.xml")
```

For Silverlight:

You can save the current report of OlapDataManger as an xml file for their future use by serializing the report with **XmlSerializer**.

The following code snippet will illustrate the saving of the current report set as an xml file:

```
[C#]

private void SaveReport()
{
   SaveFileDialog dlg = new SaveFileDialog();
```

```
   dlg.Filter = "XML files (*.xml)|*.xml";

   bool? b = dlg.ShowDialog();

   if (b.HasValue && b.Value)
   {
      using (Stream stream = dlg.OpenFile())
      {
         System.Xml.Serialization.XmlSerializer serializer = new System.Xml.Se
rialization.XmlSerializer(typeof(OlapReport));
         serializer.Serialize(stream, this.olapDataManager.CurrentReport);

      }
   }
}
```

[VB]

```
Private Sub SaveReport()
   Dim dlg As SaveFileDialog = New SaveFileDialog()
   dlg.Filter = "XML files (*.xml)|*.xml"

   Dim b As Nullable(Of Boolean) = dlg.ShowDialog()

   If b.HasValue AndAlso b.Value Then
        Using stream As Stream = dlg.OpenFile()
            Dim serializer As System.Xml.Serialization.XmlSerializer = New
System.Xml.Serialization.XmlSerializer(GetType(OlapReport))
            serializer.Serialize(stream, Me.olapDataManager.CurrentReport)
        End Using
   End If
End Sub
```

5.12 Load xml report file

You can load the xml report set by using the LoadReport method.

The following code snippet will illustrate the loading of the report:


```
[C#]


olapDataManager.LoadReport(@"C:\SampleReport\RevenueAnalysis.xml");
```

```
[VB]


olapDataManager.LoadReport("C:\SampleReport\RevenueAnalysis.xml")
```

For Silverlight:

The saved report file can be used with OlapDataManager by serializing it to type OlapReport with **XmlSerializer.**

The following code snippet will illustrate the loading of a saved xml report file:

```
[C#]


 private void LoadReport()
 {
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "XML files (*.xml)|*.xml";
    bool? b = dlg.ShowDialog();

    if (b.HasValue && b.Value)
    {
        OlapReport report = null;

        using (FileStream stream = dlg.File.OpenRead())
        {
           System.Xml.Serialization.XmlSerializer serializer = new Syst
em.Xml.Serialization.XmlSerializer(typeof(OlapReportCollection));
           OlapReportCollection olapReportCollection = serializer.Deser
ialize(stream) as OlapReportCollection;
          report = olapReportCollection[0];
        }
        olapDataManager.SetCurrentReport(report);
    }
 }
```

```
[VB]


Private Sub LoadReport()
      Dim dlg As OpenFileDialog = New OpenFileDialog()
```



```
      dlg.Filter = "XML files (*.xml)|*.xml"
      Dim b As Nullable(Of Boolean) = dlg.ShowDialog()

      If b.HasValue AndAlso b.Value Then
            Dim report As OlapReport = Nothing

            Using stream As FileStream = dlg.File.OpenRead()
               Dim serializer As System.Xml.Serialization.XmlSerializer =
New System.Xml.Serialization.XmlSerializer(GetType(OlapReportCollection))
               Dim olapReportCollection As OlapReportCollection =
TryCast(serializer.Deserialize(stream), OlapReportCollection)
              report = olapReportCollection(0)
            End Using
            olapDataManager.SetCurrentReport(report)
      End If
 End Sub
```

5.13 Rename and remove a report

Once the report collection is loaded with reports, you can rename or remove any reports in that collection by using the RenameReport and RemoveReport methods.

5.13.1 RenameReport

A report in the report collection of OlapDataManager can be renamed by invoking RenameReport method with arguments such as, index of the report and new name for the report or with old name and new name of the report. The following code snippet will illustrate this:

[C#]

```
olapDataManager.RenameReport(2, "SalesAnalysisOn2003");
olapDataManager.RenameReport("RevenueAnalysis", "RevenueAnalysisOn2003"
);
```

[VB]

```
olapDataManager.RenameReport(2, "SalesAnalysisOn2003")
olapDataManager.RenameReport("RevenueAnalysis", "RevenueAnalysisOn2003"
)
```

5.13.2 RemoveReport

A report in the report collection of the OlapDataManager can be removed by invoking the RemoveReport method. This method will accept the report name as argument and remove that report from report collection of OlapDataManager.

The following code snippet will illustrate the removal of a report:

```
[C#]
olapDataManager.RemoveReport("SalesAnalysisOn2005");
```

```
[VB]
olapDataManager.RemoveReport("SalesAnalysisOn2005")
```

5.14 Get the reports in the OlapDataManager as a stream

You can get the report collection in the OlapDataManager as a stream by using GetReportAsStream method. This method will return the current report collection of the OlapDataManager as a stream.

The following code snippet will explain obtaining the report as a stream:

```
[C#]
Stream reportStream = olapDataManager.GetReportAsStream();
```

```
[VB]
Dim reportStream As Stream = olapDataManager.GetReportAsStream()
```

5.15 Communicate the OLAP control with the base

Each and every control has an **OlapDataManager** property. Through this property, the control sends and receives information to and from the base.

Below steps explains how to load data in an OLAP control:

1. Give the connection information and **OlapReport** to the **OlapDataManager**.
2. Assign this **OlapDataManager** to the control's **OlapDataManager** property.
3. By invoking the control's **DataBind()** method virtually when setting the value for the OlapDataManager property, the data processing will begin and the output is displayed in the Control.

5.16 Add the elements to an Axis

After creating the element, add the element to the specific axis. The **OlapReport** contains the axis as **CategoricalElements**, **SeriesElement** and **SliceElements**. By adding the created elements to any of these elements group, you can specify the axis position of the elements.

The following codes will describe the adding of the elements to categorical, series element:

[C#]

```
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);

///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
```

[VB]

```
'''Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
'''Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn)

'''Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow)
```

5.17 Apply the Filter through filter element

The filter element will get information such as filter condition and filter value from the user, from the filter expression and then get the elements on which the filter has to apply.

The following codes describe the creation of the filter element and its application:

[C#]

```
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
```

```
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
 Sales Amount" });

FilterElement filterElement = new FilterElement(AxisPosition.Categorica
l);
filterElement.Elements.Add(measureElementColumn);
filterElement.Elements.Add(dimensionElementColumn);
filterElement.FilterCase = FilterCase.GreaterThan;
filterElement.FilterValue.Add(new MeasureElement { Name = "Internet Sal
es Amount", Visible = true });
filterElement.FilterValue.Add(new FilterValue { Filter Value = 2700000.
00 });
filterElement.IsFilterCondition = true;
/// Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.IsFilterOrSortOn = true;
///Adding Measure Element
olapReport.FilterElements.Add(filterElement);
```

[VB]

```
Dim dimensionElementColumn As DimensionElement = New DimensionElement()

'Specifying the Name for the Dimension Element

dimensionElementColumn.Name = "Customer"


Dim measureElementColumn As MeasureElements = New MeasureElements()

measureElementColumn.Elements.Add(New MeasureElement With {.Name = "Internet
Sales Amount"})


Dim filterElement As FilterElement = New
FilterElement(AxisPosition.Categorical)

filterElement.Elements.Add(measureElementColumn)

filterElement.Elements.Add(dimensionElementColumn)

filterElement.FilterCase = FilterCase.GreaterThan


filterElement.FilterValue.Add(New MeasureElement With {.Name = "Internet
Sales Amount", .Visible = True})


filterElement.FilterValue.Add(New FilterValue With {.Filter Value =
```



```
2700000.0})
filterElement.IsFilterCondition = True
''' Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn)
olapReport.CategoricalElements.IsFilterOrSortOn = True
'''Adding Measure Element
olapReport.FilterElements.Add(filterElement)
```

5.18 Show/hide the Expander buttons in OLAP controls

There is a property in **OlapReport** called **ShowExpanders,** which is used to show/hide the expander buttons in the OLAP controls. By using this property, we can disable or enable the drill down/up behavior of the OLAP control.

To show the Expanders:

[C#]

```
//// Displaying the Expander button in Controls
olapReport.ShowExpanders = true;
```

[VB]

```
'''Displaying the Expander button in Controls
olapReport.ShowExpanders = True
```

To hide the Expanders:

[C#]

```
//// Displaying the Expander button in Controls
olapReport.ShowExpanders = false;
```

[VB]

```
'''Displaying the Expander button in Controls
olapReport.ShowExpanders = false
```

5.19 Process OlapReport Internally

Once the OlapReport is created and bound to the OlapDataManager, the data processing begins. By binding the report, the given report will be set as the current report of the OlapDataManager and it will invoke the two important methods that let the way to generate the **MDX query** and **CellSet**.

- NotifyReportChanged
- NotifyElementModified

NotifyReportChanged

After initialization the data processing begins. When the **NotifyReportChanged** is invoked, it triggers the **ReportChaned** event, which will be handled by the control level.

NotifyElementModified

The **NotifyElementModified** method begins the processing by invoking the **ExecuteCellSet()** method, which create the **CellSet** and **PivotEngine** based on the OlapReport.

The **ExecuteCellSet()** method checks whether the user has given the MDX query. If the query exists, it will invoke an overloaded method of the **ExecuteCellSet(string *query*)** which in turn calls the **ExecuteCellSet(string query, bool b1, bool b2)** of DataProvider class. The given query will be executed in the DataProvider class and the **CellSet** will be produced.

If the query does not exist, the overloaded method of **ExecuteCellSet** (**MDXQuerySpecification querySpecification)** will get invoked. This method will invoke the **MDXQuerySpecification** creation and from there the query creation will be invoked and the query will be returned. The **ExeccuteCellSet()** method receives the query and passes it to the data provider's **ExecuteCellset** method. The query will be executed there on the connected database and a **CellSet** will be returned. From the **CellSet**, the **PivotEngine** will be created, from which the controls can get their input.

5.20 Handle Drill Down/Up Process

Whenever we collapse or expand the controls like a grid or chart, the level member items will change and the query will be regenerated to create the new **CellSet**.

The important methods that identify the drill-down members are:

- ToggleExpandableState()
- UpdateDrillDowItems()
- DrillUpDown()

Whenever the drill-down button is clicked, the **ToggleExpandableState()** method in the **OlapDataManager** will be invoked by the control. From there, the **UpdateDrillDowlItems** will be called by passing the arguments and there it checks the unique name and call the **DrillUpDown()** method, which accepts the hierarchy element as argument. This method is a recursive method and has an overload method that accepts the member element as an argument. By recursively iterating the drill-down level, all the members in that level will be created and added with its parent for the query creation.

Once the drill-down member is updated, the **NotifyElementModified()** will be invoked to generate the new query.

Sequential Diagrams

The following screen shot explains sequential diagram for drill/down process:

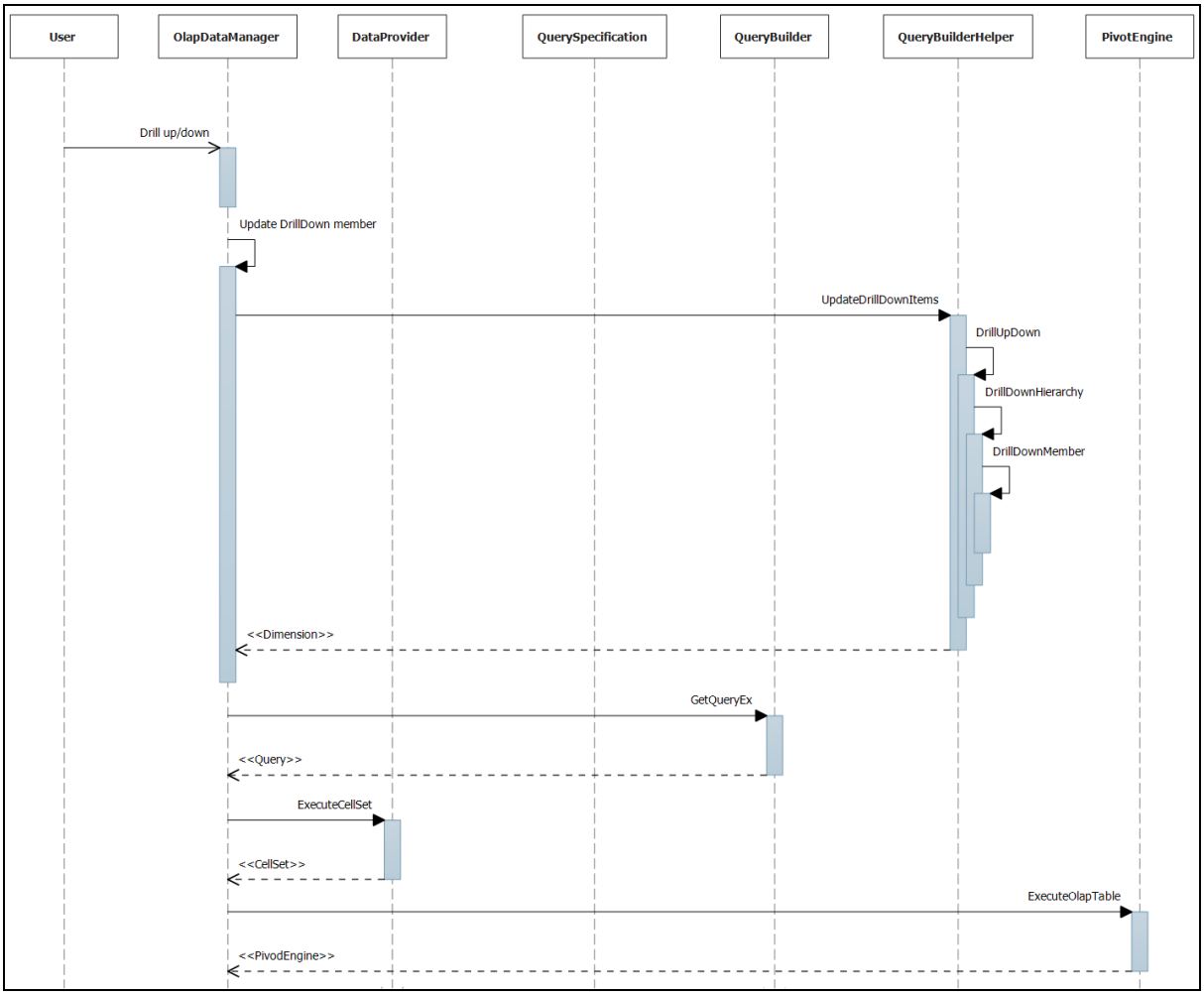

Figure 12: OLAP base sequential diagram

5.21 Connect WCF Service by an additional Binding Type in Silverlight application

OLAP Silverlight controls can be bound to *BasicHttpBinding* support.

 ***Note: Incorrect specification of endpoint address will lead to Initialize Component Error. The endpoint address should be set with respect to the type of bindings.***

Use case Scenario

User can create Web service using the *BasicHttpBinding* feature and bind the service to OLAP Silverlight controls.

BasicHttpBinding

The following are steps to create a BasicHttpBinding:

Include the Basic Http Binding and Service endpoint address in the Web.Config file as given in the following code:

```
[XML]
 <!--<Bindings>-->
     <bindings>
     <basicHttpBinding>
       <binding name="binaryHttpBinding" maxBufferSize="2147483647"
          maxReceivedMessageSize="2147483647">
         <readerQuotas maxDepth="2147483647"/>
         <security mode="None" />
       </binding>
     </basicHttpBinding>
   </bindings>

   <!--<Endpoint Address>-->
   <services>
      <service behaviorConfiguration="Services.OlapManagerBehavior"
name="Services.OlapManager">
        <endpoint address="basic" binding="basicHttpBinding"
                  bindingConfiguration="binaryHttpBinding"

contract="Syncfusion.OlapSilverlight.Manager.IOlapDataProvider" />
      </service>
   </services>
```


Instantiate the WCF service using Basic Http Binding and End Point Address values:

- Declare the *IOlapDataProvider* for Service instantiation as given in the following code:

[C#]

```
  // Declaring the IOlapDataProvider for service instantiation
IOlapDataProvider DataProvider = null;
```

[VB]

```
' Declaring the IOlapDataProvider for service instantiation
   Dim DataProvider As IOlapDataProvider = Nothing
```

- Specify the *basicHttpBinding* and Instantiate the *DataProvider* from the *ChannelFactory* as given in the following code:

[C#]

```
private void InitializeConnection()
        {
            Binding basicHttpBinding = new
BasicHttpBinding(BasicHttpSecurityMode.None)
            { MaxBufferSize = 2147483647, MaxReceivedMessageSize =
2147483647 };
            var Uri = App.Current.Host.Source.ToString();
            ////Address for Basic HTTP binding in corresponding Web
configuration file
            EndpointAddress address = new EndpointAddress (new Uri (Uri +
             "../../../../Services/OlapManager.svc/basic"));
            ChannelFactory<IOlapDataProvider> clientChannel = new
            ChannelFactory<IOlapDataProvider>(basicHttpBinding, address);
            DataProvider = clientChannel.CreateChannel();
        }
```



```
[VB]

Private Sub InitializeConnection()

Dim basicHttpBinding As System.ServiceModel.Channels.Binding = New
BasicHttpBinding (BasicHttpSecurityMode.None With {.MaxReceivedMessageSize =
2147483647})

  'Address for Basic HTTP binding in corresponding Web configuration file

Dim address As EndpointAddress = New EndpointAddress (New

Uri(App.Current.Host.Source & "../../../../OlapManager.svc/basic"))

Dim clientChannel As ChannelFactory(Of IOlapDataProvider) = New
ChannelFactory(Of IOlapDataProvider)(basicHttpBinding, address)

DataProvider = clientChannel.CreateChannel()

End Sub
```

5.22 Retrieve the MDX Query of a CurrentReport

The MDX query of a current report is used to display data in Grid/Chart control and it can be retrieved by calling the GetMdxQuery() method.

The following code explains how to retrieve MDX Query from the OlapDataManager:

```
[C#]

olapDataManager.GetMDXQuery();
```

```
[VB]

olapDataManager.GetMDXQuery()
```

In Silverlight:

```
[C#]

string currentMdxQuery = null;
//// Invoke the service call to retrieve the MDX query from the Server
based on current report.
_olapDataManager.GetMdxQuery(_olapDataManager.CurrentReport);
_olapDataManager.MdxQueryObtained += () =>
{
    ////MDX Query retrieved.
    currentMdxQuery = _olapDataManager.CurrentReport.CurrentMdxQuery;
};
```



```
[VB]
Dim currentMdxQuery As String
'Invoke the service call to retrieve the MDX query from the Server based on current report.
_olapDataManager.GetMdxQuery(_olapDataManager.CurrentReport)
_olapDataManager.MdxQueryObtained += Function()
'MDX Query retrieved.
currentMdxQuery = _olapDataManager.CurrentReport.CurrentMdxQuery
End Function
```

5.23 Add UseWhereClauseForSlicing to an Application

The user can add the UseWhereClauseForSlicing property to an application by setting the property to a Boolean value. To perform the slicing operation using the 'Where' clause, set the property to *true*. To perform the slicing operation using the 'Select' clause, set the property to *false*. By default, the value of the UseWhereClauseForSlicing property is *true*.

To perform slicing operation using 'Where' clause:

```
[C#]
OlapDataManager.UseWhereClauseForSlicing = true;
```

```
[VB]
OlapDataManager.UseWhereClauseForSlicing = True
```

To perform slicing operation using 'Select' clause:

```
[C#]
this.olapGridControl1.OlapDataManager.UseWhereClauseForSlicing = false;
```

```
[VB]
Me.olapGridControl1.OlapDataManager.UseWhereClauseForSlicing = False
```

5.24 Edit MDX Query before Its Execution

MDX Query can be edited before its execution to retrieve the CellSet through handling the **BeforeMdxQueryExecute** event of **OlapDataManager**. The following code example illustrates this.

```
[C#]

olapDataManager.BeforeMdxQueryExecute += new
QueryExecuteEventHandler(olapDataManager_BeforeMdxQueryExecute);


void olapDataManager BeforeMdxQueryExecute(object sender,
QueryExecutingEventArgs e)
{
     e.MdxQuery = "Edit MDX query here";
}
```

```
[VB]

Private olapDataManager.BeforeMdxQueryExecute += New
QueryExecuteEventHandler(AddressOf olapDataManager_BeforeMdxQueryExecute)

Private Sub olapDataManager BeforeMdxQueryExecute(ByVal sender As Object,
ByVal e As QueryExecutingEventArgs)
        e.MdxQuery = "Edit MDX query here"
End Sub
```

5.25 Host BI Silverlight component in ASP.NET MVC Web Project

The following steps explain how to add the Silverlight components in MVC project:

1. Open **Visual Studio IDE**.
2. Go to **File → New → Project** and create a new Silverlight application.

A dialog window opens as shown below:

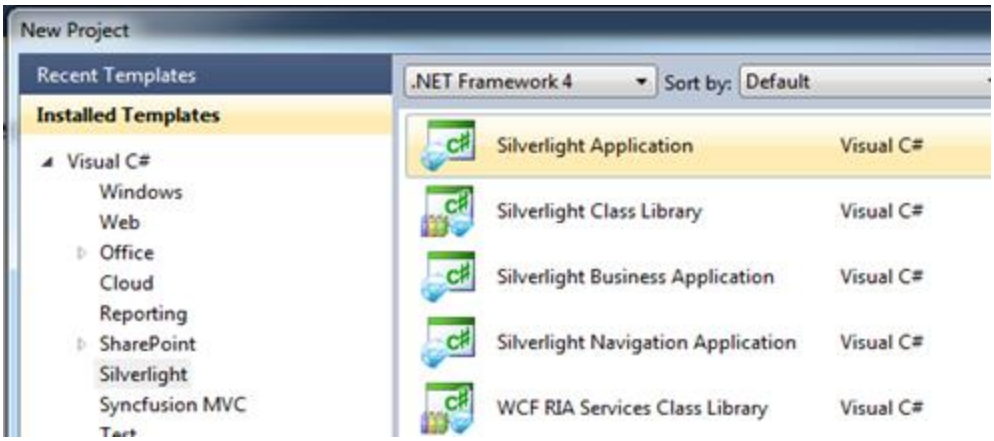

Figure 13: New Project Dialog Window

3. Select *Silverlight Application* from the **New Project dialog window** and click **OK**.

The New Silverlight Application dialog opens as shown in the following screenshot:

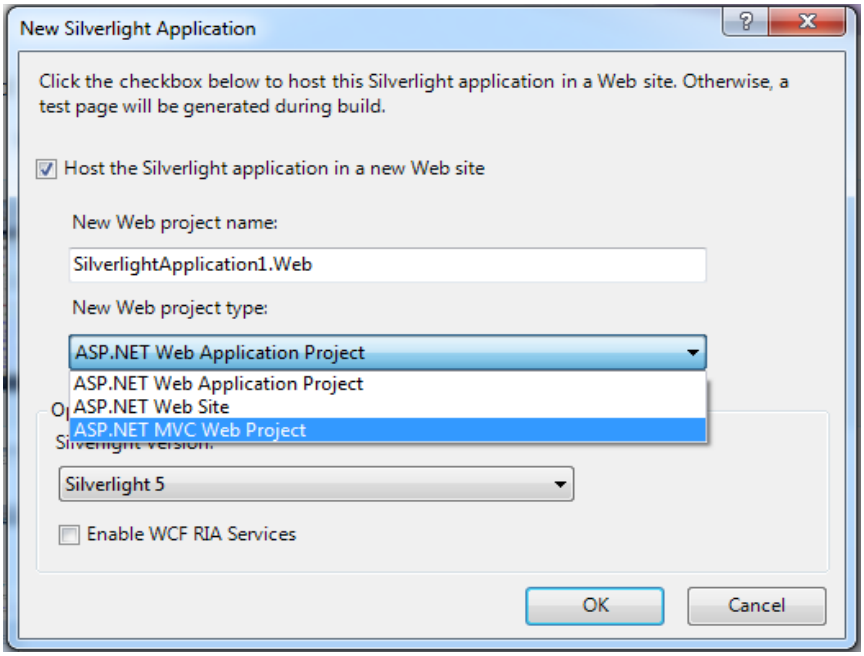

Figure 14: Web Project Selection Window

The **Solution Explorer** window shows the Silverlight application with MVC project.

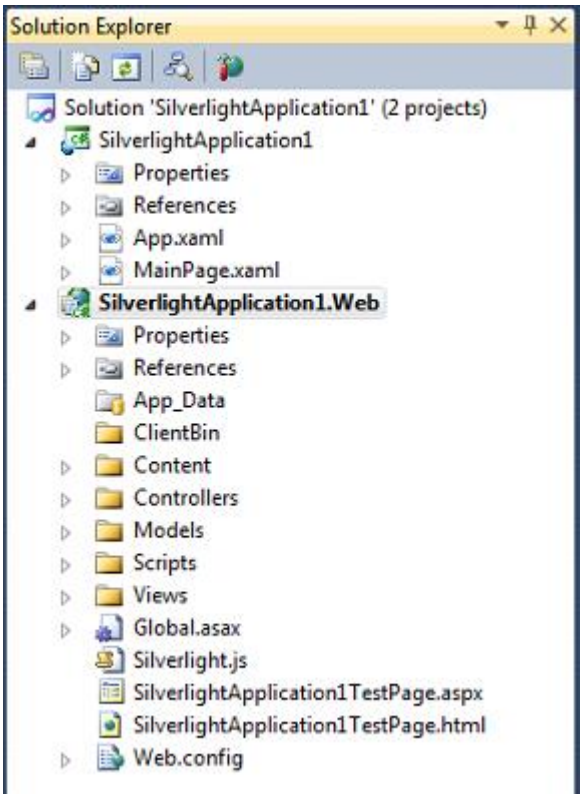

Figure 15: Solution Explorer with Silverlight and MVC Projects

4. Double-click to open the Main.xaml which is found under the Silverlight project in Solution Explorer as shown below:

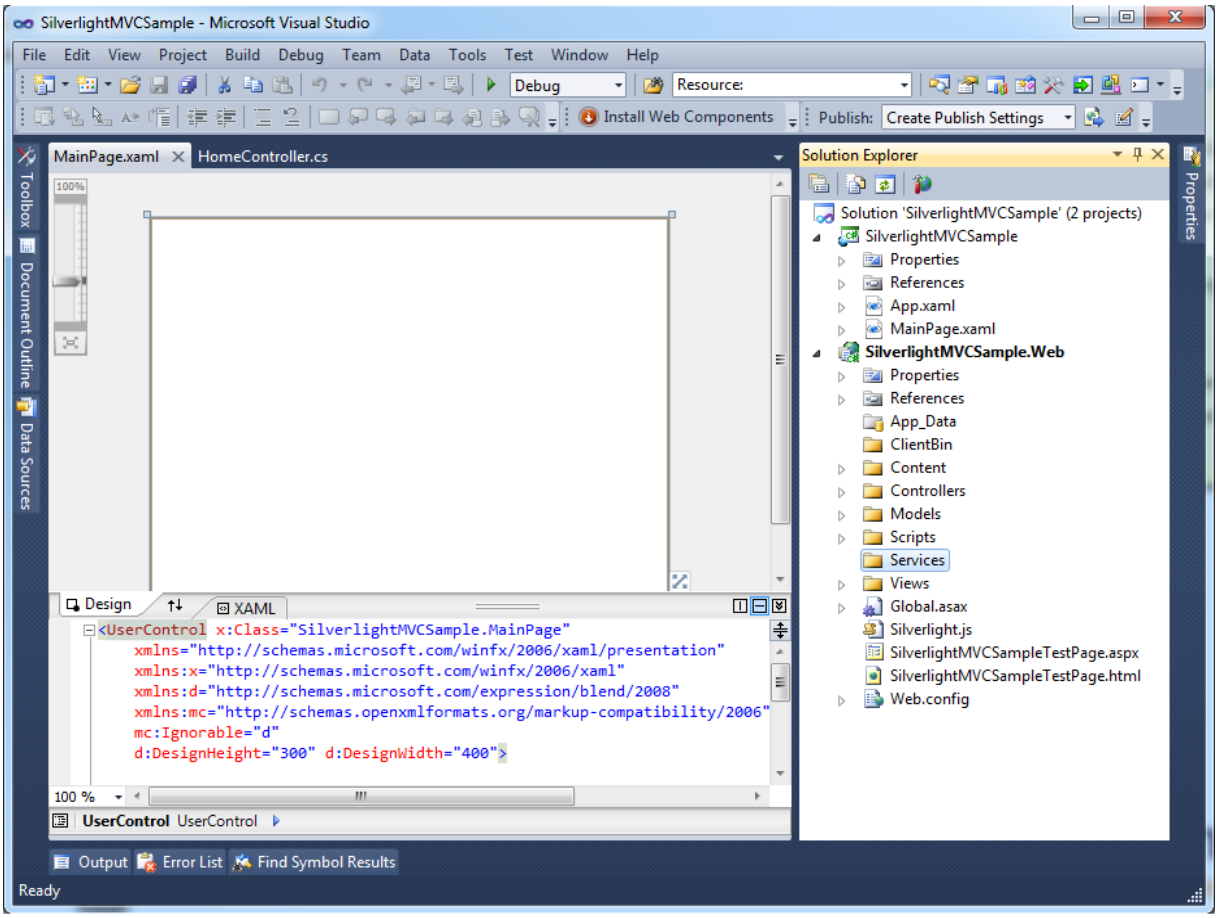

Figure 16: Designer Page

5. Drag and drop the **OlapGrid** from the toolbox to the MainPage.xaml.

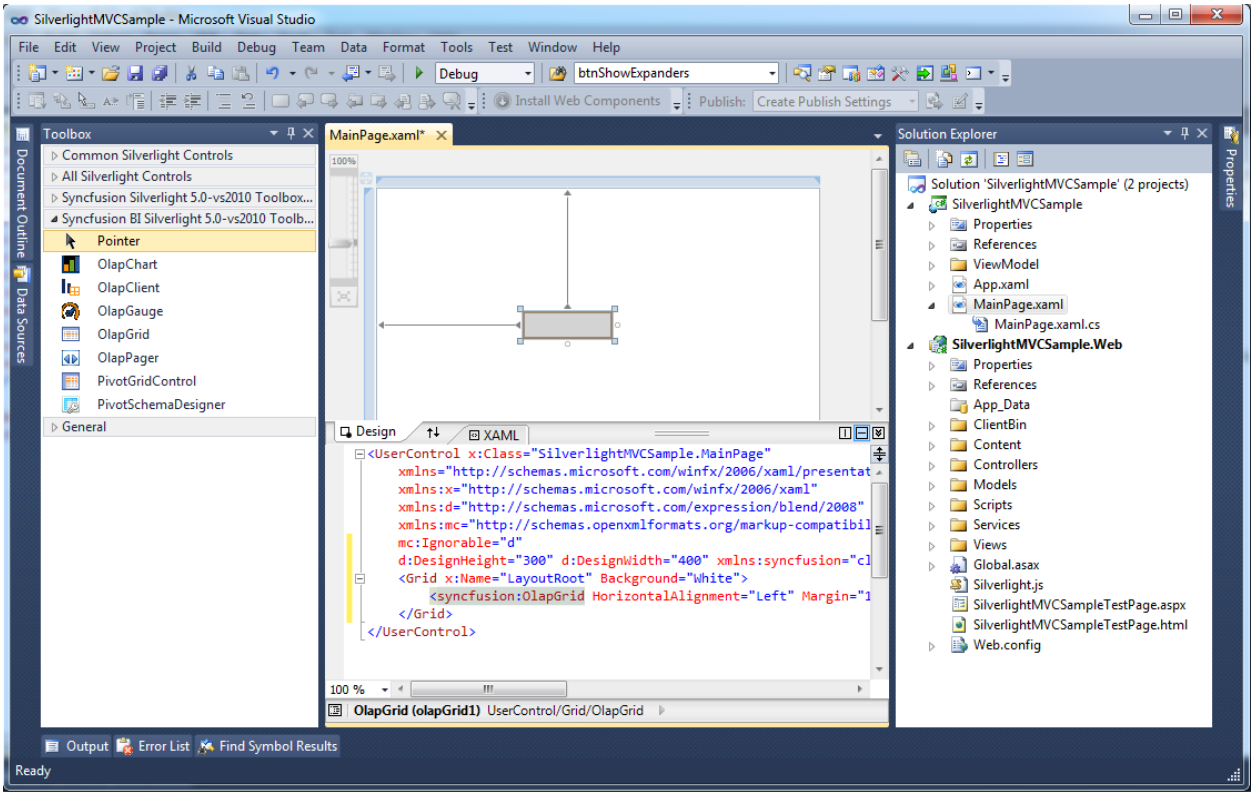

Figure 17: OlapGrid in Designer Page

6. Add the following two assemblies as references to the web project:
    - Syncfusion.Olap.Base
    - Syncfusion.OlapSilverlight.BaseWrapper

7. Add a WCF Service to the web project by right-clicking the **Project → Add New Item → WCF Service**.
8. Name the service as **OlapManager** and delete the **IOlapManager.cs** file as the service has to be inherited with the IOlapDataProvider.

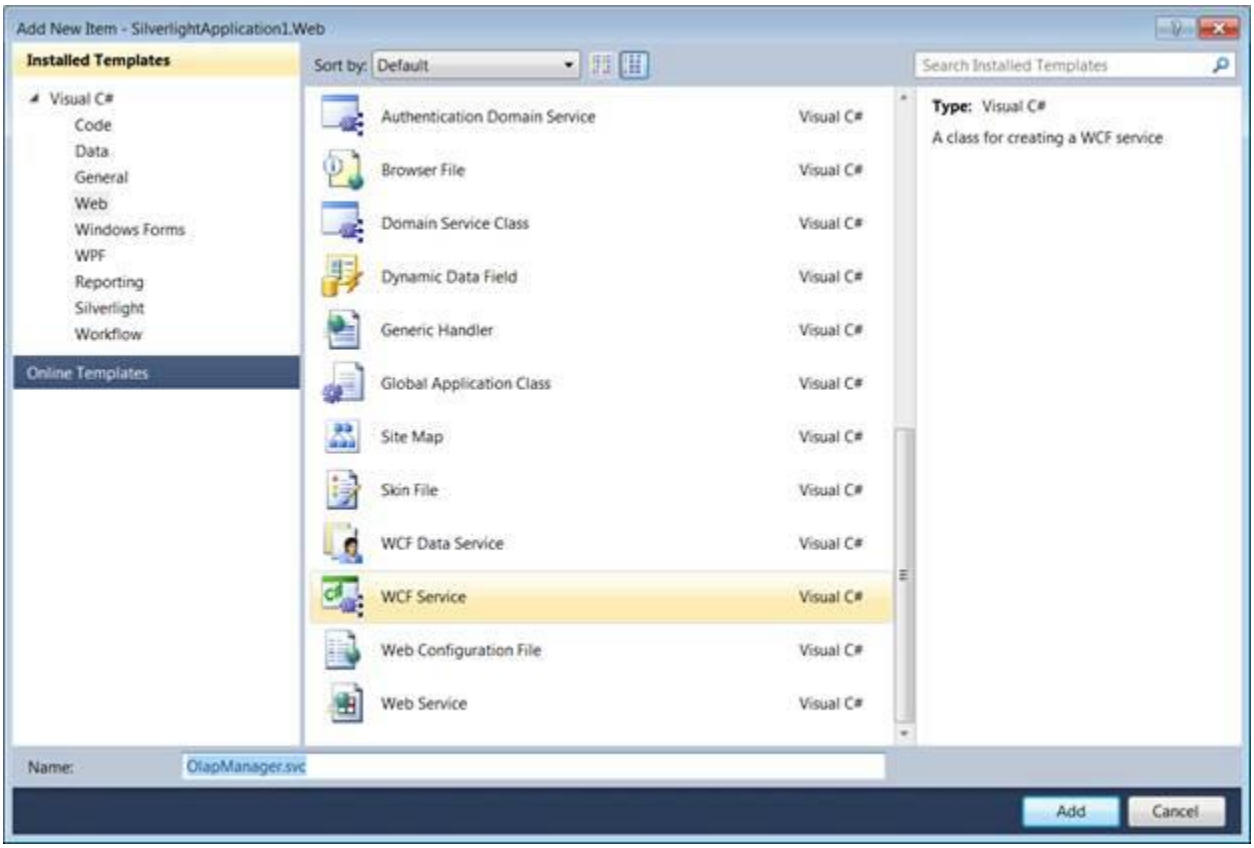

Figure 18: Add New Item Dialog (WCF Service)

9. Inherit the newly added WCF service with the IOlapDataProvider and explicitly implement the IOlapDataProvider.
10. The connection to the database is done with the help of the WCF service. The service has to be created and instantiated as described in the below code snippet.

The WCF Service has to implement the IOlapDataProvider interface. To implement this interface, you require the OlapDataProvider, which can be instantiated by passing the connection string.

The interface can be implemented as shown in the following code snippet:

```
[C#]
public class OlapManager : IOlapDataProvider
{
        Syncfusion.OlapSilverlight.Manager.OlapDataProvider
dataManager;

        /// <summary>
```



```
        /// Initializes a new instance of the <see cref="OlapManager"/>
class.
        /// </summary>
        public OlapManager()
        {
            string connectionString = "DataSource=localhost;Initial
Catalog=Adventure Works DW";
            // Instantiating the OlapDataProvider with connection
string.
            dataManager = new OlapDataProvider(connectionString);
        }
        #region IOlapDataProvider Members
        /// <summary>
        /// Executing the CellSet by passing OlapReport.
        /// </summary>
        /// <param name="report">The report.</param>
        /// <returns> The CellSet </returns>
        public Syncfusion.OlapSilverlight.Data.CellSet
ExecuteOlapReport(Syncfusion.OlapSilverlight.Reports.OlapReport report)
        {
            Syncfusion.OlapSilverlight.Data.CellSet cellSet =
this.dataManager.ExecuteOlapReport(report);
            // Closing the provider connection.
            this.dataManager.DataProvider.CloseConnection();
            return cellSet;
        }
        /// <summary>
        /// Executing the CellSet by passing MDX Query.
        /// </summary>
        /// <param name="mdxQuery">The MDX query.</param>
        /// <returns> The CellSet </returns>
        public Syncfusion.OlapSilverlight.Data.CellSet
ExecuteMdxQuery(string mdxQuery)
        {
            Syncfusion.OlapSilverlight.Data.CellSet cellSet =
this.dataManager.ExecuteMdxQuery(mdxQuery);
```



```
            // Closing the provider connection.
            this.dataManager.DataProvider.CloseConnection();
            return cellSet;
        }

        public MemberCollection GetChildMembers(string
memberUniqueName, string cubeName)
        {
            throw new NotImplementedException();
        }
        public CubeSchema GetCubeSchema(string cubeName)
        {
            throw new NotImplementedException();
        }
        public CubeInfoCollection GetCubes()
        {
            throw new NotImplementedException();
        }
        public MemberCollection GetLevelMembers(string levelUniqueName,
string cubeName)
        {
            throw new NotImplementedException();
        }
        #endregion
}
```

[VB]

```
Public Class OlapManager
      Implements IOlapDataProvider
            Private dataManager As
Syncfusion.OlapSilverlight.Manager.OlapDataProvider
            ''' <summary>
            ''' Initializes a new instance of the <see
cref="OlapManager"/> class.
            ''' </summary>
```



```
            Public Sub New()
                  Dim connectionString As String =
"DataSource=localhost;Initial Catalog=Adventure Works DW"
                  ' Instantiating the OlapDataProvider with connection
string
                  dataManager = New OlapDataProvider(connectionString)
            End Sub
             #Region "IOlapDataProvider Members"
             ''' <summary>
            ''' Executing the CellSet by passing OlapReport
            ''' </summary>
            ''' <param name="report">The report.</param>
            ''' <returns></returns>
            Public Function ExecuteOlapReport(ByVal report As
Syncfusion.OlapSilverlight.Reports.OlapReport) As
Syncfusion.OlapSilverlight.Data.CellSet
                  Dim cellSet As
Syncfusion.OlapSilverlight.Data.CellSet =
Me.dataManager.ExecuteOlapReport(report)
                  ' Closing the provider connection
                  Me.dataManager.DataProvider.CloseConnection()
                  Return cellSet
            End Function
             ''' <summary>
            ''' Executing the CellSet by passing MDX Query
            ''' </summary>
            ''' <param name="mdxQuery">The MDX query.</param>
            ''' <returns> The CellSet </returns>
            Public Function ExecuteMdxQuery(ByVal mdxQuery As String)
As Syncfusion.OlapSilverlight.Data.CellSet
                  Dim cellSet As
Syncfusion.OlapSilverlight.Data.CellSet =
Me.dataManager.ExecuteMdxQuery(mdxQuery)
                  'Closing the provider connection.
                  Me.dataManager.DataProvider.CloseConnection()
                  Return cellSet
            End Function
```



```
            Public Function GetChildMembers(ByVal memberUniqueName As
String, ByVal cubeName As String) As MemberCollection
                  Throw New NotImplementedException()
            End Function

            Public Function GetCubeSchema(ByVal cubeName As String) As
CubeSchema
                  Throw New NotImplementedException()
            End Function

            Public Function GetCubes() As CubeInfoCollection
                  Throw New NotImplementedException()
            End Function

            Public Function GetLevelMembers(ByVal levelUniqueName As
String, ByVal cubeName As String) As MemberCollection
                  Throw New NotImplementedException()
            End Function

            #End Region
 End Class
```

11. Include the custom binding and the service endpoint address in the Web.Config file under the ServiceModel section.

[Web.Config]

```
     <!--Binding-->
      <bindings>
        <customBinding>
          <binding name="binaryHttpBinding">
            <binaryMessageEncoding/>
            <httpTransport maxReceivedMessageSize="2147483647"/>
          </binding>
        </customBinding>
```



```
      </bindings>

      <!—Endpoint Address-->

      <services>

        <service name="SilverlightApplication1.Web.OlapManager" >

          <endpoint address="binary" binding="customBinding" bindingCon
figuration="binaryHttpBinding" contract="Syncfusion.OlapSilverlight.Man
ager.IOlapDataProvider">
          </endpoint>

        </service>

      </services>
```

12. Add the **System.ServiceModel** assembly as a reference for the Silverlight project.
13. Add the following namespace in MainPage.xaml.cs:
    - System.ServiceModel
    - System.ServiceModel.Channels
    - Syncfusion.OlapSilverlight.Reports
    - Syncfusion.Silverlight.Grid
    - Syncfusion.OlapSilverlight.Manager
    - Syncfusion.OlapSilverlight.Engine

14. Instantiate the service from MainPage.xaml.cs which is in the Silverlight Project.
15. Declare the IOlapDataProvider for service instantiation.

[C#]

```
// Declaring the IOlapDataProvider for service instantiation.
IOlapDataProvider DataProvider = null;
```

[VB]

```
'Declaring the IOlapDataProvider for service instantiation.

Dim DataProvider As IOlapDataProvider = Nothing
```

16. Specify the custom binding and instantiate the **DataProvider** from the **ChannelFactory**.

[C#]

```
private void InitializeConnection()
{

      System.ServiceModel.Channels.Binding customBinding = new CustomBi
```



```
nding(new BinaryMessageEncodingBindingElement(), new HttpTransportBindi
ngElement { MaxReceivedMessageSize = 2147483647 });

            EndpointAddress address = new EndpointAddress(new Uri(App.C
urrent.Host.Source + "../../../../OlapManager.svc/binary"));

     ChannelFactory<IOlapDataProvider> clientChannel = new ChannelFacto
ry<IOlapDataProvider>(customBinding, address);
     DataProvider = clientChannel.CreateChannel();
}
```

```
[VB]

Private Sub InitializeConnection()

                  Dim customBinding As
System.ServiceModel.Channels.Binding = New CustomBinding(New
BinaryMessageEncodingBindingElement(), New HttpTransportBindingElement
With {.MaxReceivedMessageSize = 2147483647})

                  Dim address As EndpointAddress = New
EndpointAddress(New Uri(App.Current.Host.Source &
"../../../../OlapManager.svc/binary"))

                  Dim clientChannel As ChannelFactory(Of
IOlapDataProvider) = New ChannelFactory(Of
IOlapDataProvider)(customBinding, address)

                  DataProvider = clientChannel.CreateChannel()

End Sub
```

17. Create the Report.

For creating reports there is a report object called OlapReport. The OlapReport object contains CategoricalItems, SeriesItems, SlicerItems, and FilterItems.

The OlapReport is associated with the OlapDataManager as the current report property. When a report is set to the current report, an event triggers and the control renders based on the current report that is set.

18. Bind the data to OlapGridData.

```
[C#]

private void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    // Initialize the service connection.
    this.InitializeConnection();
    // Instantiating the OlapDataManager.
    OlapDataManager m_OlapDataManager =new OlapDataManager();
```



```
    // Specifying the DataProvider for OlapDataManager.
    m_OlapDataManager.DataProvider = this.DataProvider;
    // Set current report for OlapDataManager.
    m_OlapDataManager.SetCurrentReport(CreateOlapReport());
    // Specifying the OlapDataManager for OlapGrid.
    this.olapGrid1.OlapDataManager = m_OlapDataManager;
    // Data Binding.
    this.olapGrid1.DataBind();
}
```

[VB]

```
Private Sub MainPage Loaded(ByVal sender As Object, ByVal e As
RoutedEventArgs)

      'Initialize the service connection.

      Me.InitializeConnection()

      'Instantiating the OlapDataManager.

      Dim m_OlapDataManager As OlapDataManager = New
OlapDataManager()

      'Specifying the DataProvider for OlapDataManager.

      m_OlapDataManager.DataProvider = Me.DataProvider

      'Set current report for OlapDataManager.

      m_OlapDataManager.SetCurrentReport(CreateOlapReport())

      ' Specifying the OlapDataManager for OlapGrid.

      Me.olapGrid1.OlapDataManager = m_OlapDataManager

      ' Data Binding.

      Me.olapGrid1.DataBind()

End Sub
```

Click here for Sample Report

	Internet Sales Amount						
	Australia	Canada	France	Germany	United Kingdom	United States	Total
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

Figure 19: OlapGrid Control with OLAP Data

Index

