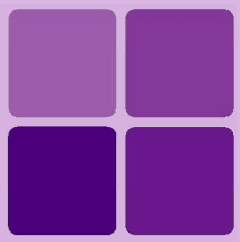

Intellicus DotNet Client Developers' Guide

Intellicus Enterprise Reporting and BI Platform

intellicus
Enterprise Reporting and
Business Insights Platform

info@intellicus.com
www.intellicus.com

Dated: - April 2014

Acknowledgements

Intellicus acknowledges using of third-party libraries to extend support to the functionalities that they provide.

For details, visit: http://www.intellicus.com/acknowledgements.htm .

Contents

Intellicus Integration Architectural Options

Intellicus DotNet Client supports IIS web server for aspx execution.

The Intellicus DotNet Client files include Intellicus ASPX Pages and Intellicus .NET API packages. Other supporting files include java script files, image files, property configuration files, etc.

These files are available at <web root folder>\ intellicus.

Integration Architectural Options

There are two recommended methods for integrating Intellicus client into a host web application.

- Intellicus portal can be run separately and can be interacted with HTTP calls, OR
- Intellicus portal aspx and .NET APIs can be copied inside the HOST application server.

Note: The options are with integrating the client components. In either case, the Report Server will run as a separate standalone server process.

Option-1: Intellicus Running on Separate Webserver

In this method Intellicus web client components are deployed in a separate application server and managed separately.

The client components run in the memory space of a separate IIS. During streaming of report output to the browser, the client components won't consume host applications' threads, temporary file space etc.

Still, this integration option can provide embedded reporting, by placing reporting output as an integral part of the host application page.

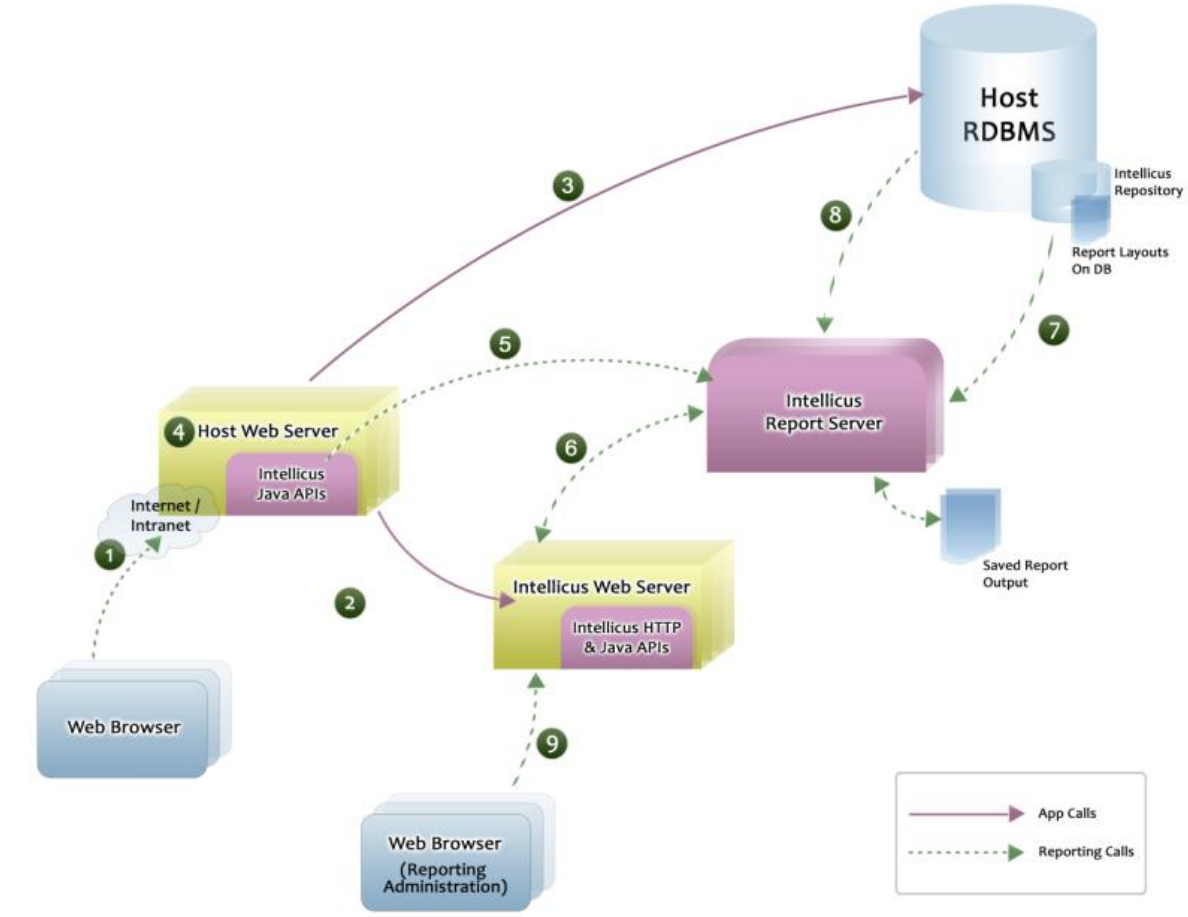

Figure 1: Integrating as a separate portal

Use cases (As numbered in the diagram)

1. Host application user login to their (Host) application.
2. User clicks Reports option for getting a report layout list etc. from the web browser and thus hit the Intellicus portal.
3. Host application user access Client database for their normal actions other than reports.
4. Client application communicates with Intellicus .NET APIs (includes core.dll, ReportClient.properties) deployed on Client app server.
5. Client uses Intellicus .NET APIs for performing Intellicus administration task programmatically.
6. Intellicus portal communicates with report server for report execution, administration etc.
7. Intellicus report server communicates with Client database for fetching report layouts.(Intellicus creates its own repository tables in the database)
8. Intellicus report server communicates with Client database and fetches data using report SQL from Client database.

9. Host application user logs in through Intellicus portal for Admin Screens using Intellicus portal login screen.

Deploying ASPX and dlls

Step 1- Install Intellicus Report Server

Run Setup.exe and Install Intellicus Report Server.

You can install Intellicus Report Server on a separate machine or on the same machine as the host application server is running.

The setup.exe installs the report server as well as the Intellicus client components.

Note: By default, Intellicus suite listens to HTTP requests on port 80. If this number conflicts with any of host application port numbers, then either of ports should be changed.

Step 2- Configure HTTP port number

Report Web Service will start automatically on port 80. To change Intellicus portal HTTP port number,

1. Open IIS management console.
2. Change port using binding properties say 8000.

The Intellicus can be now accessed as localhost:8000/intellicus.

Step 3- Setting Intellicus User Context

When it comes to integrating Intellicus with a host application it is desired that the user gets seamless experience. That means, user should not only get the same look and feel, but if the application needs user authentication, he/she should not be asked to log into Intellicus again.

Single sign-on refers to one time authentication performed by the host application. Intellicus does not perform any authentication check for the Users accessing Intellicus from Host Application as these are already authenticated. This means that User can access Intellicus without going through Intellicus Login page

Please refer IntellicusSingleSign-on.doc for more details on single sign on.

Note: IntellicusSingleSign-On.doc will be provided with Intellicus setup. Path: <Intellicus_Install_Path>\Docs\Manuals

Option-2: Intellicus Running inside a Host Application

In this method Intellicus web client components are deployed inside the host application server and managed along with the host application.

During streaming of report output to the browser, the client components run in the memory space of host application server and consume threads, temporary file space etc.

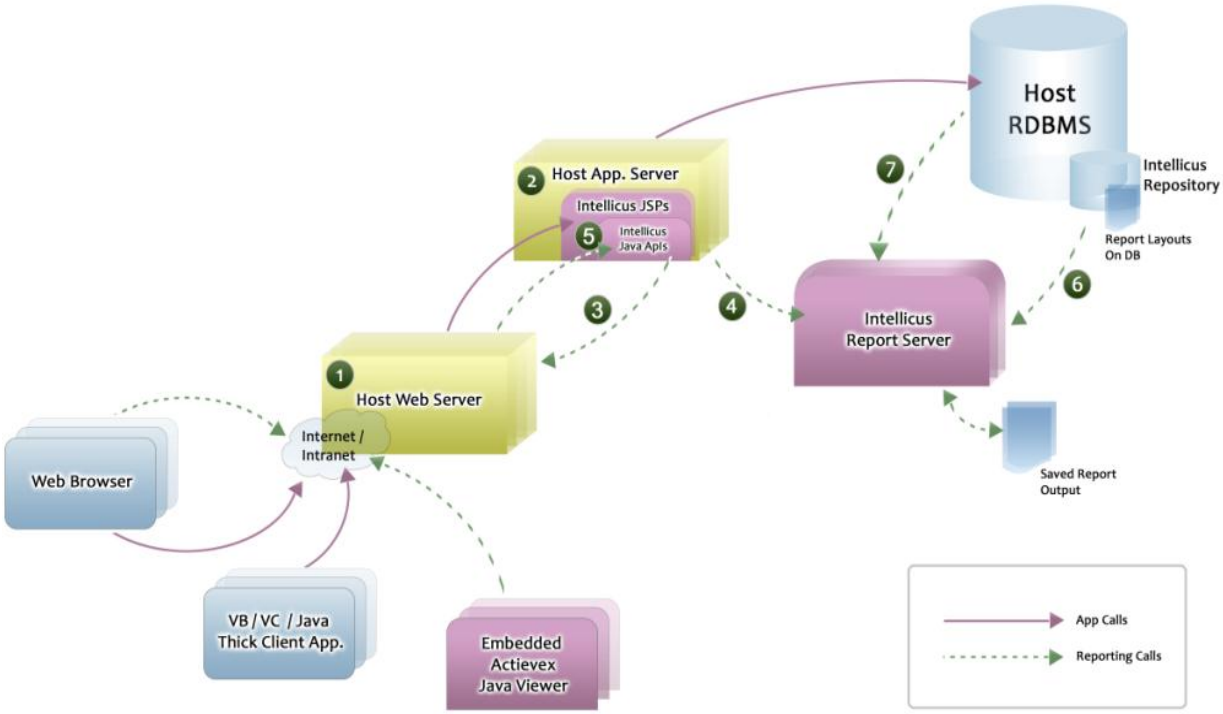

Figure 2: Integrated as embedded in host application

Use cases

1. Client application is a web-based application, which is deployed on some App Server. Intellicus Web Application is running on the same server.
2. Intellicus web Application is embedded with the Host web application.
3. The Host Web Application accesses the Intellicus pages.
4. The Host Web Application accesses the Intellicus Report Server through Intellicus Web Application.
5. The Host application uses Intellicus .NET APIs for creating/mapping Intellicus users/roles etc (Point 6 in the diagram). Host application communicates with Intellicus .NET APIs (includes core.dll, ReportClient.properties) deployed on their App server.

6. Intellicus Report Server connects to Client database; with a single-user connection pool (Intellicus creates its own repository tables in the database). Intellicus repository is also formed on the same database (may be under a different schema).
7. Intellicus report server communicates with Client database and fetches data using report SQL from Client database.

Steps to deploy Intellicus as a Embedded Application inside Host Application

Please refer "IntellicusDotNetClientIntegrationGuide.pdf" document.

HTML Look and Feel

All Intellicus aspx files use the style sheet- 'intellicus.css' to generate the HTML look-and-feel. You can change 'intellicus.css' to use your cascading style sheet file (CSS) to give the look-and-feel according to your application.

Note: Intellicus provides professional services for developing the UI skin. Please contact Intellics support for the same.

HTTP API

When Intellicus and Host application are running on different web servers as described in deployment scenario 1, then Single Sign-On need to be implemented.

Please refer IntellicusSingleSign-on.doc for more details on single sign on.

Note: IntellicusSingleSign-On.doc will be provided with Intellicus setup. Path: <Intellicus_Install_path>\Docs\Manuals

For deployment scenario 2, in which Intellicus is embedded inside Host application, Single Sign-On is not required.

Intellicus functionalities can be accessed from within an application. This is done through HTTP APIs. APIs are available for following functionalities:

- Getting category and report list
- Run, Schedule, Publish, E-Mail report
- Dashboard, Widget listing, Setting/Getting Dashboard Preferences
- Query/Parameter Object Editor
- Adhoc reporting

General steps of integration are:

- Take Intellicus Token by implementing single sign-on
- Call HTTP API for the needed functionality
- Receive response from Intellicus
- Display the information in host application

Prerequisite

If the host application's users are expected to authenticate in order to enter into the application, host application will need to integrate with Intellicus using single sign-on (user authentication by host application).

Intellicus needs to make sure that a request received from a host application is from a user already authenticated by the host application. Intellicus provides token exchange mechanism for user identification.

Before calling Intellicus HTTP APIs, we assume that token had been generated by Intellicus and received by host application with the help of single sign on.

Method to get a token for Single Sign-On

This method calls Intellicus Controller API and passes the user credentials and other hidden/business/request parameters to Intellicus.

Name: getIntellicusToken
Class: SingleSignOn

Syntax:

public String getIntellicusToken()

Returns:

TokenString: Received token from Intellicus

Method to access HTTP API

This method is called after getting a token from Intellicus. This method sets the name of Intellicus HTTP API where request is being redirected.

Name: redirectToIntellicus
Class: SingleSignOn class

Syntax:

```
public void redirectToIntellicus(String onSuccess, String intellicusToken)
```

Parameters:

- **onSuccess:** Relative URL of the requested Intellicus HTTP API.
- **intellicusToken:** Token received from Intellicus after user authentication.

CategoryList.aspx

The objective of the API is to fetch the all the categories.

REQ_CATEGORY_ID
(Optional)

This parameter is used to show only specified category.

Example: REQ_CATEGORY_ID= 4F9245A7-D639-4F99-604D-F32641B77725

REPORT_TYPE
(Optional)

This parameter is used to list the given type of reports in the specified category. Possible values for this parameter are STANDARD/STUDIO, ADHOC, OLAP and All. In case this parameter is not specified both types of reports will be listed.

Example: REPORT_TYPE= STUDIO.

Combination of the above parameters makes a relative URL that invokes category listing of Intellicus. For example:

```
String onSuccess
=/core/CategoryList.aspx?REPORT_TYPE=STUDIO&REQ_CATEGORY_ID=4F924
5A7-D639-4F99-604D-F32641B77725
```

ReportListForCategory.aspx

The objective of the API is to fetch the reports for a particular category. It also acts as an interface to perform any operations on the reports. This API is configurable on the basis of access rights and license.

With this API, user having the required access rights can perform all the report operations like run a report in desired format, schedule it, view saved reports and its description.

This API governs following behaviors:

- If the user is Super Admin or Admin he can perform all the activities.
- If the user is an end-user, and no access rights are granted, no icons will be visible.
- To quick run or run the report, 'run report' and 'publish' access rights are needed.
- To publish a report, 'view saved reports' right is needed.

- For scheduling, 'run report' and 'schedule report' rights are required.
- For editing Adhoc report, 'publish layout' rights are needed.

CATEGORY_ID
(Mandatory)

This is the id of the category in which the report being run exists. This API will list all the reports belonging to the specified category.

Example: CATEGORY_ID=4F9245A7-D639-4F99-604D-F32641B77725

CAT_NAME
(Optional)

This is the Category Menu Name to which the Report belongs.

Example: CAT_NAME=DemoCategory

REPORT_TYPE
(Optional)

This parameter is used when user wants to list which type of reports available in a specified category. Possible values for this parameter are **STANDARD/STUDIO**, **ADHOC**, **OLAP** and **ALL.**

In case this parameter is not specified, both types of reports will be listed.

Example: REPORT_TYPE=STANDARD

SEARCH_STRING
(Optional)

This parameter is used to display a list of reports having report name that is given in the SEARCH_STRING parameter.

Example: SEARCH_STRING=Country Sales

This API will list of all reports which contains 'Country Sales' in report name

FROM_DATE
(Optional)

This parameter is used for giving list of all reports which is updated from specified date.

Example: FROM_DATE = 06/06/2007

This API will list of all reports which updated from 06/06/2007 date.

TO_DATE
(Optional)

This parameter is used for giving list of all reports which is updated up to specified date.

Example: TO_DATE = 06/06/2008

This API will list of all reports which updated up to 06/06/2008 date.

If FROM_DATE and TO_DATE both parameters are given in HTTP API then it will fetch list of all reports which updated between these two dates

SHOW_RERUN
(Optional)

This parameter is used to enable/disable Re-Run option in the report listing page.

Example: SHOW_RERUN=true

This API will enable Re-Run Option in report listing.

SHOW_DELETE
(Optional)

This parameter is used to enable/disable delete option in the report listing page.

Example: SHOW_DELETE=true

This API will enable delete Option in report listing.

Combination of above parameters constructs a URL to get report listing of Intellicus reports. For example:

```
String onSuccess
=”/core/ReportListForCategory.aspx?REPORT_TYPE=ADHOC&CATEGORY_ID=
4F9245A7-D639-4F99-604D-
```

```
F32641B77725&SHOW_DELETE=true&FROM_DATE=06/06/2007&TO_DATE=06/06/2009&SHOW_RERUN=true"
```

Sample URLs:

For getting list of all the reports in the category

```
/core/ReportListForCategory.aspx?CATEGORYID=96EF065A-92DE-5F64-E2AF-C4139396DD6B
```

For getting list of Standards reports in the category

```
/core/ReportListForCategory.aspx?CATEGORYID=96EF065A-92DE-5F64-E2AF-C4139396DD6B&REPORT_TYPE=STUDIO
```

For getting list of Adhoc reports in the category

```
/core/ReportListForCategory.aspx?CATEGORYID=96EF065A-92DE-5F64-E2AF-C4139396DD6B&REPORT_TYPE=ADHOC
```

RepositoryExplorer.aspx

The objective of this API is to show the repository explorer to the User. This API is configurable on the basis of access rights and license.

With this API, user having the required access rights can view explorer for different entities like Reports, Categories, Queries, Parameter objects, Dashboards, Favorites etc.

ENTITYTYPE
(Optional)

This parameter takes entity that is to be shown in the Explorer.

Example: ENTITYTYPE = REPORT

Valid Values:
ENTITYTYPE = REPORT
ENTITYTYPE = FAVORITES
ENTITYTYPE = QUERY
ENTITYTYPE = PARAMETER
ENTITYTYPE = DASHBOARD2
ENTITYTYPE = DASHBOARD_WIDGET

In case this parameter is not specified, it takes "Repository" as default and shows Repository Explorer.

EXPLORER_TITLE
(Optional)

This parameter takes name that is to be shown in the Explorer Title.

Example: EXLPORER_TITLE = Reports

Combination of above parameters constructs a URL to get Object Explorer for Query Objects. For example:

```
String onSuccess
=”/core/RepositoryExplorer.aspx?ENTITYTYPE=QUERY&EXPLORER_TITLE=Q
ueries
```

Now pass above onSuccess parameter to redirectToIntellicus method of SingleSignOn class.

InteraController.aspx

This API is the main controller of Intellicus report server. All the reports related requests to the report server are passed through this API. This controller is used for both Standard and Adhoc reports. Depending on the Action code given, this controller will ask report server to do the required task.

InteraController.aspx is used to:

- Quick run a report
- Execute and view a report in a specific output format
- Execute and deliver a report as email, publish or printout
- View published report
- Schedule a report

This jsp accepts system parameters and business parameters.

System Parameters

ACTION_CODE
(Mandatory)

This parameter is to specify the action that this API will initiate. A list of action-codes and actions is provided in the table given below.

Action Code	Description
000	Takes navigation to system parameter page. From here a user can select report delivery option, database connection, output format and other options before running the report.
001	If the report has user parameters, it takes navigation to Intellicus user parameters page. User can specify values for parameters from this screen and can execute the report. If report does not have any parameters, it executes report directly.
010	To Run Report with previous run report parameter values.
002	Executes report and navigates to report viewer screen.
003	Request to show all the system, report parameter in one jsp.
300	Request to show all the system, eport parameter in one jsp.
004	Open Scheduled Repots in viewer.
005	Open published report in viewer.
400	Submission of edit parameter form and request for updated report.
500	Save dialog submission and request to save report.
600	Delete report layout.
700	Delete saved report.
800	Generate dynamic report.
802	Dynamic report generation for drilldown reports.
900	Promptable information check and execution of dynamic adhoc report with no parameter.
901	Execution of adhoc report after taking promptable information from AdHocWizardRun.aspx.
902	Promptable information check and execution of saved adhoc report with no parameter.

Example: ACTION_CODE=002

REPORT_ID
(Mandatory)

This is the Id of the report being run. User can view report id from Intellicus portal's "Deploy Categories and Reports" page.

Example: REPORT_ID=96EF065A-92DE-5F64-E2AF-C4139396DD6B

DSGN_MODE
(Optional)

This parameter is to specify designer of the report. The way report is executed, also depends on its design mode.

Valid values are:

- **STUDIO:** Specify this value when report being run is designed in Intellicus Studio (these reports are also known as Standard Reports).
- **ADHOC:** Specify this value when report being run is designed in Adhoc designer (these reports are also known as Adhoc Reports).

Default value: STUDIO
Example: DSGN_MODE=STUDIO

MENU_NAME
(Optional)

This is the name of the report being run. Report name is used in the UI for identifying the report.

Example: MENU_NAME=Sales by Country

CATEGORY_ID
(Optional)

This is the id of the category in which the report being run exists. User can view category id from Intellicus portal's "Deploy Categories and Reports" page.

Example: CATEGORY_ID=4F9245A7-D639-4F99-604D-F32641B77725

REPORT_FORMAT
(Optional)

This is the output format in which report is to view. Possible values for this parameter are **rdf, pdf, htm, xls, txt, rtf, ppt, dhtm(iHTML), csv, doc, xml, rwt(Rawtext), dhtm2(SMART)**

Default value: htm
Example: REPORT_FORMAT=htm

REPORT_CONN_NAME
(Optional)

This parameter is to specify name of the database connection to be used for running the report. Specified connection must exist in report server.

Example: REPORT_CONN_NAME=ReportDB

OPERATION_TYPE
(Optional)

This parameter is to specify type of the operation requested. Possible values are VIEW, EMAIL, SAVE, PRINT, PRINT_AT_SERVER.

- **VIEW:** Generated report is sent to the report viewer for display.
- **EMAIL:** Generated report is E-mailed as an attachment or hyperlink.
- **SAVE:** Report output is saved as snapshot instead of being displayed.
- **PRINT:** Report is sent to the local printer instead of being displayed.
- **PRINT_AT_SERVER:** Report is sent to the printer on report server instead of being displayed.
- **PRINT_LOCALLY:** Report is sent to the local printer instead of being displayed.

Default value: VIEW
Example: OPERATION_TYPE=VIEW

HTM_MULTIPAGEOUTPUT
(Optional)

This parameter is used to specify whether report output should be in multiple pages or single page. Possible values are true and false.

- **True:** Displays the report using multiple HTML pages.
- **False:** Displays report using a single HTML page by merging all report pages into one.

This parameter is used when OPERATION_TYPE is VIEW and REPORT_FORMAT is HTM.

Default value: true
Example: HTM_MULTIPAGEOUTPUT=true

HTM_SHOWTOOLBAR
(Optional)

This parameter is used to specify whether toolbar in the html report output should be displayed or not. Possible values are 0,1 and 2 for

SHOW_NEVER, SHOW_ALWAYS, and SHOW_WHEN_MULTIPAGE respectively.

- **SHOW_NEVER:** Never shows the toolbar in HTML viewer. The viewer cannot navigate to further pages if the report has many pages.
- **SHOW_ALWAYS:** Always shows the toolbar in HTML viewer.
- **SHOW_WHEN_MULTIPAGE:** The toolbar is shown only when the report generates more than one page. Otherwise, when the report generates only one page the toolbar is hidden.

This parameter is used when OPERATION_TYPE is *VIEW* and REPORT_FORMAT is *HTM*.

Default value: SHOW_ALWAYS
Example: HTM_SHOWTOOLBAR= 1

SHOW_NEVER=0
SHOW_ALWAYS=1
SHOW_WHEN_MULTIPAGE=2

IRLDATA
(Optional)

It is used to get the IRL xml from the request object.

ARLDATA
(Optional)

It is used to get the ARL xml from the request object.

REQUESTTYPE
(Optional)

It is used to specify the type of Request related to particular Report whether to Cancel the Running Report or not.

Example: REQUESTTYPE = CANCEL

EXECUTIONTYPE
(Optional)

It is used to specify the Execution Type whether Run, Run In Background, or Scheduled

Example:
EXECUTIONTYPE = : ALL
EXECUTIONTYPE = **DIRECT**: Run
EXECUTIONTYPE = **SCHD**: Scheduled Report
EXECUTIONTYPE = **ASYNC**: Run in Background

STATUS
(Optional)

It is used to specify the Status of Report.

Example:
STATUS = : All
STATUS = **UNDERPROCESS**: Running
STATUS = **COMPLETED**: Completed

FROMDATE
(Optional)

This parameter is used for giving list of all reports updated from specified date.

Example: FROMDATE = 06/06/2007

This API will list of all reports updated from 06/06/2007 date.

TODATE
(Optional)

This parameter is used for giving list of all reports updated up to specified date.

Example: TODATE = 06/06/2008

This API will list of all reports updated up to 06/06/2008 date.

If both parameters, FROMDATE and TODATE are given in HTTP API then it will fetch list of all reports updated between these two dates.

ORPHAN
(Optional)

This specifies whether Orphan option is selected in Look in Category i.e. FRO MCATEGORY or from ReportLayout i.e. FROMREPORT.

Example: ORPHAN = **FROMCATEGORY**

APP_PRO_STATUS
(Optional)

This specifies the approval process status value.

Example: APP_PRO_STATUS = **All**

IS_DIRECT_EXEC
(Optional)

This specifies whether the Report should directly Run or show Parameters.

Example: IS_DIRECT_EXEC = False i.e. Show Parameters and then run the Report.

Business Parameters
(Optional)

Apart from system parameters and user parameters, host application can pass business parameters to Intellicus. These parameters can be used for enforcing authorization. Also business parameters can be used in report SQL for filtering records.

These parameters are used to specify query parameter names (and their respective values) in the HTTP URL.

Suppose we have a report having certain parameters, based on which it will fetch some records. For example a report named "country details" fetches records on the basis of country name. Parameter name is "prmCountry".

So we will send prmCountry parameter and its value in the URL.

Example: prmCountry='Brazil'

The URL to run a report (in HTML) accepting prmCountry as a run time parameter, the URL would be:

```
String onSuccess
="/InteraController.aspx?
ACTION_CODE=002&OPERATION_TYPE=VIEW&DSGN_MODE=STUDIO&REPO
RT_ID=2030DDEA-841B-E360-3CA0-
5954AA945B92&REPORT_FORMAT=htm&prmCountry='Brazil'
"
```

Sample URLs:

Run-time System parameters page

```
/InteraController.aspx?ACTION_CODE=000&REPORT_ID=96EF065A
-92DE-5F64-E2AF-C4139396DD6B
```

Input parameters page

```
/InteraController.aspx?ACTION CODE=001&REPORT ID=96EF065A
-92DE-5F64-E2AF-C4139396DD6B
```

For running standard report

```
/InteraController.aspx?ACTION_CODE=002&OPERATION_TYPE=VIEW&DSGN_MODE=STUDIO&REPORT_ID=2030DDEA-841B-E360-3CA0-5954AA945B92&REPORT_FORMAT=htm
```

For running Adhoc report

```
/InteraController.aspx?ACTION_CODE=002&OPERATION_TYPE=VIEW&DSGN_MODE=ADHOC&REPORT_ID=2030DDEA-841B-E360-3CA0-5954AA945B92&REPORT_FORMAT=htm
```

For running Adhoc report in pdf format

```
/InteraController.aspx?ACTION_CODE=002&OPERATION_TYPE=VIEW&DSGN_MODE=ADHOC&REPORT_ID=2030DDEA-841B-E360-3CA0-5954AA945B92&REPORT_FORMAT=pdf
```

OlapViewer.aspx

The objective of API is to show the list of OLAP reports available in the Repository.

This API takes below parameter

LAYOUT_ID
(Optional)
This parameter is used for opening a saved layout. When this parameter is specified, layout saved with this id should be opened and viewed in specified web browser's view port. Any valid Cube Layout ID saved in repository

```
 String                        onSuccess                        =
/olap/OlapViewer.aspx?LAYOUT ID=F46DB80D-C532-5069-F7A7-A43D726CB663
```

VIEW_MODE
(Optional)
This parameter is used to specify the view in which user wants to open the Viewer.

Possible Values

- GRID: To open the Viewer in Grid view

- CHART: To open the Viewer in Chart view
- DUAL: To open the Viewer in Dual Panel view

```
String                    onSuccess                    =
/olap/OlapViewer.aspx?LAYOUT_ID=F46DB80D-C532-5069-F7A7-
A43D726CB663&VIEW_MODE=CHART
```

DATA_ACTIONS
(Optional)
This parameter is used to specify whether User should be allowed to change data on Grid and Chart.

Possible Values

- TRUE: Allow Data Refreshing.
- FALSE: No Data Refreshing.
-

Default Value

- TRUE

```
String                    onSuccess                    =
/olap/OlapViewer.aspx?LAYOUT_ID=F46DB80D-C532-5069-F7A7-
A43D726CB663&DATA_ACTIONS=FALSE
```

EXPLORER
(Optional)
This parameter is used to specify the desired state of Explorer Panel in the viewer.

Possible Values

- SHOW: To show Explorer Panel in expanded form.
- HIDE: To hide Explorer Panel
- COLLAPSED: To show Explorer Panel in collapsed form.

Default Value

- SHOW

Exceptions

- If *DATA_ACTIONS* =FALSE, then its value is always forced to HIDE

DISPLAY_TITLE
(Optional)
This parameter is used to specify whether title for layout should be displayed or not.

Possible Values

- SHOW: To show the title
- HIDE: To hide the title

Default Value

- SHOW

LAYOUT_ACTIONS
(Optional)
This parameter is used to specify whether Save and Open buttons should be displayed or not.

Possible Values

- TRUE: To show Save and Open buttons (Default)
- FALSE: To hide Save and Open buttons

Default Value

- TRUE

CONN_LIST
(Optional)
This parameter is used to specify whether Connection List should be displayed or not and if displayed it should be enabled or not.

Possible Values

- SHOW: To show enabled Connection list.
- HIDE: To hide connection list.
- DISABLE: To disable connection list.

Default Value

- SHOW

CO_LIST
(Optional)
This parameter is used to specify whether Cube Object List should be displayed or not and if displayed it should be enabled or not.

Possible Values

- SHOW: To show enabled Cube Object list.
- HIDE: To hide Cube Object list.
- DISABLE: To disable Cube Object list.

Default Value

- SHOW

OLAP_CONN_NAME
(Optional)

This parameter is used to specify the name of connection which should be selected by default in connection listing if present.

CO_NAME
(Optional)
This parameter is used to specify the name of Cube Object which should be selected by default in Cube Object listing.

TOOLBAR
(Optional)
This parameter is used to specify whether Toolbar should be displayed or not.

Possible Values

- SHOW: To show the Toolbar.
- HIDE: To hide the Toolbar.

Default Value

- SHOW

TOOLBAR_OPTIONS
(Optional)
This parameter is used to specify the toolbar options to be displayed.

The value is a comma-separated list containing the options which should be displayed in the toolbar.

- GRIDVIEW
- CHARTVIEW
- DUALVIEW
- SELECTCHART
- SWAPAXES
- CLEAR
- ACTION
- EXPORT
- ALL

Default Value

- ALL

SLICER
(Optional)
This parameter is used to specify whether Slicer Bar should be displayed or not.
Possible Values

- SHOW: To show the Slicer.
- HIDE: To hide the Slicer.

Default Value

- SHOW

SavedReportList.aspx

The objective of API is to show the list of saved reports available for the given report. Each saved report provides us with following details:

- Saved Report Name
- Published By
- Generation Time
- Expiry Time
- Comments

In addition to the above details, following options are also available:

- Option to view each saved report in any of the supported report output format.
- Option to view the comments provided by user(s) on that saved report. If collaboration is disabled in the license then when user clicks on the comment icon a message would pop-up indicating the same.
- Option to delete any of the saved report-instance(s).
- The Administrator (Super-Admin and Admin) are also provided with the option of viewing privately saved reports of other users.

REPORT_ID
(Mandatory)

This is the Id of the report for which saved reports list is requested.

Example: REPORT_ID=96EF065A-92DE-5F64-E2AF-C4139396DD6B

ISPUBLIC
(Optional)

This is to specify whether the report is public or private. Possible values for this parameter are true and false. In case this parameter is not specified both types of reports will be listed.

Example: ISPUBLIC=true

Combination of above parameters makes a URL that invokes saved report listing of Intellicus.

MENU_NAME
(Optional)

This is the name of the report being run. Report name is used in the UI for identifying the report.

Example: MENU_NAME=Sales by Country

CATEGORY_ID
(Optional)

This is the id of the category in which the report being run exists. User can view category id from Intellicus portal's "Manage Folders and Reports" page.

Example: CATEGORY_ID=4F9245A7-D639-4F99-604D-F32641B77725

FROM_DATE
(Optional)

This parameter is used for giving list of all reports updated from specified date.
(FROM_DATE shuold be in MM/DD/YYYY format).

Example: FROM_DATE = 06/06/2007

This API will list of all reports which updated from 06/06/2007 date.

TO_DATE
(Optional)

This parameter is used for giving list of all reports updated up to specified date.
(TO_DATE shuold be in MM/DD/YYYY format).

Example: TO_DATE = 06/06/2008

This API will list of all reports updated up to 06/06/2008 date.

If FROM_DATE and TO_DATE both parameters are given in HTTP API then it will fetch list of all reports updated between these two dates.

DEPTH
(Optional)

This parameter is to specify level of searching i.e. whether COMPLETE or CURRENT_LEVEL.
COMPLETE = -1 (Searches in complete multilevel category hierarchy)
CURRENT_LEVEL = 0 (Searches in current Category of multilevel hierarchy)

ACCESSRIGHTS
(Optional)

This parameter is to specify Access Rights for the Report.

CATACCESSRights
(Optional)

This parameter is to specify Access Rights for the given Category.

CALLEDFROM
(Optional)

It specifies the page, from where it is called.

Example: CALLEDFROM = VIEWER

ALL_USER_RPTS
(Optional)

This is the Request Parameter that is used to decide whether to get all the user's published report or not.

For example: **ALL_USER_RPTS** = true

```
String onSuccess = /core/ SavedReportList.aspx?
REPORT_ID=96EF065A-92DE-5F64-E2AF-C4139396DD6B&
ISPUBLIC=true&DEPTH=-1
```

Now pass above onSuccess parameter to redirectToIntellicus method of SingleSignOn class.

Dashboard

Objective of API: To load any dashboard by default or for editing any existing one. There are three types of APIs for dashboard:

- Dashboard Viewer API
- Widget Designer API
- Dashboard Preferences

DashboardViewer.aspx

This API takes below parameter.

DASHBOARD_ID: The unique identifier of the Dashboard that should be loaded for editing in the Dashboard viewer.

This parameter makes a URL that invokes Dashboard Viewer page of Intellicus.

For example:

```
String onSuccess =
/dashboard/DashBoardViewer.aspx?DASHBOARD_ID=DE65F377-
5E46-153B-A56E-54E1073D59B2
```

SELECTED_DASHBOARD_ID: This parameter shall be respected only if the dashboard with this id is in user's preferences.

For example:

```
String onSuccess =
/dashboard/DashBoardViewer.aspx?SELECTED_DASHBOARD_ID=B95
4111B-1881-21CE-1958-0DB411187785
```

SHOW_FULL_SCREEN: This option is to show full screen option or not. If the value is false then it will not show the full screen option.

This parameter makes a URL that invokes Dashboard Viewer page with/without SHOW_FULL_SCREEN icon

For example:

```
String onSuccess =
/dashboard/DashBoardViewer.aspx?DASHBOARD_ID=DE65F377-
5E46-153B-A56E-54E1073D59B2&SHOW_FULL_SCREEN=true
```

WidgetDesigner.aspx

This API takes following parameters:

WIDGET_ID: The unique identifier of the Widget that should be loaded for editing.

This parameter makes a URL that invokes Widget Designer page of Intellicus.

SHOW_TAB (Optional)**:** This parameter is used to display Report Tabs on Widget Designer.

Possible Values: ALL/REALTIME/PREGEN
Default Value: ALL

DEFAULT_TAB (Optional)**:** This parameter is used to select default Report tab (Real Time or Pre-generated) on loading of Widget Designer.

Possible Values: REALTIME/PREGEN
Default Value: REALTIME

For example:

```
String onSuccess =
/dashboard/WidgetDesigner.aspx?WIDGET_ID=1262845995956819
2168336132613160&SHOW_TAB=ALL&DEFAULT_TAB=PREGEN
```

DashboardPreferences.aspx

Below URL invokes the Dashboard Preferences page of Intellicus.

For example:

```
String onSuccess =/dashboard/DashBoardPreferences.aspx
```

AdhocWizard.aspx

Objective of the API: To load the Ad Hoc report designer and facilitate the user to perform all the desired activities on that UI. Moreover it is used to design new report as well as open and edit a saved report.

Parameters

- **DEFAULTEXPANDTABS:** All or None.
- **SYS_TEMPLATE_NAME:** is send as request to show template. It needs no validations and no pre defined values.
- **CATEGORY_ID:** Saves report in predefined category.(Only when designing a new Report and saving it)
- **REPORT_ID:** Opens the Report with given Report_Id for Editing.
- **LINK_TO_IDENTIFIER:** Provides tab identifier for launching use case on click of Query Editor.
- **REPORT_FORMAT**: This request parameter is used for governing default value for report format
- **WIZARD_LAYOUT_TYPE**: This request level parameter is used for defining layout type of Adhoc Wizard.

  Possible Values: TABBED/ACCORDION
- **SHOW_TAB*:*** HTTP request level parameter for selecting particular tab or section of Adhoc Wizard. This could be comma separated values.

Possible Values: SELECT/GROUP/FILTER/TOTAL/SORT/HIGHLIGHT/ MATRIX/CHART/NETWORK_GRAPH

- **SHOW_CHART_TYPE***:* HTTP request level parameter for showing specified chart type as default selected.

  Possible Values: PIE/BAR/LINE/CURVE/CURVEAREA/AREA/SCATTER/RADAR/BUBBLE

There are no mandatory parameters required in this case.

Combination of above parameters makes a URL that invokes AdhocWizard report design page of Intellicus.

For example:

Creating a new report

```
String onSuccess=/custom/AdHocWizard.aspx?
DEFAULTEXPANDTABS=All
```

Editing an existing report

```
String onSuccess=/custom/
AdHocWizard.aspx?DEFAULTEXPANDTABS=All&CATEGORY_ID=4F9245
A7-D639-4F99-604D-F32641B77725
```

Now pass above onSuccess parameter to redirectToIntellicus method of SingleSignOn class.

SelectAdhocSource.aspx

Objective of the API: To load the Ad Hoc report visualizer with Data source selection selection. This page is called when we open Adhoc visualizer from Navigation.

Adhoc Visualizer with Data Source selection Screen

```
String onSuccess=/custom/SelectAdhocSource.aspx
```

Now pass above onSuccess parameter to redirectToIntellicus method of SingleSignOn class.

AdhocVisualizer.aspx

Objective of the API: To load the Ad Hoc report visualizer and facilitate the user to perform all the desired activities on that UI. Adhoc Visualizer allows user to directly view the report data with all columns as default and then the user has provision to change certain things like selected columns, filters, grid view, chart view or Map view, apply grouping, sorting, filters, highlighting, total etc. Also the user can select existing report and view from this visualizer.

Parameters

VIEW_USECASE:

(Optional)

This parameter takes value of VIEW_USECASE parameter to allow user to open visualizer for various usecases like NEW for opening new report with given QUERY_ID or SAVED for opening existing report for given REPORTID etc.

Possible Values

- NEW- Used to design a new Report.QUERY_ID is mandatory. (Default)
- SAVED- Open Existing Report(REPORTID is mandatory)
- PUBLISHED- Open Saved instance of Report(REPORTOID is mandatory)
- REQUEST- To Open Adhoc Visualizer using ARLDATA.
- RUNINBACKGROUND- Run Smart Report in background(REPORTID is mandatory)
- AUDIT- Called for Auditted Reports (REPORTOID is mandatory)
- CHANGEDATASOURCE- To Open Report with different Data Source

Adhoc Visualizer with Data Source loaded for designing New Report

```
String onSuccess=/custom/AdhocVisualizer.aspx?
VIEW_USECASE=NEW&QUERY_ID=126077293450092032129192910671882
```

Adhoc Visualizer opened for Existing Report

```
String onSuccess=/custom/AdhocVisualizer.aspx?
VIEW_USECASE=SAVED&REPORT_ID=AF654266-1477-EE8A-BFB1-5D0A2A59A4BE
```

QUERY_ID:

This parameter is mandatory for VIEW_USECASE as NEW. It takes Query Id which is to be loaded in Adhoc Visualizer.

Open Adhoc Visualizer with Data Source loaded

```
http://localhost:91/intellicus/custom/AdhocVisualizer.aspx?VIEW
_USECASE=NEW&QUERY_ID=1260772934500920312919291067188 2
```

REPORTID:

This parameter is mandatory for VIEW_USECASE as SAVED. It takes Report Id so as to load the existing Report in Adhoc Visualizer.

Open Adhoc Visualizer with Report loaded

```
http://localhost:91/intellicus/custom/AdhocVisualizer.aspx?VIEW
_USECASE=SAVED&REPORT_ID=AF654266-1477-EE8A-BFB1-5D0A2A59A4BE
```

EDIT_MODE:

(Optional)

This parameter takes value of **EDIT_MODE** parameter to allow user to open visualizer in Edit mode or View Mode.

Possible Values

- TRUE: Opened with Edit Mode
- FALSE: Opened in View Mode.

Adhoc Visualizer with Data Source loaded in view mode

```
String onSuccess=/custom/AdhocVisualizer.aspx?
QUERY_ID=1260772934500920312919291067188 2&EDIT_MODE=FALSE
```

Adhoc Visualizer with Data Source loaded in Edit mode

```
String onSuccess=/custom/AdhocVisualizer.aspx?
QUERY_ID=1260772934500920312919291067188 2&EDIT_MODE=TRUE
```

Combination of above parameters makes a URL that invokes AdhocWizard report design page of Intellicus.

Now pass above onSuccess parameter to redirectToIntellicus method of SingleSignOn class.

QueryObjectList.aspx

To display the detail of a query object.

Parameters

QUERY_ ID
(Optional)

To specify the query object's Id to view details.

```
onSuccess =
/custom/reportobjects/QueryObjectList.aspx?QUERY_ID=1260773289269
86838754445
```

QUERY_NAME
(Optional)

To specify the query object's name to view its details.

```
onSuccess =
/custom/reportobjects/QueryObjectList.aspx?QUERY_NAME=Sales
Detail
```

ParameterObjectList.aspx

To display the detail of a query object.

Parameters

PARAMETER_NAME
(Optional)

To specify the Parameter object's name to view its details.

```
onSuccess =
/custom/reportobjects/ParameterObjectList.aspx?PARAMETER NAME=prm
Country
```

PARAMETER_ID
(Optional)

To specify the Parameter Id of the parameter to view its details.

```
/custom/reportobjects/ParameterObjectList.aspx?PARAMETER_ID=12646
774660505127018091194654991
```

PrintSettingList.aspx

This API is used to call page that have functionalities of working with print related settings. When a report is associated with a print setting, report is printed as per the preferences set in the associated print setting.

For example:

```
onSuccess =
/core/PrintSettingsList.aspx?PrintSettingsName=PrintSetting1
```

Preferences.aspx

This API provides UI to set following user preferences:

- Change password
- Set default connection
- Set portal preferences
- Set parameter preferences
- Set dashboard preferences
- Messenger preferences

For example:

```
onSuccess = personalization/Preferences.aspx
```

.NET API

Intellicus .NET APIs can be used for performing activities like User management, and Report Management in Intellicus.

To use the .NET APIs of Intellicus, following dlls should be placed in bin folder of host application.

- Intellicus.core.dll
- log4net.dll
- Intellicus.XercesUtilities.dll
- Ionic.Zip.dll
- Intellicus.NExcel.dll

These dlls are provided with intellicus setup.

Mandatory Step to Use .NET APIs

1. The Development on host application is started.
2. All the steps of installation and integration of Intellicus DotNet Client are complete.
3. Reference of following Intellicus DotNetClient dll are added in host application
   - Intellicus.core.dll
   - log4net.dll
   - Intellicus.XercesUtilities.dll
   - Ionic.Zip.dll
   - Intellicus.NExcel.dll
4. Store user information in session

   Intellicus Report Server needs UserInfo details before it can respond to a user request. To do this, an instance of UserInfo class should be created and validated when user logs-in into the host application. After the successful login UserInfo object should be stored in the Session as mentioned in the following sample code.

```
//Using the required Intellicus DotNet Client namespaces.
using UserInfo = com.intellica.client.common.UserInfo;
using TimeZoneConfig =
com.intellica.client.config.TimeZoneConfig;
```

```
using SecurityManager =
com.intellica.client.security.SecurityManager;
using BaseException =
com.intellica.client.exception.BaseException;
using InteraConstants =
com.intellica.client.common.InteraConstants;
using ResourceBundler =
com.intellica.client.config.ResourceBundler;
using ConfigManager=com.intellica.client.config.ConfigManager;

// Create UserInfo object.
UserInfo userInfo = new
UserInfo(txtUserName.Value,txtPassword.Value,txtOrg.Value);

// Set desired locale.
userInfo.Locale = txtLocale.Value;
if (!"".Equals(txtLocale.Value))
{
  userInfo.LocaleStatus = 1;
}

//Set desired timezone.
TimeZoneConfig timeZoneConfig = new TimeZoneConfig();
timeZoneConfig.TimeZoneId = txtTimeZone.Value;
if (!"".Equals(txtTimeZone.Value))
{
  userInfo.TimeZoneStatus = 1;
}

//Set other time zone properties.
userInfo.TimeZoneConfig = timeZoneConfig;
userInfo.BrowserTimeZone = BROWSER_TZ.Value;

//Set session id.
userInfo.SessionId = Session.SessionID;

//Declare authenticated user info.
UserInfo authenticatedUserInfo;

try
{
//Get an instance of SecurityManager class.
SecurityManager securityManager = SecurityManager.Instance;
```

```
// Get the authenticated UserInfo object.
authenticatedUserInfo = securityManager.getUserDetail(userInfo);

// Set properties again for further requests usage.
authenticatedUserInfo.Password = txtPassword.Value;
authenticatedUserInfo.Locale = txtLocale.Value;
authenticatedUserInfo.TimeZoneConfig = timeZoneConfig;
authenticatedUserInfo.BrowserTimeZone = BROWSER_TZ.Value;
authenticatedUserInfo.LocaleStatus = userInfo.LocaleStatus;
authenticatedUserInfo.TimeZoneStatus = userInfo.TimeZoneStatus;
authenticatedUserInfo.SessionId = Session.SessionID;

// Get desired resource bundler object.
ResourceBundler bundler =
ConfigManager.getResourceBundler(txtLocale.Value);

//Add bundler into session.
Session.Add(InteraConstants.InitParams.RESOURCE_BUNDLE, bundler);

//Add UserInfo object into session.
Session.Add(InteraConstants.STR_USERINFO, authenticatedUserInfo);

// Reading Intellicus Relative Path from Web.Config.
// This value should be set in Intellicus Integration.
string strIntellicusPath =
System.Configuration.ConfigurationSettings.AppSettings["Intellicu
s Relative Path"];

//Redirecting to desired Intellicus web page, for e.g. category
listing page.
Response.Redirect(Request.ApplicationPath + strIntellicusPath
+"/core/CategoryList.aspx",false);
```

Use Cases

User Management Actions

Using

```
//User Management usings
using com.intellica.client.security.SecurityManager;
using com.intellica.client.exception.ISecurityException;
using com.intellica.client.security.Authentication;
```

```
//Organization specific usings
using com.intellica.client.security.OrgInfo;

//User/Role specific usings
using com.intellica.client.security.RoleInfo;
using
com.intellica.client.common.Enums.SecurityTypes.ReportPrivileges;
using com.intellica.client.common.Enums;
```

Organization

GetOrgById

This .NET API is used to get an Organization for given Organization Id.

Steps :

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get the Organization by OrgId and get its various details.

Method: getOrganizationById

```
public getOrganizationById(String orgId,UserInfo userInfo)
```

Method will return the organization Info for given orgId

Parameters:

- **orgId**: Id of the requested OrgInfo object
- **userInfo**: The current user details

Returns:
OrgInfo object having the given orgId (may return null)

```
//This will get the Information about any existing Organization
String organisationId="Test";
OrgInfo orgObj = sMgr.getOrganizationById( organisationId,
                                           requestorUserInfo);
```

```
Response.Write ("OrganizationName: "+orgObj.OrgId);
Response.Write("Description: "+orgObj.OrgDescription);
Response.Write ("OrganizationMapType: "+orgObj.MapType);
Response.Write("Authorization mode: "
            +orgObj.AuthorizeMode);
Response.Write("AuthenticationObject:
            "+orgObj.AuthenticationObj);
```

Get Users list for Organization

This .NET API is used to get list of users for the given organization in the Intellicus Repository. User is required to provide the organisation id, whose list of users, the requesting user wants to retrieve.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr = SecurityManager.Instance;
```

4. Get the Arraylist containing all users in the given Organization

Method: getUserListForOrganization

```
public ArrayList getUserListForOrganization(String orgId,
 userInfo)
```

This method returns the list of users for the given Organization ID.

Parameters:

OrgId: of the userInfo list to be retrieved.

Returns:

ArrayList of UserInfo Bean class

```
ArrayList arrList=new ArrayList();
String orgId="Intellica";
arrList = sMgr.getUserListForOrganization(orgId,
requestorUserInfo);
```

Get Organization List

This .NET API is used to get the list of all organizations present in the Intellicus repository.

Steps:

1. Initialize Report Client.

2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get the Arraylist containing all users in the given Organization

Method: getOrgList

```
public ArrayList getOrgList(UserInfo userInfo)
```

This method returns the list of all organizations created in Intellicus repository.

Returns:

ArrayList of OrgInfo objects.

```
//An arraylist created to hold the list of organizations in it.
ArrayList arrList=new ArrayList();

//Method returns list of all organizations in Intellicus repository
arrList=sMgr.getOrgList(requestorUserInfo);
```

Add Organization

This .NET API is used to add a new organization in the Intellicus Repository and set the authentication mode.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create instance of OrgInfo.

```
String organisationId="Test";
OrgInfo addOrg=new OrgInfo(organisationId);
```

4. Set Authentication mode.

Method: setAuthenticateMode

public void setAuthenticateMode(int authenticateMode)

Authentication Mode Id values
1= Intellicus

(com.intellica.client.common.Enums.SecurityTypes.Authentication.*REPORTINGSYS)*

2= External Application
(com.intellica.client.common.Enums.SecurityTypes.Authentication.
EXTERNALAPP)

3= Host Application
(com.intellica.client.common.Enums.SecurityTypes.Authentication. *HOSTAPP)*

4= Callback
(com.intellica.client.common.Enums.SecurityTypes.Authentication. *CALLBACK)*

This identifies who will authenticate the users/roles belonging to this organization.

Parameters:
`authenticateMode` - : Authentication Mode Id's value

```
Authentication authInfo = new Authentication();
authInfo.setAuthenticateMode(com.intellica.client.common.Enums.SecurityTypes.Authentication.HOSTAPP);
addOrg.setAuthenticationObj(authInfo);
```

5. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

6. Add the Organization.

Method: addOrganization

```
public void addOrganization( orgInfo, userInfo)
```

This method adds a new Organization at the Report Engine.

Parameters:

- **OrgInfo:** The Organization details
- **UserInfo:** The User details

```
sMgr.addOrganization(addOrg, requestorUserInfo);
```

Modify Organization

This .NET API is used to modify the Organization's details.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Get the Organization i.e. to be deleted By its Organization Id.

```
String organisationId="Test";
OrgInfo modOrg=sMgr.getOrganizationById( organisationId,
requestorUserInfo);
```

4. Set the attributes of Organization

```
modOrg.setCanDelete(true);
String description="THIS IS A TEST ORGANIZATION";
modOrg.setOrgDescription(description);
```

5. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

6. Modify the Organization.

Method: modifyOrganization

```
public              void              modifyOrganization( orgInfo,
 userInfo)
```

This method modifies an Organization at the Report Engine

Parameters:

- **orgInfo**: The Organization details
- **userInfo**: The current user

```
sMgr.modifyOrganization(modOrg,requestorUserInfo);
```

Delete Organization

This .NET API is used to delete an existing organization from the Intellicus repository.

Steps:

1. Initialize Report Client.

2. Initialize Requestor UserInfo.
3. Get the Organization i.e. to be deleted By Organization Id.

```
String organizationId="Test";
OrgInfo delOrg = sMgr.getOrganizationById(organizationId,
requestorUserInfo);
```

4. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

5. Delete the Organization.

Method: deleteOrganization

```
public void deleteOrganization( orgInfo,
 userInfo)
```

This method deletes an Organization at the Report Engine from the Intellicus Repository.

Parameters:

- **orgInfo**: The Organization details
- **userInfo**: The current user details

```
sMgr.deleteOrganization(delOrg,requestorUserInfo);
```

Assign Category Privileges to Organization

This .NET API is used to assign the Access privileges of a Category to the specific Organization.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of AppInfo that contains the Organization to which the access rights are to be assigned.

```
  String userId="";
  String OrgId="TestOrg";
  //AppType for which the privileges are to be set, "" for Org,
"USER" for User and "ROLE" for Role
  String appType = "";

//Create instance of AppInfo that contains entity for which
access rights are to be assigned
AppInfo appInfo = new AppInfo(userId,OrgId,appType);
```

5. Create instance of EntityInfo which contains entity whose access rights are assigned on Category.

```
//Set desired entity : CATEGORY/REPORT/QO/PO/CUBEOBJECT
/DASHBOARD2/DASHBOARD_WIDGET/DBCONNECTION
EntityInfo entityInfo=new EntityInfo(catId,"CATEGORY");
```

6. Create object of EntityAccessRight and set accessrights

```
EntityAccessRight ear=new EntityAccessRight();
```

```
ear.setAppInfo(appInfo);
/*     access level can have below possible values-
FULLACCESS, PARTIALACCESS, DENYACCESS, NONEACCESS
  * public void setAccessLevel(int accessLevel)
  * @param accessLevel : access level can be 0/1/2/3
*/
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.PARTIALACCESS)
;
 ear.setAccessRightGrants("0,1,2");
```

7. Setting access rights for Reports, Dashboards under this category

```
//Access Rights for Reports under this category
EntityGroupAccessRight reportEAR=new EntityGroupAccessRight();
reportEAR.setType("REPORT");
reportEAR.setAccessLevel(1);//0 for deny, 1 for Full, 2 for
Partial
reportEAR.setGrants("10,12,14");
ear.addEntityGroupAccessRight(reportEAR);

//Access Rights for Dashboards under this category
EntityGroupAccessRight dashboardEAR=new EntityGroupAccessRight();
dashboardEAR.setType("DASHBOARD2");
```

```
dashboardEAR.setAccessLevel(1);//0 for deny, 1 for Full, 2 for
Partial
ear.addEntityGroupAccessRight(dashboardEAR);

ear.setOpCode("REPLACE");
```

Method: grantEntityPrivileges

This API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight,
UserInfo userInfo)
```

Parameters:

- **entity -** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class..
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **UserInfo:** Details of current user who is providing access rights.
- **Throws**: ISecurityException: - If the request cannot be performed successfully. This happens - 1.if connection can't be established with the engine or 2.if read or write operation cant be performed from or to the engine. 3.If some error has occured while executing the request on the engine. 4.If the response xml obtained from server cannot be parsed

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, requestorUserInfo);
```

Assign Connection Privileges to Organization

This .NET API is used to assign the Connection privileges to the specific Organization.

Steps:

1. Initialize Report Client.

2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of AppInfo that contains the UserId, Organization of the user to which the access rights are to be assigned.

```
  String appId="";
  String OrgId="TestOrg";
  //AppType for which the privileges are to be set, "" for Org,
"USER" for User and "ROLE" for Role
  String appType = "";

//Create instance of AppInfo that contains entity for which
access rights are to be assigned
AppInfo appInfo = new AppInfo(userId,OrgId,appType);
```

5. Create instance of EntityInfo which contains entity whose access rights are assigned on Category.

```
// Connection whose privileges are to be set for the specified
Organization
String entityId="DemoReportDB";//Category Id on which
AccessRights will be given to the user;

//Set desired entity :
CATEGORY/REPORT/QO/PO/CUBEOBJECT/DASHBOARD2/DASHBOARD_WIDGET/DBCO
NNECTION
EntityInfo entityInfo=new EntityInfo(entityId,"DBCONNECTION");
```

6. Create object of EntityAccessRight and set accessrights

```
EntityAccessRight ear=new EntityAccessRight();
```

```
ear.setAppInfo(appInfo);
/*      access level can have below values possible.
 *      Full Access
 *  Deny Access
**/
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.FULLACCESS);
ear.setOpCode("REPLACE");
```

Method: grantEntityPrivileges

This API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight,
UserInfo userInfo
```

Parameters:

- **entity -** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class..
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **UserInfo:** Details of current user who is providing access rights.
- **Throws**: ISecurityException: - If the request cannot be performed successfully. This happens - 1.if connection can't be established with the engine or 2.if read or write operation cant be performed from or to the engine. 3.If some error has occured while executing the request on the engine. 4.If the response xml obtained from server cannot be parsed

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, requestorUserInfo);
```

User

Get User By Id

This .NET API is used to get User's information for the given userid.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/User Management/GetUserById.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get the User by User Id.

Method: getUserById

```
public                                    getUserById(String userId,
String orgId,UserInfo userInfo)
```

This method returns the UserInfo class object having the given userId of the given orgId

Parameters:

- **userId**: Id of the requested UserInfo object.
- **orgId**: Organization Id of the requested user.
- **userInfo**: The current user details.

Returns:

UserInfo: object having the given userId (may return null)

```
String userId = "Mary";
String orgId = "MyOrg";

UserInfo userInfo=sMgr.getUserById(userId, orgId ,
                                   requestorUserInfo);
```

Add User (Create User)

This .NET API is used to create a new User in Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of new User and set its attributes.

```
String username = "Mary";
```

```
String password = "123456";
String orgId = "Org1";

UserInfo addUserInfo = new UserInfo(username, password, orgId);

// Optionally, make the user admin for that organization.
addUserInfo.setAdmin(true);

// Optionally, make the user Super admin.
addUserInfo.setSuperAdmin(true);
```

5. Add the User in Intellicus Repository.

Method: addUser

```
public void addUser( newUserInfo, userInfo)
```

Adds a new User at the Report Engine.

Parameters:

- **newUserInfo**: The new user or target user details.
- **userInfo**: The current user details.

```
sMgr.addUser(addUserInfo,requestorUserInfo);
```

Assign Category Privileges to User

This .NET API is used to assign the Access privileges of a Category to the specific Organization.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of AppInfo that contains the UserId, Organization of the user to which the access rights are to be assigned.
   Also set AppType="USER"

```
String userId="Mary";
```

```
  String OrgId="TestOrg";
  //AppType for which the privileges are to be set, "" for Org,
"USER" for User and "ROLE" for Role
  String appType = "USER";

//Create instance of AppInfo that contains entity for which
access rights are to be assigned
AppInfo appInfo = new AppInfo(userId,OrgId,appType);
```

5. Create instance of EntityInfo which contains entity whose access rights are assigned on Category.

```
//Set desired entity : CATEGORY/REPORT/QO/PO/CUBEOBJECT
/DASHBOARD2/DASHBOARD_WIDGET/DBCONNECTION
EntityInfo entityInfo=new EntityInfo(catId,"CATEGORY");
```

6. Create object of EntityAccessRight and set accessrights

```
EntityAccessRight ear=new EntityAccessRight();
```

```
ear.setAppInfo(appInfo);
/*     access level can have below possible values-
FULLACCESS, PARTIALACCESS, DENYACCESS, NONEACCESS
  * public void setAccessLevel(int accessLevel)
  * @param accessLevel : access level can be 0/1/2/3
*/
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.PARTIALACCESS)
;
 ear.setAccessRightGrants("0,1,2");
```

7. Setting access rights for Reports under this category

```
//Access Rights for Reports under this category
EntityGroupAccessRight reportEAR=new EntityGroupAccessRight();
reportEAR.setType("REPORT");
reportEAR.setAccessLevel(1);//0 for deny, 1 for Full, 2 for
Partial
reportEAR.setGrants("10,12,14");

ear.addEntityGroupAccessRight(reportEAR);

ear.setOpCode("REPLACE");
```

Method: grantEntityPrivileges

This API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight,
UserInfo userInfo).
```

Parameters:

- **entity -** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class..
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **UserInfo:** Details of current user who is providing access rights.
- **Throws**: ISecurityException: - If the request cannot be performed successfully. This happens - 1.if connection can't be established with the engine or 2.if read or write operation cant be performed from or to the engine. 3.If some error has occured while executing the request on the engine. 4.If the response xml obtained from server cannot be parsed

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, requestorUserInfo);
```

Assign Report Privileges To User

This .NET API is used to set the Report Access Privileges for a particular user. The user would require to provide its identity i.e. user Info and the Report for which the access rights are to be given.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of AppInfo that contains the UserId, Organization of the user to which the access rights on report are to be assigned.
   Also set AppType="USER"
   //Set appId = <ROLE_NAME> and appType="ROLE"

```
  String appId="Mary";
  String OrgId="TestOrg";
  //AppType for which the privileges are to be set, "" for Org,
"USER" for User and "ROLE" for Role
  String appType = "USER";

//Create instance of AppInfo that contains entity for which
access rights are to be assigned
AppInfo appInfo = new AppInfo(appId,OrgId,appType);
```

5. Create instance of EntityInfo which contains entityID/ReportID for which access rights are to be assigned

```
String reportId="D07131A2-87AA-154E-6E17-6079C9AFD176";
//Set desired entity : CATEGORY/REPORT/QO/PO/CUBEOBJECT
/DASHBOARD2/DASHBOARD_WIDGET/DBCONNECTION
EntityInfo entityInfo=new EntityInfo(reportId,"REPORT");
```

6. Create object of EntityAccessRight and set accessrights

```
EntityAccessRight ear=new EntityAccessRight();
```

```
ear.setAppInfo(appInfo);
/*     access level can have below possible values-
FULLACCESS, PARTIALACCESS, DENYACCESS, NONEACCESS
  * public void setAccessLevel(int accessLevel)
  * @param accessLevel : access level can be 0/1/2/3
*/
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.PARTIALACCESS)
;
ear.setAccessRightGrants("0,2,4,6,7,8,12");

ear.setOpCode("REPLACE");
```

Method: grantEntityPrivileges

This API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight,
UserInfo userInfo)
```

Parameters:

- **entity -** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class..
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **UserInfo:** Details of current user who is providing access rights.
- **Throws**: ISecurityException: - If the request cannot be performed successfully. This happens - 1.if connection can't be established with the engine or 2.if read or write operation cant be performed from or to the engine. 3.If some error has occured while executing the request on the engine. 4.If the response xml obtained from server cannot be parsed

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, requestorUserInfo);
```

Assign Entity Privileges to User

This .NET API is used to assign the Entity privileges to the specific User.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of AppInfo that contains the UserId, Organization of the user to which the access rights are to be assigned.
   Also set AppType="USER"

```
String userId="Mary";
```

```
  String OrgId="TestOrg";
  //AppType for which the privileges are to be set, "" for Org,
"USER" for User and "ROLE" for Role
  String appType = "USER";

//Create instance of AppInfo that contains entity for which
access rights are to be assigned
AppInfo appInfo = new AppInfo(userId,OrgId,appType);
```

5. Create instance of EntityInfo which contains entity whose access rights are assigned on Category.

```
// Query ID whose System Privileges are to be set for the
specified user.
String entityID="SalesQuery";

//Set desired entity : CATEGORY/REPORT/QO/PO/CUBEOBJECT
/DASHBOARD2/DASHBOARD_WIDGET/DBCONNECTION
EntityInfo entityInfo=new EntityInfo(entityID,"QO");
```

6. Create object of EntityAccessRight and set accessrights

```
EntityAccessRight ear=new EntityAccessRight();
```

```
ear.setAppInfo(appInfo);
/*     access level can have below possible values-
FULLACCESS, PARTIALACCESS, DENYACCESS, NONEACCESS
  * public void setAccessLevel(int accessLevel)
  * @param accessLevel : access level can be 0/1/2/3
*/
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.PARTIALACCESS)
;
ear.setAccessRightGrants("0,2");
```

Method: grantEntityPrivileges

This API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight,
UserInfo userInfo)
```

Parameters:

- **entity -** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class..
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **UserInfo:** Details of current user who is providing access rights.
- **Throws**: ISecurityException: - If the request cannot be performed successfully. This happens - 1.if connection can't be established with the engine or 2.if read or write operation cant be performed from or to the engine. 3.If some error has occured while executing the request on the engine. 4.If the response xml obtained from server cannot be parsed

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, requestorUserInfo);
```

Assign System Privileges to the User

This .NET API is used to assign the System Privileges to the User. The user would require to provide its identity i.e. user Info as well as of the user who is providing System Privileges.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of User for which System Privileges are to be set.

```
String userId="FinanceUser";
String organization=="MyOrg";
UserInfo userInfo = sMgr.getUserById(userId, organization,
requestorUserInfo);
```

5. Assign System Privileges to assigneeUser.

Method : setSystemPrivileges

```
public void setSystemPrivileges(String systemPrivileges)
```

Enums.SecurityTypes.SystemPrivileges.*CATEGORY_SETUP*
Enums.SecurityTypes.SystemPrivileges.*DATA_ADMIN*
Enums.SecurityTypes.SystemPrivileges.*IM_SUPPORT*
Enums.SecurityTypes.SystemPrivileges.systemPrevilegesMap
Enums.SecurityTypes.SystemPrivileges.*SCHEDULER*
Enums.SecurityTypes.SystemPrivileges.*REPORT_DESIGNER*
Enums.SecurityTypes.SystemPrivileges.*DEPLOYREPORTBUNDLE*
Enums.SecurityTypes.SystemPrivileges.*CATEGORY_SETUP_GLOBAL*
Enums.SecurityTypes.SystemPrivileges.*SCHEDULER_GLOBAL*
Enums.SecurityTypes.SystemPrivileges.*DATA_ADMIN_GLOBAL*
Enums.SecurityTypes.SystemPrivileges.*ADHOCREPORTDESIGNER*
Enums.SecurityTypes.SystemPrivileges.*WIDGET_DESIGNER*
Enums.SecurityTypes.SystemPrivileges.*GENERATE_LINK*
Enums.SecurityTypes.SystemPrivileges. *GENERATE_LINK_GLOBAL*

This method sets System Privileges in the comma-separated format.

```
userInfo.setSystemPrivileges(new
com.intellica.client.utils.Utility().getRightsFromMaskedValue(
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
CATEGORY_SETUP |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
SCHEDULER |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
SCHEDULER_GLOBAL |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
WIDGET_DESIGNER |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
REPORT_DESIGNER |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
ADHOCREPORTDESIGNER |
Enums.SecurityTypes.SystemPrivileges.GENERATE_LINK));

sMgr.modifyUser(userInfo,requestorUserInfo);
```

Assign Role To User

This .NET API is used to assign a specified existing role to the application user.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/User Management/ AssignRoleToUser.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of User for which System Privileges are to be set.

```
String userId ="User1"; // Id of the application user
String orgId ="MyOrg";

UserInfo userInfo=sMgr.getUserById(userId, orgId
,requestorUserInfo);
```

5. Create instance of RoleInfo that is to be assigned to the User.

```
String roleId ="Manager"; // Id of the role to be Assigned
RoleInfo roleInfo  =  sMgr.getRoleById(roleId, orgId
,requestorUserInfo);
```

6. Grant this Role Info to the assignee User.

Method: grantRoleToUser

```
public void grantRoleToUser( targetUInfo,
                             userInfo)
```

This method grants new role to User and keeps previous roles of that user intact.

Parameters:

- **TargetUInfo:** com.intellica.client.common.UserInfo class object to which roles has to be assigned.
- **RInfo:** com.intellica.client.security.RoleInfo object to be granted.
- **UserInfo:** The details of the requesting user.

```
sMgr.grantRoleToUser(userInfo,roleInfo,requestorUserInfo);
```

Assign Roles To User

This .NET API is used to assign multiple roles to the application user.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/User Management/ AssignRolesToUser.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of User for which System Privileges are to be set.

```
String userId="TestUser";// Id of the application user
String orgId="MyOrg";
UserInfo userInfo=sMgr.getUserById(userId, orgId
,requestorUserInfo);
```

5. Create instance of RoleInfo that is to be assigned to the User.

```
String roleId1 = "Manager";//Id of the role1 to be Assigned
String roleId2 = "Administrator";//Id of the role2 to be assigned
RoleInfo roleInfo1  =  sMgr.getRoleById(roleId1, orgId
,requestorUserInfo);
RoleInfo roleInfo2  =  sMgr.getRoleById(roleId2, orgId
,requestorUserInfo);
```

6. Grant these Roles to the assignee User.

Method: assignRolesToUser

```
public void assignRolesToUser( targetUInfo,
                               ArrayList roleInfos,
                                userInfo)
```

This method attaches a User with more than one role.

Parameters:

- **targetUInfo**: The user to be attached with a role.
- **roleInfos**: Array List of Roles to be attached with user.
- **userInfo**: The details of the current user.

```
ArrayList roleInfoArr = new ArrayList();
roleInfoArr.add(roleInfo1);//Add roleInfo object role1 to Array
roleInfoArr.add(roleInfo2);//Add roleInfo object "role2" to Array

sMgr.assignRolesToUser(userInfo,roleInfoArr,requestorUserInfo);
```

Revoke Role from User

This .NET API is to revoke specified role from user

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get User by User Id whose role is to be revoked.

```
String userId = "User1"; // Id of the application User
String orgId= "MyOrg";
UserInfo userInfo=sMgr.getUserById(userId, orgId
,requestorUserInfo);
```

5. Create instance of RoleInfo that is to be revoked for the given User.

```
String roleId = "Manager"; // Id of the role to be Assigned
RoleInfo roleInfo=sMgr.getRoleById(roleId, orgId
,requestorUserInfo);
```

6. Revoke the Role from assignee User.

Method: revokeRoleFromUser

```
public void revokeRoleFromUser(UserInfo targetUInfo,
                               RoleInfo rInfo,
                               UserInfo userInfo)
```

This method revokes or removes role assigned previously to the user.

Parameters:

- **TargetUInfo:** com.intellica.client.common.UserInfo class object to which role has to be revoked.

- **rInfo**: com.intellica.client.security.RoleInfo object to be revoked from the user.
- **userInfo**: The details of the current User.

```
sMgr.revokeRoleFromUser(userInfo,roleInfo,requestorUserInfo);
```

User Mapping

This .NET API is used to map the host application user with Intellicus user in the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get User by User Id to which , new user is to be mapped

```
String userId="Smith";// Id of the application user
String password="";
String orgId="MyOrg";
UserInfo userInfo  = new UserInfo(userId, password , orgId);
```

5. Map the new User implementing mapAppIdToUser method.

```
String newUserId="John";
sMgr.mapAppIdToUser(newUserId,userInfo, requestorUserInfo);
```

Details of Method: mapAppIdToUser

```
public void mapAppIdToUser(String appId,
                           UserInfo targetUInfo,
                           UserInfo userInfo)
```

This method adds a mapping between an AppId and the user.

Parameters:

- **AppId:** is the application Id.
- **TargetUInfo:** is the targeted User.
- **UserInfo:** the current user details.

Delete User

This .NET API is used to delete an existing user with given userId from the Intellicus Repository.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/User Management/ DeleteUser.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get UserInfo instance of the user that is to be deleted.

```
String userId="FinanaceUser";
String orgId="FinanceOrg";//Organization Id in which user
                                exists.
UserInfo targetUserInfo=sMgr.getUserById(userId,orgId,
                                    requestorUserInfo);
```

5. Delete the User.

Method: deleteUser

```
public void deleteUser(UserInfo targetUInfo,
                       UserInfo userInfo)
```

This method deletes an existing User's Info at the Report Engine.

Parameters:

- **TargetUInfo:** The UserInfo object of the user to be deleted.
- **UserInfo:** The details of the current user.

```
sMgr.deleteUser(targetUserInfo,requestorUserInfo);
```

Role

Get Role List

This .NET API is used to get the list of all Roles present in Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get the role list.

Method: getRoleList

```
public ArrayList getRoleList(UserInfo userInfo)
```

This method returns the list of all roles created in Intellicus repository.

Returns:

ArrayList of roleInfo Bean class RoleInfo

```
roleList=sMgr.getRoleList(requestorUserInfo);
```

Create Role (Add Role)

This .NET API is used to add a new role in the existing Organization.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of RoleInfo for the role to be added in the Organization.

```
String roleName = "MANAGER";
String orgId = "Org1";
RoleInfo newRole = new RoleInfo(roleName, orgId );
```

5. Add the role in the organization.

Method : addRole

```
public void addRole(RoleInfo roleInfo,
                    UserInfo userInfo)
```

This method adds a new Role at the Report Engine.

Parameters:

- **roleInfo:** The details of new Role.
- **userInfo:** The current user.

```
sMgr.addRole(newRole,requestoruserInfo);
```

Assign Category Privileges To Role

This .NET API is used to assign the Access privileges of a Category to the existing Role.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create EntityInfo and AppInfo

```
//RoleID of the Role for which AccessRights are to be assigned
String roleId     ="Manager";//role  Id  to  be  checked  whether
exists or not in the Intellicus server
String orgId    ="hostOrg";
String categoryId ="B2996F87-FE41-8D19-6C67-B3C22803DD99";

EntityInfo entityInfo=new EntityInfo(categoryId,"CATEGORY");
AppInfo appInfo = new AppInfo(roleId,orgId,"ROLE");
```

5. Create EntityAccessRight Object and set this AppInfo and EntityInfo in its object.

```
EntityAccessRight ear=new EntityAccessRight();

/*public void setAppInfo(AppInfo appInfo)Parameters:
 *appInfo - : AppInfo object.
 * */
```

```
ear.setAppInfo(appInfo);
/*      access level can have three values possible.
 *      Deny Access      = 0
 *  Full Access          = 1
 *      Partial Access   = 2
 *  None Access    = 3*/
/**
 * public void setAccessLevel(int accessLevel)
 * @param accessLevel : access level can be 0/1/2
 */
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.FULLACCESS);
```

6. Assign Category Privileges to assigneeUser using assignCategoryPrivilegesToUser.

Method: grantEntityPrivileges

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight, UserInfo userInfo)
```

API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Parameters:

- **entity:** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class.
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **userInfo:** Details of current user who is providing access rights.

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, adminUserInfo);
```

Assign Report Privileges to Role

This .NET API is used to assign the Access privileges for a Report to the Role.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create EntityInfo and AppInfo

```
//RoleID of the Role for which AccessRights are to be assigned
String roleId="Manager";
String organisationId="Intellica";
String reportId="638CB7A6-F504-0ECD-008F-10B25253DCA8";

EntityInfo entityInfo=new EntityInfo(reportId,"REPORT");
AppInfo appInfo = new AppInfo(roleId,organisationId,"ROLE");
```

5. Create EntityAccessRight Object and set this AppInfo and EntityInfo in its object.

```
EntityAccessRight ear=new EntityAccessRight();

/*public void setAppInfo(AppInfo appInfo)Parameters:
 *appInfo - : AppInfo object.
 * */
ear.setAppInfo(appInfo);
/*      access level can have three values possible.
 *      Deny Access      = 0
 *  Full Access          = 1
 *      Partial Access   = 2
 *  None Access    = 3*/
/**
 * public void setAccessLevel(int accessLevel)
 * @param accessLevel : access level can be 0/1/2
 */
ear.setAccessLevel(Enums.SecurityTypes.AccessLevel.FULLACCESS);
```

6. Assign Category Privileges to assigneeUser using assignCategoryPrivilegesToUser.

Method: grantEntityPrivileges

Syntax

```
public void grantEntityPrivileges(EntityInfo entity,
EntityAccessRight entityAccessRight, UserInfo userInfo)
```

API allows the user to assign access rights information to a user/role/organization or Everyone on an entity.

Parameters:

- **entity:** The entity object.This object must be created by setting entityId and entityType. entityTypes supported are defined in EntityTypeNames class.
- **entityAccessRight:** The entityAccessRight object. This object should contain the AppInfo object with the credentials of the user to which grants are to be assigned and access level as defined in Enums.SecurityTypes.AccessLevel
- **userInfo:** Details of current user who is providing access rights.

Example

Given below is the example of actual implementation of this method:

```
sMgr.grantEntityPrivileges(entityInfo, ear, adminUserInfo);
```

Delete Role

This .NET API is used to delete a particular Role of an existing Organization in Intellicus Repository.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/Role Management/ DeleteRole.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations.

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Create instance of RoleInfo for the role which is to be deleted from the Organization.

```
//Role ID i.e. to be deleted
String roleId     ="Manager";// Id of the role to be deleted
//Organization from which the given role is to be deleted.
```

```
String orgId ="Test";

RoleInfo roleInfo=new RoleInfo(roleId,orgId);
```

5. Assign the Report Privileges to the Role for given Report Id in a particular Category.

Method: deleteRole

```
public void deleteRole(RoleInfo roleInfo,
                       UserInfo userInfo)
```

This method deletes a Role at the Report Engine.

Parameters:

- **RoleInfo:** The role to be deleted.
- **userInfo**: The current user details.

```
sMgr.deleteRole(roleInfo,requestorUserInfo);
```

Report Management Actions

Categories

Using

```
using com.intellica.client.common.UserInfo;

using com.intellica.client.layout.LayoutManager;
using com.intellica.client.reportutils.Category;
using com.intellica.client.exception.
         LayoutHandlerException;
```

Get Category List

This .NET API retrieves the list of all categories from the Intellicus repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Get the list of all Categories in Repository.

Method: **getCategoryList**

```
public getCategoryList(UserInfo requestorUserInfo)
```

This method returns a Hashtable of Category objects. The method requests report server and returns the report categories present in the report repository, both public and private to requestor user. The public categories will be restricted to those categories that the requestorUserInfo has access to read. Each category object provides getIsPublic bool method to identify the scope of category.

Parameters:

RequestorUserInfo: UserInfo for authorization. Pass getEmptyInstance() in case of report server runs in Security-Off mode.

Returns:

Hashtable of Category objects.

```
Hashtable vecCatList= new Hashtable();
vecCatList=lm.getCategoryList(requestorUserInfo);
```

The other three APIs for getting Category List are-

1. Returns a Hashtable of categories from the report server on the basis of filters

```
public Hashtable getCategoryList(Filter filter,UserInfo userInfo)
```

2. Returns a Hashtable of report categories from the report server with given access rights

```
public Hashtable getCategoryList(int accessRight, UserInfo
userInfo)
```

3. Returns a Hashtable of categories from the report server on the basis of filters

```
public Hashtable
getCategoryList(com.impetus.intera.layout.adhoc.data.Filters
filters,UserInfo userInfo)
```

All these returns Hashtable containing Category objects.

Add a new Category

This .NET API is used to add a new Category in the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Create an instance of Category i.e to be added in Repository and set its various attributes of Categories.

```
String catName="NewDemoCategory";
String catDescription = "This is the new category just created";
String catId = "123456";
Category newCategory=new Category(catName);
//Set the various properties of the Category
newCategory.setIsPublic(true);
newCategory.setIsHidden(false);
newCategory.setDescription(catDescription);
//To set the category id
newCategory.setCategoryId(catId);
newCategory.setMenuName(catName);
```

5. Add this Category in the Repository.

Method: **addCategory**

```
public Category addCategory(Category category,
                            UserInfo userInfo)
```

This method adds new category to the Intellicus repository.

```
//Adds new category to the Intellicus repository.
lm.addCategory(newCategory,requestorUserInfo);
```

Delete Category

This .NET API is used to delete an Existing Category from the Intellicus repository along with the Reports present in that Category.User need to provide the Category ID of the Category which is to be deleted.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/Category Management/ DeleteCategory.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Delete the Category with given Category Id.

This method is used to delete the existing category from the report engine's report layout repository along with the reports contained in it.

Method: deleteCategory

```
public void deleteCategory(String catId,
                           bool unConditional,
                           UserInfo userInfo)
```

Parameters:

- **CatId:** Category ID of the Existing Category that is to be deleted.
- **UnConditional:** If true than deletes the category even if report layouts are present in the category along with all its reports.
- **UserInfo:** UserInfo object authorization.

```
//Category ID of the Category that is to be deleted.
  String categoryID="DemoCategory";

lm.deleteCategory(categoryID, true, requestorUserInfo);
```

GetSubCategories

This .NET API is used to delete an Existing Category from the Intellicus repository along with the Reports present in that Category.User need to provide the Category ID of the Category which is to be deleted.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Get the sub categories for given Category Id.

This method is used to delete the existing category from the report engine's report layout repository along with the reports contained in it.

Method: getSubCategories

```
public LinkedHashMap getSubCategories()
```

Parameters:

- **CatId:** Category ID of the Category whose categories are to be obtained
- **UserInfo:** UserInfo object authorization.

```
//Category ID of the Category that is to be deleted.
String categoryID="DemoCategory";
categoryObj = lm.getCategory(categoryID, requestorUserInfo);

//now, fill the sub-categories in the category object.
Filter filter = new Filter();
filter.setFilterField("ENTITYTYPE", "CATEGORY");

lm.populateEntityListForCategory(categoryObj,filter,requestorUser
Info);
LinkedHashMap subcategories = categoryObj.getSubCategories();
```

The other related APIs for Category Management are-

1. Public void **deleteSubCategory**(String subCategoryId)
2. public void **setSubCategories**(LinkedHashMap subCategories)
3. public Category **getParentCategory**()
4. public void **setParentCategory**(Category parentCategory)

Report Operations

Using

```
using com.intellica.client.common.UserInfo;

using com.intellica.client.layout.LayoutManager;
using com.intellica.client.reportutils.Report;
using com.intellica.client.exception.
         LayoutHandlerException;
```

Run Report

This .NET API is used to execute a Report with given Report Id.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.

3. Create an instance of reportExecuter i.e. a class having methods for executing a report.

```
ReportExecutor report = new ReportExecutor();
```

4. Set the System Parameters required for running the Report.

```
//Instantiate a collection object for setting system parameters
HashContainer hcSysParams = new HashContainer();

//fill up the system parameter values
//Report ID which is required to run
String reportId    = "Customer Report";

// This will set Id of the report.
hcSysParams.put("REPORT_ID",reportId);
// In database mode only reportId needed.don't prefix //category
id
// This will set the output format for the report.report //format
can be PDF/XLS/HTML

  hcSysParams.put("REPORT_FORMAT",InteraConstants.ReportFormats.P
DF);
```

5. Set the User/Business Parameters

```
//Instantiate a collection object for setting user/business
//parameters
HashContainer hcUserParams =new HashContainer();
String countryName = "Australia";
String roleId = "3818";
// fill up the user params if report contains any parameter
hcUserParams.put("Country", countryName);
// Parameter Name as created in the report layout(IRL)
hcUserParams.put("Role_ID", roleId);
```

6. Instantiate the Byte Array Output Stream object.

```
ByteArrayOutputStream reportData = new ByteArrayOutputStream();
```

7. Run the Report.

Method: **runReport**

```
public void runReport(UserInfo userInfo,
                      HashContainer systemParameterHash,
```

```
                        HashContainer userParameterHash,
                        IStreamCallback pCallback,
                        java.io.ByteArrayOutputStream reportData,
                        bool isPublic)
```

This method executes a report by taking the values of system-defined parameters (and report parameters) in parameters as HashContainer and gives the generated report data in the passed outputStream. This method generates the complete report, and then returns the control to the calling code. In this case, the calling code must catch the exceptions right at the calling place. The calling code can use the output stream after the method call.

Parameters:

- **UserInfo:** userInfo object will be used for passing user information to the Report Engine. Can be left as null when application is not using Intellica Security Module.
- **SystemParameterHash:** HashContainer instance, Contains the values of system parameters.
- **UserParameterHash:** HashContainer instance, Contains the values of Report parameters. Can be left as null when there are no parameters in the report.
- **PCallback:** IStreamCallback instance, For future usage.
- **ReportData: T**he RefByteArrayOutputStream instance required for the report output.The report output in the form of byte array is added in this output stream.

```
report.runReport(requestorUserInfo,hcSysParams,hcUserParams,null,
reportData,true);
```

Get the list of Published Reports of a particular report

This .NET API is used to get the list of all published reports corresponding to given Report Id.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Set the filter for getting Saved Report List.

The various filter settings could be:

```
Enums.Filters.SavedReportList.CATID
Enums.Filters.SavedReportList.ENTITY_PROPERTIES
Enums.Filters.SavedReportList.FROMDATE
Enums.Filters.SavedReportList.GENERATOR_ID
Enums.Filters.SavedReportList.GENERATOR_TYPE
Enums.Filters.SavedReportList.ISPUBLIC
Enums.Filters.SavedReportList.ORGID
Enums.Filters.SavedReportList.ORPHANED
Enums.Filters.SavedReportList.PUBLISHSTATUS
Enums.Filters.SavedReportList.REPORTID
Enums.Filters.SavedReportList.WORKFLOW_STATE
Enums.Filters.SavedReportList.TODATE
Enums.Filters.SavedReportList.USERID
```

```
Filter filterObj = new Filter();
String reportId="SalesReport";
filterObj.setFilterField(Enums.Filters.SavedReportList.REPORTID,r
eportId);
```

5. Get the List of Saved Reports based on the filters applied.

Method: getViewSavedReport

```
public void getViewSavedReport(SavedReport savedReport,
java.io.ByteArrayOutputStream reportData,
                               HashContainer sysParams,
```

```
 userInfo)
```

This method is used to get the saved report with in the ByteArrayOutputStream passed as a parameter.

Parameters:

- **SavedReport:** Saved report object SavedReport.
- **ReportData:** ByteArrayOutputStream contains the Report data.
- **SysParams:** HashContainer for report system parameters.
- **UserInfo:** UserInfo For authorization.

```
Hashtable vecPublishedReportList   =
lm.getSavedReportList(filterObj,requestorUserInfo);
```

Report Layout Management

Using

```
using com.intellica.client.common.UserInfo;

using com.intellica.client.layout.LayoutManager;
using com.intellica.client.reportutils.Report;
using com.intellica.client.exception.
         LayoutHandlerException;
```

GetAllReportList

This .NET API is used to get the complete list of all the Reports from the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Get the Report List

Method : getReportList

```
public Hashtable getReportList(UserInfo userInfo)
                                  throws LayoutHandlerException
```

Returns a Hashtable of all report names from the report server The list will be restricted to accessible reports by this user (Applies to report server enterprise edition only)

Parameters:

- **UserInfo:** UserInfo for authorization.

Returns:
Hashtable of all report names from the report server.

Part of Sample Code implementing "getReportList"**:**

```
//Hashtable that will be used to store the Report List
```

```
Hashtable vecReportList= new Hashtable();

//This method returns a Hashtable of all report names from the report server
vecReportList=lm.getReportList(requestorUserInfo);
```

The other APIs for getting Report List are-

1. Returns a Hashtable of report objects from the report server on the basis of requesting filters.

```
public Hashtable getReportList(Filter,UserInfo)
```

2. Returns a Hashtable of all report names from the report server. The list will be restricted to accessible reports by this user

```
public Hashtable getReportList(int categoryAccessRight, int reportAccessRight,UserInfo userInfo)
```

3. Returns a Hashtable of all report names from the report server. The list will be restricted to accessible reports by this user

```
public Hashtable getReportList(int reportAccessRight,UserInfo userInfo)
```

4. This method gets the list of reports based on filters (request details) specified in ReportListRequest

```
public Hashtable getReportList(ReportListRequest reportListRequest,UserInfo userInfo)
```

MoveReport

This .NET API is used to move an existing report to another Destination Category.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Pass the Source ReportId which is to be copied and the destination category Id , to which it is copied so as to move the Report.

Method : moveReport

```
public void moveReport(String sourceReportId,
                       String destinationCategoryId,
                       Report report,
                       UserInfo userInfo)
```

Method to Move a report to the new category.

Parameters:

- **SourceReportId:** source report id, mandatory argument for moving the report.
- **DestinationCategoryId:** destination category id,where new report has to be moved.if this is null then report objects's category id will be taken as destination category.
- **Report**: instance of Report class for move.
- **UserInfo:** UserInfo object for authorization.

```
//ReportID of the Report that is to be moved to new Category
String reportID="234CBC33-EC19-E754-7490-43DA2A24728C";

//CategoryID of the Destination Category where the report needs
       to be moved
String destCategoryID="61FCA109-7E0D-D023-A19C-A054A38FC9B6";

//Method to copy the report to anther Category

lm.moveReport(reportID, destCategoryID, r,requestorUserInfo);
```

Get Report List For Category

This .NET API is used to get the complete list of Reports in given Category.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Pass the Category Id whose Reports are to be listed.

Method : getReportListForCategory

```
public Hashtable getReportListForCategory
       (String categoryId,
        UserInfo requestorUserInfo)
```

Method returns a Hashtable of Report objects. The method requests report server and returns the reports present in the report repository under given report category, both public and private to requestor user. The public reports will be restricted to those reports that the requestorUserInfo has access to read. Each report object provides getIsPublic bool method to identify the scope of report.

Parameters:

- **CategoryId:** The category Id of the category from which, the report layout list is requested. Intellicus allows category ids upto 256 charecters long.
- **RequestorUserInfo:** UserInfo for authorization. Pass getEmptyInstance() in case of report server runs in Security-Off mode.

Returns:
Hashtable of Reports.

```
Hashtable vecCatList= new Hashtable();
String categoryId = "01-Sales";
vecCatList=lm.getReportListForCategory
                 (categoryId , requestorUserInfo);
```

The other related APIs for getting Report List For Category are-

1. Returns a Hashtable of all report names from the report server.The list will be restricted to accessible reports by this user.

```
public     Hashtable     getReportListForCategoryAccessRight(int
categoryAccessRight,UserInfo userInfo)
```

2. Returns a Hashtable of all report names from the report server.The list will be restricted to accessible reports by this user.
   @**param** categoryId: The category Id for which the report layout list is requested.
   @**param** reportAccessRight masked value of the access right.

```
public Hashtable getReportListForCategoryWithAccessRights(String
categoryId,int   categoryAccessRight,   int   reportAccessRight,
UserInfo userInfo)
```

3. Returns a Hashtable of all report names from the report server.The list will be restricted to accessible reports by this user.
   @**param** categoryId: The category Id for which the report layout list is requested.
   @param categoryAccessRight masked value of the access right.

```
public Hashtable getReportListForCategoryWithCategoryRight(String
categoryId,int categoryAccessRight,UserInfo userInfo)
```

4. The method requests report server and returns the reports present in the report repository under given report category, both public and private to requestor user.

```
public Hashtable getReportListForCategoryWithReportRight(String
categoryId, int reportAccessRight,UserInfo userInfo)
```

Get Saved Report List

This .NET API is used to get the list of Saved Reports in a particular Category/Report.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Pass the Category Id/Report Id whose Saved Reports are to be listed.

Method : getSavedReportList

```
public Hashtable getSavedReportList
       (Filter filter, UserInfo requestorUserInfo)
```

Method returns a Hashtable of Report objects. The method requests report server and returns the saved reports present in the report repository under given report category/Report, both public and private to requestor user.

Parameters:

- **filter:** Filter can set the values of CATID and REPORTID.
- **RequestorUserInfo:** UserInfo for authorization.

Returns:

Hashtable of Saved Reports.

```
//Cat ID of the Category whose Reports are required to be Listed
String catId = "4F9245A7-D639-4F99-604D-F32641B77725"; //category
id to be set in filter
String reportId="All Country Sales - Linked Prompt";//Report Id
to be set in filter

Filter filter = new Filter();

  filter.setFilterField(com.intellica.client.common.Enums.Filters
.SavedReportList.CATID, catId);

  filter.setFilterField(com.intellica.client.common.Enums.Filters
.SavedReportList.REPORTID, reportId);
```

The other related APIs fro Saved Reports are:

1. public LinkedHashMap **getSavedReportsMap**()
2. public void **setSavedReportsMap**(LinkedHashMap savedReportsMap)
3. public void **deleteSavedReport**(String reportOId,String reportOutputId, UserInfo userInfo)
4. public void **updateSavedReport**(SavedReport savedReport, UserInfo userInfo)
5. public void **addSavedReport**(SavedReport savedReport)

Add Report Layout to Category

This .NET API is used to add a report to an existing Category in the Intellicus repository. This would require User to provide the Category ID, and the details related to the Report i.e. to be added.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Create an instance of Report i.e. to be added in the Category. Also, set the various attributes of report.

```
String file = "C:/UploadAll Country Sales.irl";

//Read the contents from the given file
```

```
InputStream is = new FileInputStream(file);

// Get the size of the file
long length = file.length();

if (length > Integer.MAX_VALUE) {
// File is too large
  throw new Exception("File is too Large");
}
else {
// Create the byte array to hold the data
byte[] bytes = new byte[(int) length];

// Read in the bytes
int offset = 0;
int numRead = 0;

while (offset < bytes.length  && (numRead = is.read(bytes, offset,
bytes.length- offset)) >= 0)
{
offset += numRead;
}
//Ensure all the bytes have been read in
if (offset < bytes.length) {
throw new IOException("Could not completely read the file ");
}
is.close();

String reportName = "Sample_Report";
String reportID = "Sample Report";

Report reportToAdd = new Report(reportName, reportID, bytes);

String categoryID = "Demo";
reportToAdd.setCategoryId(categoryID);
```

5. Upload the Report.

Method : uploadReport

```
public void uploadReport(Report report,
                         UserInfo userInfo)
```

This method uploads a report by taking the Report object as parameter. This method internally validates the Report object for report paramaters.

The method also varifies if report already exists in the specified category.In case report object is not validated and varified the method throws LayoutHandlerException exception.The calling code must catch the exception right at the calling place.

Parameters:

- **Report:** Report class instance containing the report layout metadata and the byte array of report layout itself.
- **UserInfo:** UserInfo object use to validate the user against the application.

```
lm.uploadReport(reportToAdd, requestorUserInfo);
```

Copy Report

This .NET API is used to copy an existing report to another destination category.

User needs to provide the ReportID that is to be copied and the destination category ID where it is to be copied.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Set the attributes for copied report.

```
Report r=new Report();
String rep_Name = "Copied_Report";
r.setMenuName(rep_Name);
```

5. Pass the Source Report Id to be copied and the destination Category where it is to be copied.

Method : **copyReport**

```
public void copyReport(String sourceReportId,
                       String destinationCategoryId,
                       Report report,
                       UserInfo userInfo)
```

This method is used to make a copy of the existing report attributes.

Parameters:

- **SourceReportId:** source report id, mandatory argument for copying the report.
- **DestinationCategoryId:** destination category id, where new report has to be copied.if this is null then source report's category id will be taken as destination category.
- **Report:** instance of Report class for copy.
- **UserInfo:** UserInfo object for authorization.

Part of Sample Code implementing "copyReport":

```
String sourceReportId = "787F11B1-74E7-6792-9ACB-408753C0470F";
String destCategoryId = "61FCA109-7E0D-D023-A19C-A054A38FC9B6";
lm.copyReport(sourceReportId, destCategoryId, r,
requestorUserInfo);
```

Get Report Details

This .NET API is used to get Report Details corresponding to the given Reoprt Id from the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Get the Report Detail for given Report Id.

Method: getReportDetails

```
public Report getReportDetails(String reportId,
                               UserInfo userInfo)
```

This method returns an object of Report class obtained from the report server.

Parameters:

- **Reported:** The report id whoes details should be obtained.
- **UserInfo:** UserInfo for authorization.

Returns:
An object of Report class obtained from the report server.

```
Report reportDetail=new Report();
String reportId = "91FEE269-3AEE-C23D-6F04-7A9979BEBE09";
reportDetail=lm.getReportDetails(reportId,requestorUserInfo);
```

Other related APIs for Report Layout Management are:

1. public **LinkedHashMap** getReportsMap()
2. public void **setReportsMap**(LinkedHashMap reportsMap)
3. public void **updateReport**(Report oldReport,Report newReport,UserInfo userInfo)
4. public void **updateReport**(Hashtable reportHashtable, UserInfo userInfo)
5. public void **addReport**(Report report)

Mass Operations

User is allowed to select more than one entity and perform the selected operation on those Entities. This could be done using below APIs.

Using

```
using com.intellica.client.common.UserInfo;

using com.intellica.client.layout.LayoutManager;
using com.intellica.client.reportutils.Report;
using com.intellica.client.exception.
         LayoutHandlerException;
```

Copy Entities

This API is used to copy entities to another location. It returns modified object with new destination location.

Let's consider here, Queries are to be copied from one Category to Destination Category.

User needs to provide the Query IDs that are to be copied and the destination category ID where they are to be copied.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.

3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Set the Query Ids that are to be copied.

```
String queryIds[] = new String
"126077293450092031291929106718 82",
"126035168096892031291929144314 93","125870323323241921681022895 83
904"};
```

5. Create the EntityOperation object and set Query Entities for above given Query Ids and add in an ArrayList i.e. entityList.

```
ArrayList entityList = new ArrayList();
EntityOperation entityObj = null;

for(int i=0;i< queryIds.length;i++){
  entityObj = new EntityOperation();

  //Entity Type could be REPORT /CATEGORY /QUERY /DASHBOARD2/
  PARAMETER /SAVEDREPORT /DASHBOARD_WIDGET /CUBEOBJECT
  entityObj.setEntityType("QUERY");

  entityObj.setEntityId(queryIds[i]);
  entityList.add(entityObj);
}
```

6. Pass the arraylist of entities that are to be copied in the destination Category.

Method: **copyEntities**

```
public ArrayList copyEntities(ArrayList entityList, String
targetCatId, UserInfo userInfo)
```

This method is used to copy entites in the destination Category.

Parameters:

- **entityList :** ArrayList of {@link com.intellica.client.common.EntityOperation} class instance.
- **targetCatId:** Destination location where this entity should be copied.
- **UserInfo:** UserInfo object for authorization.

Part of Sample Code implementing "copyEntities":

```
String destCategoryId = "DataLoader";

lm.copyEntities(entityList, destCategoryId, requestorUserInfo);
```

Delete Entities

This API is used to delete entities from the Repository.
Let's consider here, "Categories" are to be deleted from the Repository.

User needs to provide the Category IDs that are to be deleted.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations.

```
LayoutManager lm= new LayoutManager();
```

4. Set the Category Ids that are to be deleted.

```
String categoryIds[] = new String "Demo", "Sales_Category"};
```

5. Create the EntityOperation object and set Category Entities for above given Category Ids and add in an ArrayList i.e. entityList.

```
ArrayList entityList = new ArrayList();
EntityOperation entityObj = null;

for(int i=0;i< queryIds.length;i++){
  entityObj = new EntityOperation();

  //Entity Type could be REPORT /CATEGORY /QUERY /DASHBOARD2/
  PARAMETER /SAVEDREPORT /DASHBOARD_WIDGET /CUBEOBJECT
  entityObj.setEntityType("CATEGORY");
```

```
  entityObj.setEntityId(categoryIds [i]);
  entityList.add(entityObj);
}
```

6. Pass the arraylist of entities that are to be deleted from the Repository.

Method: **deleteEntities**

```
public ArrayList deleteEntities(ArrayList entityList, UserInfo
userInfo)
```

This method is used to delete entities.

Parameters:

- **entityList:** ArrayList of {@link com.intellica.client.common.EntityOperation} class instance.
- **UserInfo:** UserInfo object for authorization.

Part of Sample Code implementing "copyEntities":

```
String destCategoryId = "DataLoader";

lm. deleteEntities(entityList, requestorUserInfo);
```

Following are the APIs related to Mass Operation-

1. public ArrayList **deleteEntities**(ArrayList entityList, UserInfo userInfo)
2. public ArrayList **copyEntities**(ArrayList entityList, String destinationCatId, UserInfo userInfo)
3. public ArrayList **copyLinkEntities**(ArrayList entityList,String destinationCatId, UserInfo userInfo)
4. public ArrayList **deLinkEntities**(ArrayList entityList, UserInfo userInfo)
5. public ArrayList **moveEntities**(ArrayList entityList, String destinationCatId, UserInfo userInfo)

Dashboards

Using

```
using com.intellica.client.common.UserInfo;

using com.intellica.client.dashboard.Dashboard;
using com.intellica.client.dashboard.DashboardManager;
using com.intellica.client.reportutils.Category;
```

```
using com.intellica.client.exception.
         LayoutHandlerException;
```

Get Dashboard Details

This .NET API is used to get the complete detail about the Dashboard like its Category, Description, Access Rights etc

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Get the Dashboard details whose Dashboard Id is given.

Method: **getDashBoardDetails**

```
public DashBoard getDashBoardDetails (String dashboardId,
UserInfo userInfo)
```

This method is used to get the requested DashBoard Object whose ID is passed as an argument.

Parameters:

param dashboardId: The unique identifier of the dashboard for which the details are to be obtained from the Report Server.

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

Returns: Dashboard {@link com.intellica.client.dashboard.DashBoard2 DashBoard2} object containing dashboard details

```
Dashboard dashboard = dManager.getDashBoardDetails(DashboardId,
requestorUserInfo);
```

Get Dashboard List

This .NET API is used to get the complete list of Dashboards from the Intellicus Repository for the Requestor user

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Get the Dashboard list.

Method: **getDashBoardList**

```
public DashBoard getDashBoardList (Filter filterObj,
UserInfo userInfo)
```

This method is used to get the list of Dashboards.

Parameters:

param filterObj: Filter object to set EntityType, CategoryId, Depth.

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

Returns: Arraylist of Dashboards Objects

```
dashBoardList = dManager.getDashboardList(filterObj,
requestorUserInfo);
```

Get Dashboard Widget List

This .NET API is used to get the complete list of the Dashboard widgets from the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Get the Dashboard widget list whose Dashboard Id is given.

Method: **getDashboardWidgetList**

```
public ArrayList getDashboardWidgetList (Filter filterObj,
UserInfo userInfo)
```

Parameters:

param filterObj: Filter object to set EntityType DASHBOARD_WIDGET

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

Returns: Arraylist of Dashboard Widgets

```
dashBoardWidgetList = dManager.getDashboardWidgetList(filterObj,
requestorUserInfo);
```

Get Dashboard Widgets for Category

This .NET API is used to get the widgets present in given Category from the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Get the Dashboard widget list whose Dashboard Id is given.

Method: **getDashboardWidgetList**

```
public ArrayList getDashboardWidgetList (Filter filterObj,
UserInfo userInfo)
```

Parameters:

param filterObj: Filter object to set EntityType DASHBOARD_WIDGET

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

Returns: Arraylist of Dashboard Widgets.

```
//To get the Dashboard list
filterObj.setFilterField(Enums.Filters.EntityList.ENTITYTYPE,Enum
s.Filters.EntityList.EntityType.DASHBOARD_WIDGET);

//To get Widget list for given Category.
filterObj.setFilterField(Enums.Filters.EntityList.CATEGORYID,
categoryId);
```

```
filterObj.setFilterField(Enums.Filters.EntityList.TRAVERSAL,Enums
.Filters.EntityList.Traversal.DOWN);

dashBoardWidgetList = dManager.getDashboardWidgetList(filterObj,
requestorUserInfo);
```

Delete Dashboard

This .NET API is used to delete the Dashboard from the Intellicus Repository. Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Delete the Dashboard whose Dashboard Id is given.

Method: **getDashboardWidgetList**

deleteDashboard `(String dashBoardId, UserInfo` userInfo`)`

This method is used to delete the dashboard specified by the dashboard ID passed as an argument.

Parameters:

param dashBoardId: dashboard Id of the Dashboard i.e. to be deleted

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

```
dManager.deleteDashboard(dashBoardId, requestorUserInfo);
```

Get Dashboard Preferences

This .NET API is used to get the list of the Dashboards set in its Preferences.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Dashboard Manager class object for Dashboard related settings.

```
DashboardManager dManager = new DashboardManager();
```

4. Get the Dashboard Preferences.

Method: **getDashboardPreferences**

Public ArrayList getDashboardPreferences(String dashBoardId, UserInfo userInfo)

Getting Dashboard Preferences Object.

Parameters:

param filterObj: Filter containing ONWER_APPID, OWNER_ORGID

param userInfo: {@link com.intellica.client.common.UserInfo userInfo} object for authorization to get DashBoard.

```
dbPreferencesList = dManager.getDashBoardPreferences(filterObj,
requestorUserInfo);
```

Schedules

Scheduling of reports is very helpful for better utilization of server and printer resources. Reports that take longer to run can be scheduled to save your time.

Report that needs processing of large volume of data and need server and printer resources for long time can be scheduled to be generated over the weekend when load on servers would be relatively low. By scheduling a report, it can be sent to multiple deliverables at a time, which is otherwise not possible.

Get the list of scheduled jobs

This .NET API is used to get the list of All Scheduled Jobs.

This program:

1. Returns list of report schedules for a selected report.
2. Report schedule list can be filtered for ReportId and UserId

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/Schedules/ GetScheduleJobList.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Scheduler Manager class object. This is the controller class for all operations related to scheduler. The class provides methods which acts as an interface for sending diffrent requests related to Scheduler to report engine from the jsps.

```
SchedulerManager schdMgr      =     new SchedulerManager();
```

4. Set the filter values corresponding to respective fields for getting filtered list of Scheduled Jobs.

```
// Create a Filter class Object
Filter filterObj = new Filter();

// set filter for private schedule jobs
filterObj.setFilterField(Enums.Filters.ScheduledJob.ISPUBLIC,
false);

// set filter for dedicated(Non-shared) schedule jobs
filterObj.setFilterField(Enums.Filters.ScheduledJob.ISSHARED,
false);

//Set the ReportId filter for getting all report schedules for
this Report
String reportId    = "93F21A40-01DD-DFFC-C874-A7E30A27A127";
filterObj.setFilterField(Enums.Filters.ScheduledJob.REPORTID,repo
rtId);

String userID      = "HostUser";
filterObj.setFilterField(Enums.Filters.ScheduledJob.USERID,userID
);
```

```
//Set the orgId filter for getting list of report schedules of a
user of this organization only
String orgId = "HostOrg";
filterObj.setFilterField(Enums.Filters.ScheduledJob.ORGID,orgId);
```

5. Get the Scheduled Job List based on filter applied.

Method: getScheduledJobList

```
public ArrayList getScheduledJobList(Filter filterObj,
UserInfo userInfo) throws SchedulerException
```

This method returns the scheduleJobList in ArrayList containing HashMap as each element for each row. The HashMap contains the details required for showing the list like the schedule job name, schedule job id etc. as name value pair.

Parameters:

- **Filter:** May take below fields

i.e.

```
Enums.Filters.ScheduledJob.ISPUBLIC
Enums.Filters.ScheduledJob.ISSHARED
Enums.Filters.ScheduledJob.REPORTID
Enums.Filters.ScheduledJob.ORGID
Enums.Filters.ScheduledJob.SCHEDULE
Enums.Filters.ScheduledJob.USERID
Enums.Filters.ScheduledJob.BATCHID
```

- **UserInfo:** The userInfo object for authorization.

```
// This will return array of Scheduled Jobs list from the
Intellicus Repository
ArrayList jobList = schdMgr.getScheduledJobList(filterObj,
                                    requestorUserInfo);
```

Create a Schedule Job

This .NET API is used to create Scheduled Job that runs only once at a given time.

This program

1. Sets business parameters required to execute the report.
2. Creates a report delivery task.
3. Creates a ScheduleJob using 2).

for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SchedulerManager class object. This is the controller class for all operations related to scheduler. The class provides methods acting as an interface for sending diffrent requests related to Scheduler to report engine from the jsps.

```
SchedulerManager schdMgr      =     new SchedulerManager();
```

4. Create an instance of Task and set its various properties.

```
Task task          =     new Task();
String batch Name = "Task Once";
task.setBatchName(batch_Name);
String rep Id = "93F21A40-01DD-DFFC-C874-A7E30A27A127";
task.setReportID(rep_Id);
task.setReportFormat(InteraConstants.ReportFormats.PDF);
task.setIsPublic(false);
task.setIsShared(false);
```

5. Set the Delivery Options for Task and related properties.

For E-mail

```
task.setDeliveryOperationType(com.intellica.client.common.Enums.B
atch.DispatchOperationTypes.EMAIL);

// get EmailProperty object for setting Email attributes
EMailProperty eMailProp = task.getEMailProperty();

String mail_To = "HostUser@HostOrg.com";
String mail_Sub = "Mail Subject goes here";
String mailMsg = "Dear User, Please find <%Report_Name%>
attached";

eMailProp.setMailAttach(true);
eMailProp.setMailTO(mail_To);
eMailProp.setMailSUB(mail_Sub);
eMailProp.setMailMSG(mailMsg);
```

For FTP

Part of Sample Code implementing FTP Delivery Option**:**

```
task.setDeliveryOperationType(com.intellica.client.common.Enums.B
atch.DispatchOperationTypes.FTP);

String serverName = "ftp.intellicus.com";
String username = "upload";
String password = "download";
String folderName = "transferbin";
String fileName = "MyFile";

// get FTPProperty object for setting Email attributes
FTPProperty ftpProp = task.getFTPProperty();
ftpProp.setServerName(serverName);
ftpProp.setUserName(username);
ftpProp.setPassword(password);
ftpProp.setFolderName(folderName);
ftpProp.setFileName(fileName);
```

For Publish Report

Part of Sample Code implementing Publish Delivery Option:

```
task.setDeliveryOperationType(com.intellica.client.common.Enums.B
atch.DispatchOperationTypes.PUBLISH);

//get PublishProperty object for setting publish attributes for
//the task
PublishProperty publishProp = task.getPublishProperty();

String fileName = "Shared_Publish1";
//setter method for saveFileName member variable.
publishProp.setSaveFileName(fileName);

//setter method for isPublic member variable.
publishProp.setIsPublic(true);     //True : Set the Published
                                   //Report as Public

//setter method for periodType member variable.
//Enums.Batch.PeriodType.ENDPERIOD : valid upto end of Day, Hour,
//Month, Week, Year.
//For ENDPERIOD, call setEndPeriod to set the value to Day, Hour,
//Month, Week, Year.
//Enums.Batch.PeriodType.EXPIRYDATE  :  valid  upto  a  particular
//date
//For EXPIRYDATE, call setPublishValidUptoFixedDate to set the
//end Date.
//Enums.Batch.PeriodType.INTERVALPERIOD  :  valid  upto  fixed
//interval
//For INTERVALPERIOD, call setIntervalValue and then set the
//value for Interval Period.
publishProp.setPeriodType(Enums.Batch.PeriodType.ENDPERIOD);

//setter method for endPeriod member variable when Period Type is
//set to ENDPERIOD
//It sets the endPeriod value to either Hour/Day/Week/MonthYear.
publishProp.setEndPeriod(com.intellica.client.common.Enums.Batch.
EndPeriod.YEAR);
```

For Print Report

Part of Sample Code implementing Publish Delivery Option:

```
task.setDeliveryOperationType(com.intellica.client.common.Enums.B
atch.DispatchOperationTypes.PRINT);
//This class holds PrintProperty for Task.
//All the variables are used for making a print request.
```

```
PrintProperty printProp = task.getPrintProperty();

int printCopies = 1;
String printerName = "Microsoft XPS Document Writer";

//Number of copies to be printed
printProp.setPrintCopies(1);
//Printer name that is used for printing
printProp.setPrinterName(printerName);
//Set the number of pages to be printed
printProp.setPrintPageRange("All");
```

6. Create Scdedule for the Scheduled Job and set its Frequency Type

- Daily Freequency Type

To Set Frequency type as **Daily**
Enums.Schedule.FrequencyTypes.DAILY

```
Schedule schedule =     new Schedule();
schedule.setFrequencyType(Enums.Schedule.FrequencyTypes.DAILY);
```

- Monthly Freequency Type

To Set Frequency type as **Monthly**
Enums.Schedule.FrequencyTypes.MONTHLY

```
Schedule schedule   =     new Schedule();
schedule.setFrequencyType(Enums.Schedule.FrequencyTypes.MONTHLY);
```

- Weekly Freequency Type

To Set Frequency type as **Weekly**
Enums.Schedule.FrequencyTypes.WEEKLY

```
Schedule schedule =     new Schedule();
schedule.setFrequencyType(Enums.Schedule.FrequencyTypes.WEEKLY);
```

7. Create Scheduled Job and associate created Task and Schedule to this Scheduled Job.

```
//create a SCHEDULEDJOB
ScheduleJob  schdJob    =     new ScheduleJob(task,schedule);
```

8. Set Run Type for Scheduled Job

This is setter method for runType member variable.

public void setRunType(String runType)

```
Enums.ScheduledJob.JobTypes.NOW
Enums.ScheduledJob.JobTypes. ONCE
Enums.ScheduledJob.JobTypes.RECUR
```

Parameters:

- **RunType: T**akes String value.

Part of Sample code for "NOW" as Run-type:

```
ScheduleJob  schdJob          =     new ScheduleJob();
schdJob.setRunType(Enums.ScheduledJob.JobTypes.NOW);
```

Part of Sample code for "ONCE" as Run-type:

```
ScheduleJob  schdJob          =     new ScheduleJob();
schdJob.setRunType(Enums.ScheduledJob.JobTypes.ONCE);
schdJob.setOnceDate("10/03/2007");
schdJob.setOnceTime("10:52:10");
```

Part of Sample code for "RECUR" as Run-type:

```
ScheduleJob  schdJob          =     new ScheduleJob();
schdJob.setRunType(Enums.ScheduledJob.JobTypes.RECUR);
schdJob.setFrequencyType(Enums.Schedule.FrequencyTypes.DAILY);
schdJob.setAfterDays(2);
```

9. Add the Scheduled Job in the Repository.

Method : addScheduleJob(schdJob,userInfo)
public void addScheduleJob(ScheduleJob scheduleJob, UserInfo userInfo)

Adds ScheduleJob alongwith Schedule and Task depending on whether they are created Shared or Dedicated in the Intellicus Repository.

Parameters:

- **ScheduleJob:** The ScheduleJob object.
- **UserInfo:** The userInfo object for authorization.

```
schdMgr.addScheduleJob(schdJob,requestorUserInfo);
```

Delete Schedule Job

This .NET API is used to delete dedicated scheduledJob selected from the list.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SchedulerManager class object. This is the controller class for all operations related to scheduler. The class provides methods which acts as an interface for sending diffrent requests related to Scheduler to report engine from the jsps.

```
SchedulerManager schdMgr      =     new SchedulerManager();
```

4. Delete Scheduled Job with given scheduledJob Id.

Method : deleteScheduledJob

```
public void deleteScheduledJob(String jobID,
                               UserInfo userInfo)
```

This method deletes an existing ScheduledJob.

Parameters:

- **JobID:** Takes a String value.
- **UserInfo:** The userInfo object for authorization.

```
String schdJobId   =     "114016239210";
schdMgr.deleteScheduledJob(schdJobId,requestorUserInfo);
```

Get the input parameter names and their values set at the task creation time.

This .NET API is used to get the input Parameter names whose values are set at the Task Creation time.

This program:

1. Prepare a scheduler manager object
2. Get the task object for the specified task id
3. Get the hash map of input parameter from the task object
4. Get the input parameter names and their values set at the task creation time.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SchedulerManager class object. This is the controller class for all operations related to scheduler. The class provides methods acting as an

interface for sending different requests related to Scheduler to report engine from the jsps.

```
SchedulerManager schdMgr      =     new SchedulerManager();
```

4. Get the Task instance for given Task Id whose input parameters are to be obtained.

```
String taskId      =     "1169627272589";
Task taskObj =     schdMgr.getTask(taskId, requestorUserInfo);
```

5. Get the Input Parameters for this Task.

Method: getInputParamMap

```
public java.util.HashMap getInputParamMap()
```

getter method for inputParamMap member variable.

Returns:
inputParamMap as HashMap.

```
HashMap inputParamMap    =     taskObj.getInputParamMap();
```

Cab Deployment

Upload and Deploy cab/irb file

This .NET API is used to upload and deploy cab/irb file both.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/Cab Deployer/CabDeployer.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Set the application path and cab/irb File path.

```
//path where the intellicus web client is deployed
String applicationPath = "C:/Program Files/Intellicus ";

CabFile.setApplicationPath(applicationPath+ File.pathSeparator);

//actual path of irb file
String irbFilePath = "c:/ABC.irb";
```

4. Upload the Cab File.

```
CabFile  cabFileObj  = new
CabFile(irbFilePath,requestorUserInfo);
```

5. Deploy the Cab File.

Method: deploy(String cabPath, UserInfo userInfo)

```
public java.util.HashMap deploy(String cabPath,
                                 UserInfo userInfo)
```

This method takes the path of the cab file i.e. to be deployed and deploys it in the Intellicus Repository.

Part of Sample Code implementing "deploy":

```
cabFileObj.deploy(irbFilePath,irbFilePath+".log"
,requestorUserInfo);
```

Report Object

Query Object

Query objects contain SQL query and list of the fields along with their attributes for using in Ad hoc reports and sometimes in Standard reports.

These are designed from Intellicus web portal at Main Menu -> Repository-> Report Objects -> Query Objects screen.

These objects are stored as XML in Intellicus repository.

The Intellicus web UI provides option for Adding, Deleting and Editing Query Objects.

Almost all of the operations provided by the Intellicus web UI are also available from the .NET APIs.

This document intends to explain the .NET APIs with Query Object manipulation purpose; there would be sample code files associated with this document.

Attribute of Query Objects:

Below table lists all the attributes of query objects which can be modified through .NET APIs.

Information	Type	Description
Name	Mandatory	Must be Unique
SQL Statement	Mandatory	The SQL Statement which sources data for this QO.

Information	Type	Description
Connection Name	Optional	If provided, then this Query object becomes associated to the given connection.
Column Details	Optional As array list.	These are the details of the fields.
Column field Name	Mandatory (if column details are given)	Must be returned by above SQL. Here it is used to match the field and associate below given caption and width attributes to it.
Column data type	Mandatory, CHAR/NUMBER/DATE	Default – Field data type returned by SQL.
Column Caption	Optional, String	Default – Field name returned by SQL with spaces truncated and Case converted to Title case.
Column Width	Mandatory, Integer	
Column Hidden	Optional, Bool	Default – False
Column Hyper Link	Optional, String	
Column Key Field Name	Optional, String	
Column Group Label	Optional, String	Adds this column to a group, Group label must exist.
Column Alignment	Optional	Default- Left for Char and Date, Right for Numeric

Add Report Object

This .NET API is used for creating a new Report Object.

In this API, you would require to provide SQL query to create the Query Object.

You may also provide column details like column name, caption, data type, width, output format and hyperlink.

Refer to attributes table for the list of attributes you can set during creation.

Steps

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create query object by using the following method:

Method

```
Enums.IRO.TYPE.QUERY.QUERY
Enums.IRO.TYPE.QUERY.PARAMETER
```

```
Enums.IRO.TYPE.QUERY.nFormat
Enums.IRO.TYPE.QUERY.Format
Enums.IRO.TYPE.QUERY.nParameter
Enums.IRO.TYPE.QUERY.nQuery
Enums.IRO.TYPE.QUERY.nReportObjectType
```

```
public static ReportObject createReportObject
             (Enums.IRO.TYPE.QUERY) of ReportObject class.
```

4. **Set the SQL Query** - SQL can be set by creating com.impetus.intera.layout.sqleditor.SQLEditor class object and calling setSQL () method.

```
SQLEditor sqlEditor = qoObj.getSqlEditor();
  sqlEditor.setSQL(sqlQuery,true);
```

Parameters:

- **sqlQuery:** This is to specify the query string to be used for fetching data.

5. **Set the columns**- Columns can be set by passing ArrayList containing the QueryObjectColumn object. (QueryObjectColumn contains column attributes column name, data type caption, and width and others as mentioned in above table).

```
Public void setColumnDetails (ArrayList columnDetails)
```

Parameters:

- **ColumnDetails:** This is the array list of columns to be added in the query object.

Objects of QueryObjectColumn can be created by using the constructor of com.impetus.intera.reportobjects.QueryObjectColumn in which user provides attributes like column name, caption, width and datatype to the columns.

```
QueryObjectColumn(String columnName,
                  String caption,
                  String width,
                  String dataType)
```

```
String colName = "Product";
String caption = "Product Name";
QueryObjectColumn qoc1 = new QueryObjectColumn(colName, caption
,"","");
```

6. Optionally, You may set other attributes of this column also.

```
// To hide this column
qoc1.setHidden(true);

String groupCaption = "Job";
//To set this column in a group
qoc1.setGroupLabelCaption(groupCaption);

// To set hyperlink for this column.
String hyperLink = "http://www.google.com";
qoc1.setHyperlink(hyperLink);
```

7. Set the connection name- setConnectionName method of com.impetus.intera.reportobjects.QueryObject class can be used to set the name of the connection used by Query Object

```
public void setConnectionName(String connectionName)
```

Parameters:

- **ConnectionName:** The database connection Name to be used for query object.

5. Set if filter is mandatory-This method is used to set the isFilterMandatory.

```
public void setIsFilterMandatory(bool isFilterMandatory)
of QueryObjects class.
```

Parameters:

IsFilterMandatory: Whether mandatory filters are applied or not.

6. Set the group caption.

```
public void setGroupCaption(bool isGroupCaption)
of QueryObjects class.
```

8. Add the query object. This can be done by calling addReportObject method of ReportObjectManager.]

```
public void addReportObject(ReportObject reportObject,
                            UserInfo userInfo).
```

Parameters:

- **ReportObject:** Java object of Query object to be deleted.
- **UserInfo:** Object of UserInfo class.

Part of Sample Code implementing "Add Report Object"

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

String qoName = "TestQuery";
String queryId = "qoObjID";
String connName = "ReportDB";

qoObj.setName(qoName);
qoObj.setId(queryId);
qoObj.setCached(true);
qoObj.setConnectionType("Default");
qoObj.setConnectionName(connName);
qoObj.setSource("SQL");

reportObjectManager.addReportObject(qoObj, requestorUserInfo);
```

Add Query Object from CSV Source

This .NET API is used for creating a new Query Object from a CSV source file.

In this API, you would require to provide SQL query to create the Query Object.

You may also provide column details like column name, caption, data type, width, output format and hyperlink.

Refer to attributes table for the list of attributes you can set during creation.

Steps

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create CSV Source Object

```
//CSVSource object
CSVSource csvSource =new CSVSource();
```

4. Set path for the CSV Source

```
//Set path of file along with name under connection
```

```
//if connection is created at "E:\csv" and CSV file is in
//"E:\csv\subfolder\test.csv"
//then setpath will be csvSource.setPath("subfolder/test.csv");
csvSource.setPath("test/deptdetails_test.csv");
//Set separator if not a default separator i.e ','
csvSource.setSeparator("|");
```

5. Create query object with this CSV Source and set various attributes

```
QueryObject qoObj=QueryObject.createQueryObject("QoName01",
csvSource, "ConnectionFile", requestorUserInfo);

//To set the ID for Query Object manually
qoObj.setId("QoId01");

//Set Category ID in order to create query object in a specific
category.
qoObj.setCategoryId("Cat1");

//To set caching true or false for the Query Object
qoObj.setCached(true);

//To set the description for the Query Object
qoObj.setDescription("This is a Test Query .");
```

6. Save query Object in Repository

```
//Save Query Object
qoObj.save(Enums.QueryObject.Action.ADD, requestorUserInfo);
```

Get Query Object by Name

This .NET API is used to get a java object representing Query Object by its name.

The object can be used to fetch details and/or then to modify details and replace Query Object in the repository.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/ReportObjects/ GetReportObjectByName.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Call getReportObjectByName method of com.impetus.intera.reportobjects.ReportObjectManager by type casting returned object to com.impetus.intera.reportobjects.QueryObject

```
public ReportObject getReportObjectByName
            ( String reportObjectType,
              String reportObjectName,
              UserInfo userInfo)
```

Parameters

- **Report Object Type:** Pass *Enums.IRO.TYPE.QUERY.*
- **Report Object Name:** Name of the Query Object to be fetched.
- **User Info:** Credentials of user requesting this information.

For getting attributes of Query Object you need to type-cast the Report Object into Query Object.

```
QueryObject
qoObj=(QueryObject)reportObjectManager.getReportObjectByName(Enum
s.IRO.TYPE.QUERY,qoName,userInfo);
```

4. Call getColumnDetails method on the java object of query object returned. This method will return array list of all the columns.

Method

```
public ArrayList getColumnDetails()
```

Returns:

Returns the array list of columnDetails.

5. Call getConnectionName method on the java object of query object returned. This method is used to get the name of the connection used by Query Object

```
public String getConnectionName()
```

Returns:

Returns the name of the connection.

Call isFilterMandatory method on the java object of query object returned.

```
public bool isFilterMandatory()
```

Returns:

Returns whether mandatory filters are applied or not.

Part of Sample Code implementing "get Report Object By Name"

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

String qoName = "All Country Sales";
QueryObject
qoObj=(QueryObject)reportObjectManager.getReportObjectByName(Enum
s.IRO.TYPE.QUERY,qoName, requestorUserInfo);
```

Get Report Object List

This .NET API is used to retrieve the list of Report Objects from Intellicus repository.

Steps

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Call getReportObjectList of com.impetus.intera.reportobjects.ReportObjectManager to retrieve an array List of Query Objects from the Intellicus repository.

```
public ArrayList getReportObjectList(String roType,UserInfo
userInfo)
```

Parameters:

- **Report Object Type:** Pass Enums.IRO.TYPE.QUERY.

The returned ArrayList will have each element as a Query Object retrieved from the Intellicus repository.

Part of Sample Code implementing "get Query Object List"

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

ArrayList reportObjList=new ArrayList();

ReportObject
repObjectQuery=ReportObject.createReportObject(Enums.IRO.TYPE.QUE
RY);

reportObjList=reportObjectManager.getReportObjectList(repObjectQu
ery.getType(),requestorUserInfo);
```

Other related APIs are:

```
public ArrayList getReportObjectList(String roType, Filter filterObj, UserInfo userInfo)
```

```
public ArrayList getReportObjects(String roType, bool cached, UserInfo userInfo)
```

```
public ArrayList getReportObjects(String roType, UserInfo userInfo)
```

Get Parameter Object List

This .NET API is used to retrieve the list of Query Objects from Intellicus repository.

Steps

1. Call getReportObjectList of com.impetus.intera.reportobjects.ReportObjectManager to retrieve an array List of Query Objects from the Intellicus repository.

```
public ArrayList getReportObjectList(String roType,UserInfo
userInfo)
```

Parameters:

- **Report Object Type:** Pass Enums.IRO.TYPE.QUERY.

The returned ArrayList will have each element as a Query Object retrieved from the Intellicus repository.

Part of Sample Code implementing "get Parameter Object List".

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

ArrayList reportObjList=new ArrayList();

ReportObject
repObjectQuery=ReportObject.createReportObject(Enums.IRO.TYPE.
PARAMETER);

reportObjList=reportObjectManager.getReportObjectList(repObjectQu
ery.getType(),requestorUserInfo);
```

Delete Report Object

This .NET API is to delete an existing Report Object from Intellicus Repository.

For deleting a report object, you would require to provide the name of the Query Object to be deleted. If report object with the given name exists, then that would be deleted, otherwise it gives an error message that "No such Query Object Found".

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Get Report Object –Retrieve the Report Object to be deleted from the Intellicus Report repository. For this, call getReportObjectByName() method of com.impetus.intera.reportobjects.ReportObjectManager.

```
Public ReportObject getReportObjectByName (Enums.IRO.TYPE.QUERY,
String reportObjectName, UserInfo userInfo)
```

Parameter:

- **Report Object type:** Pass Enums.IRO.TYPE.QUERY for the Query Object.
- **ReportObjectName:** The name of the query object to be deleted.
- **UserInfo:** Object of UserInfo class.

```
//To get the Object for the Query Object Class.
QueryObject qoObj=(QueryObject)reportObjectManager.
getReportObjectByName(Enums.IRO.TYPE.QUERY,qoName,userInfo);
```

Delete the Report Object – For deleting the Report Object, call deleteReportObject method of

```
com.impetus.intera.reportobjects.ReportObjectManager

public void deleteReportObject(ReportObject reportObject,
                               UserInfo userInfo)
```

Parameters:

- **ReportObject:** .NET object of Query object to be deleted.
- **UserInfo:** Object of UserInfo class.

```
  //This will used to Delete the Query Object.
reportObjectManager.deleteReportObject(qoObj,userInfo);
```

Part of Sample Code implementing "Delete Report Object"

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

String qoName = "newqo1";
QueryObject
qoObj=(QueryObject)reportObjectManager.getReportObjectByName(Enum
s.IRO.TYPE.QUERY,qoName,userInfo);
```

```
reportObjectManager.deleteReportObject(qoObj,requestorUserInfo);
```

Replace Query Object

Query object can be replaced by calling the replaceReportObject() method of com.impetus.intera.reportobjects.ReportObjectManager.

```
  reportObjectManager.replaceReportObject(qObject,userInfo);
```

Part of Sample Code implementing "Replace Report Object"

```
ReportObjectManager reportObjectManager = new
ReportObjectManager();

String qOName="TestReplace";

QueryObject
qObject=(QueryObject)reportObjectManager.getReportObjectByName(En
ums.IRO.TYPE.QUERY,qOName,requestorUserInfo);

String colName = "BANKS.BANKID";
String caption = "BankID";
String width = "3";
String dataType = "NUMBER";
String grpLabelCaption = "Job";
//create the column to be added and set its attributes.
QueryObjectColumn qocAdd = new
QueryObjectColumn("BANKS.BANKID","BankID","3","NUMBER");
qocAdd.setAlignment(1);
qocAdd.setHidden(false);
qocAdd.setGroupLabelCaption(grpLabelCaption);

//add this column in the query object
qObject.addColumn(qocAdd);

reportObjectManager.replaceReportObject(qObject,requestorUserInfo
);
```

Other related APIs for accessing QO/PO are:

1. public LinkedHashMap getAllPOMap()
2. public LinkedHashMap getAllQOMap()

OLAP

Using

```
//OLAP related usings
using com.impetus.intera.layout.datasource.qo.QODataSource;
using com.impetus.intera.layout.upx.FetchDataSource;
using com.impetus.intera.reportobjects.CubeObject;
using com.impetus.intera.reportobjects.QueryObjectColumn;
using com.impetus.intera.reportobjects.QueryObjectFacade;
using com.impetus.intera.reportobjects.ReportObjectException;
using com.impetus.intera.reportobjects.ReportObjectManager;
using com.intellica.client.common.EntityProperties;
using com.intellica.client.common.EntityProperty;
using com.intellica.client.common.Enums;
using com.intellica.client.layout.LayoutManager;
using com.intellica.client.olap.CubeDataSource;
using com.intellica.client.olap.DimMapping;
using com.intellica.client.olap.Dimension;
using com.intellica.client.olap.Hierarchy;
using com.intellica.client.olap.Level;
using com.intellica.client.olap.Measure;
using com.intellica.client.olap.MeasureGroups;
```

Cube Object

AddCube

This .NET API is used to create a new Cube Object in the Intellicus Repository.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/OLAP/ AddCube.java for sample code of this use case.

Steps :

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. addReportObject() method of ReportObjectManager class

Method: addReportObject

This is the method of ReportObjectManager class i.e. used to add Cube Object in Repository

```
public void addReportObject(ReportObject reportObject,
                            UserInfo userInfo).
```

Parameters:

- **ReportObject:** Java object of Cube object to be added.
- **UserInfo:** Object of UserInfo class.

Code Snippet of the sample code-

Set the Query Id for which Cube Object is to be created.

```
String queryId = "SampleCubeQuery";
```

Set other details.

```
String categoryId = "CT"; //Category in which cube object is to be added
String measureName = "Count";
String measureDataField = "ResourceId";
String measureSummaryFunction = "2"; //1 for sum, 2 for count
//Set cube object properties
CubeObject cubeObj = new CubeObject();
cubeObj.setCategoryId(categoryId);  //category Id where Cube Object is to be saved.
cubeObj.setName("Cube_"+queryId);
cubeObj.setId("Cube_"+queryId);
cubeObj.setCoType("INTERNAL");
cubeObj.setCubeUniqueName("Cube_"+queryId);

FetchDataSource fetchDataSource = new FetchDataSource();

QODataSource qoDataSrc = new QODataSource();
qoDataSrc.setId(queryId);   //Query Object Id
fetchDataSource.setSource("QO");
fetchDataSource.setSourceObj(qoDataSrc);

HashMap<String, FetchDataSource> cubeQueries = new HashMap<String, FetchDataSource>();
cubeQueries.put("CUBE_QUERY_"+queryId, fetchDataSource);
cubeObj.setCubeQueries(cubeQueries);
```

Adding Dimensions

For adding dimensions, first iterate over the list of columns/fields of Query object, then add them.

```
ArrayList<Dimension> dimensionList = new ArrayList<Dimension>();
//Getting columns/fields in QO so as to iterate to create Dimensions
```

```
QueryObjectFacade                queryObjFacade                =
(QueryObjectFacade)rom.getReportObjectFacadeById(Enums.IRO.TYPE.Q
UERY, queryId, requestorUserInfo);
ArrayList<QueryObjectColumn> qoColumnsList =
queryObjFacade.getColumnList();
Iterator itr = qoColumnsList.iterator();

//iterating through the Column List
while(itr.hasNext()){
  QueryObjectColumn qoColumn  = (QueryObjectColumn)itr.next();
  Response.Write("Column Name = "+qoColumn.getColumnName());
  columnName = qoColumn.getColumnName();

  dimMappingObj = new DimMapping();
  dimMappingObj.setDimensionId("DIMENSION_"+columnName);
  dimMappingMap.put("DIMENSION_"+columnName,dimMappingObj);

  Dimension dimensionObj = new Dimension();
  dimensionObj.setDimID("DIMENSION_"+columnName);
  dimensionObj.setIsValuesRestricted(false);
  dimensionObj.setGISEnabled(false);
  dimensionObj.setUniqueName("DIMENSION_"+columnName);
  dimensionObj.setName(columnName);
  dimensionObj.setType("REGULAR");
  HashMap<String,    CubeDataSource>    dataSourceObj    =    new
HashMap<String, CubeDataSource>();
  CubeDataSource cubeDataSourceObj = new CubeDataSource();
  cubeDataSourceObj.setId("CUBE_QUERY_"+queryId);
  cubeDataSourceObj.setType("REFERENCED");
  dataSourceObj.put("CUBE_QUERY_"+queryId,cubeDataSourceObj );
  dimensionObj.setFetchSourcesForDimension(dataSourceObj);

  ArrayList<Hierarchy>         hierarchyList         =         new
ArrayList<Hierarchy>();
  Hierarchy hierachy = new Hierarchy();
  hierachy.setName("HIERARCHY");
  hierachy.setUniqueName("DIMENSION_HIERARCHY_"+columnName);
  ArrayList<Level> levelList = new ArrayList<Level>();
  Level level = new Level();
  //to allow All as member value of a dimension
  if(allowAll){
        level.setAll(true);
        level.setUniqueName("DIEMENSION_LEVEL_0");
        level.setName("Hierarchy.ALL");
```

```
        levelList.add(level);
    }
    level = new Level();
    level.setAll(false);
    level.setUniqueName("DIEMENSION_LEVEL_1");
    level.setName(columnName);
    level.setDataField(columnName);
    level.setAssociatedDimQueryId("CUBE_QUERY_"+queryId);
    levelList.add(level);

    hierachy.setLevels(levelList);
    hierarchyList.add(hierachy);
    dimensionObj.setHierarchies(hierarchyList);
    dimensionList.add(dimensionObj);
}

cubeObj.setDimMappings(dimMappingMap);
```

Adding the measure Dimension

```
//Adding Measure dimension
Dimension dimensionObj = new Dimension();
dimensionObj.setDimID("Dim_Measures");
dimensionObj.setIsValuesRestricted(false);
dimensionObj.setGISEnabled(false);
dimensionObj.setUniqueName("Measures");
dimensionObj.setName("Measures");
dimensionObj.setType("MEASURE");

ArrayList<Hierarchy> hierarchyList = new ArrayList<Hierarchy>();
Hierarchy hierachy = new Hierarchy();
hierachy.setUniqueName("Measures");
ArrayList<Level> levelList = new ArrayList<Level>();
Level level = new Level();

level.setAll(false);
level.setDataType("Regular");
level.setUniqueName("MeasuresLevel");
levelList.add(level);

hierachy.setLevels(levelList);
hierarchyList.add(hierachy);
dimensionObj.setHierarchies(hierarchyList);
dimensionList.add(dimensionObj);
```

```
cubeObj.setDimensions(dimensionList);

String measureId = "MEASURE_1";
MeasureGroups measureGrpsObj = new MeasureGroups();
Map<String, ArrayList<String>> measureMap = new HashMap<String,
ArrayList<String>>();
ArrayList<String> measureIdList = new ArrayList<String>();
measureIdList.add(measureId);
measureMap.put("MeasureGroup", measureIdList);
measureGrpsObj.setMeasureGroups(measureMap);
measureGrpsObj.setMeasureGroups(measureMap);

Map<String, CubeDataSource> dataSourceMap = new HashMap<String,
CubeDataSource>();
ArrayList<String> DataSourceId = new ArrayList<String>();
CubeDataSource cubeDataSrcObj = new CubeDataSource();
cubeDataSrcObj.setId("CUBE_QUERY_"+queryId);
cubeDataSrcObj.setType("REFERENCED");
dataSourceMap.put("CUBE_QUERY_"+queryId, cubeDataSrcObj);
measureGrpsObj.setFetchSources(dataSourceMap);

cubeObj.setMeasureGroups(measureGrpsObj);

//Adding Measures
HashMap<String,Measure>          measureList          =          new
HashMap<String,Measure>();
Measure measureObj = new Measure();

measureObj.setId(measureId);
measureObj.setUniqueName(measureId);
measureObj.setName(measureName);
measureObj.setIsDefault(true);
//measureObj.setFormatType("2");
measureObj.setFormat("0");
measureObj.setQueryID("CUBE_QUERY_"+queryId);
measureObj.setDataField(measureDataField);
measureObj.setSummaryFunction(measureSummaryFunction);
measureList.put(measureId, measureObj);
cubeObj.setMeasures(measureList);
```

Calling addReportObject() method

```
//Method used to add Cube Object in Repository
```

```
rom.addReportObject(cubeObj, requestorUserInfo);
```

Get Dimensions List

This .NET API is used to get the list of dimensions for provided Cube Object Id.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/OLAP/GetDimensionList.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. addReportObject() method of ReportObjectManager class

Method: getDimensions

This is the method of CubeObject class i.e. used to get the dimensions of requested Cube object

```
public ArrayList<Dimension> getDimensions()
```

Code Snippet of the sample code-

Set the Cube Object Id whose dimensions are to be fetched.

```
String cubeName = "SampleCubeQuery";
```

Getting Cube Object Detail to get its dimensions.

```
OLAPManager olapMgr = new OLAPManager();
CubeObject cubeObj =
(CubeObject)olapMgr.getCubeObjectDetail(cubeName, cubeName,
requestorUserInfo);
ArrayList<Dimension> dimList = cubeObj.getDimensions();
Dimension dim = null;
//iterating through the list of dimensions
for(int i=0;i<dimList.size();i++){
      dim = (Dimension)dimList.get(i);
      Response.Write("Dimension Id = "+dim.getUniqueName()+", Name =
"+dim.getName());
}
```

Delete Cube Object

This .NET API is used to delete the cube object from the Intellicus Repository for given Cube Object Id.

Refer to <Intellicus_Install_Path>/SampleCodes/.NET APIs/OLAP/DeleteCubeObject.java for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Delete the Report Object

Method: deleteReportObject

For deleting the Report Object, call deleteReportObject method of ReportObjectManager

```
com.impetus.intera.reportobjects.ReportObjectManager

public void deleteReportObject(ReportObject reportObject,
                               UserInfo userInfo)
```

Parameters:

- **ReportObject:** Java object of Cube object to be deleted.
- **UserInfo:** Object of UserInfo class.

```
//This will be used to delete the Cube Object.
reportObjectManager.deleteReportObject(qoObj,userInfo);
```

Part of Sample Code implementing "Delete Cube Object"

Set the Cube Object Id that is to be deleted.

```
//Cube object Id that is to be deleted
String cubeObjId = "Cube_QOForCube_1";
```

Getting the object of Cube and deleting it from Repository

```
//getting ReportObject corresponding to provided Cube Object Id
ReportObject rObj = (ReportObject)rom.getReportObject
(Enums.IRO.TYPE.CUBEOBJECT, cubeObjId, cubeObjId, true
,requestorUserInfo);

//Method to delete Cube
rom.deleteReportObject(rObj, requestorUserInfo);
```

Build Cube Object

This .NET API is used to build the cube for provided Cube Object.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Build the Cube Object

Method: buidCubeObject
For building the cube, call method of ReportObjectManager

```
com.impetus.intera.reportobjects.ReportObjectManager

public void buildCubeObject(String cubeBuildXML, UserInfo
userInfo) throws ReportObjectException{
```

Parameters:

- **cubeBuildXML:** XML of CubeObject Build.
- **UserInfo:** Object of UserInfo class.

Part of Sample Code implementing "Build Cube Object"

Set the Cube Object Id that is to be build.

```
//Cube object Id that is to be deleted
String cubeObjId = "Cube_QOForCube_1";
```

Getting the object of Cube and deleting it from Repository

```
String buildXML = "<CUBEOBJECT_BUILD ID=\""+cubeObjId+"\">\n";
buildXML += "\t\t\t<BUILD_INFO>\n\t\t\t\t<PROPERTIES>\n";
buildXML += "\t\t\t\t<PARAM NAME=\"BUILD_ON_HADOOP\">\n";
buildXML += "\t\t\t\t\t<VALUE><![CDATA[FALSE]]></VALUE>\n";
buildXML += "\t\t\t\t</PARAM>\n";
buildXML += "\t\t\t\t</PROPERTIES>\n";
buildXML += "\t\t\t</BUILD_INFO>\n";
buildXML += "<SCHEDULE_INFO
JOBTYPE=\"NOW\"></SCHEDULE_INFO></CUBEOBJECT_BUILD>";

//method used to build the cube object.
rom.buildCubeObject(buildXML, requestorUserInfo);
```

Get Build Status

This .NET API is used to get the build status for the given Cube.
Posiible values of build status –
"BUILDING", "COMPLETED", "PENDING", "FAILED", "BUILD_SUBMITTED"

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Get the build status for the Cube Object

Method: getCubeObjectBuildStatus
For getting the build status, call method of ReportObjectManager

```
com.impetus.intera.reportobjects.ReportObjectManager

public COBuildStatus getCubeObjectBuildStatus(Filters
filters,UserInfo userInfo) throws ReportObjectException{
```

Parameters:

- **filters** : filters to get CubeObject Build Status
- **UserInfo:** Object of UserInfo class.

Part of Sample Code implementing "Get Build Status"

Set the Cube Object Id that is to be build.

```
//Cube object Id whose build status is to get
String cubeObjId = "SalesCube";
```

Getting the object of Cube and deleting it from Repository

```
Filters filters = new Filters();
Filter filter = new Filter("CUBEOBJECT_ID", "", cubeObjId);
filters.add(filter);
COBuildStatus coBuildStatus =
rom.getCubeObjectBuildStatus(filters, requestorUserInfo);
Response.Write("Build Status = "+coBuildStatus.getStatus()
```

Cancel Build

This .NET API is used cancel the build for which build cube request is sent to Report Engine

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Get the build status for the Cube Object

Method: cancelCubeObjectBuild
For cancelling the cube build process, call method of ReportObjectManager

```
com.impetus.intera.reportobjects.ReportObjectManager

public void cancelCubeObjectBuild(HashMap requestParams,UserInfo
userInfo) throws ReportObjectException{
```

Parameters:

- **requestParams :** Request Params required to cancel Cube Object Build(REQUEST_ID)
- **UserInfo:** Object of UserInfo class.

Part of Sample Code implementing "Cancel Build"

Set the Cube Object Id that is to be build.

```
//Cube object Id whose build status is to get
String cubeObjId = "SalesDataCube";
```

Getting the object of Cube and deleting it from Repository

```
Filters filters = new Filters();
Filter filter = new Filter("CUBEOBJECT_ID", "", cubeObjId);
filters.add(filter);
COBuildStatus coBuildStatus =
rom.getCubeObjectBuildStatus(filters, requestorUserInfo);
COBuildingStatus cbStatus =
(COBuildingStatus)coBuildStatus.getCubeStatus();
buildRequestId = cbStatus.getBuildRequestId();
//if build is not yet completed, then only cancel build process
if(!buildRequestId.isEmpty()){
    requestParams.put(InteraConstants.SysParams.REQUESTID_GUID,
    buildRequestId);
    //method to cancel build process
    rom.cancelCubeObjectBuild(requestParams, requestorUserInfo);
}
```

Database connection Management

Get the list of all the DB connections present in the Intellicus Repository

This .NET API is used to get the list of all the DB connections present in the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations

```
LayoutManager lm = new LayoutManager();
```

4. Get the list of All Database Connections.

Method : getDBConnectionList

```
public java.util.HashMap getDBConnectionList(UserInfo userInfo)
```

This method returns the list of report engine to database connection names.

Parameters:

- **UserInfo:** UserInfo For authorization.

Returns:

HashMap of Database Connection names and Driver details as ArrayList.

```
HashMap map=lm.getDBConnectionList(requestorUserInfo);
```

Create DB Connection in the Intellicus Repository

This .NET API is used to create a new Database Connection in the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations

```
LayoutManager lm = new LayoutManager();
```

4. Set url for DBDriver.

```
DBDriver driverDB = new DBDriver();

//Making an object of hashmap for DBDriver
HashMap hMap = new HashMap();

String server = "192.168.33.52";
String provider = "MSSQL";
String port = "1433";
String database = "M90";
String driverVersion = "2005";

driverDB.setUrl("jdbc:sqlserver://192.168.33.52:1433;databaseName
=M90;selectMethod=cursor;user=sa;password=intellicus");
driverDB.setProvider(provider);

hMap.put("SERVER", server);
hMap.put("PORT", port);
hMap.put("DATABASE",dataBase);
hMap.put("DRIVERVERSION",driverVersion);

driverDB.setAttrHash(hMap);
```

5. Create an instance of DB Connection and set related properties

```
String userId = "sa";
String passwrd = "123456";
String connName = "MyConnection";

//Making an object of DBConnection
DBConnection dbConn = new DBConnection();
dbConn.setDbDriver(driverDB);
dbConn.setUserId(userId);
dbConn.setPassword(passwrd);
dbConn.setConnectionName(connName);
```

6. Add DB Connection to Report Server.

Method : addReportServerConnection

```
public String addReportServerConnection
          (DBConnection dbConnection, UserInfo userInfo)
          throws LayoutHandlerException
```

This method adds a new Report Server Connection in the Repository.

Parameters:

- **DbConnection:** DBConnection object which needs to be added.
- **UserInfo:** The UserInfo object.

Returns:
String: The status of operation returned by the Report Engine

```
lm.addReportServerConnection(dbConn, requestorUserInfo);
```

Create DB Connection for File Data Source

This .NET API is used to create a new Database Connection in the Intellicus Repository for File Type Data Source.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations

```
LayoutManager lm = new LayoutManager();
```

4. Set url for DBDriver.

```
DBDriver driverDB = new DBDriver();

//Making an object of hashmap for DBDriver
HashMap hMap = new HashMap();

driverDB.setUrl("jdbc:FS:////192.168.33.93/FilesForConnection/csv
");
driverDB.setProvider("FILES");
//set driver type
hMap.put("DRIVERTYPE","NETWORK PATH");
driverDB.setAttrHash(hMap);
```

5. Create an instance of DB Connection and set related properties

```
String connName = "FileConnection";

//Making an object of DBConnection
DBConnection dbConn = new DBConnection();
dbConn.setDbDriver(driverDB);
dbConn.setConnectionName(connName);
```

```
//set username
dbConn.setUserId("UserID");
//set password
dbConn.setPassword("pwd123");
//set Initial Connections
dbConn.setInitialConnections("2");
//Set incremental size
dbConn.setIncrementSize("4");
//Set Maximum number of Connections
dbConn.setMaxConnections("8");
//Set true for Read Only Connection or else set false
dbConn.setIsReadOnly(false);
//Set true to set it as Default Connection
dbConn.setIsDefault(false);
//To anable/disable Metadata Cache for the Connection
dbConn.setMetaDataCacheEnabled(true);
```

6. Add DB Connection to Report Server.

Method : addReportServerConnection

```
public String addReportServerConnection
          (DBConnection dbConnection, UserInfo userInfo)
```

This method adds a new Report Server Connection in the Repository.

Parameters:

- **DbConnection:** DBConnection object which needs to be added.
- **UserInfo:** The UserInfo object.

Returns:
String: The status of operation returned by the Report Engine

```
lm.addReportServerConnection(dbConn, requestorUserInfo);
```

Delete DB Connection from the Intellicus Repository

This .NET API is used to delete an existing Database Connection in the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.

3. Create a Layout Manager class object for report layout management related operations

```
LayoutManager lm = new LayoutManager();
```

4. Get the DB Connection list and iterate through each element of list so as to get the given connection i.e. to be deleted

If the connection exists, then delete it.

Method: deleteReportServerConnection

```
public String deleteReportServerConnection
(DBConnection dbConnection, UserInfo userInfo)
```

This method deletes the given report server connection from the repository.

Parameters:

- **DbConnection:** DBConnection object which needs to be deleted.
- **UserInfo:** The UserInfo object.

Returns:

String: The status of operation returned by the Report Engine.

Part of Sample Code implementing "deleteReportServerConnection":

```
map=lm.getDBConnectionList(requestorUserInfo);
Set s=map.keySet();
Iterator itr=s.iterator();
while(itr.hasNext())
{
  ArrayList l=(ArrayList)(map.get(itr.next()));
  //Iterating through the List
  for(int i=0;i<l.size();i++)
  {
        Object o=l.get(i);

        if("class com.intellica.client.common.DBConnection"
        .equalsIgnoreCase(String.valueOf(o.getClass())))
        {
              DBConnection dbConn=(DBConnection)l.get(i);

              // Check whether the Connection exists or not
              if(connectionName.equalsIgnoreCase
                    (dbConn.getConnectionName()))
              {
```

```
            try
            {
            //The method delete given Report Server Connection
            lmanager.deleteReportServerConnection(dbConn,
                                    requestorUserInfo);
            }catch(LayoutHandlerException lhException)
            {
                  lhException.printStackTrace();
            }
            }//end inner if
      }//end Outer if
  }//end for
}//end while
```

Audit Log

Get Audit Detail

This .NET API is used to get the Audit detail information from the Intellicus Repository.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create an Audit Manager controller class object for audit management related operations.

```
AuditManager am= new AuditManager();
```

4. Set the filter for values fields.

```
// prepare parameters for filtering
Date fromDate = new Date(2006-1900 ,2-1,01);
Date toDate = new Date(2006-1900,3-1,9);
String reportName  = "Product";
String username = "Admin";
String reportId = "91FEE269-3AEE-C23D-6F04-7A9979BEBE09";
String categoryId  = "Test";

Filter filter=new Filter();
filter.setFilterField("REPORTNAME",reportName);
filter.setFilterField("CATEGORYID",categoryId);
filter.setFilterField("FROMDATE",fromDate);
filter.setFilterField("TODATE",toDate);
filter.setFilterField("REPORTID",reportId );
```

```
filter.setFilterField("USERID",userName);
```

5. Get the Audit Log details based on filter applied.

Method : getAuditLogList

```
public ArrayList getAuditLogList
       (String fromDate,
        String toDate,
        String reportName,
        String userName,
        String reportId,
        String categoryId,
        UserInfo userInfo)
```

This method will provide the list containing audit log information.

Parameters :

Filter: Filter which can take request filters from Enums.Filters.AuditLog
i.e.
Enums.Filters.AuditLog.REPORTNAME
Enums.Filters.AuditLog.CATEGORYID
Enums.Filters.AuditLog.FROMDATE
Enums.Filters.AuditLog.TODATE
Enums.Filters.AuditLog.REPORTID
Enums.Filters.AuditLog.USERID
userInfo: The User Information Object

Returns:
ArrayList of Audit Log information of various reports.Each element of list is object of class AuditLogData.

```
ArrayList arrayList=am.getAuditLogList(filter,requestorUserInfo);
```

Delete Audit Detail

This .NET API is used to delete Audit Log information.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Audit Manager controller class object for audit management related operations.

```
AuditManager am= new AuditManager();
```

4. Set the filter for values fields that are to be deleted in Audit Details.

```
Date fromDate = new Date(2006-1900 ,2-1,01);
Date toDate  = new Date(2006-1900,3-1,9
String reportName  = "Product";
String username = "Admin";
String categoryId  = "Test";
String reportId    = "91FEE269-3AEE-C23D-6F04-7A9979BEBE09";

Filter filter=new Filter();
filter.setFilterField("REPORTNAME",reportName);
filter.setFilterField("CATEGORYID",categoryId);
filter.setFilterField("FROMDATE",fromDate);
filter.setFilterField("TODATE",toDate);
filter.setFilterField("REPORTID",reportId );
filter.setFilterField("USERID",userName);
```

5. Delete the Audit Log List as per filter

Method : deleteAuditLogList

```
public void deleteAuditLogList(Filter filter, UserInfo userInfo)
```

This method will delete audit log information from the Report Engine.

Parameters:

- **Filter:** Filter which can take request filters from Enums.Filters.AuditLog

i.e.
Enums.Filters.AuditLog.REPORTNAME
Enums.Filters.AuditLog.CATEGORYID
Enums.Filters.AuditLog.FROMDATE
Enums.Filters.AuditLog.TODATE
Enums.Filters.AuditLog.REPORTID
Enums.Filters.AuditLog.USERID
UserInfo: The User Information Object

```
am.deleteAuditLogList(filter,requestorUserInfo);
```

Data Masking

Save Dtaa Masking

This .NET API is used to save the data masking details.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create an instance of SecurityManager.

```
SecurityManager sMgr = SecurityManager.Instance;
```

4. Set the various required details, like Connection name, table name, column name, masking character etc.

```
MaskedData maskedData = new MaskedData();
ArrayList<ConnMaskedDetails>        connList        =        new
ArrayList<ConnMaskedDetails>();
ConnMaskedDetails connMaskedDetailsObj = new ConnMaskedDetails();
connMaskedDetailsObj.setConnectionName("DemoReportDB");
//Connection Name
ArrayList<DBSchema> dbSchemaList = new ArrayList<DBSchema>();
DBSchema dbSchemaObj = new DBSchema();
DBMaskedEntities dbMaskedEntitiesObj = new DBMaskedEntities();
DBMaskedEntity dbMaskedEntityObj = new DBMaskedEntity();
dbMaskedEntityObj.setDBEntityName("BRANCH"); //Table name
dbMaskedEntityObj.setDBEntityType("0");

Column columnObj = new Column();
columnObj.setColumnId("365372309");  //Some  GUID  for  setting
Column Id
columnObj.setColumnName("BRANCH_ID"); //column Name
columnObj.setColumnMaskChar("#"); //Mask character
columnObj.setMaskLevel("1");  //For  Masking  only  on  provided
connection, 0 for Mask for All Connections
columnObj.setMaskType(0); //for masking completely set o, and for
masking partially set 1
columnObj.setColumnOpcode("ADD");//ADD for adding data masking on
a  column,  UPDATE  for  updating  data  masking  on  already  masked
column

ArrayList<ColGrant> colGrantList = new ArrayList<ColGrant>();
ColGrant colGrantObj = new ColGrant();
colGrantObj.setOrgId("HostOrg");  //Organization  for  exceptional
users and roles
colGrantObj.addRoleId("Admin");    //Role
colGrantObj.addUserId("John");     //User
colGrantObj.addUserId("Mary");     //User
colGrantList.add(colGrantObj);
columnObj.setColGrantList(colGrantList);

dbMaskedEntityObj.addColumn(columnObj);
dbMaskedEntitiesObj.addMaskedEntity(dbMaskedEntityObj);
dbSchemaObj.setDbMaskedEntities(dbMaskedEntitiesObj);
```

```
dbSchemaList.add(dbSchemaObj);

connMaskedDetailsObj.setDbSchemaList(dbSchemaList);
connList.add(connMaskedDetailsObj);
maskedData.setConnList(connList);
```

5. Save the data masking details.

Method : saveColsSecurityInfo

```
public void saveColsSecurityInfo(String strCLRSaveList, UserInfo
userInfo)
```

Parameters :

maskedData : Object of {@link com.intellica.client.security.MaskedData} with all the column masking* details filled.
userInfo: userInfo object keeping the detail about the current user

Get Masked Columns

This .NET API is used to get the Masked columns details for specified connection

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create an instance of SecurityManager.

```
SecurityManager sMgr = SecurityManager.Instance;
```

4. Get the masking details.

Method: getColsSecurityInfo

Returns the security information of all columns as ArrayList of ColSecurityInfo class.

```
public MaskedData getColsSecurityInfo(Filter filter, UserInfo
userInfo)
        throws ISecurityException
```

Parameters :

filter : Filter object to get the masked details of given filter creteria only.

- Possible values of filter are {@link com.intellica.client.common.Enums.Filters.ColumnMaskDetails}.

userInfo: userInfo object keeping the detail about the current user

5. Below is the code snippet for iterating through the masked column list-

```
MaskedData maskedData = sMgr.getColsSecurityInfo(filter,
requestorUserInfo);

ArrayList connList = maskedData.getConnList();
//Iterating over the connections to get masked columns for that
connection
for(int i=0;i<connList.size();i++){
  ConnMaskedDetails connMaskedDetailObj =
(ConnMaskedDetails)connList.get(i);
  ArrayList dbSchemaList = connMaskedDetailObj.getDbSchemaList();
  for(int j=0;j<dbSchemaList.size();j++){
        DBSchema dbSchemaObj = (DBSchema)dbSchemaList.get(j);
        DBMaskedEntities dbMaskedEntitiesObj =
dbSchemaObj.getDbMaskedEntities();
        ArrayList dbMaskedEntityList =
dbMaskedEntitiesObj.getMaskedEntityList();
        for(int k=0;k<dbMaskedEntityList.size();k++){
              DBMaskedEntity dbMaskedEntityObj =
(DBMaskedEntity)dbMaskedEntityList.get(k);
              Response.Write("Table Name =
"+dbMaskedEntityObj.getDBEntityName());  //Table name
              Response.Write("----------------------------");
              ArrayList columnList =
dbMaskedEntityObj.getColumnList();
              //Iterating over the masked column list
              for(int
l=0;l<dbMaskedEntityObj.getColumnListSize();l++){
                    Column colObj =
(Column)dbMaskedEntityObj.getColumn(l);
                    Response.Write("Column Name =
"+colObj.getColumnName()+", MaskType = "+colObj.getMaskType()+",
MaskLevel = "+colObj.getMaskLevel()+", MaskChar =
"+colObj.getColumnMaskChar());
              }
  Response.Write("=======================================");
        }
  }
}
```

ReporServerProperty

GetReportEnginePortAndIP

This .NET API is used to get Report Engine's IP Address and Port number of the Report Engine.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a Layout Manager class object for report layout management related operations

```
LayoutManager lm = new LayoutManager();
```

4. Get the IP of Report Engine.

Method : getReportEngineIP

```
public String getReportEngineIP()
```

Get report Engine IP, sets through any of the constructor or if not then returns the default value from ConfigManager class.

Returns:
String: Report Engine IP

```
// method used to get the IP Address of Report Engine
// Returns ReportEngine IP as a String value.
String reportEngineIP=lm.getReportEngineIP();
```

5. Get the port of Report Engine.

Method : getReportEnginePort

```
public int getReportEnginePort()
```

Get report Engine Port, sets through any of the constructor or if not then returns the default value from ConfigManager class.

Returns:
String : report Engine Port

```
  // method used to get the Port of Report Engine
  // Returns report Engine Port as Integer value
  int reportEnginePort=lm.getReportEnginePort();
```

GetReportServerProperty

This .NET API is used to get SMTP Server Values of the Report Engine.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Get the Property values of Report Server.

Method : getReportServerPropValue

```
public String getReportServerPropValue
          (ArrayList reportServerPropList,
           String key,UserInfo userInfo)
```

This method gets the value of Report Server Property from the Report Server for given value of key

Parameters:

- **UserInfo:** The UserInfo object.
- **ReportServerPropList:** ArrayList of all Report Server Properties
- **key:** Name of property whose value is required

Returns:
String representing required property value. If no property is identified with given key then this Method returns null.

Part of Sample Code implementing "getReportServerPropValue" :

```
//This returns arraylist of values of Report Server properties
ArrayList arrList =
sMgr.getReportServerPropDetails(requestorUserInfo);
```

5. Get all the property values of Report Server.

```
//Set the key for the server property.This key is the name     of
the property as specified in the ReportEngine.properties       at
< Intellicus_Install_path >\ReportEngine\Config folder
String key="SMTP SERVER";//Value of Report Server property
```

```
String value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write ("SMTP_SERVER Value : "+value);

key="LISTENER_PORT";
value=sMgr.getReportServerPropValue(arrList ,key,
                          requestorUserInfo);
Response.Write (" LISTENER_PORT Value : "+value);

key="DATABASE_CONNECTION_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                               requestorUserInfo);
Response.Write ("DATABASE_CONNECTION_TIMEOUT Value :
  "+value);

key="SECURITY_FEATURES";
value=sMgr.getReportServerPropValue(arrList ,key,
                               requestorUserInfo);
Response.Write ("SECURITY_FEATURES Value : "+value);

key ="AUDIT_LOG";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write ("AUDIT_LOG Value : "+value);

key = "QUEUE_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write ("QUEUE_SIZE Value : "+value);

key = "REMOTE_SESSION_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                               requestorUserInfo);
Response.Write ("REMOTE_SESSION_TIMEOUT Value : "+value);

key = "RTF_FIELD_CONTROL_MAP";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write ("RTF_FIELD_CONTROL_MAP Value : "+value);

key = "DATA_SOURCE_FETCH_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                    requestorUserInfo);
Response.Write ("DATA_SOURCE_FETCH_SIZE Value : "+value);
```

```
key = "AUDITLOG_PURGE_TIME";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write ("AUDITLOG_PURGE_TIME Value : "+value);

key = "CACHE_PURGE_TIME";
value=sMgr.getReportServerPropValue(arrList ,key,
                                requestorUserInfo);
Response.Write ("CACHE_PURGE_TIME Value : "+value);

key = "AUTHORIZATION_CACHE_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                                requestorUserInfo);
Response.Write("AUTHORIZATION_CACHE_TIMEOUT Value :
                                                  "+value);
key = "SCHD_JOB_DISPATCH_QUEUE_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Response.Write("SCHD_JOB_DISPATCH_QUEUE_SIZE Value :
                                    "+value);
```

SetReportServerProperty

This .NET API is used to set the SMTP Server Values of Report Engine.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a SecurityManager class object for getting the controller information for all Administration related operations

```
SecurityManager sMgr=SecurityManager.Instance;
```

4. Set the Property values of Report Server by putting them in a Hashmap.

```
HashMap ServerPropHMap= new HashMap();

  String key="SMTP SERVER";
  String value="192.168.100.20";
  ServerPropHMap.put(key,value);

  key="DATABASE CONNECTION TIMEOUT";
  value="800";
```

```
ServerPropHMap.put(key,value);

key="SECURITY_FEATURES";
value="enabled";
ServerPropHMap.put(key,value);

key="QUEUE_SIZE";
value="900";
ServerPropHMap.put(key,value);

key="LOG";
value="../logs";
ServerPropHMap.put(key,value);

key="PAGE_CHUNKSIZE";
value="10";
ServerPropHMap.put(key,value);

key="LOG_LEVEL";
value="INFO";
ServerPropHMap.put(key,value);

key="USER_THREADS";
value="5";
ServerPropHMap.put(key,value);

key="REPOSITORY_CACHE_TIMEOUT";
value="15";
ServerPropHMap.put(key,value);

key="EXIT_ON_ERROR";
value="disable";
ServerPropHMap.put(key,value);
```

5. Save these property values set in last step for Report Server

Method : saveReportServerProperties

```
public String saveReportServerProperties
      (java.util.HashMap reportServerPropMap,
       UserInfo userInfo)
```

This method saves the modified server properties and returns the status sent by the Report Server.

Parameters:

- **ReportServerPropMap:** HashMap which keeps the property name as key and its its concerning value as the value.
- **UserInfo:** The UserInfo object.

Returns:

String: Having the status return by the Report Engine after modifying the property file.

Part of Sample Code implementing "setReportServerPropValue" :

```
sMgr.saveReportServerProperties(ServerPropHMap,
                                        requestorUserInfo);
```

ReporServerConnectivity

TestServerConnectivity

This .NET API is used to check whether the Report Server is running on the specified IP and Port or not.

Steps:

1. Initialize Report Client.
2. Create a Security controller class object using its factory for user management related operations

```
SecurityManager sMgr=SecurityManager.Instance;
```

3. Get the Security Mode of Report Engine.

Method : getSecurityMode

```
public bool getSecurityMode()
```

This method returns the security mode read from the report engine.

Returns:
If the security mode is on on the report engine, it returns true. In other cases, it returns false.

```
bool securityMode  = false;

//If it gives Exception, then failed to connect Report Server
securityMode=sMgr.getSecurityMode();
```

User Preferences

GetUserPreferences

This .NET API is used to get the default User Preferences set by the User.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a PersonalizationManager class object for UserPrefernces and Dashboard related settings.

```
PersonalizationManager pm=new PersonalizationManager();
```

4. Get the User Preferences in the instance of Preferences.

Method: getUserPreferences

```
public Preferences getUserPreferences
       (UserInfo userInfo)
```

This method returns Preferences for the given user.

Parameters:

- **UserInfo:** The Information about the user to be used for authorization. Also the user whose Preferences will be returned.

Returns:
Preferences for the user.

```
Preferences userPref = null;
userPref=  pm.getUserPreferences(requestorUserInfo);
```

5. Get all the User Preferences

```
//Returns the defaultConnection
String defaultCon = userPref.getDefaultConnection();

//Returns the user's preferred default portal theme
String  portalTheme = userPref.getDefaultPortalTheme();

//Method returns default stylesheet of the user
```

```
 String defaultStylesheet = userPref.getDefaultStylesheet();

 //Method returns the user's preferred deliver location
 String deliveryLocation = userPref.getDeliveryLocation();

 //Method returns the user's preferred report format
 String defaultFormat = userPref.getFormat();

 //Method returns the user's preferred language
 String prefLanguage = userPref.getPrefLanguage();

//Method returns the number of recent reports to be shown
int reportCount = userPref.getReportCount();

//Method returns the Bool value whether to show the inbox
Bool showInbox = userPref.getShowInbox();

//This Method returns default template name used for all
   reports
String templateName = userPref.getTemplateName();

// Method returns the Bool value whether to show the recent
   reports or not
Bool showRecentReports = userPref.getShowRecentReports();

Response.Write("Default Connection  :  "+defaultCon);
Response.Write("Default Portal Theme  :  "+portalTheme);
Response.Write("Default Stylesheet:"+defaultStylesheet);
Response.Write("Delivery Location  :  "+deliveryLocation);
Response.Write("Default Format  :  "+defaultFormat);
Response.Write("Default Preferred Language  :
   "+prefLanguage);
Response.Write("Report Count  :  "+reportCount);
Response.Write("Show Inbox  :  "+showInbox);
Response.Write("Template Name  :  "+templateName);
Response.Write("Show Recent Reports  :
        "+showRecentReports);
```

setUserPreferences

This .NET API is used to set the default User Preferences by the User.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a PersonalizationManager class object for UserPrefernces and Dashboard related settings.

```
PersonalizationManager pm=new PersonalizationManager();
```

4. Get the User Preferences in the instance of Preferences.

```
Preferences pref = new Preferences();
pref=  pmanager.getUserPreferences(requestorUserInfo);
```

5. Set the new value for User Preferences.

```
int reportCount = 12;
String templateName = "Beach";
String email = "hostUser@hostOrg.com";
String defConnection = "MyConnection";

pref.setReportCount(reportCount);
pref.setTemplateName(templateName);
pref.setShowRecentReports(true);
pref.setDefaultConnection(defConnection);
pref.setDefaultPortalTheme("Default");
pref.setEmail(email);
pref.setFormat(InteraConstants.ReportFormats.PDF);
pref.setPrefLanguage("EN_US");
```

6. Update these User Preferences

Method: updateUserPreferences

```
public Preferences updateUserPreferences
       (Preferences userPref,
        UserInfo userInfo)
```

This method updates the user preferences to the repository.

Parameters:

- **UserPref:** The user preferences prefernces details
- **UserInfo:** userInfo object for authorization.

Returns:
Preferences

```
//Method used to update the user preferences to the repository
pmanager.updateUserPreferences(pref, requestorUserInfo);
```