Intellicus DotNet Client Developer's Guide

Intellicus Enterprise Reporting and BI Platform Release 5.X

info@intellicus.com
www.intellicus.com

Dated: - September 2011.

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

HTTP API

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

- Getting category and report list
- Run, Schedule, Publish, E-Mail report
- Adhoc reporting

General steps of integration are:

- Store User Info in session
- Call HTTP API for the needed functionality

Using HTTP-APIs in Developer Environment

Prerequisites

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

Store User Info 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.

```
//Import 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["Intell
icus Relative Path"];

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

Important: Above sample code uses DotNet Client APIs. Please refer DotNet API section for more details.

CategoryList.aspx

The objective of the API is to fetch 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. 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=4F
9245A7-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, scheduling, view saved reports and view its description.

This API governs following behaviors:

- If the user is Super Admin or Admin then user 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' right is needed.

CATEGORY_ID
(Mandatory)

This is the id of the category in which the report being run exists. This API lists 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 are available in a specified category. Possible values for this parameter are STANDARD/STUDIO/ADHOC.

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

Example: REPORT_TYPE=STANDARD

SEARCH_STRING
(Optional)

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

Example: SEARCH_STRING=Country Sales

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

FROM_DATE (Optional)

This parameter gives list of all reports which are updated from specified date.

Example: FROM_DATE = 06/06/2009

This API lists all reports updated from 06/06/2009 date.

TO_DATE
(Optional)

This parameter gives list of all reports updated up to specified date.

Example: TO_DATE = 06/06/2009

This API will list of all reports which updated up to 06/06/2009 date.

If FROM_DATE and TO_DATE, both parameters are given in HTTP API then it will fetch list of all reports which are 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 (SHOW_RERUN=true) 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 (SHOW_DELETE=true) 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
```

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.

This API is located at <Install_path>intellicus.

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 aspx 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 description (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.
002	Executes report and navigates to report viewer screen.
003	Request to show all the system, report parameter in one aspx.
300	Executing Reports in Asynchronous mode.
700	Delete saved report.
900	Promptable information check and execution of dynamic adhoc report with no parameter.

Example: ACTION_CODE=002

REPORT_ID
(Mandatory)

This is the Id of the report being run.

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, are designed in Intellicus Studio (these reports are also known as Standard Reports).
- **ADHOC:** Specify this value when report being run, are 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.
Example: CATEGORY_ID=4F9245A7-D639-4F99-604D-F32641B77725

REPORT_FORMAT
(Optional)

This is the output format in which report can be viewed. Possible values for this parameter are **PDF, HTM, RDF, XLS, CSV, DHTM, DOC, TXT**

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, PRINT_LOCALLY.

- **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 or 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 or 2 for SHOW_NEVER, SHOW_ALWAYS, and SHOW_WHEN_MULTIPAGE respectively.

- **SHOW_NEVER:** It 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 gives the list of all reports updated from specified date.

Example: FROMDATE = 06/06/2009

This API will list of all reports updated from 06/06/2009 date.

TODATE
(Optional)

This parameter gives list of all reports updated up to specified date.

Example: TODATE = 06/06/2008

This API lists 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. FROMCATEGORY 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 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'

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&REPORT_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&DSG
N_MODE=STUDIO&REPORT_ID=2030DDEA-841B-E360-3CA0-
5954AA945B92&REPORT_FORMAT=htm
```

For running Adhoc report

```
/InteraController.aspx?ACTION_CODE=002&OPERATION_TYPE=VIEW&DSG
N_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
```

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 = 06/06/2009

This API lists all reports which updated from 06/06/2009 date.

TO_DATE
(Optional)

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

Example: TO_DATE = 06/06/2009

This API lists all reports updated up to 06/06/2009 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
```

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 sent 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

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&REPORT_ID=6FA78172-C4D0-9A6D-9CDF-
4F70161803D1
```

Editing an existing report

```
String onSuccess=/custom/
AdHocWizard.aspx?DEFAULTEXPANDTABS=All&CATEGORY_ID=4F9245A7-
D639-4F99-604D-F32641B77725
```

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
```

AccessRight.aspx

This API provides UI to manage users' access rights for a report (design, view, schedule, etc). The user needs to have appropriate access rights on the report. Access is given to users as well as roles. If access right is given to a role, all the users having that role will automatically get that access rights of the role.

With this API following behaviours are governed:

- Set Access Rights at Category level.
- Set Access Rights at Reports level.

Available option to give access right:-

- ***Full Access***: Complete Access Rights on all the reports under the selected category.
- ***Deny Access***: No Access Rights to any of the reports under the selected category.
- ***Partial Access***: Selective Access Rights for all the reports under the selected category.

User Credentials

USERINFO_USERID
(Mandatory)

This parameter is used for specifying user id of the user requesting the operation. This user id should exist in Intellicus also.

Example: USERINFO_USERID=Admin

USERINFO_PWD
(Mandatory)

This parameter is used for specifying password of the user requesting the operation. Password is not required for the Organization, which has set authentication mode as "Host Application".

Example: USERINFO_PWD=Admin

USERINFO_ORGID
(Mandatory)

This parameter is used for specifying Intellicus organization id to which user requesting the operation belongs.

Example: USERINFO_ORGID=Intellica

UserManagement.aspx

This API provides UI to set User related activities:

User related activities

- Adding a new user.
- Change settings for a user.
- Delete a user.
- Suspend a user.

Role related activities

- Setting up a new role.
- Change settings for a role.
- Delete a role.
- Suspend a role

User Credentials

USERINFO_USERID
(Mandatory)

This parameter is used for specifying user id of the user requesting the operation. This user id should exist in Intellicus also.

Example: USERINFO_USERID=Admin

USERINFO_PWD
(Mandatory)

This parameter is used for specifying password of the user requesting the operation. Password is not required for the Organization, which has set authentication mode as "Host Application".

Example: USERINFO_PWD=Admin

USERINFO_ORGID
(Mandatory)

This parameter is used for specifying Intellicus organization id to which user requesting the operation belongs.

Example: USERINFO_ORGID=Intellica

SecurityMapping.aspx

This API provides UI to work with mapping setup for an organization:

- Add a mapping.
- Delete a mapping.

User Credentials

USERINFO_USERID
(Mandatory)

This parameter is used for specifying user id of the user requesting the operation. This user id should exist in Intellicus also.

Example: USERINFO_USERID=Admin

USERINFO_PWD
(Mandatory)

This parameter is used for specifying password of the user requesting the operation. Password is not required for the Organization, which has set authentication mode as "Host Application".

Example: USERINFO_PWD=Admin

USERINFO_ORGID
(Mandatory)

This parameter is used for specifying Intellicus organization id to which user requesting the operation belongs.

Example: USERINFO_ORGID=Intellica

ReportBundleDeployer.aspx

This API provides UI to deploy bundle / package (a cab file). On this screen, we can perform following operations:

- Uploading a cab file.
- Deploying a cab file.

User Credentials

USERINFO_USERID
(Mandatory)

This parameter is used for specifying user id of the user requesting the operation. This user id should exist in Intellicus also.

Example: USERINFO_USERID=Admin

USERINFO_PWD
(Mandatory)

This parameter is used for specifying password of the user requesting the operation. Password is not required for the Organization, which has set authentication mode as "Host Application".

Example: USERINFO_PWD=Admin

USERINFO_ORGID
(Mandatory)

This parameter is used for specifying Intellicus organization id to which user requesting the operation belongs.

Example: USERINFO_ORGID=Intellica

Org.aspx

This API provides UI to set Organization related information:

- Filter list of Organization - List of organization can be filtered by Starting character (Starts with) and Characters that appear anywhere in the organization name (Contains).
- Adding a new organization.
- Change settings for the organization.
- Delete the organization.

User Credentials

USERINFO_USERID
(Mandatory)

This parameter is used for specifying user id of the user requesting the operation. This user id should exist in Intellicus also.

Example: USERINFO_USERID=Admin

USERINFO_PWD
(Mandatory)

This parameter is used for specifying password of the user requesting the operation. Password is not required for the Organization, which has set authentication mode as "Host Application".

Example: USERINFO_PWD=Admin

USERINFO_ORGID
(Mandatory)

This parameter is used for specifying Intellicus organization id to which user requesting the operation belongs.

Example: USERINFO_ORGID=Intellica

QuickJob.aspx

This API is used to add new job.

OPCODE
(Mandatory)

This parameter is used for specifying operation code.
For adding a new job its value should be Add.
Example:

```
core/QuickJob.aspx?OPCODE=ADD
```

Sample URLs (HTTP API's)

InteraController.aspx

To get system parameters page for Standard Report

```
intellicus/InteraController.aspx?ACTION_CODE=000&REPORT_ID=4B22DE
7A-90F6-E193-3B82-CD8574D9BD98
```

To get Input parameters page

```
intellicus/InteraController.aspx?ACTION_CODE=001&REPORT_ID=4B22DE
7A-90F6-E193-3B82-CD8574D9BD98
```

To run Adhoc report

```
intellicus/InteraController.aspx?ACTION_CODE=002&OPERATION_TYPE=V
IEW&DSGN_MODE=ADHOC&REPORT_ID=6D227356-C327-E28F-4556-
967B02B0A909&REPORT_FORMAT=htm
```

ReportListForCategory.aspx

For getting list of all the reports in the category.

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

CategoryList.aspx

For getting list of all the categories.

```
intellicus/core/CategoryList.aspx?
```

AdhocWizard.aspx

For editing existing Adhoc Report

```
intellicus/custom/AdHocWizard.aspx?REPORT_ID=E85A9025-6AB9-4EDB-
0DE3-E0533FAFAB04&CATEGORY_ID=4F9245A7-D639-4F99-604D-
F32641B77725
```

SavedReportList.aspx

For getting list of all saved reports for the specified report Id.

```
intellicus/core/SavedReportList.aspx?REPORT_ID=2030DDEA-841B-
E360-3CA0-5954AA945B92
```

Preferences.aspx

For working with password, default data connection and portal preferences.

```
intellicus/personalization/Preferences.aspx?
```

AccessRight.aspx

For providing access rights to user/role on category and reports.

```
intellicus/security/AccessRight.aspx
```

UserManagement.aspx

For performing user/role related activities.

```
intellicus/security/UserManagement.aspx?
```

SecurityMapping.aspx

For performing mapping setup for an organization.

```
intellicus/security/SecurityMapping.aspx?
```

ReportBundleDeployer.aspx

To deploy bundle / package (a cab file).

```
intellicus/security/ReportBundleDeployer.aspx
```

QuickJob.aspx

To add a new job.

```
intellicus/core/QuickJob.aspx?OPCODE=ADD
```

Org.aspx

For performing organization related activities.

```
intellicus/security/Org.aspx
```

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
- ACTION
- SWAPAXES
- CLEAR
- 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

DASHBOARD

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

- Dashboard Viewer API
- Widget Designer API
- Dashboard Preferences

List of Parameters: The parameters used here will be divided into two:

- For Dashboard ViewerAPI

Dashboard Viewer

This API takes below parameter.

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

Above 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=B954111B-
1881-21CE-1958-0DB411187785
```

SHOW_FULL_SCREEN: This option is to show full screen option or not. Show the icon, if not false.

Above 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
```

Widget Designer

This API takes one parameter.

WIDGET **_ID:** The unique identifier of the Widget that should be loaded for editing.

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

For example:

```
String onSuccess =
/dashboard/WidgetDesigner.aspx?WIDGET_ID=126284599595681921683361
32613160
```

Dashboard Preferences

Below URL invokes the Dashboard Preferences page of Intellicus.

For example:

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

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
```

DotNet API

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

To use the DotNet APIs of Intellicus, reference of following dlls should be added in host application.

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

These dlls are provided with Intellicus DotNetClient setup. Following is given the path for dll files:

<Intellicus DotNet Client Install path>\bin

Mandatory Step to Use DotNet APIs

Initialize Report Client

Configuration

Intellicus client DotNet APIs is configured by a file "ReportClient.properties".

This file is provided with Intellicus DotNetClient setup and after installation available at the following location:

<Intellicus DotNet Client Install path>\client\config

In case, if host application is not integrated with Intellicus DotNetClient so this file should be copied in the host application.

During, initialization action the report client component reads the configuration file and keeps the configuration in memory.

Importing namespaces

Host DotNet application class has to import the following classes, to initialize the Report Client.

```
using ReportClient = com.intellica.client.init.ReportClient;
```

The initialization action is to be performed once in the lifetime of application.The initialization would be done according to the configurations set in the ReportClient.properties file.

Init

```
System.IO.Stream is_Renamed = new
System.IO.FileStream("<ReportClient.properties_AbsolutePath>",
System.IO.FileMode.Open, System.IO.FileAccess.Read);
ReportClient.init(is_Renamed);
```

Other constructers are also available to initialize the Report Client.

Init using IP and Port

```
String reportServerIP = "127.0.0.1";
int reportServerPort = 50015;

ReportClient.init(reportServerIP, reportServerPort);
```

Init using HTTP Communication

```
String httpConnectionURL = "http://localhost/intellicus";
ReportClient.initForURL(httpConnectionURL);
```

Initialize Requestor User context

Requestor User

A Requestor User is the user, who is requesting any action to the Intellicus system.

A com.intellica.client.common.UserInfo class object represents a user in Intellicus.

The UserInfo class has the following attributes.

Attribute	Description
userId	User ID
password	Password
orgID	Organization ID
sessionId	Session ID
securityDescriptor	Security Descriptor string
customerId	Customer ID for service provide deployments
location	Location ID
locale	Locale
dbName	Database Name

Intellicus mode of authentication uses **User ID**, **Password** and **Org ID** as mandatory attributes to authenticate a user and authorize that user's actions.

A host application that takes over authentication responsibilities can use any of the above attributes to authenticate.

Importing namespaces

DotNet Application class has to import the following class to store the login user information and check for Authorization.

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

UserInfo

The UserInfo object acts as a carrier to all above attributes in such case.

So, in all the use cases discussed below, host application has to create an object of UserInfo class and pass it in all the method calls.

```
UserInfo requestorUserInfo = new
        UserInfo("John","john","Org1");
```

Note: For performing any admin activity like user management at Intellicus, requestor user should be admin user at Intellicus.

Use Cases

User Management Actions

Import

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

//Organization specific imports
using OrgInfo = com.intellica.client.security.OrgInfo;

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

Organization

GetOrgById

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

Refer to SampleCodes/DotNetAPI/Organization Management/ GetOrgById.cs 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 Organization by OrgId and get its various details.

Method: getOrganizationById

```
public OrgInfo 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
System.String organisationId = "TestOrg";
            OrgInfo                    orgObj                    =
sMgr.getOrganizationById(organisationId, requesterUserInfo);
Logger.debug("OrganizationName: " + orgObj.OrgId);
Logger.debug("OrganizationDescription:             "             +
orgObj.OrgDescription);
Logger.debug("OrganizationMapType: " + orgObj.MapType);
Logger.debug("OrganizationAuthorization      mode:       "       +
orgObj.AuthorizeMode);
Logger.debug("OrganizationAuthenticationObject:         "        +
orgObj.AuthenticationObj);
```

Get Users list for Organization

This DotNet 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.

Refer to SampleCodes/DotNetAPI/Organization Management /GetUserListForOrg.cs 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 Arraylist containing all users in the given Organization

Method: getUserListForOrganization

```
Public ArrayList getUserListForOrganization(string orgId,
UserInfo 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 objects

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

Get Organization List

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

Refer to SampleCodes/DotNetAPI/Organization Management/GetOrgList.cs 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 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 DotNet API is used to add a new organization in the Intellicus Repository and set the authentication mode.

Refer to SampleCodes/DotNetAPI/Organization Management/AddOrg.cs for sample code of this use case.

Steps:

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

```
System.String organisationId = "TestOrg";
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.Se
curityTypes.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 orgInfo,
                            UserInfo 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 DotNet API is used to modify the Organization's details.

Refer to SampleCodes/DotNetAPI/Organization Management / ModifyOrg.cs for sample code of this use case.

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.CanDelete = true;
String description="THIS IS A TEST ORGANIZATION";
modOrg.OrgDescription = 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 orgInfo,
                               UserInfo 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 DotNet API is used to delete an existing organization from the Intellicus repository.

Refer to SampleCodes/DotNetAPI/Organization Management /DeleteOrg.cs for sample code of this use case.

Steps:

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

```
String organizationId="TestOrg";
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 orgInfo,
                               UserInfo 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);
```

User

Get User By Id

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

Refer to SampleCodes/DotNetAPI/User Management/GetUserById.cs 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 UserInfo 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 DotNet API is used to create a new User in Intellicus Repository.

Refer to SampleCodes/DotNetAPI/User Management/AddUser.cs 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 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.Admin = true;

// Optionally, make the user Super admin.
addUserInfo.SuperAdmin = true;
```

5. Add the User in Intellicus Repository.

Method: addUser

```
public void addUser(UserInfo newUserInfo,
                    UserInfo 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 the User

This DotNet API is used to assign the Access privileges of a Category to the User.

Refer to SampleCodes/DotNetAPI/User Management/ AssignCategoryPrivilegesToUser.cs 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 to which Category Rights are to be assigned.

```
String userId = "Mary";
String orgId = "MyOrg";
String password = "abcd";
UserInfo adminUserInfo = new UserInfo(userId , password , orgId);
```

5. Create EntityInfo and AppInfo

```
// UserId of the user whose Category Privileges are to be set
  String userId="testUser";
  String OrgId="hostOrg";
//Category Id on which AccessRights will be given to the user;
  String categoryId="B2996F87-FE41-8D19-6C67-B3C22803DD99";

  EntityInfo entityInfo=new EntityInfo(categoryId,"CATEGORY");
  AppInfo appInfo = new AppInfo(userId,OrgId,"USER");
```

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

```
EntityAccessRight ear=new EntityAccessRight();

ear.AppInfo = appInfo;
ear.AccessLevel = Enums.SecurityTypes.AccessLevel.FULLACCESS;
```

7. 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 User

This DotNet 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.

Refer to SampleCodes/DotNetAPI/User Management/ AssignReportPrivilegesToUser.cs 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 to which Report Rights are to be assigned.

```
String userId = "Mary";
String password = "";
String orgId = "MyOrg";
UserInfo UserInfo=new UserInfo(userId , password, orgId);
```

5. Create EntityInfo and AppInfo

```
String reportId="638CB7A6-F504-0ECD-008F-10B25253DCA8";

EntityInfo entityInfo=new EntityInfo(reportId,"REPORT");
AppInfo appInfo = new AppInfo(userId,userOrganisation,"USER");
```

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

```
EntityAccessRight ear=new EntityAccessRight();
ear.AccessLevel = Enums.SecurityTypes.AccessLevel.FULLACCESS;
```

7. 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 System Privileges to the User

This DotNet 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.

Refer to SampleCodes/DotNetAPI/User Management/ AssignSystemPrivilegesToUser.cs 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="FinanceUser";
String organization=="MyOrg";
UserInfo userInfo = sMgr.getUserById(userId, organization,
requestorUserInfo);
```

5. Assign System Privileges to assigneeUser.

Property : SystemPrivileges

```
SystemPrivileges = <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

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

```
userInfo.SystemPrivileges =
com.intellica.client.utils.Utility.getRightsFromMaskedValue(com.i
ntellica.client.common.Enums.SecurityTypes.SystemPrivileges.CATEG
ORY_SETUP |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
SCHEDULER |
com.intellica.client.common.Enums.SecurityTypes.SystemPrivileges.
REPORT_DESIGNER);

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

Assign Role To User

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

Refer to SampleCodes/DotNetAPI/User Management/AssignRoleToUser.cs 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(UserInfo targetUInfo,
                            RoleInfo rInfo,
                            UserInfo 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 DotNet API is used to assign multiple roles to the application user.

Refer to SampleCodes/DotNetAPI/User Management/ AssignRolesToUser.cs 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(UserInfo targetUInfo,
                              ArrayList roleInfos,
                              UserInfo 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 DotNet API is to revoke specified role from user

Refer to SampleCodes/ DotNetAPI/User Management/RevokeRoleFromUser.cs 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 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)
                        throws ISecurityException
```

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

Refer to SampleCodes/DotNetAPI/User Management/UserMapping.cs 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 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 DotNet API is used to delete an existing user with given userId from the Intellicus Repository.

Refer to SampleCodes/DotNetAPI/User Management/DeleteUser.cs 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 DotNet API is used to get the list of all Roles present in Intellicus Repository.

Refer to SampleCodes/Java APIs/Role Management/ GetRoleList.cs 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 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 obj

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

Create Role (Add Role)

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

Refer to SampleCodes/DotNetAPI/Role Management/AddRole.cs 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 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 DotNet API is used to assign the Access privileges of a Category to the existing Role.

Refer to SampleCodes/DotNetAPI/Role Management/ AssignCategoryPrivilegesToRole.cs 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 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();
ear.AccessLevel = 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 DotNet API is used to assign the Access privileges for a Report to the Role.

Refer to SampleCodes/ DotNetAPI /Role Management/ AssignReportPrivilegesToRole.cs 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 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();
ear.AccessLevel = 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 DotNet API is used to delete a particular Role of an existing Organization in Intellicus Repository.

Refer to SampleCodes/DotNetAPI/Role Management/ DeleteRole.cs 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.getInstance();
```

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

Import

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

Get Category List

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

Refer to SampleCodes/DotNetAPI/Category Management/ GetCategoryList.cs 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. Get the list of all Categories in Repository.

Method: **getCategoryList**

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

This method returns the ArrayList 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 IsPublic Boolean property to identify the scope of category.

Parameters:

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

Returns:

ArrayList of Category objects.

```
ArrayList CatList= new ArrayList();
CatList=lm.getCategoryList(requestorUserInfo);
```

The other two APIs for getting Category List are-

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

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

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

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

All these returns ArrayList containing Category objects.

Add a new Category

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

Refer to SampleCodes/DotNetAPI/Category Management/ CreateNewCategory.cs 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. 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.IsPublic = true;
newCategory.IsHidden = false;
newCategory.Description = catDescription;
//To set the category id
newCategory.CategoryId = catId;
newCategory.MenuName = 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 DotNet 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 SampleCodes/DotNetAPI/Category Management/DeleteCategory.cs 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 DotNet API is used to delete an Existing Category from the Intellicus repository along with the Reports present in that Category. User needs to provide the Category ID of the Category which is to be deleted.

Refer to SampleCodes/DotNetAPI/Category Management/DeleteCategory.cs 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. 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.

Property:SubCategories

Returns OrderedDictionary having collection of categorie object

```
//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);
```

```
OrderedDictionary subcategories = categoryObj.SubCategories;
```

Report Layout Management

Import

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

GetAllReportList

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

Refer to SampleCodes/DotNetAPI/Report Layout Management/ GetAllReportList.cs 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. Get the Report List

Method : getReportList

```
public ArrayList getReportList(UserInfo userInfo)
```

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

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

```
// ArrayList that will be used to store the Report List
ArrayList reportList= new ArrayList();
```

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

The other APIs for getting Report List are-

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

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

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

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

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

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

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

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

MoveReport

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

Refer to SampleCodes/DotNetAPI/Report Layout Management/ MoveReport.cs 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. 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 the report to a 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 DotNet API is used to get the complete list of Reports in given Category.

Refer to SampleCodes/DotNetAPI/Report Layout Management/ GetReportListForCategory.cs 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. Pass the Category Id whose Reports are to be listed.

Method : getReportListForCategory

```
public ArrayList getReportListForCategory
       (string categoryId,
        UserInfo requestorUserInfo)
```

Method returns the ArrayList 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 IsPublic bool property 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 EmptyInstance in case of report server runs in Security-Off mode.

Returns:
ArrayList of Reports.

```
arrayList catList= new ArrayList();
String categoryId = "01-Sales";
catList=lm.getReportListForCategory
                 (categoryId , requestorUserInfo);
```

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

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

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

2. Returns a ArrayList 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 ArrayList getReportListForCategoryWithAccessRights(String
categoryId,int   categoryAccessRight,   int   reportAccessRight,
UserInfo userInfo)
```

3. Returns a vector 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 ArrayList 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 ArrayList getReportListForCategoryWithReportRight(String
categoryId, int reportAccessRight,UserInfo userInfo)
```

Get Saved Report List

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

Refer to SampleCodes/DotNetAPI/Report Layout Management/GetSavedReportList.cs 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. Pass the Category Id/Report Id whose Saved Reports are to be listed.

Method : getSavedReportList

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

Method returns the ArrayList 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:

ArrayList 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);
```

Add Report Layout to Category

This DotNet 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.

Refer to SampleCodes/DotNetAPI/Report Layout Management/ AddReportLayoutToCategory.cs 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. 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";
System.IO.FileInfo file = new System.IO.FileInfo(f);

//Read the contents from the given file
System.IO.Stream            is_Renamed            =            new
System.IO.FileStream(file.FullName,       System.IO.FileMode.Open,
System.IO.FileAccess.Read);
// 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.CategoryId = 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 DotNet 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.

Refer to SampleCodes/DotNetAPI/Report Layout Management/ CopyReport.cs 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. Set the attributes for copied report.

```
Report r=new Report();
String rep_Name = "Copied_Report";
r.MenuName = 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 DotNet API is used to get Report Details corresponding to the given Reoprt Id from the Intellicus Repository.

Refer to SampleCodes/DotNetAPI/Report Layout Management/ GetReportDetails.cs 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. 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 void **updateReport**(Report oldReport,Report newReport,UserInfo userInfo)

2. public void **updateReport**(ArrayList reportVector, UserInfo userInfo)
3. public void **addReport**(Report report)

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 objects are stored as XML in Intellicus repository.

Query objects can be added, deleted and updated using DotNet API.

This document intends to explain the DotNet 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 Java APIs.

Information	Type	Description
Name	Mandatory	Must be Unique
SQL Statement	Mandatory	The SQL Statement which sources data for this QO.
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, Boolean	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 DotNet API is used for creating a new Report Object.

Refer to SampleCodes/DotNetAPI/ReportObjects/ AddReportObject.cs for sample code of this use case.

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.SqlEditor;
  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 System.Collections.ArrayList ColumnDetails
```

value:

- **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.Hidden = true;

String groupCaption = "Job";
//To set this column in a group
qoc1.GroupLabelCaption = groupCaption;

// To set hyperlink for this column.
String hyperLink = "http://www.google.com";
qoc1.Hyperlink=hyperLink;
```

7. Set the connection name- ConnectionName property of com.impetus.intera.reportobjects.QueryObject class can be used to set the name of the connection used by Query Object

```
public System.String ConnectionName
```

value:

- **ConnectionName:** The database connection Name to be used for query object.

5. Set if filter is mandatory-This property is used to set the isFilterMandatory.

```
public bool FilterMandatory of QueryObjects class.
```

value:

IsFilterMandatory: Whether mandatory filters are applied or not.

6. Set the group caption.

```
public bool GroupCaption
```

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.Name = qoName;
qoObj.Id = queryId;
qoObj.Cached = true;
qoObj.ConnectionType = "Default";
qoObj.ConnectionName = connName;
qoObj.Source = "SQL";

reportObjectManager.addReportObject(qoObj, requestorUserInfo);
```

Get Query Object by Name

This DotNet 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 SampleCodes/DotNetAPI/ReportObjects/GetReportObjectByName.cs 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 ColumnDetails property on the DotNet object of query object returned. This property will return array list of all the columns.

Method

```
public ArrayList ColumnDetails
```

Returns:

Returns the array list of columnDetails.

5. Call ConnectionName property on the DotNet object of query object returned. This property is used to get the name of the connection used by Query Object

```
public string ConnectionName
```

Returns:

Returns the name of the connection.

Call isFilterMandatory property on the DotNet object of query object returned.

```
public boolean 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 DotNet API is used to retrieve the list of Report Objects from Intellicus repository.

Refer to SampleCodes/ DotNetAPI/ReportObjects/ GetReportObjectList.cs for sample code of this use case.

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, boolean cached,
UserInfo userInfo)
```

```
public ArrayList getReportObjects(String roType, UserInfo
userInfo)
```

Get Parameter Object List

This DotNet API is used to retrieve the list of Query Objects from Intellicus repository.

Refer to SampleCodes/DotNetAPI/ReportObjects/GetReportObjectList.cs for sample code of this use case.

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.Type,requestorUserInfo);
```

Delete Report Object

This DotNet 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".

Refer to SampleCodes/DotNetAPI/ReportObjects/DeleteReportObject.cs for sample code of this use case.

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:** DotNet 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.Alignment = 1;
qocAdd.Hidden = false;
qocAdd.GroupLabelCaption = grpLabelCaption;

//add this column in the query object
qObject.addColumn(qocAdd);

reportObjectManager.replaceReportObject(qObject,requestorUserInfo
);
```

Other related Property for accessing QO/PO are:

1. public OrderedDictionary AllPOMap
2. public OrderedDictionary AllQOMap

Database connection Management

Get the list of all the DB connections present in the Intellicus Repository

This DotNet API is used to get the list of all the DB connections present in the Intellicus Repository.

Refer to SampleCodes/DotNetAPI/DBConnection Mangement/GetDBConnectionList.cs 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. Get the list of All Database Connections.

Method : getDBConnectionList

```
public Hashtable getDBConnectionList(UserInfo userInfo)
```

This method returns the list of report engine to database connection names.

Parameters:

- **UserInfo:** UserInfo For authorization.

Returns:

Hashtable of Database Connection names and Driver details as ArrayList.

```
Hashtable map=lm.getDBConnectionList(requestorUserInfo);
```

Create DB Connection in the Intellicus Repository

This DotNet API is used to create a new Database Connection in the Intellicus Repository.

Refer to SampleCodes/DotNetAPI/DBConnection Mangement/CreateDBConnection.cs 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. Set url for DBDriver.

```
DBDriver driverDB = new DBDriver();

//Making an object of hashmap for DBDriver
Hashtable hMap = new Hashtable();

String server = "192.168.33.52";
String provider = "MSSQL";
String port = "1433";
String database = "M90";
String driverVersion = "2005";

driverDB.Url                                                    =
"jdbc:sqlserver://192.168.33.52:1433;databaseName=M90;selectMetho
d=cursor;user=sa;password=intellicus";
driverDB.Provider = provider;

hMap["SERVER"]= server;
hMap["PORT"]=port;
hMap["DATABASE"]=dataBase;
hMap["DRIVERVERSION"]=driverVersion;

driverDB.AttrHash = 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.DbDriver =driverDB;
dbConn.UserId =userId;
dbConn.Password = passwrd;
dbConn.ConnectionName =connName;
```

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 DotNet API is used to delete an existing Database Connection in the Intellicus Repository.

Refer to SampleCodes/DotNetAPI/DBConnection Mangement/DeleteDBConnection.cs 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. 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 = lmanager.getDBConnectionList(filter, requestorUserInfo);
foreach (DictionaryEntry de in map)
   {
    System.Collections.ArrayList                l                =
(System.Collections.ArrayList)(map[de.Key]);

                //Iterating through the List
                for (int i = 0; i < l.Count; i++)
                {
                    System.Object o = l[i];

                    // ArrayList contains both DBConnection and
DBProvider class type
                    // When class type is DBConnection
                    if
("com.intellica.client.common.DBConnection".Equals(System.Convert
.ToString(o.GetType().ToString()),StringComparison.CurrentCulture
IgnoreCase))
                    {
                        //This class stores the information
regarding a database connection
                        DBConnection dbConn = (DBConnection)l[i];

                        // Check whether the Connection i.e. to
be deleted exists or not
                        if
(connectionName.Equals(dbConn.ConnectionName))
                        {
                            try
```

```
                                {
                                    // The method delete given Report
Server Connection
                                    // Syntax : public String
deleteReportServerConnection(DBConnection dbConnection, UserInfo
userInfo)
                                    // @ param dbConnection :
DBConnection object which needs to be deleted.
                                    // @ param userInfo : The
UserInfo object

lmanager.deleteReportServerConnection(dbConn, requestorUserInfo);
                                }
                                catch (LayoutHandlerException
lhException)
                                {
                                    Logger.error(lhException);
                                }
                            } //end inner if
                        } //end Outer if
                    } //end for
                }                   }
                 }//end inner if
            }//end Outer if
      }//end for
}//end while
```

ReporServerProperty

GetReportEnginePortAndIP

This DotNet API is used to get Report Engine's IP Address and Port number of the Report Engine.

Refer to SampleCodes/DotNetAPI/ ReportServerProperty/ GetReportEnginePortAndIP.cs 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. Get the IP of Report Engine.

Property: ReportEngineIP

```
Public string ReportEngineIP
```

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

```
// property used to get the IP Address of Report Engine
// Returns ReportEngine IP as a String value.
String reportEngineIP=lm.ReportEngineIP;
```

5. Get the port of Report Engine.

Property : ReportEnginePort

```
public int ReportEnginePort
```

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

```
  // property used to get the Port of Report Engine
  // Returns report Engine Port as Integer value
  int reportEnginePort=lm.ReportEnginePort;
```

GetReportServerProperty

This DotNet API is used to get SMTP Server Values of the Report Engine.

Refer to SampleCodes/DotNetAPI/ ReportServerProperty/GetReportServerProperty.cs 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 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);
Logger.debug("SMTP_SERVER Value : "+value);

key="LISTENER_PORT";
value=sMgr.getReportServerPropValue(arrList ,key,
                         requestorUserInfo);
Logger.debug(" LISTENER_PORT Value : "+value);

key="DATABASE_CONNECTION_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                              requestorUserInfo);
Logger.debug("DATABASE_CONNECTION_TIMEOUT Value :
  "+value);

key="SECURITY_FEATURES";
value=sMgr.getReportServerPropValue(arrList ,key,
                              requestorUserInfo);
Logger.debug("SECURITY_FEATURES Value : "+value);
```

```
key ="AUDIT_LOG";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Logger.debug("AUDIT_LOG Value : "+value);

key = "QUEUE_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Logger.debug("QUEUE_SIZE Value : "+value);

key = "REMOTE_SESSION_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                                requestorUserInfo);
Logger.debug("REMOTE_SESSION_TIMEOUT Value : "+value);

key = "RTF_FIELD_CONTROL_MAP";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Logger.debug("RTF_FIELD_CONTROL_MAP Value : "+value);

key = "DATA_SOURCE_FETCH_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                    requestorUserInfo);
Logger.debug("DATA_SOURCE_FETCH_SIZE Value : "+value);

key = "AUDITLOG_PURGE_TIME";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Logger.debug("AUDITLOG_PURGE_TIME Value : "+value);

key = "CACHE_PURGE_TIME";
value=sMgr.getReportServerPropValue(arrList ,key,
                                requestorUserInfo);
Logger.debug("CACHE_PURGE_TIME Value : "+value);

key = "AUTHORIZATION_CACHE_TIMEOUT";
value=sMgr.getReportServerPropValue(arrList ,key,
                                requestorUserInfo);
Logger.debug("AUTHORIZATION_CACHE_TIMEOUT Value :
                                          "+value);
key = "SCHD_JOB_DISPATCH_QUEUE_SIZE";
value=sMgr.getReportServerPropValue(arrList ,key,
                                    requestorUserInfo);
Logger.debug("SCHD_JOB_DISPATCH_QUEUE_SIZE Value :
                                "+value);
```

SetReportServerProperty

This DotNet API is used to set the SMTP Server Values of Report Engine.

Refer to SampleCodes/DotNetAPI/ReportServerProperty/ SetReportServerProperty.cs 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. Set the Property values of Report Server by putting them in a Hashmap.

```
Hashtable ServerPropHMap= new Hashtable();

  String key="SMTP_SERVER";
  String value="192.168.100.20";
  ServerPropHMap[key] = value;

  key="DATABASE_CONNECTION_TIMEOUT";
  value="800";
  ServerPropHMap[key] = value;

  key="SECURITY_FEATURES";
  value="enabled";
  ServerPropHMap[key]=value;

  key="QUEUE_SIZE";
  value="900";
  ServerPropHMap[key]=value;

  key="LOG";
  value="../logs";
  ServerPropHMap[key]=value;

  key="PAGE_CHUNKSIZE";
  value="10";
  ServerPropHMap[key]=value;

  key="LOG_LEVEL";
  value="INFO";
  ServerPropHMap[key]=value;

  key="USER_THREADS";
```

```
value="5";
ServerPropHMap[key]=value;

key="REPOSITORY_CACHE_TIMEOUT";
value="15";
ServerPropHMap[key]=value;

key="EXIT_ON_ERROR";
value="disable";
ServerPropHMap[key]=value;
```

5. Save these property values set in last step for Report Server

Method : saveReportServerProperties

```
public string saveReportServerProperties
      (Hashtable 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 DotNet API is used to check whether the Report Server is running on the specified IP and Port or not.

Refer to SampleCodes/DotNetAPI/TestServerConnectivity/ TestServerConnectivity.cs for sample code of this use case.

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.

Property : SecurityMode

```
public boolean SecurityMode
```

This property 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.

```
boolean securityMode    = false;

//If it gives Exception, then failed to connect Report Server
securityMode=sMgr.SecurityMode;
```

User Preferences

GetUserPreferences

This DotNet API is used to get the default User Preferences set by the User.

Refer to SampleCodes/DotNetAPI/User Preferences/GetUserPreferences.cs for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a PersonalizationManager class object for UserPrefernces.

```
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. At the time of return this UserInfo contains the information regarding the user Preferences also.

Returns:
Preferences for the user.

```
Preferences userPref = null;
userPref=  pm.getUserPreferences(requestorUserInfo);
```

5. Get all the User Preferences

```
 //Returns the defaultConnection
 String defaultCon = userPref.DefaultConnection;

 //Returns the user's preferred default portal theme
 String  portalTheme = userPref.DefaultPortalTheme;

 //Method returns default stylesheet of the user
 String defaultStylesheet = userPref.DefaultStylesheet;

 //Method returns the user's preferred deliver location
 String deliveryLocation = userPref.DeliveryLocation;

 //Method returns the user's preferred report format
 String defaultFormat = userPref.Format;

 //Method returns the user's preferred language
 String prefLanguage = userPref.PrefLanguage;

//Method returns the number of recent reports to be shown
int reportCount = userPref.ReportCount;

//Method returns the Boolean value whether to show the inbox
Boolean showInbox = userPref.ShowInbox;

//This Method returns default template name used for all
   reports
String templateName = userPref.TemplateName;

// Method returns the Boolean value whether to show the recent
   reports or not
Bool showRecentReports = userPref.ShowRecentReports;
```

```
Logger.debug("Default Connection  :  "+defaultCon);
Logger.debug("Default Portal Theme  :  "+portalTheme);
Logger.debug("Default Stylesheet:"+defaultStylesheet);
Logger.debug("Delivery Location  :  "+deliveryLocation);
Logger.debug("Default Format  :  "+defaultFormat);
Logger.debug("Default Preferred Language  :
   "+prefLanguage);
Logger.debug("Report Count  :  "+reportCount);
Logger.debug("Show Inbox  :  "+showInbox);
Logger.debug("Template Name  :  "+templateName);
Logger.debug("Show Recent Reports  :
        "+showRecentReports);
```

setUserPreferences

This DotNet API is used to set the default User Preferences by the User.

Refer to SampleCodes/ DotNetAPI/User Preferences/SetUserPreferences.cs for sample code of this use case.

Steps:

1. Initialize Report Client.
2. Initialize Requestor UserInfo.
3. Create a PersonalizationManager class object for UserPrefernces

```
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.ReportCount = reportCount;
pref.TemplateName =templateName;
pref.ShowRecentReports = true;
pref.DefaultConnection=defConnection;
pref.DefaultPortalTheme="Default";
pref.Email=email;
pref.Format=InteraConstants.ReportFormats.PDF;
```

```
pref.PrefLanguage="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);
```