Intellicus COM API Guide

Intellicus Enterprise Reporting and BI Platform

intellicus
Enterprise Reporting and
Business Insights Platform

info@intellicus.com
www.intellicus.com

Dated: - September 2010.

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

1

CVista

Important:

MSDEV will crash if the resource dialog box (which contains the CVista ActiveX control) is open and one closes it and then re-compiles or closes the workspace or MSDEV.

Workaround - One should not open this dialog box in resource editor unless it is needed.

Assertion occurs when a report run through a 'debug build' executable, fails to execute.

The Basics

CVista is an ActiveX control used for viewing reports. CVista is intellicus's proprietary format.

APIs Exposed

CreateReportInstance(long ReportType, [optional]VARIANT IRLData,[optional] VARIANT bIsReportVisible)

This API is used to create a new Instance of a report. It takes following parameters:

Mandatory Parameter

ReportType: Report type. Possible value-enums are:

- eRT_STATIC if the report is a static report
- eRT_DYNAMIC if the report is dynamic report
- eRT_SAVEDREPORT if the report is a published report

Optional Parameters

IRLData: Content of IRL file.

BIsReportVisible: To set if the report should be visible or not.

- True: To make the report visible.
- False: To hide the report.

ExportReport(long ReportID, [optional]VARIANT eOutputFormat, [optional]VARIANT bstrOutputFileName)

This API is used to export a report. It takes following parameters:

Mandatory Parameters

ReportID: Report ID of the report to be exported. Pass –1 to export the active report.

Optional Parameters

eOutPutFormat: Output format in which the report should be exported.

bstrOutputFileName: Name of output file. File will be saved with this name.

HandleKeys (long KeyCode, long Shift)

This API is called to handle keyboard events on Cvista level. This API was exposed due to TAB Control used in Studio which was capturing Arrow keys event. Due to which CVista was not able to handle arrow key events for scrolling when used in studio.

Initialize (long SourceType, BSTR bstrSourceName)

This API is used to configure CVista before executing a report. It has following parameters:

Mandatory Parameters

SourceType: Type of source from where report will come.

BstrSourceName: Name of source from where report will come.

SourceType uses e-nums. Value for bstrSourceName should be passed based on the value of SourceType.

SourceType and corresponding source name can be any of the following:

SourceType	bstrSourceName
eST_ENGINE	Ip:Port
eST_URL	URL
eST_REPORTFILE	File name

InitializeDefault

This API Provides the Default Initialization to CVista.

LoggingInfo(BSTR bstrFilePath, long level)

This is an API used to set logging information.

Mandatory Parameters

bstrFilePath: To set log file path.

level: To set log level.

OpenReport(BSTR bstrReportName)

This API is used to open a report saved on a file system. It throws exception to the host application if any error occurs at the time of opening the report. It has following parameter:

BstrReportName: The report name including its absolute path.

PrintReport(short ReportID , [optional] VARIANT bstrPrinterName)

This API is used to print a report. It has following parameters:

Mandatory Parameter

This is an API exposed by CVista to print the report specified by ReportID. It takes another optional parameter Printer Name, which is not used as of now. It is reserved for future use. Passing ReportID value as -1 will print the active report.

RemoveAllReports

This is an API exposed by CVista for Deleting All the Report instance.

RemoveReport(long ReportID)

This is an API exposed by CVista for Deleting the Report instance specified by ReportID. Passing ReportID value as -1 will remove the active report.

SaveReport(long ReportID , [optional] VARIANT bstrOutputFileName)

This API saves the Report specified by ReportID to report output to a file on the Disk. Path is specified by the *bstrOutputFileName* parameter. Passing ReportID value as -1 will save the active report.

SetReportConnection(long ReportID, BSTR bstrReportTitle)

This is an API exposed by CVista for setting the connection name for the report specified by ReportID. Passing ReportID value as -1 will set the connection name for the most active report. If failed, an exception is thrown to the host application.

SetReportParameters(IReportParameters* pReportParameters)

This API is required for setting the ReportParameters for the Active Report.

SetReportTitle(long ReportID, BSTR bstrReportTitle)

This API is exposed by CVista for setting the report title for the report specified by ReportID. Passing ReportID value as -1 will set the report title for the most active report. If failed, an exception is thrown to the host application.

SetShowParamDialog(VARIANT_BOOL bShowParamDialog)

This API is required for setting whether to show Parameter Dialog or not.

SetSysParam(long Name, BSTR Value)

This is an API exposed by CVista for Setting the Values of System parameters for a report. It takes name of system parameter as enum specified bye SysConstants enum. This API sets the system parameter for the most active report.

SetUserInfo(IUserInfo* pUserInfo, VARIANT_BOOL bIsSecurityEnabled)

This API is required for setting the User Info and security options. If failed, an exception is thrown to the host application.

ShowNextPage(long ReportID)

This API is called when the user wants to navigate to the next page of the report specified by ReportID. Passing ReportID value as -1 will navigate to next page of the most active report.

ShowPage(long ReportID, long PageNo)

This API is exposed by CVista to navigate to the specific page for report specified by ReportID. Passing ReportID value as -1 will navigate to page specified by PageNo of the most active report.

ShowPrevPage(long ReportID)

This API is called when the user wants to navigate to the previous page of the report specified by ReportID. Passing ReportID value as -1 will navigate to previous page of the most active report.

ShowReport

This API initiates the display of most active report. If failed, an exception is thrown to the host application.

ShowSavedReport(BSTR bstrReportName,BSTR bstrOutputID, BSTR bstrReportOID)

This API shows the saved report. Saved Report is specified by OutputID and ReportOID. Report Name is specified by bstrReportName. If failed, an exception is thrown to the host application.

StopReportExecution(long ReportID)

This is an API exposed by CVista for stopping the execution of report specified by ReportID. Passing value of ReportID as -1 will stop the execution of the most active report.

2

CVistaParam

The Basics

CVistaParam is an active X control. With the help of CVistaParam user can provide parameters value at run time. Parameters are the conditions provided for retrieval of reports. User can provide the parameter value at runtime and retrieve the related data from the database. CVistaParam provides a mechanism by which user can provide parameter value at run time. After taking values of parameters from user, CVistaParam updates report parameters accordingly and sets the result back to the user.

APIs exposed

The APIs exposed by CVistaParam are:

GetReportParameters

API for Validating the ReportParameters as well as serves as the getter methods for the ReportParameters object after updating its state after accepting the inputs from the User.

IsParametersInitialized

This API Checks whether the parameters are initialized for the control or not.

IsValid(VARIANT_BOOL bShowMsg)

This API is used for checking whether all the mandatory parameters have been set or not. The parameter is used to specify whether the Control itself or the host application is responsible for showing the Error message. Returns a Boolean TRUE is all the Mandatory parameters have been set else returns a FALSE.

SetConnectionName(BSTR bstrConnectionName)

This API sets the Connection name. Input to this API is name of the connection.

SetCriteriaTabCaption(BSTR bstrCaption)

This API Sets the Criteria tab Caption. Input to this API is caption.

SetReportParameters(IReportParameters* pReportParameters)

This API is used for setting the ReportParameters Object by the Host Application. This API is also responsible for drawing the controls for the parameter Form.

SetSortTabCaption(BSTR bstrCaption)

This API Sets the Sort Tab Caption. Input to this API is caption

SetUserInfo(IUserInfo* pUserInfo)

This API is used to set the user information. The input to this API is an object of IUserInfo.

3

REReqLib

The Basics

The REReqLib.Dll contains common data structures which are commonly used by the Desktop Studio, CVista and CVistaparam. By this means it has reduced code redundancy.

ISystemParameters

This is the one of the category of parameters that has to be passed on to the user. These parameters are always present and govern the basic behaviour of report execution. E.g. which format has to be generated upon execution; whether the report has to be printed, saved or viewed; if printed then which printer to use and so on. As before, the user cannot add any new parameters but can only query the existing value or modify it.

One important note about the System parameters is that it uses an enum defined in the ReCOMCLient type library, called SYSTEM_CONSTANTS. This enum contains a list of many constants in use by System Parameters functionality. **Not all constants qualify to be used as parameter names** though.

Following APIs and properties are supported by ISystemParameters:

ParamNames

This get property returns a list of all parameter names as strings. I.e. it returns the list as an array of VARIANTS. Remember that the names sent here are strings**, and the argument ParamName taken by PutValue method should essentially be one of the constants declared in SYSTEM_CONSTANTS. This is a slight discrepancy which will l have to be removed in the near future.**

Value (long eParamName)

This get property returns the current value of the specified parameter. Remember that the argument has to be a constant belonging to the SYSTEM_CONSTANTS enum. However, also remember that all the constants in that enum are not valid parameter names. If the numeric constant specified is found to be invalid, then an exception is thrown.
Also remember, that unlike UserParameters, SortParameters, FilterParameters, a value picked from the list returned by 'ParamNames' will not do since

ParamNames returns strings and this function takes a numeric constant. This inconsistency will be removed ASAP.

PutValue(long eParamName, BSTR bstrValue)

This **method** specifies the new value for a particular parameter. The parameter name has to a valid Numeric constant, if not, then an exception will be thrown. The **bstrValue** should be a ready to insert **string**. Read the paragraph above to know more about ParamConstant.

String (long eParamName)

This get property takes a valid numeric constant as a parameter. This parameter is mapped to a corresponding string and that string is then returned as a VARIANT. E.g. suppose the numeric Constant you passed was : SYSTEM_PARAM_OPERATION_TYPE then the string that you get might be : "OPERATION TYPE". If the numeric constant passed is not correctly mapped to any corresponding string, an exception is thrown.

PrinterNames

This get property returns a list of all printer names available at the server. Any of the string can be directly used as a value for the parameter SYS_PARAMS_PRINTER_NAME.

DefaultPrinter

This get property returns a single string and this specifies the name of the server's default printer.

Exists (VARIANT *Number)

This get property takes a numeric constant as a parameter and verifies whether it is one of the valid System Parameter numeric constants. I.e. if it can be used to specify a system parameter. It returns a Boolean answer as a VARIANT.

GetXML

This API returns the system parameters XML.

PopulateServerPrinterList (VARIANT *varpNodeList)

This API populates the Server printer list. It takes XML DOM node pointer as input.

UpdateSystemParameters(ISystemParameters *pSysParams)

This API is deprecated.

GetParameterIndex(BSTR bstrParameterName)

This API returns the enum corresponding to parameter name. Parameter name is specified by the input parameter bstrParameterName.

IUserParameters

This is the one of the categories of parameters that have to be sent along with the execution request. By far, it is also the most complicated. These are parameters, which are dependent on the report. I.e. these parameters are specific to the report that the end-user has chosen for execution.

The object **UserParameters** is basically an aggregation of **UserParameter** objects.

This interface also provides necessary interfaces to iterate **IUserParameter** objects.

Following APIs and properties are supported by IUserParameters:

ParamNames

This get property, like all its counterparts in other interfaces, returns a list of valid parameter names (a safearray as a VARIANT). Any name from this list can be used to retrieve or set the value of that particular parameter.

UserParameter (VARIANT *paramName)

This get/set property returns/sets an object of the type IUserParameter. It takes a parameter name as an argument, i.e. it returns the UserParameter object corresponding to that particular parameter name. If the parameter name is not found then an exception is thrown. For details, see **IUserParameter** class section.

Add(IUserParameter* ptrParam, BSTR key)

This API adds a userparameter to the userparameters Collection. It takes Userparameter and key as input parameter. Key is not used but it is reserved for future use.

GetGroupedParameters(IUserParameters **pMandatoryParams,IUserParameters **pOptionalParams)

This API seperates the UserParameters Collections into two userParameters collections which are Mandatory and optional.

GetXML(BSTR *pbstrXMLString)

This API returns the UserParameters XML.

IsParametersUpdated()

This API Checks whether the values for the Parmeters collection are updated or not.

IsParametersVisible()

This API returns Weather any parameter in the parameters collection is Visible or not.

LoadXML(BSTR bstrXMLString,long eParamSource)

This API loads the userparameters XML and populates Userparameters collection.

Remove(long Index)

This API removes the userparameter from the specified index passed as input parameter.

UpdateUserParameters(IUserParameters *pUserParams)

This API Updates the UserParamters Collections with the help of UserParameters Collection passed as input Paramter to this API.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the **IUserParameter** object from specified index. For details, see **IUserParameter** class section.

Count

This property returns the total number of **IUserParameter** objects in its collection.

IUserParameter

This interface maintains detailed information about a particular user parameter object. Although, the parameter finally evaluates to just one 'value', some display related information is needed for the parameter, which is what this object contains.

Following APIs and properties are supported by IUserParameter:

Name

This get/set property returns/sets the name of the parameter as a string (although not important, note that this name will be the same as the name that is used to get this user parameter object from UserParameters object).

FilterParameters

This is a get property which returns the FilterParameters object. For details, see IFilterParameters class section.

Format

Get/set property for format of input parameter.

IsParameterGrouped

This is a get bool property specifying if parameter is grouped or not.

IsRestrictToList

This is a get bool property specifying whether the user should be allowed to select a value from available values in a combo or should be allowed to key in a value that is not present in combo.

IsSearchEnabled

This is a get bool property specifying if the parameter is search parameter or not.

IsShadowParam

This is a get bool property specifying if the parameter contains shadow params.

IsTreeView

This is get bool property specifying if the parameter is of treeview type parameter or not.

ParameterGroups

This is a get property which returns the object of ParameterGroups. For details, see IParameterGroups class section.

TreeNode

This is a get property which returns TreeNode object which is used for treeview type parameter. For details, see ITreeNode class section.

UnCheckedValue

This is a get property which is used for Check box (Boolean type) parameter. If the value of UneckedValue matches with the default valur the check box parameter should be unchecked

GetEnclosedBy

This API returns the enclosedby string which is used to enclose the parameter value to separate it from other parameter values

GetSeperator

This API returns the separator

GetSQLStmt

This API returns the SQL statement contained by the userparameter

GetUpdatedValue

This API returns the updated value of the parameter

CheckedValue

This is a get property which is used for Check box (Boolean type) parameter. If the value of CheckedValue matches with the default valur the check box parameter should be checked

ComboType

This get/set property returns/sets the combo type.If the InputType property evaluates to IT_COMBO, then this property further specifies what type of combo box the parameter should be displayed with. I.e. this property returns an integer that should be compared against the constants declared in the enum ComboType. The ComboType is predefs by default, so make sure that you also

check the value of InputType before reaching any conclusion about the type of input mechanism.

InputType	ComboType	Final Input Mechanism
IT_TEXT	DON'T CARE	Text Box
IT_COMBO	CT_PREDEF	Predefined Combo box
IT_COMBO	CT_SQL	Combo box populated through SQL query.
IT_OPTION	CT_PREDEF	Predefined RadioListBox
IT_OPTION	CT_SQL	RadioListBox populated through SQL query.

DataType

This get/set property returns/sets an integer as a VARIANT that specifies what the data type of the parameter is. The integer returned can be compared against an enum defined in the type library: DataType. This property can have one of the following values from enum **DataType**

- DT_CHAR
- DT_NUMBER
- DT_DATE
- DT_BOOL

DefaultValue

This get/set property returns/sets the default value of the parameter (if any) as a string.

InputType

This get/set property returns/sets a VARIANT containing an integer to ascertain the input type of the parameter e.g. text box or a combo box. This integer can be compared using constants declared in the enum InputType declared in type library. This property can have one of the following values from enum **InputType**

- IT_TEXT
- IT_COMBO
- IT_OPTION

IsMandatory

This get/set property returns/sets a Boolean value which specifies whether the parameter is mandatory or not. The Boolean is returned inside a VARIANT.

IsVisible

This get/set property again, returns/sets a VARIANT containing a Boolean expression saying whether the parameter is initially visible or not.

Predefs

This get/set property returns/sets an object of type **IPredefs**, which contains a list of predefined values to populate the Combo box. This object is returned only if the InputType is IT_COMBO and the Combo Type is CT_PREDEF. If not, then an exception is thrown on using this property. For details, see **IPredefs** class section.

Prompt

This get/set property returns/sets the prompt string for the Parameter. (BSTR as a VARIANT).

Size

This get/set property returns/sets the size as an integer.

Value

this is a get/set property gets or sets the current value of the parameter. (BSTR as a VARIANT).

ValueColumn

This get/set property is available only if the user parameter is a SQL combo box. If yes, then this property returns/sets the name of the recordset column which is to be used to pick up the actual values from. It returns a string. (See next).

DisplayColumn

This get/set property is available only if the user parameter is a SQL combo box. If yes, then this property returns the name of the recordset column from which the values to display are taken.

The SQL combo box works in the following way: The combo box that is finally dislayed on the UI has a drop down menu. The menu contains strings, one of which the user will choose as the parameter value. However, these strings (the ones that are being displayed) can be mapped (one-to-one) to another list of parameters that are internally sent to the engine but are not displayed. Both the lists are lists are obtained in the form of a recordset containing two columns. Thus, the column of the recordset that contains the values to be displayed on UI is the DisplayColumn, and the column that contains the values which are actually to be sent internally is called the valueColumn. It is possible that both of these

be the same if what is displayed is what is to be sent.

Combo_IsMultiselect

This get/set property sets or returns the bool value indicating whether combo type parameter is a multiselect type parameter (list box) or single select type parameter(combo box).

UpdateUserParameter

Updates the Value and default values for a parameter.

ShadowParamName

This get property returns the shadow param name.

ShadowParamValue

This put property sets the shadow param value.

IPredefs

This interface helps in maintaining a list of display string – Value pairs to be used for Combo boxes. Following APIs and properties are supported by IPredefs:

Keys

This get property returns an array of all Display Strings. I.e. an array of VARIANTS in a VARIANT. A string from this list is used to get the 'value' implied by the display string.

Value (VARIANT *DisplayString)

This get property returns the implied value of the given display string as a VARIANT. A list of valid display string is returned by the Keys method. An invalid display string produces an exception.

ILicenseDetails

LicenseDetails object contains information about the license. REComClient's LicenseManager fetches the data related to License and gives the response XML to LicenseDetails object. LicenseDetails object parses the XML and Fills the information.

Following APIs and properties are supported by ILicenseDetails:

ExpiryDate

This is a get property which returns ExpiryDate of the license.

LicenseeID

This is a get property which returns Licensee ID of the license.

IsPrintSupported

This is a get property which returns a bool value specifying whether printing is supported in this license.

IsPrintAtServerSupported

This is a get property which returns a bool value specifying whether printing at server is supported in this license.

IsSecurityEnabled

This is a get property which returns a bool value specifying whether security is enabled in this license.

IsScheduleSupported

This is a get property which returns a bool value specifying whether scheduling is supported in this license.

IsOLAPSupported

This is a get property which returns a bool value specifying whether OLAP is supported in this license.

IsColumnMaskSupported

This is a get property which returns a bool value specifying whether column masking is supported in this license. Column masking is a feature in which particular columns data are masked with some characters.

IsAdhocWizardSupported

This is a get property which returns a bool value specifying whether Adhoc Wizard is supported in this license.

IsCollaborationSupported

This is a get property which returns a bool value specifying whether collaboration is supported in this license.

IsCabDeploySupported

This is a get property which returns a bool value specifying whether Cab deployement is supported in this license.

IsMailReportSupported

This is a get property which returns a bool value specifying whether mailing a report is supported in this license.

IsAdHocPowerViewerSupported

This is a get property which returns a bool value specifying whether Adhoc power viewer feature is supported in this license.

IsAuditLogSupported

This is a get property which returns a bool value specifying whether audit logging supported is supported in this license.

IsSecurityCallBackSupported

This is a get property which returns a bool value specifying whether Security call back is supported in this license.

IsDashboardSupported

This is a get property which returns a bool value specifying whether Dashboard feature is supported in this license.

ProductEdition

This is a get property which returns Product Edition.

ProductLicenseType

This is a get property which returns Product License Type.

LicenseeName

This is a get property which returns LicenseName.

StartDate

This is a get property which returns StartDate of the Licnese.

ServerVersion

This is a get property which returns the Report server version.

ServerProcessorFamily

This is a get property which returns the processor family of the machine on which report server is running.

ServerLocation

This is a get property which returns the server location on which report server is running.

ExpiryDate

This is a get property which returns the expiry date of the license.

CPUCount

This is a get property which returns no of CPUs present on the machine.

StudioSeats

This is a get property which returns the no of studio.

NoOfWebStudioUsers

This is a get property which returns the maximum no of users who can use.

NoOfDBConnections

This is a get property which returns the maximum no of Databse connections that can be permitted in this license.

NoOfStudioUsers

This is a get property which returns the maximum no of users who can use studio.

GetNoOfOrganizations

This is a get property which returns the maximum no of organizations which can be permitted in this license.

IsPublishReportSupported

This is a get property which returns the bool value specifying whether publishing report is supported.

ServerOperatingSystem

This is a get property which returns the Operating system running on server.

ThreadCount

This is a get property which returns the maximum no of permitted threads.

NoOfUsers

This is a get property which returns the maximum no of users permitted in this license.

IsSecurityAccessRightsEnabled

This is a get property which returns the bool value specifying whether security access rights are enabled.

IsMetaLayerEnalbed

This is a get property which returns the bool value specifying whether meta layer is enabled.

IsPersonalizationEnabled

This is a get property which returns the bool value specifying whether Personalization is enabled in this license.

IsSecurityUserMapSupported

This is a get property which returns the bool value specifying whether Security user map is supported in this license.

IsLoadBalancerSupported

This is a get property which returns the bool value specifying whether load balancer is supported in this license.

ServerMachineID

This is a get property which returns the server machine's ID.

ServerProductName

This is a get property which returns the server produce name.

IsDesignerAPIsSupported

This is a get property which returns the bool value specifying whether Designer APIs are supported in this release.

IsExportFormatSupported(short eReportFormat, VARIANT_BOOL *pVal)

This API determines if exporting in the Report format specified by input parameter eReportFormat is supported in this license or not. If yes, this API returns TRUE otherwise FALSE.

IsOutputFormatSupported(short eReportFormat, VARIANT_BOOL *pVal)

This API determines if Report output format specified by input parameter eReportFormat is supported in this license or not. If yes, this API returns TRUE otherwise FALSE.

ParseXML(BSTR bstrXML)

This API parses the LicenseDetails XML and fills the information related to license details.

IParameterGroups

This interface is a collection of IParameterGroup objects. It maintains the collection of IParameterGroup. This interface also provides necessary interfaces to iterate IParameterGroup objects. Following APIs and properties are supported by IParameterGroups:

ParameterGroup (BSTR bstrGroupName)

This 'get' property returns the ParameterGroup object. It takes Group name as input parameter to identify the ParameterGroup object from the collection. For details, see **IParameterGroup** class section.

ParamGroupNames

This 'get' property returns the array of parameter group names, which acts as a key to retrieve the **ParameterGroup** object.

Name

This get property returns the groups name.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IParameterGroup object from specified index. For details, see **IParameterGroup** class section.

Count

This property returns the total number of **IParameterGroup** objects in its collection.

LoadXML(BSTR bstrXML)

This API loads the XML passes as input parameter. It parses the XML and populates ParameterGroup objects.

IParameterGroup

ParameterGroup object contains the information about user preferences. ParameterGroup object is used for default selection of values in user parameter. A User might have his own preference of select values for an exported user parameter and details of these select values are contained by ParameterGroup object. Following APIs and properties are supported by IParameterGroups:

Name

This is a get/set property which returns/sets the name of parameter group.

ID

This is a get/set property which returns/sets the ID for the parameter group.

IsPublic

This is a get/set property which returns/sets bool value which specifies whether this parameter group is public or not.

IsUsed

This is a get/set property which returns/sets bool value which says whether this parameter group is a preference of user (selected by user) or not. If value is TRUE, the values contained by this group must be selected in user parameter.

Values

This is a get property which returns array of strings.

LoadXML

This API loads the XML and populates Name, ID, IsPublic, IsUsed, Values, OwnerAppID, OwnerOrgID.

IReportParameters

ReportParameters is a house keeper of SystemParameters object, UserParameters object, FilterParameters object and SortParameters object. Following APIs and properties are supported by IReportParameters:

FilterParameters

This get/set property sets or returns the FilterParameters object. For details, see **IFilterParameters** class section.

UserParameters

This get/set property sets or returns the UserParameters object. For details, see **IUserParameters** class section.

SystemParameters

This get/set property sets or returns the SystemParameters object. For details, see **ISystemParameters** class section.

SortParameters

This get/set property sets or returns the SortParameters object. For details, see **ISortParameters** class section.

LoadXML(BSTR bstrXMLString, long eParamSource);

This API loads the XML and populates SystemParameters, UserParameters and FilterParameters object. bstrXMLString is a input parameter string XML.

GetXML(BSTR* pbstrXMLString)

This API returns the ReportParameters XML. The XML consists of SystemParameters' XML, UserParameters' XML, SortParameters' XML and FilterParameters' XML.

UpdateReportParameters(IReportParameters* pRptParameters)

This API updates the ReportParameters.

ResetFilterParameters

This API resets the FilterParameters object.

ISearchColumns

ISearchColumns contains a collection of ISearchColumn objects. This interface also provides necessary interfaces to iterate **ISearchColumn** objects. Following APIs and properties are supported by ISearchColumns:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ISearchColumn object from specified index. For details, see **ISearchColumn** class section.

Count

This property returns the total number of **ISearchColumn** objects in its collection.

ISearchColumn

Following APIs and properties are supported by ISearchColumn:

Name

This get/set Property sets or returns the Name of SearchColumn.

DataType

This get/set Property sets or returns the datatype of the Column. This property can have one of the following values from enum **DataType**

- DT_CHAR
- DT_NUMBER
- DT_DATE
- DT_BOOL

Prompt

This get/set Property sets or returns the Caption text (prompt text) of the Column.

ISortFields

ISortFields object contains the collection of **IReportField** object. This interface also provides necessary interfaces to iterate **IReportField** objects.

Following APIs and properties are supported by ISortFields:

SortFieldNames

This is a get property which returns the array of Field Names. These field names can be used to retrieve **IReportField** object.

SortField(BSTR bstrParamName)

This is a get/set property which returns/sets IReportField object from the collection using the key bstrParamName which is passed as parameter. If object does not exist, it throws an exception. For details, see **IReportField** class section.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the **IReportField** object from specified index. For details, see IReportField class section.

Count

This property returns the total number of **IReportField** objects in its collection.

Remove([in] long Index)

This API removes the IReportField object from the collection contained at the specified index which is passed as input parameter. If Index is invalid , an exception is thrown.

GetXML

This API returns SortFields XML.

Add(IReportField* ptrParam, BSTR key);

This API adds IReportField object to the ReportFields Collection. Key is a optional parameter which is not used as of now. It is reserved for future use.

UpdateSortFields

This API updates the SortFields for the Report using the Source SortFields Passed as Parameter to this Function.

IReportField

Following APIs and properties are supported by IReportField:

FieldName

This get/set property sets or returns Report Field Name.

SortType

This get/set Property sets or returns Type of sorting. This property can have one of the following values from enum SortOrder.

- ASCENDING
- DESCENDING

UpdateReportField(IReportField *pReportField)

This API does nothing.

IReportFields

ReportFields contains the collection of Report Field object. This interface also provides necessary interfaces to iterate IReportField objects. Following APIs and properties are supported by IReportFields:

ReportField(BSTR bstrParamName)

This is a get/set property which returns/sets IReportField object from the collection using the bstrParamName key which is passed as input parameter. For details, see **IReportField** class section.

ReportFieldNames

This is a get property which returns array of Report Field Names.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the **IReportField** object from specified index. For details, see **IReportField** class section.

Count

This property returns the total number of **IReportField** objects in its collection.

Remove([in] long Index);

This API Removes the IReportField object from the collection contained at the specified index which is passed as input parameter. If Index is invalid, an exception is thrown.

Add(IReportField* ptrParam, BSTR key)

This API adds IReportField object to the ReportFields Collection. Key is a optional parameter which is not used as of now. It is reserved for future use.

LoadXML(BSTR bstrXMLString, long eParamSource)

This API loads the XML and populates IReportField objects. eParamSource is a optional input parameter which is used to identify the type of parameter source. eParamSource can have one of the following values from enum **ParamSource .**

- PS_IRL
- PS_PARAMS
- PS_USERPARAMS
- PS_SQLEDITOR

Default value of eParameSource is PS_IRL.

GetSortCount

This API returns the Sort Parameter Count for the Report.

IsPromptEnabled

This API checks if the Prompt is enabled for the report or not.

ITreeNode

This class is basically meant for the tree view control information. One ITreeNode represents one tree node in tree view control. Following APIs and properties are supported by ITreeNode:

ColumnName

This get property returns the column name corresponding to the node.

ID

This get property returns the ID of the node. ID of root node is always 0.

DisplayText

This get property returns the display text of the node.

ValueText

This get property returns the value text for the node.

ParentNode

This get property returns the parent node which is an object of ITreeNode.

IsLeafNode

This get/set property specifies whether the node is a leaf node or not.

IsValueNode

This get/set property specifies whether the node is a value node or not.

IUserFilters

IUserFilters contains a collection of IUserFilter object. This interface also provides necessary interfaces to iterate IUserFilter objects. Following APIs and properties are supported by IUserFilters:

Add(IUserFilter* ptrFilter, BSTR key)

This API adds IUserFilter object to the UserFilters Collection. Key is a optional parameter which is not used as of now. It is reserved for future use.

Remove(long Index)

This API removes the IReportField object from the collection contained at the specified index which is passed as input parameter. If Index is invalid, an exception is thrown.

GetXML([out,retval] BSTR* bstrXML)

This API returns the UserFilters XML. The XML returned by this API does not contain <Filters> tag. The caller of this API has to manually append this tag if required.

IUserFilter

UserFilter class is used for filtering purposes. If one needs to apply filter, It needs to create UserFilter object, set the appropriate values in the properties of UserFilter object and add this UserFilter object to UserFilters collection. For each filter User must create new UserFilter object. Modifying the added UserFilter object to the collection will modify the content of added UserFilter object also. Following APIs and properties are supported by IUserFilter:

Name

This get/set property sets or returns the name of the Filter.

Operator

This get/set property sets or returns the operator.

Value

This get/set property sets or returns the value of the Filter.

IUserInfo

UserInfo object contains the Metadata Information. Following APIs and properties are supported by IUserInfo:

AppID

This get/set Property sets or returns the Application ID

OrgID

This get/set property sets or returns the Organization ID

Password

This get/set property sets or returns the password.

SessionID

This get/set property sets or returns the Session ID.

SD

This get/set property sets or returns the Security Descriptor.

CustomerID

This get/set property sets or returns the Customer ID.

Location

This get/set property sets or returns the Location.

Locale

This get/set property sets or returns Locale.

Timestamp

This get/set property sets or returns TimeStamp.

DBName

This get/set property sets or returns Database Name.

GUID

This get/set property sets or returns GUID.

RunTimeUserID

This get/set property sets or returns RunTime UserID.

RunTimePassword

This get/set property sets or returns RunTime Password.

GetMetadataTag

This API returns the Metadata XML string for the request.

IFilterParameters

FilterParameters contains a collection of FilterParameter object. This interface also provides necessary interfaces to iterate IFilterParameter objects. Following APIs and properties are supported by IFilterParameters:

FieldFilters

This get/set property sets or returns FieldFilters object. For details, see **IFieldFilters** class section.

OptionalFilters

This get/set property sets or returns OptionalFilters object. For details, see **IOptionalFilters** class section.

GetXML

This API returns FilterParameters XML.

LoadXML(BSTR bstrXMLString,long eParamSource)

This API loads FilterParameters XML and populates FilterParameter object.

GetFiltersCount(short eFilterType)

This API returns the Filters count based on the type of the filterparameters. Filter parameters can be of two types user parameters and Adhoc parameters. The value of eFilterType can have one of the following values from enum **FilterParamType.**

- USEPARAM
- ADHOC

ResetFieldFilters()

This API resets the FieldFilters object.

IFilterParameter

FilterParameter object contains the information related to filtering data purposes. FilterParameters are used for Adhoc filters as well as search parameters. Following APIs and properties are supported by IFilterParameter:

FirstValue

This get/set property sets or returns First operand. User can set it's input as initial or first operand in this property.

SecondValue

This get/set property sets or returns second operand. User can set it's input as last or second operand in this property.

DataType

This get/set property sets or returns the data type of the field. This property can have one of the following values from enum **DataType.**

- DT_CHAR
- DT_NUMBER
- DT_DATE
- DT_BOOL

FieldType

This get/set property sets or returns the type of field.

FetchData

This get/set property sets or returns bool value specifying whether data has been fetched from database or not. If True, the data can be populated using Predefs property.

ParamName

This get/set property sets or returns the parameter name.

ColumnName

This get/set property sets or returns the column name.

Operation

This get/set property sets or returns the operator between two operand.

Predefs

This get/set property sets or returns the operand values if FetchData property is TRUE.

GetXML

This API returns FilterParameter XML.

UpdateFilterParameter

This API updates the FilterParameter for the Report using the Source FilterParameter passed as Parameter to this Function.

IFieldFilters

FieldFilters contains the collection of FilterParameter objects. It maintains the collection of FilterParameter objects. This interface also provides necessary interfaces to iterate **IFieldFilter** objects. Following APIs and properties are supported by IFieldFilters:

MaxRows

This get/set property sets or returns the maximum number of rows to be returned in the recordset.

FilterParam(BSTR bstrParamName)

This get property returns the FilterParameter from the collection using the Filter parameter name specified by input parameter bstrParamName.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the **IFilterParameter** object from specified index. For details, see **IFieldFilter** class section.

Count

This property returns the total number of **IFilterParameter** objects in its collection.

GetXML

This API returns FilterParameters XML.

Add(IFilterParameter *ptrFieldParam, BSTR key);

This API adds FilterParameter object into the collection. Currently key is not used, it is reserved for future use. FilterParameter's name is used as key to insert into collection.

IOptionalFilters

This interface does the housekeeping of **IOptionalFilter** objects. This interface also provides necessary interfaces to iterate **IOptionalFilter** objects. Following APIs and properties are supported by IOptionalFilters:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the **IOptionalFilter** object from specified index. For details, see **IOptionalFilter** class section.

Count

This property returns the total number of **IOptionalFilter** objects in its collection.

Add(IFilterParameter* ptrOpFilter, BSTR key)

This API adds FilterParameter object into the collection. Currently key is not used, it is reserved for future use. FilterParameter's name is used as key to insert into collection.

LoadXML(BSTR bstrXMLString,long eParamSource)

This API loads FilterParameters' XML and populates FilterParameter objects.

ISortParameters

This interface contains all the information related with sort parameters. Following APIs and properties are supported by ISortParameteres:

ReportFields

This get property returns the **ReportFields** for Parameters. For details, see **IReportFields** class section.

SortFields

This get/set property returns/sets the SortFields for Parameters. For details, see **ISortFields** class section.

GetSortCount

This API returns the No. of SortParameters.

GetXML

This API returns the XML String for Sort Parameters.

IsSortParamDynamic

This API checks whether the Parameters are to be Prompted while viewing the report or not.

LoadXML

This API loads the SortParameters Object with the help of the XML String.

UpdateSortParameters

This API updates the SortParameters for the Report using the Source SortParameters Passed as Parameter to this Function.

Enums exported from ReReqLib

```
enum ProviderType
{
	PT_OCI,
	PT_ODBC,
	PT_MSSQL,
	PT_MY_SQL,
	PT_OTHERS,
	PT_SYBASE,
	PT_OCI_TNS,
	PT_POSTGRES,
	PT_INFORMIX,
	PT_ORACLE_THIN,
	PT_SOURCEFORGE,
	PT_ORACLE,
	PT_CSV,
	PT_XLS,
};

enum enumProviderType_OTHERS_Props
{
	EPT_OTHERS_DRIVER_CLASS_NAME=0,
};

enum enumProviderType_CSV_Props
{
	EPT_CSV_FILETYPES,
	EPT_CSV_ISEXTERNAL,
};

enum enumProviderType_XLS_Props
{
	EPT_XLS_FILETYPES,
	EPT_XLS_ISEXTERNAL,
};

enum DBMetadataMode
{
	DB_MD_TABLES,
	DB_MD_VIEWS,
```

```
        DB_MD_STOREDPROCEDURES,
        DB_MD_SYNONYMS,
        DB_MD_SYNONYMS_TABLE_OR_VIEW,
        DB_MD_SYNONYMS_FOR_SP,
        DB_MD_ALL
};

enum InputType
{
        IT_TEXT,
        IT_COMBO,
        IT_OPTION,
};

enum enumParamGroupsType
{
        EPGT_NONE,
        EPGT_PARAMETER ,
        EGPT_QUERY,
};

enum SortOrder
{
        ASCENDING,
        DESCENDING
};

enum FilterParamType
{
        USEPARAM,
        ADHOC,
};

enum DataType
{
        DT_CHAR,
        DT_NUMBER,
        DT_DATE,
        DT_BOOL
};

enum FieldFormat
{
        FF_NONE,
        FF_NUMBER,
        FF_STRING,
        FF_DATE,
        FF_BINARY,
        FF_TIME,
};

enum ComboType
{
        CT_PREDEF,
        CT_SQL
};
```

```
enum DateGroupBy
{
	DTG_NONE,
	DTG_DAY,
	DTG_WEEK,
	DTG_MONTH,
	DTG_QTR,
	DTG_YEAR,
};


enum ParamSource
{
	PS_IRL,
	PS_PARAMS,
	PS_USERPARAMS,
	PS_SQLEDITOR,
};

enum Appearance
{
	V3D,
	VFLAT,
};

enum Direction
{
	VERTICAL,
	HORIZONTAL,
};

enum ParamRenderingStyle
{
	INDEXBASED,
	GROUPED,
};

enum DateFilterOperatorTypes
{
	DT_BETWEEN,
	DT_GREATER,
	DT_LESS,
	DT_EQUAL,
	DT_LESSOREQUAL,
	DT_GREATEROREQUAL,
	DT_NOTEQUAL,
	DT_ISNULLORBLANK,
	DT_LAST,
	DT_CURRENT,
	DT_NEXT,
};

enum NumberFilterOperatorTypes
{
	NM_GREATER,
	NM_LESS,
	NM_EQUAL,
```

```
        NM_LESSOREQUAL,
        NM_GREATEROREQUAL,
        NM_BETWEEN,
        NM_NOTEQUAL,
        NM_ISNULLORBLANK,
};

enum CharFilterOperatorTypes
{
        CH_EQUAL,
        CH_GREATER,
        CH_LESS,
        CH_LESSOREQUAL,
        CH_GREATEROREQUAL,
        CH_BETWEEN,
        CH_NOTEQUAL,
        CH_ISNULLORBLANK,
        CH_STARTS_WITH,
        CH_CONTAINS,
        CH_IN_LIST,
        CH_EQUALSIGNORECASE,
        CH_STARTSWITHIGNORECASE,
        CH_CONTAINSIGNORECASE,
};


enum enumMetaData_DataSetType
{
        EDST_CATEGORY,
        EDST_IRL,
        EDST_SAVEDREPORT,
        EDST_CATEGORYLIST,
        EDST_REPORTLAYOUT,
        EDST_REPORTLAYOUTLIST,
        EDST_SAVEDREPORTLIST,
        EDST_REPORT,
        EDST_UPX,
        EDST_CPX,
        EDST_RPX,
        EDST_RECORDSET,
        EDST_EDIT,
        EDST_PRINTERLIST,
        EDST_ERRORMAP,
        EDST_DYNAMICIRL,
        EDST_ORG,
        EDST_USER,
        EDST_ROLE,
        EDST_USERANDROLE,
        EDST_ACCESSRIGHT,
        EDST_MAPPING,
        EDST_DBCONNECTION,
        EDST_DBMETADATA,
        EDST_LOGIN,
        EDST_SECURITYMODE,
        EDST_AUDITLOG,
        EDST_AUDITEDREPORT,
        EDST_CL_GRANT,
```

```
        EDST_ORG_ADV_FEATURES,
        EDST_ORGCREDENTIALS,
        EDST_SELECT_OPTIONS,
        EDST_FONTDETAIL,
        EDST_SERVERPROP,
        EDST_LICENSEINFO,
        EDST_LICENSEREQINFO,
        EDST_PRINTSETTING,
        EDST_IRO,
        EDST_C_REGSTATELOOKUP,
        EDST_C_REGRECONFIG,
        EDST_POSTVIEW,
        EDST_COMMENT,
        EDST_CONNECTCLIENT,
        EDST_DISCONNECTCLIENT,
        EDST_CLIENTLICENSEEDETAILS,
        EDST_LICENSEDETAILS,
        EDST_CHART,
        EDST_TEMPLATE,
        EDST_RSSTATUS,
        EDST_IRO_PREFS,
        EDST_IRO_USER_PREFS,
        EDST_MYREPORTLAYOUTLIST,
        EDST_MYCATEGORYLIST,
        EDST_PUBLICTOMYREPORT,
        EDST_MYREPORTLAYOUT,
        EDST_MYCATEGORY,
        EDST_DASHBOARD,
        EDST_RECENTREPORTLIST,
        EDST_MYREPORT,
        EDST_DYNAMICARL,
        EDST_SCHEDULE_LIST,
        EDST_BATCH_LIST,
        EDST_JOB_LIST,
        EDST_SCHEDULE,
        EDST_BATCH,
        EDST_JOB,
        EDST_DEDICATEDJOB,
        EDST_HISTORYLIST,
        EDST_HISTORYDETAIL,
        EDST_FSDATAFILES,
        EDST_FSHEADER,
        EDST_REQUEST,
};

enum enumMetaData_OperationTypes
{
        EOT_REQUEST,
        EOT_RESPONSE,
};

enum enumMetaData_RequestOpCodes
{
        EROC_GETREPORT,
        EROC_GET,
        EROC_DELETE,
        EROC_REPLACE,
```

```
        EROC_ADD,
        EROC_EXEC,
        EROC_GETLIST,
        EROC_GETINFO,
        EROC_REGENERATE,
        EROC_GETUPX,
        EROC_GETCPX,
        EROC_GETRPX,
        EROC_GETIRL,
        EROC_UPDATE,
        EROC_LOOKUP,
        EROC_RECONFIG,
        EROC_PRINT,
        EROC_SAVE,
        EROC_TEST,
        EROC_GETLINE,
        EROC_ADDFILE,
        EROC_CANCEL,
        EROC_EMAIL,
        EROC_GETCONFIGFILE,
        EROC_COPY,
        EROC_MOVE,
        EROC_COPYLINK,
};

enum SysConstants
{
        BinaryVersion,
        ProtocolVersion,
        DEFAULT_DATE_FORMAT,
        REPORT_CLEAN_UP_TIME,
        REPORT_CLEAN_UP_INTERVAL,
        CLIENT_CLEAN_UP_INTERVAL,
        SCHEDULAR_INTERVAL,
        REPORT_SESSION_TIME_OUT,
        MSSQL_DB,
        ORACLE_DB,
        DATABASE_MODE,
        FILE_MODE,
        REMOTE_MODE,
        STREAM_MODE,
        FILE_SEPARATOR,
        POOL_NAME,
        STR_USERINFO,
        MIN_CHUNK_SIZE,
        DATE_SEPARATOR,
        DEFAULT_NULL_VALUE,
        defaultLogFileTrimSize,
        DEFAULT_REPORT_EXEC_PRIORITY,
        OpType_Seperator,
        SCKT_DC,
        SYS_PARAMS,
        REPORT_PARAMS,
        CONF_SYS_PARAMS,
        PAGE_SETTINGS,
        SESSION_ID,
        EDITABLE,
```

```
        EDIT_FIELDS,
        TOTAL_PAGES,
        REPORT_STREAMED,
        ACTION_CODE,/**/

        // SYSTEM PARAMETERS
        SYS_PARAMS_OPERATION_TYPE,
        SYS_PARAMS_OPERATION_CODE,
        SYS_PARAMS_REPORT_ID,
        SYS_PARAMS_CATEGORY_ID,
        SYS_PARAMS_REPORT_FORMAT,
        SYS_PARAMS_MENU_NAME,
        SYS_PARAMS_REPORT_NAME,
        SYS_PARAMS_PRINTER_NAME,
        SYS_PARAMS_PRINT_COPIES,
        SYS_PARAMS_PRINT_PAGES,
        SYS_PARAMS_OUTPUT_FILE_NAME,
        SYS_PARAMS_REPORT_CONN_NAME,
        SYS_PARAMS_REPORTOID,
        SYS_PARAMS_OUTPUT_MODE,
        SYS_PARAMS_TIME_STAMP,
        SYS_PARAMS_LAYOUT_OPERATION_TYPE,
        SYS_PARAMS_PRIORITY,
        SYS_PARAMS_FORMAT_PROPERTIES,

        SYS_PARAMS_EMAIL_TO,
        SYS_PARAMS_EMAIL_CC,
        SYS_PARAMS_EMAIL_BCC,
        SYS_PARAMS_EMAIL_SUBJECT,
        SYS_PARAMS_EMAIL_TEXT,
        SYS_PARAMS_EMAIL_FROM,
        SYS_PARAMS_EMAIL_REPLY_TO,

        SYS_PARAMS_SAVE_FILE_NAME,
        SYS_PARAMS_SAVE_SERVER,
        SYS_PARAMS_SAVE_FOLDER,
        SYS_PARAMS_OPERATION_PROPERTIES,
        SYS_PARAMS_LAST_ACCESS_TIME,
        SYS_PARAMS_STOP_RUNNING_REPORT,
        SYS_PARAMS_THREADLAGTIME,
        SYS_PARAMS_PAGE_COUNT,
        SYS_PARAMS_PRINTSETTINGNAME,
        SYS_PARAMS_REFRESH_DATA,
        SYS_PARAMS_IS_PUBLIC,
        SYS_PARAMS_ARL_DATA,
        //     New Added
        SYS_PARAMS_IMPLICIT_OPERATION,
        SYS_PARAMS_PREFETCH_DRILLDOWN,
};

enum PromptPosition
{

        ePromptPos_TOP,
        ePromptPos_LEFT,
};
```

```
enum REPORTFORMAT
{
	// public static class ReportFormats
	reccFormatIntera,
	reccFormatPDF,
	reccFormatHTML,
	reccFormatMHTML,
	reccFormatExcel,
	reccFormatText,
	reccFormatRTF,
	reccFormatTIF,
	reccFormatPPT,
	reccFormatDOC,
	reccFormatCSV,
	reccFormatXML
};

enum OPERATION_TYPE
{
	reccOperationView,
	reccOperationSave,
	reccOperationPrint,
	reccOperationPrintAtServer,
	reccOperationEMail
};

enum PDFPROPS
{
	reccPDFVersion,
	reccPDFJPGQuality,
	reccPDFShowBookMarks,
	reccPDFSemiDelimetedNeverEmbedFonts,
	reccPDFMultipageOutput
};

enum EXCELPROPS
{
	reccExcelVersion,
	reccExcelMultiSheet,
	reccExcelDoubleBoundaries,
	reccExcelMinColumnWidth,
	reccExcelMinRowHeight,
	reccExcelGenPageBreaks,
	reccExcelShowMarginSpace,
	reccExcelTrimEmptySpace,
	reccExcelBorderSpacing,
	reccExcelAutoRowHeight,
	reccExcelShowPageHeadersAndFooters,
	reccExcelRepeatPageHeadersAndFooters
};

enum MHTMLPROPS
{
	reccMHTMLCreateCSSFile,
	reccMHTMLVersion,
	reccMHTMLTableOfContents,
```

```
        reccMHTMLJPEGQuality,
        reccMHTMLMultiPageOutput,
        reccMHTMLTitle,
        reccMHTMLShowTOC
};

enum HTMLPROPS
{
        reccHTMLCreateCSSFile,
        reccHTMLVersion,
        reccHTMLTableOfContents,
        reccHTMLJPEGQuality,
        reccHTMLMultiPageOutput,
        reccHTMLTitle,
        reccHTMLShowTOC,
        reccHTMLShowToolBar,
};

enum TEXTPROPS
{
        reccTextTextDelimeter,
        reccTextSuppressEmptyLine,
        reccTextUNICODE,
        reccTextPageDelimiter
};

enum CSVPROPS
{
        reccCSVColumnSeperator,
};

enum enumSaveReportExpiryPeriodType
{
        ESREP_BY_EXPIRYTIME,
        ESREP_BY_SAVEINTERVAL,
        ESREP_BY_SAVEENDPERIOD,
};

enum enumOperations
{
        EO_EMAIL,
        EO_PRINT,
        EO_FTP,
        EO_PUBLISH,
};
```

4

RComClient

This document is a detailed description of the API provided by ReCOmCLient.dll.

Categories of APIs

Layout Management

This category of tasks constitutes the tasks that are required to handle the user interface. For example, getting a list of all report categories available, adding, deleting or replacing these categories, getting a list of all reports or getting a list of reports belonging to a particular category etc.

Report Execution

This category of tasks deals with the operations that are performed when a particular report has been selected for execution. Execution is not just a one step process but involves a sequence of procedures to be followed. At certain situations, decisions also have to be made.

User Role Organization Management

This category of tasks involves the querying of organization lists, user lists, and role lists etc. It helps in finding out the access rights of different users and roles and performing managerial actions (like adding, updating deleting) on organizations, users and roles.

Database Management

This category provides functionality that helps in interacting with the engine for operations pertaining to databases. For example: getting a list of database connections supported by the engine, querying the metadata of these databases, adding a database connection, executing commands etc. This helps in reducing the processing at the client and bringing the execution logic to the engine.

Other categories

- AccessRights Management
- Report Utility Classes

- System Files Management
- Session Level Management
- Published Reports
- Report Object Management
- License Management
- Chart Services

The Basics

There are a couple of things that need to be done no matter what use-case you intend to use. They include the functioning of the following classes:

1. IEngineObjectFactory
2. ILayoutManager
3. IUserSession
4. ISecurityManager
5. IUserMgmt
6. IReportExecuter
7. IDBMgmt
8. IChartService
9. ILicenseManager
10. IReportObjectsManager
11. IReportUtility

IEngineObjectFactory

The interface "**IEngineObjectFactory**" is the beginning of all things. This interface has to be created first from the dll. As its name suggests, its responsibility is to create the instances of all objects that the user might not be able to correctly create. It has the following functions:

Initialize ([in] VARIANT *pConfigFilePath)

This function has to be called with a stirng parameter containing the IP address and port where Report server is running. Syntax for this string is "IP:PORT". If ("") NULL string is passed then corresponding IP address and port will be fetched from registry.

LayoutManager ([in] IUserInfo* pUserInfo, [out, retval] ILayoutManager* *pVal)

This is a "get" property, which returns an object for the interface ILayoutManager. I.e. the decision, which class to instantiate for the

ILayoutManager is made by the object factory. This call requires the UserInfo object for its internal communication with the report engine.

UserMgmt (UserInfo)

This is a "get" property, which returns an object for the interface IUserMgmt. I.e. the decision, which class to instantiate for the IUserMgmt is made by the object factory.

ReportExecuter ([in] IUserInfo* pUserInfo, [out, retval] IReportExecuter* *pVal)

This is a "get" property, which returns an object for the interface IReportExecuter. I.e. the decision, which class to instantiate for the IReportExecuter is made by the object factory.

DBMgmt ([in] IUserInfo* pUserInfo, [out, retval] IDBMgmt* *pVal)

This is a "get" property, which returns an object for the interface IDBMgmt. I.e. the decision, which class to instantiate for the IDBMgmt is made by the object factory.

ChartService([in] IUserInfo* pUserInfo, [out, retval] IChartService* *pVal)

This is a "get" property, which returns an object for the interface IChartService. This call requires the UserInfo object for its internal communication with the report engine.

LicenseManager(IUserInfo *pUserInfo, [out, retval] ILicenseManager* *pVal)

This is a "get" property, which returns an object for the interface ILicenseManager. This call requires the UserInfo object for its internal communication with the report engine.

ReportObjectsManager([in]IUserInfo *pUserInfo, [out, retval] IReportObjectsManager **pVal)

This is a "get" property, which returns an object for the interface IReportObjectsManager. This call requires the UserInfo object for its internal communication with the report engine.

ReportUtility([in]IUserInfo* pUserInfo, [out, retval] IReportUtility **pVal)

This is a "get" property, which returns an object for the interface IReportUtility. This call requires the UserInfo object for its internal communication with the report engine.

SecurityManager(IUserInfo* pUserInfo, [out, retval] ISecurityManager* *pVal)

This is a "get" property, which returns an object for the interface ISecurityManager. This call requires the UserInfo object for its internal communication with the report engine.

UserSession ([out, retval] IUserSession* *pVal)

This is a "get" property, which creates an object for the interface IUserSession and returns the object.

SystemFilesManager ([in] IUserInfo* pUserInfo, [out, retval] ISystemFilesManager* *pVal)

This is a "get" property, which returns an object for the interface ISystemFilesManager. This call requires the UserInfo object for its internal communication with the report engine.

Layout Management

The list of interfaces provided for layout management is:

12. ILayoutManager
13. ICategories
14. ICategory
15. Ireport

ILayoutManager

This class provides different functions, which help in implementing each use case for Layout management.

CategoryList([out, retval] ICategories** pVal)

This is a "get" property, which returns ICategories object which contains a collection of different ICategory objects. This collection is index based starting from zero. Category object can be also obtained by Category's name. ICategories class provides necessary APIs for iterating through the collection.

AddCategory ([in] ICategory* pCategory)

This method adds a new category to Repository. It takes ICategory object as input parameter. This API prepares the request XML using ICategory object and sends it.

DeleteCategory ([in] BSTR bstrCatID, [in] VARIANT_BOOL Condition)

This get property takes two parameters. The category ID of the category to be deleted. This will be obtained from previous call to CategoryList. The condition is a Boolean value which indicates if the category has to be deleted unconditionally or not. If **yes** then the category will be deleted even if it contains reports inside it and if **no** then the category will be deleted only if it does not have any reports inside.

ReplaceCategory([in] BSTR bstrOldCategoryID,[in] BSTR bstrNewCatMenuName)

This get property essentially renames a category. The first argument specifies the Category ID of the category to be renamed (it will be obtained from a previous call to CategoryList). The second argument specifies the new name to be given to the category.

ReportLayoutList([out, retval] IReports** pVal)

This is a "get" property, which returns IReports object which contains a collection of different IReport objects. This collection is index based starting from zero. IReport object can be also obtained by Report's name. IReports class provides necessary APIs for iterating through the collection.

ReportListForCategory([in] BSTR bstrCategoryID, [out, retval] IReports** pVal)

This is a "get" property, which returns IReports object which contains a collection of different IReport objects. But the reports that are enumerated belong to the category whose ID is specified by the argument. This collection is index based starting from zero. IReport object can be also obtained by Report's name. IReports class provides necessary APIs for iterating through the collection.

AddReport([in] VARIANT vrReportID, [in] VARIANT vrCategoryID, [in] VARIANT vrMenuName, [in]VARIANT vrReportName, [in] VARIANT vrIRLData,[out,retval] VARIANT* pRetVal)

This function takes the IRL data and adds it as a report using the Report name mentioned. The menu name is the name to be displayed at the report selection menu (Database mode). Report ID is an optional parameter which the programmer can use if he wants an ID of his choice. If not, it should be left blank. In case the ID requested is already in use, an error will be thrown. The CatID specifies which category the report should be added to. The IRL is passed

as a string. This API returns the Report object for the report which has been uploaded.

DeleteReport([in] BSTR bstrReportID)

This function deletes the report of the ID that is passed as parameter. If the ID is not found then an exception is thrown.

ReplaceReport([in] VARIANT vrReportID, [in] VARIANT vrCategoryID, [in] VARIANT vrMenuName, [in]VARIANT vrReportName, [in] VARIANT vrIRLData)

This function takes IRL data which is used to replace another report. The report ID and category ID identify which report is to be replaced and the menu name is the new menu name of the report. The IRL is passed as a string.

GetReport(BSTR bstrReportID, BSTR bstrCategoryID, [out,retval]BSTR *vrIRLData)

This function returns the IRL data as string for the report identified by the given report Id and category Id passes as parameters.

XMLDataSourceMetaData([in] IXMLDataSource* pXMLDataSource, [out, retval] IXMLDataSourceMetaData* *pVal)

This property takes the object of IXMLDataSource as input. IXMLDataSource object contains necessary information about which type of XML DataSource Metadata is required. Using this information, Request is made to report server to fetch XML DataSource Metadata information and IXMLDataSourceMetaData class object is created and populated. This property returns this created object IXMLDataSourceMetaData.

get_Report(BSTR strReportID, [out, retval] IReport* *pVal)

This get Property fetches the information of report of given Report ID and then creates IReport object, populates it and returns the object.

GetCategoryListWithFilters ([in] IUserFilters* pUserFilters,[out,retval] ICategories** pCategories)

This API gives the category lists (ICategories object) on the basis of user provided filters passed as input parameter. pUserFilters is used to prepare the request XML. Request is made to report server, ICategories object is created and Response XML is populated using ICagetories object. If failed, an exception is thrown to host application.

GetReportLayoutListWithFilters([in] IUserFilters* pUserFilters, [out,retval] IReports** pReports)

This function gets IReports object on the basis of user provided filters passed as input parameter. pUserFilters is used to prepare the request XML. Requst is made to report server and IReports object is created and Response XML is populated using ICagetories object. If failed, an exception is thrown to host application.

AddReport2 ([in] IDispatch* pUserFilters,[in] BSTR strIRLData,[out,retval] IReport** pVal)

This function takes the IRL data and adds it as a report using the Report name mentioned. pUserFilter contains the filters. pUserFilters is a collection of UserFilter object. Each object is one filter. So one needs to create object for each filter that he wants to apply and add it to pUserFilters. Filter name is set in the 'Name' property of UserFilter class. It can have following values as string.

UserFilter.Name values for AddReort2
"REPORTID"
"CATEGORYID"
"MENUNAME"
"REPORTNAME"
"DEPL_TYPE"
"PRINTSETTINGNAME"
"FORMAT"
"PRIORITY"
"CACHE"
"CACHE_LEVEL"
"CACHE_FREQ"
"CACHE_EXP"
"CONNECTION"
"ISPUBLIC"
"ISHIDDEN"
"SRC_REPORTID"
"DEST_CATEGORYID"
"CONTENT_TYPE"
"DSGN_MODE"
"DESC"

"OWNERAPPID"
"OWNERORGID"

The menu name is the name to be displayed at the report selection menu (Database mode). Report ID is an optional parameter which the programmer can use if he wants an ID of his choice. If not, it should be left blank. In case the ID requested is already in use, an error will be thrown. The CatID specifies which category the report should be added to. The IRL is passed as a string. This API returns the Report object for the report which has been uploaded.

ReplaceReport2 ([in] IDispatch* pUserFilters,[in] BSTR strIRLData)

This function takes IRL data which is used to replace another report.
pUserFilter contains the filters. pUserFilters is a collection of UserFilter object. Each object is one filter. So one needs to create object for each filter that he wants to apply and add it to pUserFilters. Filter name is set in the 'Name' property of UserFilter class. It can have following values as string.

UserFilter.Name values for AddReort2
"REPORTID"
"CATEGORYID"
"MENUNAME"
"REPORTNAME"
"DEPL_TYPE"
"PRINTSETTINGNAME"
"FORMAT"
"PRIORITY"
"CACHE"
"CACHE_LEVEL"
"CACHE_FREQ"
"CACHE_EXP"
"CONNECTION"
"ISPUBLIC"
"ISHIDDEN"
"SRC_REPORTID"
"DEST_CATEGORYID"
"CONTENT_TYPE"
"DSGN_MODE"

"DESC"
"OWNERAPPID"
"OWNERORGID"

The report ID and category ID identify which report is to be replaced and the menu name is the new menu name of the report. The IRL is passed as a string.

Icategories

This class is basically meant for housekeeping the individual ICategory objects. It has two methods and two properties:

Category(VARIANT varCategory,ICategory* *pParam)

This is a get property which returns Category of category name passed as parameter.

CategoryNames(VARIANT *pVal)

This is a get property which returns an array of category names

Add(ICategory *pParam, VARIANT *key)

This Method adds Category object to internal collection of Category objects. When a category object is added to collection at the same time category name is added to a vector array which is used to get category object from index obtained from this vector array using category name.

Remove (VARIANT *Index)

This Method deletes Category object from specified Index. When object at this index is removed, at the same time category name of this object which is stored at this Index is also removed.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the corresponding ICategory object from specified index passed as input parameter.

Count

This property returns the total number of ICategory objects in its collection.

ICategory

This interface contains properties to obtain information about a category. The function CategoryList of ILayoutManager contains a list of such objects inside it's own collection.

Following properties and APIs are exposed from this interface.

SetID(BSTR bstrCatID)

This method sets the Category ID for this category.

SetName(BSTR bstrCatName)

This method sets the name of category.

ID

This property returns the category ID.

Name

This property returns the name of the category.

IsPublic

This Property Specifies that whether category is public or private. It can be set or retrieved.

OwnerAppID

This property sets or returns the Owner Application ID.

OwnerOrgID

This property sets or returns the Owner Organization ID.

IReports

This class is used for housekeeping of IReport objects. This class maintains the collection of IReport objects. The collection is based on zero based index. Report object can be obtained either by index or by Report name.

Add

This method adds an object of IReport to the internal collection of IReport objects.

Report

This "get" property returns the IReport object for a given report name.

ReportNames

This "get" property returns the Report names array for all the IReport objects.

Remove

This method removes the IReport object from the collection at specified index.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the corresponding IReport object from specified index passed as input parameter.

Count

This property returns the total number of IReport objects in its collection.

IReport

This interface contains properties to obtain information about the reports. The functions ReportLayoutList and ReportListForCategory of ILayoutManager return a list of such objects inside a dictionary. The interface has five get properties:

ID

This get property returns the Report Id.

Name

This get property returns the report name.

MenuName

This get property returns the report Menu name.

CatMenuName

This get property returns the menu name of the parent category.

CatID

This get property returns the ID of the parent category.

AccessRights

This property gets/sets the access rights for Report. To set access rights, it takes Access rights string as input.

Category_AccessRights:

This property gets/sets the category access rights. To set access rights, it takes Access rights string as input.

Date

This property gets/sets the date in string format.

Deployment_Type

This property gets/sets the Deployment_Type. The value of Deployement_Type can be either of the following from ReportDeployment_Type enum.

1. STANDARD
2. CUSTOM

Title

This property gets/sets the Title.

Date

This property gets/sets the date.

Description

This property returns/sets the Description of the report

DesignMode

This property gets/sets the DesignMode. The value of DesignMode can be either of the following from enumDesignMode enum.

3. EDM_STUDIO
4. EDM_ADHOC

IsAccessForCategory(long eAccessRightsForCategory)

This get property returns bool value specifying whether access right specified by eAccessRightsForCategory enum is valid or not.

IsAccessForReport(long eAccessRightsForReports)

This get property returns bool value specifying whether access right specified by eAccessRightsForReports enum is valid or not.

IsPublic

This get property returns bool value specifying whether report has public acess or private access.

ReportFormat

This get property returns the Report format.

SourceReportID

This get/set property sets or returns the Source Report ID which is used for the case of hyperlinked report.

SetCatID (BSTR bstrCatID)

This API sets the category ID.

SetCatMenuName (BSTR bstrCatMenuName)

This API sets the Category's menu name.

SetID(BSTR bstrID)

This API sets the ID of the report.

SetMenuName (BSTR bstrName)

This API sets the menu name of Report.

SetName (BSTR bstrName)

This API sets the name of the report.

Version

This property gets/sets the version.

Report Execution

This category of tasks helps in the correct execution of a report after the report has been selected by the user from the user interface.

The different interfaces provided are:

- IReportExecuter.
- ICustomParameters
- ISystemParameters
- IUserParameters
- IUserParameter
- IPredefs
- IreportPageSettings

IReportExecuter

This is the interface that is the most important for report execution. It encapsulates the entire logic and at times returns objects of the other interfaces mentioned above to provide and accept data from the user. The functions and properties provided are:

Initialize ([in]BSTR bstrReportID, [in]BSTR bstrCategoryID)

This is the first method that should be called in order to use the report executer object. The report executer needs to be initialized, the first stage of initialization is performed by the EngineObjectFactory itself when it returns the reference, however, the **user must call this** function with the proper values in order to complete the initialization. The **bstrReportID** and **bstrCategoryID** are, the report ID of the report to be executed and the ID of the category in which the report is contained, respectively.

CustomParameters([out, retval] ICustomParameters* *pVal)

This get property returns an object of the type ICustomParameters. See the explanation of this class for greater details.

ReportParameters([out, retval] IReportParameters* *pVal)

This get property returns an object of the type IReportParameters. See the explanation of this class for greater details.

StartExecution

This function starts the execution of the report that is identified by the reportID and catID parameters passed to the initialize function. More precisely, it sends the request XML to the engine and receives and parses the Page settings XML which can optionally be retrieved using PageSettings get property. If an error occurs, then it throws a COM exception.

Note that initialization, setting of user, custom and system Parameters should precede this call.

PageSettings([out, retval]IReportPageSettings* *pVal)

This get property returns an object of the type IPageSettings. See the explanation of this class for greater details.

Chunk ([out]VARIANT *pDataSize, [out, retval] VARIANT *pVal)

This is the get property that returns the data read from the engine. The data returned is in the form of a VARIANT containing a safe array. The safe array contains binary data that can directly be sent into the Intera Viewer. The method also takes another argument that is filled up within the function. This argument is returned with an **8 character** string which specifies the size of the chunk in bytes. The argument is of type variant which internally contains a BSTR.

The engine does not send the entire report in one go, but in chunks, thus this function has to be placed in a loop and called the right number of times in order to get the complete data. The safearray is of zero length when the data is complete. Thus, the user should place the Chunk function call in a loop that runs until the size of the safearray returned is zero.

ExecuteReport

This function is only for debugging purposes.

SetParameter (VARIANT vrName,BSTR value,[optional] VARIANT vrType)

It sets the value in system parameter if vrName's variant type is VT_I4. if vrName's variant type is VT_BSTR, then the value of UserParameter object is changed if it exists in UserParameters collection and if it does not exist then new UserParameter is created and it is added to UserParameters collection.

ICustomParameters

There are three categories of parameters that have to be passed on to the engine for report execution. Custom parameter is one of those three categories. This interface helps in managing these parameters.

The essential fact to remember is that the user cannot add any parameter of his choice but can only query the values of existing parameters or can modify their values.

The different functions and properties provided by this interface are:

ParamNames

This get property returns a list of all parameter names. The user can later on use each individual parameter name to query what the current value of that parameter is or use the parameter name to set its value. The variable returned is a VARIANT that contains a safe array. Or for all practical purposes, it returns a VARIANT containing an array of VARIANTS!

Value (VARIANT *ParamName)

This get property returns the current value of the specified parameter. Note that the parameter name should belong to the list of names returned by the ParamNames property or should be a name confirmed using the Exists property. An invalid parameter name will cause an exception.

PutValue (VARIANT ParamName, VARIANT* Value)

This is a **method** and not a property. This method takes two arguments, one to specify which parameter we want to set and the second specifies the value that the given parameter has to be set to. The value must be a string. Again, all the notes given about the parameter name in the above paragraph apply here as well.

Exists (VARIANT *ParamName)

This get property ascertains whether a particular string is a parameter in the list of Custom parameters. The string that has to be confirmed is passed as an argument and a VARIANT containing a Boolean answer is returned.

GetXML

This function reads the dictionary and creates a string that can be inserted into the XML request for Execution of a report.

IReportPageSettings

This interface helps in maintaining the page settings pertaining to a report after its execution. This object is filled upon receiving the first response after report execution and is optional to use.

Properties are:

PaperBin

This is the get property which returns the member variable for paper bin.

TopMargin

This is the get property which returns top margin.

RightMargin

This is the get property which returns Right margin.

PaperWidth

This is the get property which returns paper width.

PaperSize

This is the get property which returns paper size.

PaperHeight

This is the get property which returns paper height.

Orientation

This is the get property which returns orientation.

MirrorMargins

This is the get property which returns mirror margin.

LeftMargin

This is the get property which returns Left margin.

Gutter

This is the get property which returns Gutter.

Duplex

This is the get property which returns Duplex.

Collate

This is the get property which returns Collate.

BottomMargin

This is the get property which returns Bottom Margin.

All these are get properties that return VARIANTS corresponding to the attribute of page settings that the property name suggests.

User/Role/Organization Management

The user management use cases are implemented using the following set of interfaces:

5. IUserMgmt
6. IOrganizations
7. IOrganization
8. IUsers
9. IUser
10. IRoles
11. IRole
12. IOrgCredentials
13. IOrgCredential
14. ISecurityManager

ISecurityManager

This class does user management related stuffs and responsible for managing users, roles and privileges related stuffs. This class adds, deletes and replaces users, roles and privileges.

AddMapEntities

This API maps Application user to Intellicus User in organization.

AddRole

This API adds a single role to repository. It takes IRole object which is to be added.

AddRoles

This API adds multiple roles to repository at one time.It takes IRoles object which contains a collection of IRole objects.

AddUser

This API adds a single user to repository. It takes IUser object which is to be added.

AddUsers

This API adds multiple users to repository at one time.It takes IUsers object which contains a collection of IUser objects.

AssignCategoryPrivilegesToRole

This API assigns category privileges to role. This API takes IRole object to which privileges are to be assigned, category ID of category whose privileges has to be assigned, accessLevel which can take one of the following value mentioned below and accessRights which can take any combination of values from enum AccessRightsForCategory mentioned below.

enum AccessLevels

- DENYACCESS
- FULLACCESS
- PARTIALACCESS

enum AccessRightsForCategory

- VIEW_REPORTS
- VIEW_REPORTS_SECURED
- SAVE_REPORTS
- SAVE_REPORTS_SECURED
- EXPORT_REPORTS
- EXPORT_REPORTS_SECURED
- PRINT_REPORTS

- PRINT_REPORTS_SECURED
- PRINT_REPORTS_AT_SERVER
- PRINT_REPORTS_AT_SERVER_SECURED
- SCHEDULED_REPORTS
- PUBLISH_LAYOUTS
- PUBLISH_OUTPUTS

AssignCategoryPrivilegesToUser

This API assigns category privileges to user. This API takes IUser object to which privileges are to be assigned, category ID of category whose privileges has to be assigned, accessLevel which can take one of the following value mentioned below and accessRights which can take any combination of values from enum AccessRightsForCategory mentioned below.

enum AccessLevels

- DENYACCESS
- FULLACCESS
- PARTIALACCESS

enum AccessRightsForCategory

- VIEW_REPORTS
- VIEW_REPORTS_SECURED
- SAVE_REPORTS
- SAVE_REPORTS_SECURED
- EXPORT_REPORTS
- EXPORT_REPORTS_SECURED
- PRINT_REPORTS
- PRINT_REPORTS_SECURED
- PRINT_REPORTS_AT_SERVER
- PRINT_REPORTS_AT_SERVER_SECURED
- SCHEDULED_REPORTS
- PUBLISH_LAYOUTS
- PUBLISH_OUTPUTS

AssignReportPrivilegesToRole

This API assigns Report privileges to role. This API takes IRole object to which privileges are to be assigned, Report ID of Report whose privileges has to be assigned, Category ID of Cateogory to which report belongs and accessRights which can take any combination of values from enum AccessRightsForReports mentioned below.

Enum AccessRightsForReports

- VIEW_REPORT
- VIEW_REPORT_SECURED
- SAVE_REPORT
- SAVE_REPORT_SECURED
- EXPORT_REPORT
- EXPORT_REPORT_SECURED
- PRINT_REPORT
- PRINT_REPORT_SECURED
- PRINT_REPORT_AT_SERVER
- PRINT_REPORT_AT_SERVER_SECURED
- PUBLISH_OUTPUT

AssignReportPrivilegesToUser

This API assigns Report privileges to user. This API takes IUser object to which privileges are to be assigned, Report ID of Report whose privileges has to be assigned, Category ID of Cateogory to which report belongs and accessRights which can take any combination of values from enum AccessRightsForReports mentioned below.

Enum AccessRightsForReports

- VIEW_REPORT
- VIEW_REPORT_SECURED
- SAVE_REPORT
- SAVE_REPORT_SECURED
- EXPORT_REPORT
- EXPORT_REPORT_SECURED
- PRINT_REPORT
- PRINT_REPORT_SECURED
- PRINT_REPORT_AT_SERVER

- PRINT_REPORT_AT_SERVER_SECURED
- PUBLISH_OUTPUT

CreateAsRole

This API creates a new role from an existing role. Parameters are described as below which are passed as input to this API.

- bstrDRoleID – Destination role ID
- bstrDRoleName – Destination Role Name
- bstrDOrgID – Destination Organization ID
- bstrDDesc – Destination Role Description
- bstrSRoleID – Source Role ID
- bstrSOrgID – Source Organization ID

CreateAsUser

This API creates a new user from an existing user. Parameters are described as below which are passed as input to this API.

- bstrDUserID– Destination user ID
- bstrDUserName– Destination User Name
- bstrDPasswd- Destination Password.
- bstrDOrgID – Destination Organization ID
- bstrDDesc – Destination User Description
- bstrSUserID– Source User ID
- bstrSOrgID – Source Organization ID

DeleteMapEntities

This API deletes mapping between Application User ID and Intellicus User ID for the organization specified by bstrOrgID.

DeleteRole

This API deletes a single role from repository. It takes IRole object which is to be deleted.

DeleteRoles

This API deletes multiple roles from repository at one time.It takes IRoles object which contains a collection of IRole objects.

DeleteUser

This API deletes a single user from repository. It takes IUser object which is to be deleted.

DeleteUsers

This API adds multiple users from repository at one time.It takes IUsers object which contains a collection of IUser objects.

ReportServerConf

This get property returns the object of IReportServerConf. This API retrieves the report server configuration information from report server and populates IReportServerConf object.

AccessRightsManager

This get property returns the object of IAccessRightsManager. For details, see IAccessRightsManager class section.

GetOrganizationList

This get property returns the object of IOrganizations object. This API retrieves the organization list from repository through report server and populates IOrganizations object which is a collection of IOrganization objects.for details, see IOrganizations class section.

GetOrgsWithCredentials

This get property returns the object of IOrganizations object which contains organization credential details also. This API retrieves the organization list with credential information from repository through report server and populates IOrganizations object which is a collection of IOrganization objects.for details, see IOrganizations class section.

GetRoleList

This get property returns the object of IRoles object. This API retrieves the all the roles from repository through report server and populates IRoles object which is a collection of IRole objects.for details, see IRoles class section.

GetRoleListForOrg

This get property returns the object of IRoles object. This API retrieves the roles from repository through report server for specified organization of organization id which is passed as input parameter and populates IRoles object which is a collection of IRole objects.for details, see IRoles class section.

GetUserList

This get property returns the object of IUsers object. This API retrieves the all the users from repository through report server and populates IUsers object which is a collection of IUser objects.for details, see IUsers class section.

GetUserListForOrg

This get property returns the object of IUsers object. This API retrieves the users from repository through report server for specified organization of organization id which is passed as input parameter and populates IUsers object which is a collection of IUser objects.for details, see IUsers class section.

RenameRole

This API renames role ID of role of specified organization. This API takes following parameters described as below.

- bstrSrcID – Source Role ID
- bstrOrgID – Organization ID of organization to which role belongs.
- bstrDestID – new Role ID which is to be assigned.

RenameUser

This API renames user ID of role of specified organization. This API takes following parameters described as below.

- bstrSUserID– Source User ID
- bstrOrgID – Organization ID of organization to which role belongs.
- bstrDUserID– new User ID which is to be assigned.

ReplaceMapEntities

This API modifies mapping between Application user and Intellicus User of specified Organization.

ReplaceRole

This API replaces a single role onto repository with passed new one. It takes IRole object which is to be deleted.

ReplaceRoles

This API replaces multiple roles onto repository at one time. It takes IRoles object which contains a collection of IRole objects.

ReplaceUser

This API replaces a single user from repository with passed new one. It takes IUser object which is to be deleted.

ReplaceUsers

This API replaces multiple users onto repository at one time. It takes IUsers object which contains a collection of IUser objects.

IUserMgmt

This is the central interface for the user management set of use cases. Its instance can be obtained from the engine object factory using the UserMgmt get property. The UserAuthorization object will have to be passed. The functions and properties provided are:

OrganizationList

This get property returns an object of type IOrganizations which in turn keeps a set of IOrganization objects, each of which stores information about the different organizations supported by the engine database.

UserList

This get property returns an object of type IUsers which in turn keeps a set of IUser objects which maintain information about individual users registered on the database.

RoleList

This get property returns an object of type IRoles which in turn keeps a set of IRole objects which maintain information about individual roles added in the database.

IOrganizations

This class is basically meant for housekeeping the individual IOrganization objects. It has two methods:

OrganizationNames

This get property returns an array of VARIANTS that contain different organizations names. These organization names will be used as keys to obtain the actual IOrganization object from this object.

Organization (BSTR bstrOrgName)

This get property takes an Organization name (as string) as a parameter and returns the corresponding IOrganization object.

Add

This Method adds new organization to it's internal collection of objects to add the organization to repository. It takes Organization object as input parameter.
User needs to call commit() for final transaction.

Remove

This Method adds organization to it's internal collection of objects to remove the organization from repository. It takes Organization object as input parameter.
User needs to call commit() for final transaction.

Replace

This Method adds organization to it's internal collection of objects to replace the organization onto repository. It takes Organization object as input parameter.
User needs to call commit() for final transaction.

Commit

This method commits the final transaction to add, delete, replace the organizations onto repository. This method prepares request XML with necessary details of organizations to be added,deleted and replaced and sends it to server.

IOrganization

This class is meant for storing organization information. Following properties or APIs are exposed from this interface.

AuthenticateMode

This Get/Set property sets or returns the Authentication mode. Authentication mode can be either of the following from enum enumAuthenticationMode.

- AUTH_INTELLICUS
- AUTH_EXTERNAL

- AUTH_HOST
- AUTH_CALLBACK

Authentication

This Get property returns Authentication object. For details, see Authentication class section.

AuthenticationDetail

This Get/Set property returns or sets the AuthenticationDetail string.

Authorization

This Get property returns the Authorization object. For details, see Authorization class section.

AuthorizationDetail

This Get/Set property returns or sets the AuthorizationDetail string.

AuthorizeMode

This Get/Set property returns or sets the AuthorizeMode string.

CanDelete

This Get/Set property returns or sets the bool value specifying if the organization can be deleted or not. If user exists in that organization then this organization can not be deleted. In this case, The value will be false.

Description

This Get/Set property returns or sets the Description string.

GlobalFilter

This Get property returns GlobalFilter object. For details, see GlobalFilter class section.

IsDefault

This Get/Set property returns or sets bool value specifying is this organization is default organization.

IsFilterApplicable

This Get/Set property returns or sets bool value specifying is filter is applicable for this organization.

MapType

This Get/Set property returns or sets the map type.

Name

This Get/Set property returns or sets the organization's name.

OrgCredentials

This Get/Set property returns or sets OrgCredentials object. For details, see OrgCredentials class section.

OrgID

This Get/Set property returns or sets the organization's ID.

PwdExpiration

This Get/Set property returns or sets the Password expiration.

PwdLength

This Get/Set property returns or sets the password length.

PwdNeverExpire

This Get/Set property returns or sets the Bool value specifying whether password will expire or it will be for life time**.**

SpecialPwd

This Get/Set property returns or sets bool value specifying whether password is special one or not.

IAuthentication

IAuthentication class contains the information related to authentication required for organization.

This class provides the following properties:

ExternalAppAttribs

This is a "get" property that provides the object of IExternalAppAttribs.

CallBackAttribs

This is a "get" property that provides the object of ICallBackAttribs. For Details, see ICallBackAttribs class section.

AuthenticaionMode

This is a get and set property which returns the authentication mode, like Intellicus, External, Host or Callback Mechanism. This property can have one of the following value from enum 'enumAuthenticationMode'.

- AUTH_INTELLICUS
- AUTH_EXTERNAL
- AUTH_HOST
- AUTH_CALLBACK

IAuthorization

This class provides the following properties:

AuthorizationMode

This get/set property returns or sets the authorization mode. Currently authorization mode is available through Intellicus. This property can have value from one of the following values from enum 'enumAuthorizationMode'

- AUTHORIZATION_INTELLICUS = 1

IGlobalFilter

This Class contains information related to filtering purposes. This class provides the following properties:

CallBackAttribs

This is a "get" property that provides the object of ICallBackAttribs. For details, see ICallBackAttribs class section.

SystemFilterAttribs

This is a "get" property that provides the object of ISystemFilterAttribs. For details, see ISystemFilterAttribs class section.

GlobalFilterMode

This property sets or returns the Global filter mode. This property can have value from one of the following values from enum 'enumGlobalFilterMode'.

- GFT_INTELLICUS
- GFT_CALLBACK

IsApplicable

This get/set bool property specifies whether the global filter applied or not.

ICallBackAttribs

This class is basically meant for housekeeping the individual ICallBackAttrib objects. It maintains a collection of the ICallBackAttrib objects. This class provides the following properties:

CallbackType

This property returns the callback type, like Local, Socket, RMI. This property can have value from one of the following values from enum 'enumCallbackType'.

- CALLBACK_LOCAL
- CALLBACK_SOCKET
- CALLBACK_RMI

CallBackAttrib

This is a "get" property that returns the object of ICallBackAttrib. For details, see ICallBackAttrib class section.

ImplementerAttribs

This is a "get" property that returns the object of IImplementerAttribs. For Details, See ImplementerAttribs class section.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ICallBackAttrib object from specified index.

Count

This property returns the total number of ICallBackAttrib objects in its collection.

This class provides one API:

Add

This method adds the ICallBackAttrib object to the collection of ICallBackAttrib objects. CallBackAttrib's name is used as key in the collection.

ICallBackAttrib

This interface contains properties to obtain information about the callback attributes. It has following get and set properties:

Name

This is only a "get" property which returns the name of the attribute.

Value

This property sets/returns the value of attribute.

Type

This property sets/returns the type of the attribute. This property can have value from one of the following values from enum enumCallBackAttrib.

- CBA_SERVER_IP
- CBA_SERVER_PORT

IExternalAppAttribs

This class is basically meant for housekeeping the individual IExternalAppAttrib objects. It maintains a collection of the IExternalAppAttrib objects. This class provides the following properties:

ExternalAppType

This property sets/returns the External application type, like WinNT, UNIX, DAP or NDIS. This property can have one of the following values from enum enumAuthExtAppType

- AUTH_EXT_APP_WINNT
- AUTH_EXT_APP_UNIX
- AUTH_EXT_APP_LDAP
- AUTH_EXT_APP_NDIS

ExternalAppAttrib

This is a "get" property that returns the object of IExternalAppAttrib. For Details, see IExternalAppAttrib class section.

Add

This method adds the IExternalAppAttrib object to the collection of IExternalAppAttrib objects. ExternalAppAttrib's name is used as key in the collection.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IExternalAppAttrib object from specified index.

Count

This property returns the total number of IExternalAppAttrib objects in its collection.

IExternalAppAttrib

This interface contains properties to obtain information about the callback attributes. It has following get and set properties:

Name

This is only a "get" property which returns the name of the attribute.

Value

This property sets/returns the value of attribute.

Type

This property sets/returns the type of the attribute. This property can have one of the following values from enum 'enumExternalAppAttrib'.

- EAA_SERVER_IP
- EAA_SERVER_PORT
- EAA_USERDN

IImplementerAttribs

This class is basically meant for housekeeping the individual IImplementerAttrib objects. It maintains a collection of the IImplementerAttrib objects. This class provides the following properties:

ImplementerType

This property returns the Implementer attribute type, like Java Class, Native Lib, Com DLL. This property can have one of the following values from enum enumImplementerType.

- IMPL_JAVA_CLASS
- IMPL_NATIVE_LIB
- IMPL_COM_DLL

ImplementerAttrib

This is a "get" property that returns the object of IImplementerAttrib. For Details, see IImplementerAttrib class section.

Add

This method adds the IImplementerAttrib object to the collection of IImplementerAttrib objects and also to IndexNameMap where the name and object are mapped from collection.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the corresponding item.

Count

This property returns the total number of objects in its collection.

IImplementerAttrib

This interface contains properties to obtain information about the Implementer attributes. It has following get and set properties:

Name

This is only a "get" property which returns the name of the attribute.

Value

This property returns the value of attribute.

Type

This property returns the type of the attribute. This property can have one of the following values from enum enumImplementerAttrib

- IA_PATH

ISystemFilterAttribs

This class is basically meant for housekeeping the individual ISystemFilterAttrib objects. It maintains a collection of the ISystemFilterAttrib objects. This class provides the following properties:

SystemFilterName

This property sets/returns the system filter name.

SystemFilterAttrib

This is a "get" property that returns the object of ISystemFilterAttrib. For Details, see ISystemFilterAttrib class section.

SelectOptions

This is a "get" property that returns the object of ISelectOptions. For Details, see ISelectOptions class section.

Add

This method adds the ISystemFilterAttrib object to the collection of ISystemFilterAttrib objects. SystemFilterAttrib's name is used as key in the collection.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ISystemFilterAttrib object from specified index.

Count

This property returns the total number of ISystemFilterAttrib objects in its collection.

ISystemFilterAttrib

This interface contains properties to obtain information about the System filters. It has following get and set properties:

Name

This is only a "get" property which returns the name of the filter attribute.

Value

This property sets/returns the value of attribute.

Type

This property sets/returns the type of the filter attribute. This property can have one of the following values from enum enumSystemFilterAttrib.

- SFA_FILTER_COLUMN_NAME
- SFA_IGNORE_IF_NOT_PRESENT

IUsers

This class is basically meant for housekeeping the individual IUser objects. This class contains a collection of IUser objects. This class provides necessary interfaces to iterate through the IUser objects.

UserNames

This get property returns an array of VARIANTS that contain different users' names. These user names will be used as keys to obtain the actual IUser object from this object.

User

This get property takes a User name (as VARIANT) as a parameter and returns the corresponding IUser object.

Add

This API adds a User to Users Collection. User's Name is used as Key in the collection.

Remove

This API removes a user from the Users collection from the specified index

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IUser object from specified index.

Count

This property returns the total number of IUser objects in its collection.

IUser

This class is meant for storing user information. Functions provided are:

Description

This Get/Set property returns or sets the description of the user.

Is_Admin

This Get/Set property returns or sets bool value specifying whether this user is Admin or not.

Is_Password_Blank

This Get/Set property returns or sets bool value specifying whether password is blank for this user or not.

Is_SuperAdmin

This Get/Set property returns or sets bool value specifying whether this user is super admin or not.

OrgID

This Get/Set property returns or sets organization ID.

Password

This Get/Set property returns or sets the password.

RoleIDs

This Get/Set property returns or sets the Role IDs.

Status

This Get/Set property returns or sets the status.

System_Privileges

This Get/Set property returns or sets the system privileges string

UserID

This Get/Set property returns or sets the User ID.

IsFirstLogin

This Get/Set property returns bool value specifying whether this user has logged in for the first time or not. If for first time then he should be prompted to change the password. This property was used in earlier versions. Now this property has been deprecated and should not be used.

UserName

This Get/Set property returns or sets the user name.

IRoles

This class is basically meant for housekeeping the individual IRole objects. This class contains a collection of IRole objects. This class provides necessary interfaces to iterate through the IRole objects.

RoleNames

This get property returns an array of VARIANTS that contains different roles' names. These role names will be used as keys to obtain the actual IRole object from this object.

Role

This get property takes a role name (as VARIANT) as a parameter and returns the corresponding IRole object.

Add

This API Adds a role to Roles collection. Role's Name is used as key to add in the collection.

Remove

Remove a role object from Roles collection from the specified index.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IUser object from specified index.

Count

This property returns the total number of IUser objects in its collection.

IRole

This class contains role information. Following properties are exposed from this class.

Description

This Get/Set property returns or sets the description.

Is_Admin

This Get/Set property returns or sets the bool value specifying if the role for Admin.

Level

This Get/Set property returns/sets the role level.

OrgID

This Get/Set property returns/sets the organization ID.

RoleID

This Get/Set property returns/sets the role ID.

RoleName

This Get/Set property returns/sets the role name.

Status

This Get/Set property returns/sets the status.

TimeStamp

This get/set property returns/sets the timestamp

System_Privileges

This Get/Set property returns/sets the system privileges.

IOrgCredentials

This class is basically meant for housekeeping the individual IOrgCredential objects. This class contains a collection of IOrgCredential objects. This class provides necessary interfaces to iterate through the IOrgCredential objects.

Add

This method adds IOrgCredential object to internal collection. IOrgCredential's name is used as key in the collection.

OrgCredential

This get property retrieves the IOrgCredential object of specified name from collection.

OrgCredentialNames

This get property retrieves the array of name of organizational credential objects. This name can be used to retrieve individual IOrgCredential object.

Remove

This method removes IOrgCredential object from specified index.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IOrgCredential object from specified index.

Count

This property returns the total number of IOrgCredential objects in its collection.

IOrgCredential

This class is used to store the information about organizational credential.

DefaultValue

This get property gives the default value of organizational credential.

DisplayName

This get property gives the display name for organizational credential.

Name

This get property gives the name of organizational credential.

Type

This get property gives the type of organizational credential.

Values

This get property gives the value of organizational credential.

Database Management

Database management involves actions which are independent of all other aspects of reporting like layout management, report execution etc and deal only with database maintenance and access.

The various use cases belonging to this category of tasks are satisfied by:

- IDBMgmt
- IDBConnections
- IDBConnection
- IDBProvider

IDBMgmt

This is the primary interface for the database functionality. It contains methods for satisfying all use cases related to database operations. The methods/properties provided are:

Connections([out, retval] IDBConnections* *pVal)

This is a get property which does not take any argument. It makes a server trip and returns the details about the DB connections at the engine in the form of an object hierarchy. I.e. the information is returned in a hierarchical fashion. Calling this property returns an object of the interface IDBConnections (Note the 's'). The IDBConnections object contains multiple IDBConnection (Note: no 's') objects which in turn contain information about the individual connections. These individual IDBConnection objects can be retrieved from IDBConnections (mechanism will be discussed shortly).

DBMetadata ([in]VARIANT vrDBName, [in] long lMode, [out,retval] _Recordset* *pRecordset)

(Returns Recordset object)

This get property returns the metadata for the database specified by DBName. This metadata is put inside a hierarchical, shaped recordset. That is, this function call only returns the structure of the different tables/views etc stored on the engine configuration.

The recordset that is returned is called "Entity Recordset". It is named so because it contains brief details about the top level entities of a database; namely tables, views, stored procedures and Synonyms. The 'brief details' are name and type. The name is what the entity's identifier and the type helps us in determining which one of the mentioned entity types it is. Depending on what kind of object (entity) it is, the object will have a different set of details and hence we could not specify them in the Entity RS itself. However, there is a field called 'child' which contains details. This child is actually a sub-recordset. The structure of the child recordset for each object type is show in the diagram.

The child recordset contains finer details about the object, in case of tables and views these details will be columns and their specifications like datatype etc, in case of stored procedures the details will be parameter names and parameter specifications. Synonyms have not yet been implemented.
The child element in the child recordset is a sub-sub-recordset which will contain yet another level of details. I.e. the child recordset of table-details RS will be called 'column details RS" since it will contain column data information.
Thus the naming convention of the recordsets should also be clear by now. If not, then please read this part of the document again.
Additional details which may arise in the future; like security constraints, can be accommodated later thanks to the flexible structure.

ExecuteCmd ([in] VARIANT vrDBName, [in] VARIANT vrCmd,[in] VARIANT vrMaxRows, [out,retval] _Recordset** pRecordset)

This method remotely executes a query at the engine and returns the resultant recordset. As parameters, it requires the command (query/stored procedure) to be executed, the name of the database against which the command will be executed and maximum no of rows which should be returned as resultset. Value of vrMaxRows as 0 will give all the rows in the result set.
The recordset which is returned contains the data.

NewDBConnection([out, retval] IDBConnection* *pVal)

This function creates a new DBConneciton object, initializes it and returns it.

DBMetadataAndCache([in]VARIANT vrDBName, [in] long lMode, [in] BSTR strFilePath, [out, retval] _Recordset* *pVal)

This property is same like DBMetadata property except that it caches the Metadata on the local file system at the path specified by input parameter strFilePath and returns recordset containing DBMetaData information.

ReportParameters

This get/set property returns or sets the IReportParameters object.

LoadDBMetadataFromCache([in] BSTR strFilePath,[out,retval] _Recordset** pVal)

This API prepares the DBMetaData recordset using the cached value rather Then retrieving it through report server. DBMetaData is cached when DBMetadataAndCache is used to retrieve the metadata.

IDBConnections

This interface is returned by the 'Connections' get property of IDBMgmt. It is basically a container for house-keeping multiple DBConnection objects. It allows the programmer to retrieve one DBConnection object at a time. The DBConection

object in turn contains the further details about each connection. The properties/methods provided by this interface are:

DBConnectionNames

This get property returns an array containing the names of all connections. These connection names may be used for displaying at the UI or for retrieving a singe DBConnection object using the next property.

DBConnection (Connection Name)

This get property takes one argument. It takes a connection name and then returns a DBConnection object complete with all details for the specified connection.

Add

This function adds the input database connection object to the database connection pool.

Remove

This function removes the database connection object based on the input index.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IDBConnection object from specified index.

Count

This property returns the total number of IDBConnection objects in its collection.

IDBConnection

This is the object interface that contains details about a single connection. The DBConnections object seen above was basically a container and house-keeper for multiple DBConnection objects. Along with some atomic/primitive properties, this object further contains an object called IDBProvider. This DBProvider object contains details that are specific to the database provider. For example, Oracle requires a host name, port number and a TNS name, whereas ODBC simply requires a DSN string.

The get/set properties and methods provided by DBConnection are:

Name

This set of get/put properties help in retrieving or setting the name of the connection.

IsDefault

This set of get/put properties help in retrieving/setting a bool value, which determines whether the connection is the default connection or not.

IsRepository

This set of get/put properties help in retrieving/setting a bool value which determines if the connection in question will also serve as the repository database or not.

Provider

This get property returns a DBProvider object which stores provider specific information.

DBPassword

This get/set property returns / sets the Data Base password.

DBUserID

This get/set property returns/ sets the database user ID.

IsRunTimeCredential

This get property returns bool value which tells whether run time credentials required will be used for authentication or not. If this property is TRUE then RTUserID and RTPassword properties are used by IUserInfo class of REReqLib.

IsValid

This get/set property returns/sets the bool value which specifies whether database connection is valid or not.

TestDBConnction

This method tests the DB Connection using the information provided to the properties like DBUserID, DB connection name, DBPassword etc. If it is valid, this method just returns otherwise it throws exception to the host application.

IDBProvider

This interface house keeps information specific to the database provider. To keep the information extensible, the database parameters have not been coded as variables/properties. Instead, the different provider parameters exist as key-value pairs in a dictionary and this interface is essentially a wrapper over the dictionary. For example, lets say oracle type 4 provides 3 parameters that the user can use, namely, HOST, PORT and TNS. Then instead of creating different get/set properties (which would make the interface rigid) HOST and host name, PORT and the port number and TNS and TNS name exist as key value pairs inside the dictionary. The interface provides enumerated values which can be used to identify the type of the provider and it provides get/put properties which enable retrieval of parameters.

ProviderType

This set of get/put properties helps in retrieving/setting an enumerated integer which identifies the type of the provider. The enums are named as : PT_<xxx> where <xxx> is a placeholder for the provider type. These enums are exposed by ReCOMClient type library.

PropertyNames

This get property returns an array containing the different parameter names for the provider. These parameter names can be used for displaying on the UI or for getting/setting individual parameter values from the next get/put property.

PropertyValue(VARIANT property_name)

This set of get/put properties helps in retrieving/setting the value of a specific provider parameter ('property'). These properties take the name of the parameter ('property') to be used as a parameter.

DriverType

This get property returns the type of driver. This property can have one of the following values from enum enumDriverTypes:

- EODT_NONE
- EODT_ORACLE_OCI
- EODT_ORACLE_OCI_TNS
- EODT_ORACLE_THIN

DriverVersion

This get property return the version of driver.

AccessRights APIs

IAccessRights

This class is basically meant for housekeeping the individual IAccessRight objects. It maintains a collection of the IAccessRight objects. This class also provides necessary interfaces to iterate IAccessRights objects. Access rights can be applied for two type of entities. One is User and second is Role. It provides the following properties and APIs.

AccessRight

This is a "get" property that returns the object of IAccessRight. For details, see IAccessRight class section.

ID

This property returns/sets the ID of the access right.

Type

This property specifies the type of the access right, like user or role. This property can have one of the following values from enum enumAccessRightsType

- EART_USER
- EART_ROLE

OpCode

This property returns/sets the operation code, like "Add", "Replace" or "Delete". This opcode tells which operation to perform on access rights.

OrgID

This property returns/sets the organization ID.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IAccessRight object from specified index.

Count

This property returns the total number of IAccessRight objects in its collection.

IaccessRight

This interface contains properties to obtain information about the access right. It has following get and set properties:

EntityType

This property returns/sets the type of entity for which the access rights to be applied, like category, report or system. This property can have one of the following values from enum enumAccessEntityType

- EET_CATEGORY
- EET_REPORT
- EET_SYSTEM

CatID

This property returns/sets the category ID.

ReportID

This property returns/sets the report ID.

AccessLevel

This property returns/sets the access level as deny, full or partial access. This property can have one of the following values from enum AccessLevels

- DENYACCESS
- FULLACCESS
- PARTIALACCESS

IsAccessForCategory

This property returns bool value specifying whether access rights specified by parameter eCategoryPriveledges for category granted or not.

IsAccessForReport

This property returns bool value specifying whether access rights specified by parameter eReportPriveledges for report granted or not.

AccessRights

This property is a "put" property that is used to specify new access rights. Access rights are set in string comma separated format.

Report Utility classes

IReportUtility

This class provides the following get property:

get_SystemFiles([in] long eSystemFileType, [out, retval] ISystemFiles* *pVal)

This property returns the list of all system file corresponding to a file type specified by eSystemFileType from the report server. It provides a list of XML Data Source files from report server.

VerifyScript([in] BSTR bstrRequestData,[out,retval] BSTR* bstrResponseData)

This API verifies script. Script code is passed as parameter which is verified by report server. And response is sent back which contains information whether script is valid or not. If any error occurs, exception is thrown to host application.

System Files Management

ISystemFilesManager

This class basically manages System Files placed onto repository. It provides functionality to retrieve System Files. This class has following properties

SystemFilesList([in] long eSystemFileType, [out, retval] ISystemFiles* *pVal)

This get property returns the ISystemFiles object which contains the collection of ISystemFile objects. This ISystemFile object contains only name of system file. It does not contain any data associated with that system file. To retrieve the data use ' 'DetailedSystemFile' or 'DetailedSystemFiles' property. eSystemFileType can take one of the following values from enum enumSystemFileType.

- ESFT_ADHOC_TEMPLATE_IRL
- ESFT_ADHOC_TEMPLATE_STYLE
- ESFT_XML_DATASOURCE
- ESFT_CHART_TEMPLATE

DetailedSystemFile([in] long eSystemFileType, [in,out] ISystemFile* *pSystemFile)

This get property is meant to provide the details (file content) of system file. This property provides file content of single system file. It takes the system file type and SystemFile object (by reference) as input. One needs to provide system file name in 'SystemFileName' property of SystemFile object which is passed as parameter. This property provides the file content in the 'FileData' property of same SystemFile object which is passed as input parameter by reference. eSystemFileType can take one of the following values from enum enumSystemFileType.

- ESFT_ADHOC_TEMPLATE_IRL
- ESFT_ADHOC_TEMPLATE_STYLE
- ESFT_XML_DATASOURCE
- ESFT_CHART_TEMPLATE

DetailedSystemFiles([in] long eSystemFileType, [in,out] ISystemFiles* *pSystemFiles)

This get property is meant to provide the details (file content) of multiple system files at one time. It takes the system file type and SystemFiles object (by reference) as input. One needs to add SystemFile objects into SystemFiles collection whose contents need to be retrieved. SystemFile object which is being

added to collection should be with system file name set in 'SystemFileName' property of SystemFile object. file content in stored in the 'FileData' property of same SystemFile object. If user passes the object of SystemFiles without adding any SystemFile object to it, this property returns the SystemFiles object with collection of all the available system files of type eSystemFileType. eSystemFileType can take one of the following values from enum enumSystemFileType.

- ESFT_ADHOC_TEMPLATE_IRL
- ESFT_ADHOC_TEMPLATE_STYLE
- ESFT_XML_DATASOURCE
- ESFT_CHART_TEMPLATE

ISystemFiles

This class is basically meant for housekeeping the individual ISystemFile objects. It maintains a collection of the ISystemFile objects. It also provides necessary interfaces to iterate ISystemFile objects. It also provides the following properties:

SystemFileType

This property returns the file type. This property can have one of the following values from enum enumSystemFileType

- ESFT_ADHOC_TEMPLATE_IRL
- ESFT_ADHOC_TEMPLATE_STYLE
- ESFT_XML_DATASOURCE
- ESFT_CHART_TEMPLATE

SystemFile

This is a "get" property that returns the object of ISystemFile of the name mentioned by system file name.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ISystemFile object from specified index.

Count

This property returns the total number of ISystemFile objects in its collection.

Add(ISystemFile* pSystemFile)

This API adds a SystemFile object into the collection.

ISystemFile

This interface contains properties to obtain information about the system file. It has following get and set properties:

SystemFileName

This property returns/sets the system file name.

FileData

This get property returns the system file content in string format. This property is applicable only if ResponseCode is 0.

ResponseCode

This get property returns the response code. If response code is other than 0 then there must be some error while retrieving the system file. In this case, FileData would not contain any data, instead Message property would contain message.

Message

This get property returns the message. This property is applicable only if ResponseCode is other than 0.

IXMLDataSourceMetaData

This interface contains properties to obtain information about the XML data source metadata elements. It provides the following get property:

XMLDataSourceElements

This property returns the object of IXMLDataSourceElements. For details, see IXMLDataSourceElements class section.

IXMLDataSourceElements

This class is basically meant for housekeeping the individual IXMLDataSourceElement objects. It maintains a collection of the IXMLDataSourceElement objects. It also provides necessary interfaces to iterate IXMLDataSourceElement objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IXMLDataSourceElement object from specified index.

Count

This property returns the total number of IXMLDataSourceElement objects in its collection.

IXMLDataSourceElement

This interface contains properties to obtain information about the XML data source elements. It has following get and set properties:

ElementType

This property returns/sets the element type.

Path

This property returns/sets the path of the element.

Session Level Management

IUserSession

This class is responsible for maintaining user sessions. This class contains all the information required for maintaining user session.

AppID

This "get" property returns the application ID.

Description

This "get" property returns the description related to user session.

Is_Admin

This "get" property returns bool value true if this particular user session belongs to admin, false otherwise.

Is_Blank_Password

This "get" property returns bool value true if the password for this particular user session is blank, false otherwise.

Is_First_Login

This "get" property returns bool value true if this is the first login, false otherwise.

Is_SuperAdmin

This "get" property returns bool value true if this particular user session belongs to super admin, false otherwise.

OrgName

This "get" property returns the organization name for this user session.

RoleIDs

This "get" property returns the role IDs for this user session.

Status

This "get" property returns the status of this user session.

System_Privileges

This "get" property returns the system privileges for this user session.

Initialize

This is the method which should be used to intialize the object to hold the authentication user information. It takes IUserInfo object as parameter which contains authentication information. This property uses the information contained by IUserInfo object and verifies the information at the server. If information is valid, this API return TRUE, otherwise FALSE. Also this API initializes IUserPreferences object and populates it.

IsLoggedIn

This method checks if the user was authenticated at the server. If yes, it returns true, otherwise false.

IsPinging

This method pings to the server and checks if the engine is running. If yes, it returns true, otherwise false.

IsSecurityEnabled

This method verifies from engine that whether engine is running in the secured mode or not. If yes, it returns true, otherwise false.

LogOut

This function is purely logical in nature. It modifies the state of the object so that it behaves as if the user is not logged in. To log in again, the Initialize function must be called.

Report Object Management

IReportObjectsManager

This class is basically meant for managing the report objects. It provides the following properties:

ParameterObjects([out, retval] IParameterObjects* *pVal)

This get property returns the IParameterObjects. For details, see IParameterObjects class section.

QueryObjects([out, retval] IQueryObjects* *pVal)

This get property returns the object of IQueryObjects. IQueryObjects contains a collection of IQueryObject. This set of IQueryObject contains only basic information about the query object. For details, see IQueryObjects class section.

TemplateObjects([out, retval] ITemplateObjects* *pVal)

This get property is deprecated. One should not use this get property.

Query Object APIs

IqueryObjects

This class is used for housekeeping of IQueryObject objects. This class maintains the collection of IQueryObject. It also provides necessary interfaces to iterate IQueryObject objects. It provides the following properties:

Add

This method adds an object of IQueryObject to the internal collection of IQueryObject to add to the repository if BeginTrans() is called otherwise QueryObject is added to repository. To add multiple QueryObjects to repository at one time, one will need to first call BeginTrans, then add IQueryObject objects using Add API to collection and then call CommitTrans() to perform final transaction to repository.

BeginTrans

This API indicates that query objects have to be kept in a list instead of directly adding it to Repository.

CommitTrans

This API performs final transaction of adding or removing Query Objects to repository.

QueryObject

This "get" property returns the query object for a given name of QueryObject.

QueryObjectNames

This "get" property returns array of Query object names. These names can be used to retrieve IQueryObject object from collection.

Remove

This method adds an object of IQueryObject to the internal collection of IQueryObject to remove from the repository if BeginTrans() is called otherwise QueryObject is directly removed from repository. QueryObject is identified by it's name which is passed as parameter. To remove multiple QueryObjects from repository at one time, one will need to first call BeginTrans, then create list of IQueryObject objects using Remove API and then call CommitTrans() to perform final transaction to repository.

Update

This method adds an object of IQueryObject to the internal collection of IQueryObject to update on the repository if BeginTrans() is called otherwise QueryObject is directly updated on repository. To update multiple QueryObjects on repository at one time, one will need to first call BeginTrans, then create list of IQueryObject objects using Remove API and then call CommitTrans() to perform final transaction to repository.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IQueryObject object from specified index.

Count

This property returns the total number of IQueryObject objects in its collection.

IQueryObject

This class provides different functions, which help in implementing each use case for managing query objects.

Columns

This is a "get" property, which returns the column value.

ConnName

This property returns/sets the connection name.

ConnType

This property returns/sets the type of connection.

Description

This is a "get" property, which returns description.

ID

This is a "get" property, which returns the ID.

IsCached

This property returns/sets the bool value which indicates whether the data is cached or not.

Name

This is a "get" property, which returns the Name.

XML

This is a "get" property, which returns the XML.

Initialize

This method initializes the member variables of the IQueryObject class.

The class also provides the following get and set properties:

IsFilterMandatory

This property specifies whether the filter is mandatory for the query object or not.

SourceType

This property specifies the type of source for the query object (like SQL/XML). This property can have one of the following values from enum enumQueryObjectSourceType

- EQOST_SQL
- EQOST_XML

ISQLEditor

This class provides different functions regarding accessing and changing the SQL editor properties. It provides the following properties:

SQLEditorLayout

This is a "get" property which returns the object of ISQLEditorLayout. For details, see ISQLEditorLayout class section.

SQLClause

This is a "get" property which returns the object of ISQLClause. For details, see ISQLClause class section.

ISQLEditorLayout

This class provides different functions regarding the tables and linkers used. It provides the following get and set properties:

VScrollValue

This property returns/sets the vertical scrolling values for the layout.

HScrollValue

This property returns/sets the horizontal scrolling values for the layout.

Tables

This is a "get" property which returns the object of ITables. For details, see ITables class section.

Linkers

This is a "get" property which returns the object of ILinkers. For details, see ILinkers class section.

ITables

This class is basically meant for housekeeping the individual ITable objects. It maintains a collection of the ITable objects. It also provides necessary interfaces to iterate ITable objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IQueryObjectFilter object from specified index.

Count

This property returns the total number of IQueryObjectFilter objects in its collection.

ITable

This interface contains properties to obtain information about the table in sql editor layout. It has following get and set properties:

Name

This property returns/sets the name of table.

Left

This property returns/sets the left of table that is, starting x-value of table.

Top

This property returns/sets the top of table that is, starting y-value of table.

Height

This property returns/sets the height of table.

Width

This property returns/sets the width of the table.

Index

This get/set property specifies the index value of table.

ILinkers

This class is basically meant for housekeeping the individual ILinker objects. It maintains a collection of the ILinker objects. It also provides necessary interfaces to iterate ILinker objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ILinker object from specified index.

Count

This property returns the total number of ILinker objects in its collection.

ILinker

This interface contains properties to obtain information about the linker in sql editor layout. It has following get and set properties:

Source

This property returns/sets linker source.

Target

This property returns/sets linker target.

ISQLClause

This class provides different functions regarding accessing the sql query. It provides the following properties:

IstoredProcParameters

This is a "get" property which returns the object of IStroredProcParameters.

IoptionaFilters

This is a "get" property which returns the object of IOptionalFilters. For details, see IOptionalFilters class section.

Select

This property returns/sets the selection part of query like if query string is "SELECT emp.* ..", then this property returns the string "emp.*".

From

This property returns/sets the table name specifies in query from where the selection fields are to be retrieved.

WhereConditions

This property returns object of ISQLCLauseConditions. For details, see ISQLClauseConditions class section.

GroupBy

This property returns/sets the "group by" clause field of the query.

OrderBy

This property returns the "order by" clause field of the query.

HavingConditions

This property returns object of ISQLCLauseConditions. For details, see ISQLClauseConditions class section.

OrderByPrompt

This property returns/sets a comma separated string when multiple fields are to be specified for the order by clause.

IsShowOrderByPrompt

This property returns or sets bool value specifying whether multiple order by clause fields are specified.

OrderByPromptCount

This property returns/sets the number of fields specifies for the order by clause.

SQLType

This property returns the type of sql supplied for the getting the key field, like SQL or STOREDPROC. This property can have one of the following values from enum enumSQLType

- EST_SQL
- EST_STOREDPROC

SQL

This property returns the sql query for retrieving the key field.

ISQLClauseConditions

This class is basically meant for housekeeping the individual ISQLClauseCondition objects. It maintains a collection of the

ISQLClauseCondition objects. It also provides necessary interfaces to iterate ISQLClauseCondition objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the ILinker object from specified index.

Count

This property returns the total number of ILinker objects in its collection.

ISQLClauseCondition

This interface contains properties to obtain information about the conditions applied in the query used either in "where" clause or "having" clause. It has following get and set properties:

Open

This get/set property specifies the starting bracket of the condition.

Close

This get/set property specifies the closing bracket of the condition.

Operand1

This property returns/sets first operand involved in the condition.

Operand2

This property returns/sets second operand involved in the condition.

Operator

This property returns/sets operator acting between the two operands in query.

Relation

This property returns/sets relational operator like AND,NOT or OR in case of multiple conditions.

IStoredProcParameters

This class is basically meant for housekeeping the individual IStoredProcParameter objects. It maintains a collection of the IStoredProcParameter objects. It also provides necessary interfaces to iterate IStoredProcParameter objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IStoredProcParameter object from specified index.

Count

This property returns the total number of IStoredProcParameter objects in its collection.

IStoredProcParameter

This interface contains properties to obtain information about the stored procedures in sql editor layout. It has following get and set properties:

ParamName

This get/set property specifies the parameter name received from the stored procedure.

ParamValue

This get/set property specifies the parameter value received from the stored procedure.

IXMLDataSource

This class contains the properties to obtain the information about the XML data source. Hence it provides the following get and set properties:

XMLRuntimeSourceType

This property returns the data source type like file or url or database. This property can have one of the following values from enum enumXMLRuntimeSrcType

- EXRST_FILE
- EXRST_URL
- EXRST_DATABASE

File

This property returns/sets the filename for data source.

URL

This property returns/sets the URL path for the XML data source.

ConnectionName

This property returns/sets the name of the connection for the data source.

SQLQuery

This property returns/sets the SQL query for getting the data from it.

XMLColumn

This proprety returns/sets the xml column for selection.

RecordPattern

This property returns/sets the pattern of the tags in xml received.

IFormulas

This class is basically meant for housekeeping the individual IFormula objects. It maintains a collection of the IFormula objects. It also provides necessary interfaces to iterate IFormula objects. It provides the following properties:

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IFormula object from specified index.

Count

This property returns the total number of IFormula objects in its collection.

Formula

This property returns the object of IFormula from the collection using its name.

IFormula

This interface contains properties to obtain information about the formula fields. It has following get and set properties:

Name

This property returns/sets the name of the formula.

ReturnType

This property returns/sets the return type of the formula.

Expression

This property returns/sets the expression representing the formula.

Report Objects

IReportObjects

This class is basically meant for housekeeping the individual IReportObject objects. It maintains a collection of the IReportObject objects. It also provides necessary interfaces to iterate IReportObject objects. It provides the following properties:

Add

This API adds IReportObject object to the ReportObject Collection. For details, see IReportObject class section.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IReportObject object from specified index. For details, see IReportObject class section.

Count

This property returns the total number of IReportObject objects in its collection.

IReportObject

IReportObject class cotains the information of report object stored on repository. It is a super type of Query Object and Parameter Object. This class is used to get detailed information about query object or parameter object. This class prepares the request xml for the same.

ID

This get/set property returns/sets the report objedt ID.

IsCached

This get/set property returns/sets whether the report object is cached or not.

Name

This get/set property returns/sets the report object name.

Type

This get/set property returns/sets the report object type. Report object type can be “Query” or “Parameter”.

Parameter Objects

IParameterObjects

This class is basically meant for housekeeping the individual IParameterObject objects. It maintains a collection of the IParameterObject objects. It also provides necessary interfaces to iterate IParameterObject objects. It provides the following properties:

Add

This method adds an object of IParameterObject to the internal collection of IParameterObject to add to the repository if BeginTrans() is called otherwise parameter object is directly added to repository To add multiple parameter objects to repository at one time, one will need to first call BeginTrans(), then create list of IParameterObject objects using Add API and then call CommitTrans() to perform final transaction to repository.

BeginTrans

This API indicates that parameter objects have to be kept in a list instead of directly adding it to Repository.

CommitTrans

This API performs final transaction of adding or removing or updating parameter Objects to repository.

ParameterObject

This get property returns the object of IParameterObject of the parameter name passed as input parameter.

ParameterObjectNames

This get property returns the array of parameter names of parameter objects. These names can be used to retrieve the IParameterObject using the ParameterObject property.

Remove

This method adds an object of IParameterObject to the internal collection of IParameterObject to remove from the repository if BeginTrans() is called otherwise parameter object is directly removed from repository. parameter object is identified by it's name which is passed as parameter. To remove multiple parameter objects from repository at one time, one will need to first call BeginTrans, then create list of IParameterObject objects using Remove API and then call CommitTrans() to perform final transaction to repository.

Update

This method adds an object of IParameterObject to the internal collection of IParameterObject to update on the repository if BeginTrans() is called otherwise parameter object is directly updated on repository. To update multiple parameter objects on repository at one time, one will need to first call BeginTrans, then create list of IParameterObject objects using Update API and then call CommitTrans() to perform final transaction to repository.

_NewEnum

This property identifies an object as supporting iteration through the IEnumVARIANT interface.

Item

This property returns the IParameterObject object from specified index. For details, see IParameterObject class section.

Count

This property returns the total number of IParameterObject objects in its collection.

IParameterObject

This interface contains properties to obtain information about a parameter object. This class contains all the information related with a parameter object.

Description

This get property returns the description of the parameter object.

ID

This get property returns the ID of the parameter object.

Name

This get property returns the Name of the parameter object.

XML

This get/set property returns/sets the XML for parameter object.

Initialize

This API does the initialization activities necessary for parameter object.

ILicenseManager

This class manages the license related activities. Right now it conatains one property which gives the object of ILicenseDetails.

LicenseDetails

This get property returns the object of ILicenseDetails class. ILicenseDetails class is a member of REREQLIBLib library. For details, see ILicenseDetails class section.

IChartService

This class provides chart related services.

Chart

This get property returns the object of IChart class. For details, see IChart class section.

Themes

This get property returns the array of names of chart themes.

IChart

This interface contains properties to obtain information about a chart. This class contains all the information related with a chart.

Height

This get property returns the height of the chart.

ImageFormat

This get property returns the image format used for chart.

Name

This get property returns the name of the chart.

Picture

This get property returns the picture associated with the chart.

Width

This get property returns the width of the chart.

Enums Exposed from REComClient

```
enum ReportDeployment_Type
{
	STANDARD,
	CUSTOM,
};

enum enumDesignMode
{
	EDM_STUDIO = 0,
	EDM_ADHOC,
};

enum AccessLevels
{
	DENYACCESS,
	FULLACCESS,
	PARTIALACCESS,
};

enum OrgCredentialType
{
	CT_TEXT,
	CT_COMBO,
	CT_OPTION,
};

enum enumAccessRightsType
{
	EART_USER = 0,
	EART_ROLE
};

enum enumAccessEntityType
{
	EET_CATEGORY = 0,
	EET_REPORT,
	EET_SYSTEM
};


enum enumSystemPriviledges
{
	ESP_CATEGORY_SETUP   ,
	ESP_SCHEDULING       ,
	ESP_REPORT_DESIGNER  ,
};
```

```
enum AccessRightsForCategory
{
	VIEW_REPORTS,
	VIEW_REPORTS_SECURED,
	SAVE_REPORTS,
	SAVE_REPORTS_SECURED,
	EXPORT_REPORTS ,
	EXPORT_REPORTS_SECURED,
	PRINT_REPORTS,
	PRINT_REPORTS_SECURED,
	PRINT_REPORTS_AT_SERVER ,
	PRINT_REPORTS_AT_SERVER_SECURED,
	SCHEDULED_REPORTS,
	PUBLISH_LAYOUTS,
	PUBLISH_OUTPUTS,
};

enum AccessRightsForReports
{
	VIEW_REPORT,
	VIEW_REPORT_SECURED,
	SAVE_REPORT,
	SAVE_REPORT_SECURED,
	EXPORT_REPORT ,
	EXPORT_REPORT_SECURED,
	PRINT_REPORT,
	PRINT_REPORT_SECURED,
	PRINT_REPORT_AT_SERVER,
	PRINT_REPORT_AT_SERVER_SECURED,
	PUBLISH_OUTPUT,
};

enum enumPortalConf
{
	//[GLOBALS]
	EPC_GLOBALS_SHOW_CONNECTION,
	EPC_GLOBALS_HELP_STYLE,
	EPC_GLOBALS_ALLOW_REFRESH_DATA,

	//[RUNREPORTS]
	EPC_RUN_RPTS_SHOW,
	EPC_RUN_RPTS_STANDARD,
	EPC_RUN_RPTS_ADHOC,
	EPC_RUN_RPTS_CALENDAR_STYLE,

	//[REPORTMANAGEMENT]
	EPC_RPT_MGMT_SHOW,
	EPC_RPT_MGMT_DEPLOY,
	EPC_RPT_MGMT_DEPLOYBUNDLER,
	EPC_RPT_MGMT_REPORTOBJECTS,
	EPC_RPT_MGMT_PRINTSETTING,

	//[DESIGNING]
	EPC_DESIGN_SHOW,
	EPC_DESIGN_WEBSTUDIO,
	EPC_DESIGN_ADHOCDESIGNER,
```

```
        //[SCHEDULING]
        EPC_SCHEDULING_SHOW,

        //[ADMINISTRATION]
        EPC_ADMIN_SHOW,
        EPC_ADMIN_RS_STATUS_REFRESH_INTERVAL,

        //[PERSONALIZATION]
        EPC_PERSONAL_SHOW,
};

enum enumAdhocConf
{
        //[Globals]
        EAC_GLOBALS_ALLOWSETTEMPLATE,
        EAC_GLOBALS_DEFAULTTEMPLATE,
        EAC_GLOBALS_ALLOWSETFORMAT,
        EAC_GLOBALS_ALLOWEDFORMATS,
        EAC_GLOBALS_DEFAULTFORMAT,
        EAC_GLOBALS_SHOWHELP,
        EAC_GLOBALS_DEFAULT_EXPAND_TABS,
        EAC_GLOBALS_FIELDLIST_RESIZABLE,
        EAC_GLOBALS_FIELDLISTWIDTH,
        EAC_GLOBALS_CALENDAR_STYLE,
        EAC_GLOBALS_ALLOW_HIDE_DETAILS,

        //[Data source]
        EAC_DATA_SRC_TYPE,
        EAC_DATA_SRC_ONEDITSHOW,
        EAC_DATA_SRC_ONEDITCHANGE,
        EAC_DATA_SRC_SELECT_TYPE,
        EAC_DATA_SRC_ALLOW_SEQ_SELECTION,
        EAC_DATA_SRC_SET_CUSTOM_WIDTH,
        EAC_DATA_SRC_ALLOW_CUSTOM_ALIGN,
        EAC_DATA_SRC_QO_FETCH_TYPE,

        //[Filters]
        EAC_FILTERS_SHOW,
        EAC_FILTERS_COUNT,
        EAC_FILTERS_SHOWMAXROWS,
        EAC_FILTERS_SHOWPROMPT,
        EAC_FILTERS_SHOWPARAMETER,

        //[FilterOperators]
        EAC_FILTEROPERATORS_CHAR,
        EAC_FILTEROPERATORS_NUMBER,
        EAC_FILTEROPERATORS_DATE,

        //[Groups]
        EAC_GROUPS_SHOW,
        EAC_GROUPS_COUNT,
        EAC_GROUPS_SHOWGROUPBY,

        //[Totals]
        EAC_TOTALS_SHOW,
        EAC_TOTALS_COUNT,
```

```
        //[Sort]
        EAC_SORT_SHOW,
        EAC_SORT_COUNT,

        //[Matrix]
        EAC_MATRIX_SHOW,

        //[Chart]
        EAC_CHART_SHOW,
        EAC_CHART_ALLOWSETPOSITION,
        EAC_CHART_DEFAULTPOSITION,
        EAC_CHART_ALLOWSETLEVEL,
        EAC_CHART_DEFAULTLEVEL,
        EAC_CHART_CHARTTYPES,
        EAC_CHART_DEFAULTCHARTTYPE,
        EAC_CHART_ALLOWSETSERIESCHARTTYPES,

        //[Viewer]
        EAC_VIEWER_REMOVE_PREV_FILTERS_FOR_FIELD,
};

enum enumReportServerProp
{
        ERSP_LISTENER_PORT,
        ERSP_DATABASE_CONNECTION_TIMEOUT,
        ERSP_INPUT_MODE,
        ERSP_SECURITY_FEATURES,
        ERSP_AUDIT_LOG,
        ERSP_INPUT_DIRECTORY,
        ERSP_QUEUE_SIZE,
        ERSP_PAGE_CHUNKSIZE,
        ERSP_LOG_LEVEL,
        ERSP_OUTPUT_DIRECTORY,
        ERSP_XML_DATA_SOURCE_DIRECTORY,
        ERSP_TEMP,
        ERSP_LOG,
        ERSP_USER_THREADS,
        ERSP_FONT_DIRECTORY,
        ERSP_CACHE_CAPACITY,
        ERSP_LEFT_MARGIN_OFFSET,
        ERSP_TOP_MARGIN_OFFSET,
        ERSP_SCALE_OFFSET,
        ERSP_CLIENT_SESSION_TIMEOUT,
        ERSP_REMOTE_SESSION_TIMEOUT,
        ERSP_RTF_FIELD_CONTROL_MAP,
        ERSP_DATA_SOURCE_FETCH_SIZE,
        ERSP_PERSONAL_DIRECTORY,
        ERSP_CSA_TIMEOUT,
        ERSP_CACHE_PATH,
        ERSP_AUDITLOG_PURGE_TIME,
        ERSP_CACHE_PURGE_TIME,
        ERSP_CACHE_PURGE_FREQUENCY,
        ERSP_STANDARD_COLORS,
        ERSP_AUTHORIZATION_CACHE_TIMEOUT,
        ERSP_REPOSITORY_CACHE_TIMEOUT,
        ERSP_SORT_AREA_SIZE,
        ERSP_FILE_DATA_SOURCE_PATH,
```

```
        ERSP_EMAIL_FROM_ADDRESS,
        ERSP_HOST_URL,
        ERSP_JOB_SUCCESS_MAIL_TO,
        ERSP_JOB_SUCCESS_MAIL_SUBJECT,
        ERSP_JOB_ERROR_MAIL_TO,
        ERSP_JOB_ERROR_MAIL_SUBJECT,
        ERSP_SCHD_JOB_DISPATCH_QUEUE_SIZE,
        ERSP_SCHD_JOB_DISPATCH_THREADS,
        ERSP_SMTP_SERVER,
        ERSP_SMTP_SERVER_USER,
        ERSP_SMTP_SERVER_PASSWORD,
        ERSP_HYPERLINK_RELATIVE_PATH,
        ERSP_DATA_CACHING,
};

enum enumHTMLToolBarConf
{
        //[GLOBALS]
        EHTMC_GLOBALS_SEQUENCE ,
        EHTMC_GLOBALS_BACK_COLOR,

        //[NAVIGATION]
        EHTMC_NAVIGATION_SHOW,
        EHTMC_NAVIGATION_SEQUENCE,

        //[PUBLISH]
        EHTMC_PUBLISH_SHOW,
        EHTMC_PUBLISH_SEQUENCE,

        //[PRINT]
        EHTMC_PRINT_SHOW,
        EHTMC_PRINT_SEQUENCE,
        EHTMC_PRINT_LEFT_MARGIN,
        EHTMC_PRINT_RIGHT_MARGIN,
        EHTMC_PRINT_TOP_MARGIN,
        EHTMC_PRINT_BOTTOM_MARGIN,


        //[EMAIL]
        EHTMC_EMAIL_SHOW,
        EHTMC_EMAIL_SEQUENCE,

        //[REFRESHDATA]
        EHTMC_REFRESHDATA_SHOW,
        EHTMC_REFRESHDATA_SEQUENCE,

        //[EXPORT]
        EHTMC_EXPORT_SHOW,
        EHTMC_EXPORT_SEQUENCE,
        EHTMC_EXPORT_DIALOG_LIST,

        //[COMMENTS]
        EHTMC_COMMENTS_SHOW_IN_NEW,
        EHTMC_COMMENTS_SHOW_IN_PUBLISHED,
        EHTMC_COMMENTS_SEQUENCE,

        //[ADHOCPOWERVIEWER]
```

```
        EHTMC_ADHOCPOWERVIEWER_ENABLED,
        EHTMC_ADHOCPOWERVIEWER_SHOWHELP,

        //[REPORT]
        EHTMC_REPORT_SHOW,
        EHTMC_REPORT_SEQUENCE,

        //[UTILITY]
        EHTMC_UTILITY_SHOW,
        EHTMC_UTILITY_SEQUENCE,
};


enum enumUserPreferences
{
        EUP_LANGUAGE,
        EUP_THEME,
        EUP_DELIVERY_LOC,
        EUP_EMAIL,
        EUP_FORMAT,
        EUP_DASHBOARD,
        EUP_SHOW_INBOX,
        EUP_SHOW_REPORTS,
        EUP_SHOW_REP_PUB,
        EUP_REP_COUNT,
        EUP_RUNTIME_USER_ID,
        EUP_RUNTIME_PASSWORD,
        EUP_DEFAULT_CONNECTION,
        EUP_IS_RUNTIME_CREDENTIALS,
};

enum enumReportClientConf
{
        ERCC_INTERA_HOME,
        ERCC_RPT_ENGINE_IP,
        ERCC_RPT_ENGINE_PORT,
        ERCC_OUTPUT_MODE,
        ERCC_RPT_PATH,
        ERCC_TEMP_PATH,
        ERCC_RELATIVE_PATH,
        ERCC_LOGGING_ENABLED,
        ERCC_LOG_LEVEL,
        ERCC_LANGUAGE,
        ERCC_USER_NAME,
        ERCC_USER_PASSWORD,
        ERCC_USER_ORG,
        ERCC_SERVER_TIME_OUT,
        ERCC_SERVER_TIME_OUT_CHUNK,
        ERCC_RPT_CLEAN_UP_INTERVAL,
        ERCC_RPT_CLEAN_UP_TIME,
        ERCC_REGISTRY_ENABLED,
        ERCC_LOADBALANCER_REGISTRIES,
        ERCC_HTML_VIEWER_TIMEOUT,
};

enum enumDriverTypes
{
```

```
        EODT_NONE = -1,
        EODT_ORACLE_OCI ,
        EODT_ORACLE_OCI_TNS,
        EODT_ORACLE_THIN,
};

enum enumSavedReportInstanceVal
{
        ESRI_NONE = 0,
        ESRI_LASTSAVED = 1,
};

enum enumExternalAppAttrib
{
        EAA_SERVER_IP,
        EAA_SERVER_PORT,
        EAA_USERDN,
};

enum enumCallBackAttrib
{
        CBA_SERVER_IP,
        CBA_SERVER_PORT,
};



enum enumImplementerAttrib
{
        IA_PATH,
};

enum enumSystemFilterAttrib
{
        SFA_FILTER_COLUMN_NAME,
        SFA_IGNORE_IF_NOT_PRESENT,
};

enum enumAuthenticationMode
{
        AUTH_INTELLICUS,
        AUTH_EXTERNAL,
        AUTH_HOST,
        AUTH_CALLBACK,
};

enum enumAuthExtAppType
{
        AUTH_EXT_APP_WINNT,
        AUTH_EXT_APP_UNIX,
        AUTH_EXT_APP_LDAP,
        AUTH_EXT_APP_NDIS,
};

enum enumCallbackType
{
        CALLBACK_LOCA,
```

```
        CALLBACK_SOCKET,
        CALLBACK_RMI,
};

enum enumImplementerType
{
        IMPL_JAVA_CLASS,
        IMPL_NATIVE_LIB,
        IMPL_COM_DLL,
};

enum enumAuthorizationMode
{
        AUTHORIZATION_INTELLICUS,
};

enum enumGlobalFilterMode
{
        GFT_INTELLICUS,
        GFT_CALLBACK,
};

enum enumSelectOptionsType
{
        SOT_USERATTR = 1,
};


enum ReportObjectOpType
{
        ADDOBJ,
        DELETEOBJ,
        REPLACEOBJ,
        ONSERVEROBJ,
        NOOP,
};

enum LOGGING_LEVELS
 {
        LOGGING_LEVEL_NOLOGGING,
        LOGGING_LEVEL_FATAL,
        LOGGING_LEVEL_ERROR,
        LOGGING_LEVEL_WARNING,
        LOGGING_LEVEL_INFO,
        LOGGING_LEVEL_DEBUG
};


enum enumWeekDays
{
        EWD_SUNDAY,
        EWD_MONDAY,
        EWD_TUESDAY,
        EWD_WEDNESDAY,
        EWD_THRUSDAY,
        EWD_FRIDAY,
        EWD_SATURDAY,
```

```
};

enum enumMonths
{
	EM_JAN,
	EM_FEB,
	EM_MAR,
	EM_APR,
	EM_MAY,
	EM_JUN,
	EM_JUL,
	EM_AUG,
	EM_SEP,
	EM_OCT,
	EM_NOV,
	EM_DEC,
};

enum enumScheduledTaskParametersType
{
	ESTP_STATIC,
	ESTP_SQL,
	ESTP_XML
};

enum enumFetchType
{
	EFT_STATIC,
	EFT_DYNAMIC
};

enum enumFetchDataSourceType
{
	EFDS_AUTOGEN,
	EFDS_PREDEF,
	EFDS_SQL
};

enum enumSQLType
{
	EST_SQL,
	EST_STOREDPROC
};

enum enumFetchDataSourceSQLType
{
	EFDSST_AUTOGEN,
	EFDSST_LOOKUP
};

enum enumFetchDataSourceXMLType
{
	EFDSXT_AUTOGEN,
	EFDSXT_LOOKUP
};

enum enumXMLRuntimeSrcType
```

```
{
	EXRST_FILE,
	EXRST_URL,
	EXRST_DATABASE
};

enum enumQueryObjectSourceType
{
	EQOST_SQL,
	EQOST_XML
};

enum enumTemplateOutputFormatType
{
	ETOFT_HTM,
	ETOFT_PDF,
	ETOFT_XLS,
	ETOFT_DHTM,
	ETOFT_RDF,
	ETOFT_TXT,
	ETOFT_CSV,
	ETOFT_DOC
};

enum enumIRLType
{
	EIT_STATIC,
	EIT_DYNAMIC,

};

enum enumDataSourceType
{
	EDST_NONE,
	EDST_QO,
	EDST_TABLE,
	EDST_METALAYER
};

enum enumFunction
{
	EF_SUM,
	EF_AVG,
	EF_COUNT,
	EF_MAX,
	EF_MIN,
	EF_STDDEV
};

enum enumLevel
{
	EL_GROUP,
	EL_PAGE,
	EL_REPORT
};

enum enumChartType
```

```
{
	ECT_LINE,
	ECT_BAR,
	ECT_PIE,
	ECT_SCATTER,
	ECT_RADAR,
	ECT_COLUMN
};

enum enumChartPosition
{
	ECP_TOP,
	ECP_BOTTOM
};

enum enumChartLevel
{
	ECL_REPORT,
	ECL_PAGE
};

enum enumColumnFieldType
{
	ECFT_SQL,
	ECFT_FORMULA
};

enum enumSystemFileType
{
	ESFT_ADHOC_TEMPLATE_IRL,
	ESFT_ADHOC_TEMPLATE_STYLE,
	ESFT_XML_DATASOURCE,
	ESFT_CHART_TEMPLATE
};

enum enumColumnAlignmentType
{
	EAFAT_LEFT,
	EAFAT_RIGHT,
	EAFAT_CENTER
};

enum enumXMLMetaDataSourceType
{
	EXMST_FILE = 0,
	EXMST_URL,
	EXMST_DATABASE
};

enum enumXMLDataSourceElementType
{
	EXDET_NONE = -1,
	EXDET_XML_TAG = 0,
	EXDET_XML_ATTRIBUTE,

};
```