Reveal

SDK API

Public API summary

C#

```
public class RevealUtility
{
    public static async Task<RVDashboard> LoadDashboard(Stream data);
}

public class RevealView : Control
{
    public RevealSettings Settings { get; set; }
    public string LocalDataFilesRootFolder { get; set; }
    public IRVDataSourceProvider DataSourceProvider { get; set; }
    public IRVAuthenticationProvider AuthenticationProvider { get; set; }
    public IRVDataProvider DataProvider { get; set; }

    public event EventHandler<VisualizationClickedEventArgs>
VisualizationDataPointClicked;

    public delegate void LinkingEventHandler(object sender,
VisualizationLinkingDashboardEventArgs e);
    public event LinkingEventHandler VisualizationLinkingDashboard;

    public bool MaximizeVisualization(RVVisualization viz);
    public bool MinimizeVisualization();
    public RVVisualization MaximizedVisualization { get; }

    public delegate void SaveDashboardEventHandler(object sender, Stream
dashboardStream);
    public event SaveDashboardEventHandler SaveDashboard;

    public event EventHandler<ImageExportedEventArgs> ImageExported;

    public RVDashboard Dashboard { get; }

    public void SetFilterSelectedValues(RVDashboardFilter filter, List<object>
selectedValues);
}

public class RevealSettings
{
    public RVDashboard Dashboard { get; set; }
    public bool CanEdit { get; set; }
    public bool IsEditing { get; set; }
    public bool ShowBackButton { get; set; }
    public bool ShowFilters { get; set; }
    public bool ShowMenu { get; set; }
    public bool ShowExportImage { get; set; }
    public bool ShowChangeVisualization { get; set; }
    public bool ShowRefresh { get; set; }
    public RVVisualization MaximizedVisualization { get; set; }
    public RVDateDashboardFilter DateFilter { get; set; }

    public RevealSettings(RVDashboard revealDashboard) { }
```

```
    public void SetFilterSelectedValues(RVDashboardFilter filter, List<object> values);
    public List<object> GetFilterSelectedValues(string filterId);
}

public class RVDashboard
{
    public IEnumerable<RVVisualization> Visualizations { get; }
    public IEnumerable<RVDashboardFilter> Filters { get; }
    public RVDateDashboardFilter DateFilter { get; }

    public RVVisualization GetVisualizationByTitle(string title);
    public RVVisualization GetVisualizationById(string id);

    public RVDashboardFilter GetFilterByTitle(string title);
    public RVDashboardFilter GetFilterById(string id);
}

public class RVVisualization
{
    public string Title { get; }
    public string Id { get; }
    public string VisualizationType { get; }
}
public class RVDashboardFilter
{
    public string Title { get; }
    public string Id { get; }
}
public class RVDateDashboardFilter
{
    public RVDateFilterType DateFilterType { get; }
    public RVDateRange Range { get; }

    public RVDateDashboardFilter(RVDateFilterType filterType);
    public RVDateDashboardFilter(RVDateFilterType filterType, RVDateRange customRange);

    public string Id { get; }
}
public enum RVDateFilterType
{
    AllTime, CustomRange, LastWeek, LastMonth, LastYear, YearToDate, QuarterToDate,
    MonthToDate, Yesterday, Today, ThisMonth, ThisQuarter, ThisYear, PreviousMonth,
    PreviousQuarter, PreviousYear, NextMonth, NextQuarter, NextYear, TrailingTwelveMonths
}
public class RVDateRange
{
    public System.DateTime? From { get; set; }
    public System.DateTime? To { get; set; }

    public RVDateRange();
    public RVDateRange(System.DateTime? fromDate, System.DateTime? toDate);
}

public class RVFilterValue
{
    public string Label { get; }
    public object Value { get; }
    public RVFilterValue(object value, string label);
```

```
}

public interface IRVDataSourceProvider
{
    Task<RVDataSourceItem> ChangeVisualizationDataSourceItemAsync(RVVisualization
visualization, RVDataSourceItem dataSourceItem);
    Task<RVDataSourceItem> ChangeDashboardFilterDataSourceItemAsync(RVDashboardFilter
globalFilter, RVDataSourceItem dataSourceItem);
}

public interface IRVAuthenticationProvider
{
    Task<NetworkCredential> ResolveCredentials(RVDashboardDataSource dataSource);
}

public interface IRVDataProvider
{
    IRVInMemoryData GetData(RVDataSourceItem dataSourceItem);
}

public interface IRVInMemoryData
{
    IEnumerable<System.Collections.Generic.IEnumerable<object>> GetData();
    IEnumerable<RVSchemaColumn> GetSchema();
}
```

Loading dashboard from rdash file

C#

```
var fileStream = File.Open(path, FileMode.Open, FileAccess.Read);

var revealView = new RevealView();
var dashboard = await RevealUtility.LoadDashboard(stream);
revealView.Settings = new RevealSettings(dashboard);
```

Change data sources and In-Memory data support

C#

```
public Task<RVDataSourceItem>ChangeVisualizationDataSourceItemAsync(RVVisualization
visualization, RVDataSourceItem dataSourceItem)
{
    var sqlServerDsi = dataSourceItem as RVSqlServerDataSourceItem;
    if (sqlServerDsi != null)
    {
        // Change SQL Server host url
        var sqlServerDS = (RVSqlServerDataSource)sqlServerDsi.DataSource;
        sqlServerDS.Host = "http://10.0.0.20";

        // Change SQL Server database and table/view
        sqlServerDsi.Database = "Adventure Works";
        sqlServerDsi.Table = "Employees";

        return Task.FromResult(sqlServerDsi);
    }

    // Fully replace a data source item with a new one
    if (visualization.Title == "")
    {
        var sqlDs = new RVSqlServerDataSource();
        sqlDs.Host = "rpluste01";
        var sqlDsi = new RVSqlServerDataSourceItem(sqlDs);
        sqlDsi.Table = "Customers";

        return Task.FromResult(sqlDsi);
    }

    // Provide in-memory data
    if (visualization.Title == "")
    {
        return Task.FromResult(new RVInMemoryDataSourceItem("myData"));
    }
}


public IRVInMemoryData GetData(RVDataSourceItem dataSourceItem)
{
    var datasetId = ((RVInMemoryDataSourceItem)dataSourceItem).DatasetId;
    if (datasetId == "myData")
    {
```

```
        var dataList = new List<MyData>()
        {
            new MyData { StringCol = "John", Number = 34, DateTime = DateTime.Now, Date =
DateTime.Now, Time = DateTime.Now},
            new MyData { StringCol = "Jack", Number = 45, DateTime = null, Date = null,
Time = null}
        };
        return new RVInMemoryData<MyData>(data);
    }
    else
    {
        throw new Exception("Invalid data requested");
    }
}
```

Passing in dashboard filter values

Initially selected values

C#

```
var dashboard = await RevealUtility.LoadDashboard(stream);
var settings = new RevealSettings(dashboard);

settings.SetFilterSelectedValues(
              dashboard.GetFilterByTitle("Country"),
              new List<object>() { "Canada" }
              );
```

Selection change after the view is displayed

C#

```
revealView.SetFilterSelectedValues(
              dashboard.GetFilterByTitle("Country"),
              new List<object>() { "Canada" }
              );
```

Callback for providing credentials to data sources

C#

```
public Task<NetworkCredential> ResolveCredentials(RVDashboardDataSource dataSource)
{
    NetworkCredential userCredential = null;
    var sqlServerDs = dataSource as RPSqlServerDataSource;
    if (sqlServerDs != null && sqlServerDs.Host == "rpluste01 ")
    {
        userCredential = new NetworkCredential("shared", "passw0rd");
    }
    return Task.FromResult(userCredential);
}
```

Event firing for Data point (or Grid Cell) clicked

C#

```
revealView.VisualizationDataPointClicked += RevealView_VisualizationDataPointClicked;

private void RevealView_VisualizationDataPointClicked(object sender,
VisualizationClickedEventArgs e)
{
}

public class VisualizationClickedEventArgs : EventArgs
{
    public RVVisualization Visualization { get; }
    public RVDataCell Cell { get; }
    public RVDataCell[] Row { get; }
}
```

Event firing for Dashboard Linking

C#

```
 revealView.VisualizationLinkingDashboard += RevealView_VisualizationLinking;

private void RevealView_VisualizationLinking(object sender,
VisualizationLinkingDashboardEventArgs e)
{
    e.Callback("MyFileId", File.Open("MyFile.rdash", FileMode.Open, FileAccess.Read));
}

public class VisualizationLinkingDashboardEventArgs : EventArgs
{
    public VisualizationLinkingCallback Callback { get; }
    public string Title { get; }
    public string Url { get; }
}

public delegate void VisualizationLinkingCallback(String dashboardId, Stream dashboard);
```

Web Client SDK

The Reveal Web Client SDK exposes a JavaScript function called *RevealView*, which is responsible for interacting with the backend and rendering a requested dashboard.

```
function RevealView(selector, revealSettings)
```

- selector – jQuery selector where the dashboard view will be injected.
- revealSettings: an instance of RevealSettings class, containing the dashboard to be rendered and the rest of the settings for RevealView.

1. Load dashboard

```
var dashboardId = "TestDashboard";
var revealSettings = new RevealSettings(dashboardId);

RevealUtility.LoadDashboard(dashboardId, function (dashboard) {
    revealSettings.dashboard = dashboard;
    new RevealView("#revealView", revealSettings);
}, function (error) {
    Console.log(error);
});
```

2. Set dashboard parameters

```
revealSettings.setFilterSelectedValues(dashboard.getFilterByTitle("Departments"),
["CPA", "Marketing"]);
```

3. Visualization data point clicked event

```
revealSettings.onVisualizationDataPointClicked = function (visualization, cell,
row) {
    console.log(visualization.title);
 }
});
```

4. Visualization linking event

```
revealSettings.onVisualizationLinkingDashboard = function (title, url, callback) {
    callback("TargetDashboardId");
 };

});
```

5. Notify the dashboard view that its container size is changed

```
revealView.updateSize();
```