



VIZOR PLATFORM API

REFERENCE FOR ADVANCED USERS AND DEVELOPERS

Contents

Vizor Platform API.....	1
Contents	2
Chapter 1 - Introduction.....	9
1.1. The DTS Model	9
DTS Record	9
DTS Fields.....	10
1.2. General Guidelines for Customizations.....	10
Suggestions	10
1.3. Web View Types	10
Normal Web View.....	10
Wizard Web View.....	11
Form Web View.....	11
Chapter 2 - Dts Fields	12
2.1. Data Types.....	12
2.2. Container Types.....	13
2.3. Relationship Types.....	15
2.4. Choice Tables.....	17
Chapter 3 - Web Browser API.....	19
3.1. Guidelines when making customizations to Web Views.....	19
Suggestions	19
Things to Avoid	19
3.2. CustomSettings.js	20
3.3. TmplCensusMain.htm.....	20
3.4. tmplRecord.htm.....	21

3.5.	Web Field HTML Templates.....	22
	Field-Control Type Naming Convention.....	22
	Named (arbitrary) HTML Template	23
3.6.	Web Field Template Tags/ Variables.....	24
3.7.	Common Field HTML Templates	26
3.8.	Vector Web Session	27
	Close the Web Session.....	28
	Check if the session is modal	28
3.9.	Web Session URL	29
	CurrentVDName	29
	BaseVDName	29
3.10.	Standard Object Model Hierarchy.....	29
3.11.	Vector Libraries	30
3.12.	HTML Documents.....	30
	Creating a new HTML document.....	30
	The Main HTML document.....	31
3.13.	Current Record and the RecordMgr (CMqRecordMgr)	31
3.14.	VNRecord	32
3.15.	Field Values of a Record	33
3.16.	VNFormField.....	34
3.17.	CMqRecord	34
3.18.	VNRecForm.....	34
3.19.	Make HTTP request calls	35
3.20.	Opening Windows with the Framework.....	35
3.21.	Debugging	35
3.22.	Summary List.....	35

3.23. Examples of Customizations	37
Change the height of a frame for a given view.....	37
Open the New Email dialog with a specific Email Template.....	37
Chapter 4 - Web View Extensions.....	40
4.1. Interface.....	40
4.2. Give it a Name.....	40
4.3. Cancelling the Event.....	41
4.4. Adding a WV Extension.....	41
Chapter 5 - Web Server API.....	42
5.1. Issue Service.....	42
5.2. Interactive test.....	42
5.3. PowerShell consumer.....	43
Best practices.....	44
5.4. UI Localized messages sent from Server API.....	45
Chapter 6 - Application Server COM API.....	46
6.1. Concepts.....	46
References.....	46
6.2. The CenApp object.....	46
6.3. Open a Session	47
6.4. Get a Project object	47
6.5. Open a Project	48
6.6. Update Issue	50
6.7. Add Attachment.....	50
IMqCenApp.OpenSession()	52
IMqCenApp.OpenProject()	52
IMqRecordsMgr.Initialize()	53

IMqRecordsMgr. Update ()	54
IMqRecordsMgr.AddAttachment()	54
6.8. Add, Delete or Update (save) DTS records	54
6.9. CenApp Variables.....	54
6.10. MultiChoice Fields.....	55
6.11. SingleChoice Fields.....	55
6.12. SQL Engine.....	56
6.13. SQL Expressions	56
TblRuleCondition, tblSystemQueries.....	57
TblCondExpr (model)	58
Format Types for SQL Expression Components.....	59
Chapter 7 - Web View Settings & Configurations	61
7.1. Reports.....	61
7.2. Summary list / Drill down	61
7.3. Details Form - Tabs	61
Chapter 8 - Macros.....	63
8.1. Macros.....	63
8.2. Evaluating a macro.....	64
8.3. Working with CustomCode.bas.....	65
Arguments used in MqCustomCode.....	66
Debugging.....	67
Chapter 9 - Data Model (Database Schema).....	68
9.1. Licenses Database.....	68
TblMdbSystem	68
TblProject Access.....	68
TblProjects.....	69

tblSystemMessagesLicenses	69
9.2. Users Database	70
TblgroupPrivileges	70
tblNextUserID	70
TblUser.....	71
9.3. DAT Database.....	71
TblAttachments.....	71
TblDts	71
TblFixInformation	72
TblNextDefect.....	72
TblRevisionHistory	72
TblSystemMessagesDat.....	73
9.4. DEF Database.....	73
TblDtsFields	73
TblDtsFieldValueTables.....	75
TblDtsTables.....	75
TblEmptyTableNames.....	76
TblFeatures.....	77
TblFeaturesDependent.....	77
TblFieldAccess	77
TblMacros.....	78
tblMailContents	78
tblNextObjectID	79
tblNotification.....	79
TblOperators	80
TblSystemConditions.....	80

TblSystemCustomConditions	81
TblSystemLayouts	82
TblSystemMessagesDef.....	82
TblSystemQueries	82
TblSystemReportListing.....	83
tblSystemReportPageLayouts	83
tblSystemReportPageSections	84
tblSystemReportSummaries.....	84
TblTableTypes.....	84
TblWho	85
tblWorkaround.....	86
TblTabs	86
tblRule.....	86
tblRuleCondition.....	86
tblSetValue.....	86
tblPossibleValue.....	87
Chapter 10 - Business Rules	90
10.1. File Business Rules	90
10.2. Optional Bultin-in File Business Rules.....	91
Chapter 11 - Configuring Localizations and Translations.....	93
11.1. How to change to a non-English language	93
11.2. Folders	94
Chapter 12 - Programming with Localizations, Translations and Resource Strings.....	95
12.1. Web Browser Side.....	95
How to Use it	95
Requirements.....	96

12.2.	Server Side	97
Chapter 13 -	Asset Intelligence and Calculations	99
13.1.	Asset Totals.....	99
Chapter 14 -	Emails.....	100
14.1.	Outgoing Emails	100
14.2.	Incoming Emails.....	100
Chapter 15 -	Apendices.....	101
15.1.	Department, Location and Company Fields	101

CHAPTER 1 - INTRODUCTION

This guide is intended to be a reference for the following development projects:

- Customize the user interface beyond the standard VIZOR editors capabilities
- Create custom behaviours and validations in the VIZOR forms
- Create new VIZOR projects with unique characteristics
- Customize the validation and saving of specific fields
- Create VIZOR records from an external system
- Query VIZOR information from an external system

In order to use this guide, the reader must have passed the VIZOR administration training **levels 1 and 2** and have previous knowledge of programming languages.

VIZOR offers various alternatives and interfaces of its API. You may use one or several approaches in your project, depending on your requirements.

1. Web Browser API
2. Web Server API
3. Application Server COM API
4. Web View Settings & Configurations
5. Programming Extensions within VIZOR
6. Data Layer Integration

1.1. The DTS Model

DTS Record

A DTS Record is main record being tracked in a VIZOR Project. For example, a DTS Record is a **Ticket** or **Issue** in HelpDesk, an **Asset** in Assets, a **Purchase** in Purchases & Agreements and a **User** in Users & Contacts.

The fields of the DTS Record are defined in tblDtsFields.

The data is stored in DTS Tables, typically in the main table tblDts (with a few exceptions such as tblUser for the Users Project).

DTS Fields

DTS fields are 'virtual' fields that can be located in different physical tables or even databases and can be used as part of the same 'virtual' record.

Some of the advantages of presenting these fields as coming from a single virtual table are:

- The data can be presented in the UI in a simpler way.
- The data can be queried and used in summary lists.
- The data can be queried and used in conditions for email notifications, report subscriptions, etc.

This is the heart of the schema of a VIZOR project. It lists the DTS fields which are available in web views, summary list layouts, queries, sorts, reports, notifications, workflow rules, etc.

1.2. General Guidelines for Customizations

Suggestions

If you are creating a custom VIZOR project, it is recommended to use a standard project such as HelpDesk as the base project.

Limit the customized files and customizations to the minimum and keep the customizations in separate files, that is, in files that are not part of a standard VIZOR installation. This will make it easier to upgrade the customizations for future versions of VIZOR.

Try to create single-choice fields for YES/NO fields, allowing on this way adding new values in the future (a third state) or for using workflow rules

1.3. Web View Types

Normal Web View

This Web View is accessible from the home page. It has a searchable summary list and the VN Form used to edit, view or create new DTS Records.

Wizard Web View

Designed to be only shown from another Web View.

It never shows a Summary List, only the details of a DTS record (tabs + record + command bar).

Only supported in modal window mode before v2.5.1.

After Saving or Canceling, it closes the embedded or separate window.

Form Web View

Views of this type will have a new HTML file (VNForm.htm) which is a simplified tmplRecord.htm and used instead of tmplCensusMain and all other html files in the view. It is created in the WVE using the fields that are exported in the view, based on the template file VNFormTpl.rec.

This view type is currently only supported in read-only mode, to load a record, therefore saving will not be enabled, and any fields in the view should not/will not be enabled/editable (drop downs, etc).

The SQL and XML for loading the record are precreated in view attributes to save time when loading the record.

The SQL is stored in memory the first time the view is loaded.

A new webservice loads the data based on the precreated XML and SQL.

The current client side code for loading the records will still be used. If we need to support other objects/classes, we'll need to add the reference in the VNFormTpl.rec.

To have a link to go to edit mode, in the ViewConstants.js for the form view, a variable with the URL called strEditViewURL must be created.

This type of view will not show in the quick links and is just for loading single records. There is no summary list, or other toolbars.

CHAPTER 2 - DTS FIELDS

DTS fields are 'virtual' fields that can be located in different physical tables or even databases and can be used as part of the same 'virtual' record.

Some of the advantages of presenting these fields as coming from a single virtual table are:

- The data can be presented in the UI in a simpler way.

- The data can be queried and used in summary lists.

- The data can be queried and used in conditions for email notifications, report subscriptions, etc.

This is the heart of the schema of a VIZOR project. It lists the DTS fields which are available in web views, summary list layouts, queries, sorts, reports, notifications, workflow rules, etc.

Fields are primarily defined by their data types, relationship types and container types.

2.1. Data Types

Field Data type IDs.

Data Type	ID	Notes
Boolean	1	
Currency	5	
Date	8	
Text	10	

Memo	12	
Time	101	
Number	102	
Single Choice	103	
Multi Choice	104	
SCC	105	
Related Record	106	also Child Records
DateTime	109	Date + Time
SubRecordLink	110	The main field that links to a subrecords table
Password	115	
HTML Rich Text	120	
SQLRowVersion	121	binary, timestamp type in SQL Server

2.2. Container Types

Container type IDs are unique at the moment. In the future we may allow different container types using the same ID for different data types.

List of know container types used by VIZOR*:

*Please note that others may have been created by customers or resellers.

Data Type	Container Type		
(all)	1		Default container type. It produces the default type of field from the Web View Editor.
	2	Attachments	Attachments button that shows the number of attachments and it opens a dialog with the complete list of attachments.
	3	VSS	VSS grid and controls.
	4	Report Only	Field not visible in the tabs and only used in reports. E.g. 1:M fields of the VAM integration that are needed for the reports.
	5	Grid like Related Record, Child Records, etc.	RMA Tickets RMA Parts List Only available for Data Type = 106 If Data Type = 106 AND Container Type = 5 nControlType can be used to control the columns in the grid using the id of named layout (layout editor)
	6	Hyperlink Parent Issue	E.g. Parent Issue
	7	Alternate Choice List	
	8	Single-choice Rating	Display a single-choice as a rating with images (e.g. stars)

	15	Tasks	Tasks local to the project and shown in a list/grid table.
	16	Append Memo	Memo field where new text is appended on save, always added. Usually used in the new Log Memo fields like Activity Log.
	30	Total Time with Stopwatch	Stopwatch/timer control. Includes timer icons, buttons and functionality E.g. Total Effort Time
	31	TimeSpan	The TimeSpan represents a time interval. Its two typical uses are: - Reflecting the time interval between two date and time values. For example, to store the time between the creation time and closure time. - Measuring elapsed time. For example, for the Stopwatch or the total effort time to reflect the time interval that has elapsed since the start of something an event.
	9	Rich Text Editor Memo (HTML)	The content supports Rich Text Format (via HTML) and it is edited via a special editor (CKEditor in Vizor 2.5) E.g. Description, Detailed Description when used with Rich Text.

2.3. Relationship Types

The Relationship Type defines how the DTS field related to the Main table, when executing queries.

The assumptions are:

The query will execute in the DAT database of the project (e.g. Assets01_Dat)
 So, tables and fields located in the same database are not prefixed.
 The query will be centered around the main table, usually tblDts.
 All the other tables will be joined.

How the DTS fields that are not in the main table are joined depend on the Relationship Type.

-1	Main	tblDts
1	<p>One to One, joined as defined by Main.nID = TABLE.nRelRec</p> <p>Where the joined fields are specified in tAliasSourceTable= nID;nRelRec</p> <p>separated by ; if one field is absent, "nID" is assumed</p>	tblRelIssue
2	<p>One to Many joined as defined by Main.nID = TABLE.nID</p> <p>Where the joined fields are specified in tAliasSourceTable= nID;nID</p> <p>separated by ; if one field is absent, "nID" is assumed</p>	tblAttachments
3	Many to One	tblUser
6	<p>Double Join Join of a join. That is, it can only be joined with the main table via another joined table. The first joined table is defined by the tTableName of the field specified in the nRelatedField</p> <p>The join is defined by Main.nID = TABLE1.nRelatedFieldID TABLE1.nID = TABLE2.nID</p> <p>Where the joined fields are specified in tAliasSourceTable= nID; nRelatedFieldID nID; nID</p>	Fields from a Purchase associated to a SW License, as seen from the SW License

	separated by ; and a pipe for separating the joins. if one field is absent, "nID" is assumed	
--	---	--

2.4. Choice Tables

Field can have lists of possible values, like catalogs, which are called Choice Tables. These are needed for single and multichoice fields.

What choice list is used for a given field is defined by these properties of the field (defined in tblDtsFields):

Field in tblDtsFields	Example	Notes
tFieldValuesTable	tblUser	Name of the database table where the choices come from.
nValueTableDSType	64	Numeric ID that defines the database where the table is.
tWhere	WHERE Security_Type=8	Where condition appended to the query of the specified table to get the value for the choices.
fChoiceOrder	0/false	If true, the Choice Table uses a custom sort via the additional field nChoiceID in the table. So, the sort would be similar to: ORDER BY nChoiceID
tValueTableIDFld	nNewID	If empty, the ID column/field of the choice table is nID , otherwise the name specified here. This is the value stored in the databases.
tValueTableNameFld	tNewName	If empty, the Name column/field of the choice table is tName , otherwise the name specified here. This is the string displayed to users.

CHAPTER 3 - WEB BROWSER API

The main language used is Javascript.

3.1. Guidelines when making customizations to Web Views

Suggestions

Add short, simple customizations to the CustomSettings.js file.

Try to encapsulate the new functionality in a separate library (a JS file), potentially a Web View Extension that is loaded in CustomSettings.js (see documentation).

Another recommended way is to use custom HTML field templates (HTMPL)

Consider how the customization should/will be upgraded and favour a design that facilitates it.

Apply files in the CustomizedFiles folder to the highest possible folder, that is, the folder with the maximum scope, ideally all projects, all web views.

When adding a file to the CustomizedFiles folder, always search for other instances of that file in the CustomizedFiles folder structure.

Avoid customizing html files in favor of reusable js libraries.

Document the changes in a separate file.

Things to Avoid

Avoid adding code to existing/generic files such as Start.rec, TmplCensusMain, CustomCode.js, etc.

Avoid deleting standard fields if they are not needed. Consider if they may be used in the future and if so, hide them or simply do not use them. You may need that type of field in the future and it will be much easier to use it at that time than to recreate it if it has special behaviours.

Files that cannot be added to customized files folder:

TmplCensusMain

Avoid updating this file directly. It needs to be updated in the Templates folder.

TmplGlobal.asa

It needs to be updated in the Templates folder.

Global.asa

It is generated from the TmplGlobal.asa

3.2. CustomSettings.js

If javascript or HTML code needs to be changed for a non-generic customization, the change should be done inside the file customsettings.js.

This file should include customizations, behaviours and settings of the Web view that are changed manually, outside the VIZOR editors.

The vision of this file is to consolidate all customer-modifiable settings into a well defined area/file.

3.3. TmplCensusMain.htm

Customizations, behaviours and settings of the Web view that are controlled from the VIZOR editors, typically the Web View Editor, should be set in TmplCensusMain.htm. A typical example of the approach is the following: the WebViewEditor has a Web View attribute in the UI which the user can change. The value is stored in the database and it is ultimately reflected in the TmplCensusMain.htm file to be used at runtime.

The Web View Editor (WVE) writes flags, Boolean variables, values, etc. as necessary into TmplCensusMain.htm when generating the files for the Web view. This process should be typically done using the existing tag-based model with replacement mechanism.

The template of TmplCensusMain.htm must not be customized, modified or edited in any way. That is, TmplCensusMain.htm must not be placed inside the CustomizedFiles folder.

3.4. tplRecord.htm

tplRecord.htm is the file behind the details of the main form of a Web View.

The forms can load records and save records into the databases, based on the fields included in the form.

The fields included in the form are controlled by the Web View Editor, in the Fields section.

Every field exported or published to the Web View (right side) is included in the tplRecord.htm.

The field can be visible or not depending on the field attributes and workflow rules, but the field will exist in the HTML source.

The template files to construct this file come from (unless there are customized files):

ConnectServer\CensusWeb\Templates\View\MISC

At a high level, the tplRecord.htm is constructed of:

```
Start.rec  
(Tabs)  
End.rec
```

Tabs are included in the view if at least one field is exported.

For the tabs, each tab uses a source HTML template. The default tab template is: Tab.rec

The default template for the tabs can be modified in the table DEF.tbITabs, in the field tFormName.

Each Tab file contains an HTML table with 4 columns (TDs), and the Web View Editor (WVE) replaces the tag %TabFields% with the HTML of all the fields exported in that view as defined in the WVE.

For this %TabFields%, the Web View Editor (WVE) loops through the fields and uses the default HTML of that field based on the data type and control type. A field can also use a custom HTML template (see KB article Web Fields, HTML Templates, etc.).

3.5. Web Field HTML Templates

Field templates are fundamentally HTML files that are used to represent a field in the UI, usually in a Web view tab or form. Web Field templates are used by the Web View Editor when generating a Web view.

The Web View Editor creates the HTML used in the UI embedding a Web field html template per field and resolving the tags. The tags are placeholders for variables resolved per field, such as field name, field type, associated url, etc. Most of the fields are 'glued' together into tmplRecord.htm

Field templates are HTML files that are used to represent a field in the UI, usually in a Web view tab or form. Web Field templates are used by the Web View Editor when generating a Web view. The Web View Editor creates the HTML used in the UI embedding a Web field html template per field and resolving the tags. The tags are placeholders for variables resolved per field, such as field name, field type, associated url, etc. Most of the fields are 'glued' together into tmplRecord.htm.

Characteristics:

The web field html file should be located in the misc folder of the view templates.

They can be edited with any html or text editor that respects the tags/variables (e.g. %ID%).

Field values will be loaded into HTML elements with the id: id='MqF_%ID%', where %ID% is the id of the field as specified in tblDtsFields (DEF database).

The file name of a field template can be of the following types:

Field-Control Type Naming Convention

The file name derives from the field, container type and the field attributes.

Ttypes based on the naming usually have names like:

fld_103_1_0.html

which follows the naming convention of:

fld_**data type**_**container type**_**sub type/attributes**.html

The data type, container type are defined in tblDtsFields.

<datatype>:= any of cenDataType (long number)

<containertype>:= any of cenContainerType (long number)

<subtype>:= field attributes, like read-only, disabled=1, required=2

Example:

"102_1_0"

will be number data type with default container type and not read-only

If this template doesn't exist (as if the template file is the same regardless of the field attributes of the given data type), then the more generic file template will be used:

"102_1"

If this field template doesn't exist, the next template that will be used is:

"102" (for all container types and the specific data type)

The container (and control type) can usually be changed and new values created in order to create a new UI for a field. For example, a new UI can be created for date fields that uses a custom calendar or even a calendar in a different window.

Named (arbitrary) HTML Template

An arbitrary file name for the HTML template can be specified in the Field attribute "Field HTML Template", in a Web View (Web View Editor).

Arbitrary html field templates per Web view allow developers, power users and administrators to:

- reuse templates among different data types
- display the same field in different ways for different web views.

The attribute "Field HTML Template" is optional and defines the name of the html field template file. The template file will be expected to exist in the MISC folder of the view, like:

...IssueTrackerServer\CensusWeb\Views\CensusWebVD\HelpDesk_SubmitTicket\MISC

(note the normal rules of Web view HTML files and Customized Files apply).

If the html file entered in WVE doesn't exist, then it will use the default template for that field.

That is, an html field template following the standard field type order/logic (data type + container specific first, to more generic if not found) will be used, located in the same folder.

If no html field template is found, the default 'hard coded' html from the WVE will be used.

For examples on the look of some common Field HTML templates, see the linked document "Field HTML Templates Gallery".

3.6. Web Field Template Tags/ Variables

Tag		
%ID%	ID of the field	
%CtlName%	Control name of the field (tblDtsFields)	
%AssociatedFieldID%		
%Caption%		
%CaptionClass%		
%TabID%		
%DataViewID%		
%ControlType%		
%FieldTableName%	Table name of choices	
%DataSize%		
%FldAttr%	Read only=0?	
%TDCClass_Caption%		
%TDCClass_Control%		
%ColSpan%		

%DefColSpan%		
%BrowseAlternateChoiceListPostfix%		
%ValueTableDSType%		
%SearchURL%		
%TableName%		Not suggested as it provides information about the db schema to a browser side.
%Tooltip%	Tooltip as defined in the field attribute	
%ProjectName%	Name of the project (e.g. HelpDesk)	
%ReadOnly%	1 if read-only	
%Visible%	1=visible 0= not visible	
%Hidden%	1=hidden 0= not hidden	

	simply the reverse of Visible.	
%HTMLBefore%	After the outer <td> <td> %HTMLBefore%	Yes, suggested
%HTMLAfter%	Before the outer </td> %HTMLAfter% </td>	Yes, suggested
%LinkTitle%		

Replacing field values of a record dynamically

When working in the context of a VNDtsForm, there is one Dts record. In these cases, it is often useful to declare in HTML a variable that resolves at runtime to the value of a specific field of the current Dts record.

<ValOfField>field ID</ValOfField>

3.7. Common Field HTML Templates

Template	Notes
fld_Asset-Type_Model.htmlpl	For displaying the asset type and the device model together, in a better-looking UI
fld_User_Image_Assets.htmlpl	For displaying images of a person and an asset in readonly mode and with optional arrows. Used in the Check In/ Out view.
fld_VoteQuestionFixed.htmlpl	For displaying the question in approvals but using a fixed text

	for the question, coming from the resources, instead of loaded from the question field.
fld_Image_RO_Users.html	For displaying the question in approvals but using a fixed text for the question, coming from the resources, instead of loaded from the question field.
fld_101_12_SpanTwoWidthOne.html	Time field with Span Two Columns and Width One.
fld_HiddenField.html	To force a field to be hidden regardless of the workflow/business rules.
fld_Image_RO.html	For displaying an image of in readonly mode.
fld_NoField.html	<p>For eliminating the html control of a field completely. This is useful when the html page used for the dts record already has the html element used for the field.</p> <p>Fields use bound control name for loading the data into the appropriate field. That ID must be unique in the html otherwise the LoadData process will not work properly.</p> <p>This may be needed when the complete html for the field will be in a different part of the html document. In this case, the html template for the field will not be a solution.</p> <p>In these cases, the developer may add the html control to the html template of the record (like the VNForm), to the html template of a tab or to another html field template.</p> <p>When doing so, the field needs to use this template to the Web View Editor does not generate a duplicate html control.</p>

3.8. Vector Web Session

g_VWS is the root instance of a Vector Web Session.

Typical uses of VecWebSession:

- Use the HTTPRequest to make a webcall

```
g_VWS.HTTPRequest ( )
```

- Get the fields collection

- Get the current record (VNRecord)

```
g_VWS.RecordMgr().CurrentRecord()
```

- Get the current location

This line gets the current location of the main view, so

```
g_VWS.MainWindow().BaseURL()
```

which returns a path like:

http://qa-finch/connect00/HelpDesk_HelpDesk/HTML/

- Get the current session URL object

```
g_VWS.URL()
```

See

Close the Web Session

Example to exit (log off) and close the vector session window. It handles internally if it's modal or not, embedded etc.

```
m_objRef.objControlsSs.exit();  
g_VWS.MainWindow().Close();
```

Check if the session is modal

To check if the current web session is modal (embedded into another web session) or not:

```
if(g_VWS.MainWindow().IsModal()){
```

3.9. Web Session URL

The Web Session URL object has the following methods:

CurrentVDName

Gets the current virtual directory of the session, like **connect03**. The VD is allocated per session depending on the server load.

```
alert( g_VWS.URL().CurrentVDName() );
```

BaseVDName

Gets the base virtual directory of the session, like **connect**. The Base VD is the same as when the logon started and before the load distributing allocated another VD to the user. Except for rare configurations, it is always **connect** for a regular session and **connectadmin** for a web admin session.

```
alert( g_VWS.URL().BaseVDName() );
```

3.10. Standard Object Model Hierarchy

g_VWS

- CommandBar CMqCommandBar
- HttpRequest CMqHttpRequest_js
- LoadDataMgr CMqLoadDataMgr
- MainWindow VNMainWindow
- RecordMgr CMqRecordMgr
 - LoadRecord
 - CurrentRecord VNRecord

	GetFormField	VNFormField	
	GetValueControl	CMqFormElement	
	FldVals[i]	CMqFormElement	(GetFormField.GetValueControl)
ReportMgr	VNReportManager		
SummaryList	CMqSummaryList		
ViewFields	CMqFieldCol		
WorkflowMgr	CMqWorkFlowRules		
URL	CMqWebSessionURL		

3.11. Vector Libraries

Usually under the variable named g_VNLibs, use g_VNLibs for creating objects from libraries and not for referencing instances of objects or variables stored in Main.

e.g.:

```
var g_VNLibs = parent;
var oNew = new g_VNLibs.SuperObject("bla");
```

3.12. HTML Documents

Creating a new HTML document

Follow these instructions to create a new HTML document that will be used in VIZOR.

Normal html documents should have:

A reference to the VNHTMLDocInit library:

```
<script type="text/javascript" src="../../js/VNHTMLDocInit.js"></script>
```

The library initializes the framework ensuring a local reference to the session g_VWS can be used. It also adds a reference to the JS libraries, already loaded in Main (usually the parent): variable called g_VNLibs.

Advanced Note:

The library initializes the framework creating a local reference to the global g_VWS like this:

```
var g_VWS = parent.g_VWS;  
var g_VWS = opener.g_VWS;
```

The Main HTML document

Main is the HTML document where the Vector Web Session is created and the main js libraries are loaded.

Usually it is censusmain.asp, from the html file TmplCensusMain.htm

The Main HTML document is composed of iFrames that bring the following UI elements:

- Top Toolbar
- (Query) Toolbar
- Simple Query Editor
- Summary List
- Tabs
- Record
- Command Bar
- Load Data (hidden)

3.13. Current Record and the RecordMgr (CMqRecordMgr)

The RecordMgr object:

- Has the collection of field definitions
- Has access to the current record (a VNRecord object)
- Opens other records that are not the current record (other VNRecord objects)

Example:

```
g_VWS.RecordMgr().WebViewOpenInNewSession="HelpDesk/Edit HelpDesk Ticket";
```

To get the ID of the current record, at the level of the top document (censusmain), usually referenced via g_MqRef, you can use:

objRecord.GetRecordId()

Example:

```
alert("The current record is " + objRecord.GetRecordId());
```

or

```
alert("The current record is " + g_MqRef.objRecord.GetRecordId());
```

3.14. VNRecord

The VNRecord represents one DTS record in VIZOR, such as an asset, a ticket, a purchase.

The CurrentRecord() method of the RecordsMgr returns a VNRecord:

```
var vnRec= g_VWS.RecordMgr().CurrentRecord();  
vnRec.IsNewRecord();
```

Has:

VNFormFields	Collection of form fields displayed in the main
--------------	---

	record form. The collection is controlled by what fields are exported to the view in the WVE.
GetRevisionNumber()	The revision number of the record in question, as seen from the browser at that time. The actual revision on the server could had been changed by another user.
IsNewRecord()	Boolean Yes if it's a new record being created.
GetFormField()	iFldID: ID of DTS Field in that form Gets the VNFormField of the ID passed in the argument

3.15. Field Values of a Record

Inside the a record (usually the CurrentRecord), you can use the **FldVals** method, which returns a form element object for the field specified.

FldVals(iFldID)

The returned formelement object of the FieldVal is used to get and set the values of the field in that record.

Example, to get the value of field 11 (usually Owner):

```
Var val = g_VWS.RecordMgr().CurrentRecord().FldVals(11).GetValue();
```

Example, to set the value of field 11 (usually Owner):

```
g_VWS.RecordMgr().CurrentRecord().FldVals(11).SetValue("0");
```

3.16. VNFormField

A VN Form Field is an instance of a Vector Dts Field (fields controlled by the Field Editor) in a form dialog or similar UI. The typical UI is the main Details form (tmplRecord).

A VN Form Field may include multiple DOM/HTML elements.

The typical elements included in one VN Form Field are:

Caption Control	The caption as specified in the Field Caption (Field Editor) or Web View Editor (Field Attribute)
Field Control	The HTML container of the core of the field, including the value. However, the Field control is usually not the control with the value but a TD or DIV.
Value Control	The HTML element that has the value, like an INPUT, TEXT or SELECT, etc.
Extra Control	The container HTML with the extra UI elements that support the field. It could anything, from buttons, to maps, to images, etc.
GetHTMLDoc()	Returns a reference to the HTML document where the DTS field control is.
GetFieldDef()	Return a reference to the DtsField definition.

3.17. CMqRecord

Represents the detailed record UI area in the view. Usually has tabs and buttons to Save, Finish, Cancel, etc.

3.18. VNRecForm

Includes libraries needed inside a DTS form. To be used inside a form such as tmplRecord.

3.19. Make HTTP request calls

You can use the HTTPRequest object in the VecWebSession object for making AJAX-like HTTP request calls:

```
g_VWS.HTTPRequest()
```

3.20. Opening Windows with the Framework

You can use the VIZOR framework to open dialogs (windows), embedded or not.

Use the VNWindow (library window.js) and call one of the methods, such as OpenModalWin.

```
OpenModalWin(IngWidth,IngHeight,blnNearFullscreen,intMode);
```

3.21. Debugging

Add code for logging to facilitate future debugging.

Use WriteConsole

Suggested Format:

```
[class.method or function/module] <text>
```

Example:

```
WriteConsole("[WebViewOpener.Open] Opening view...")
```

3.22. Summary List

General sequence for loading the summary list:

Initialize

[CensusMain]

g_VWS.Init()

[MqVecWebSession.js] VecWebSession

m_objSL.Init()

[CMqSummaryList.js] CMqSummaryList

...

var m_objSLMgr=new CMqSummaryListMgr()

[CMqSummaryList.js] CMqSummaryListMgr

[iframe=SummaryList]

[TmplSummaryList.htm]("/HTML/SummaryList.ASP?WCI=SummaryListTmpl&ATO=1")

onLoad()

parent.g_VWS.SummaryList().Load();

(then inner iframe loads)

[TmplSummaryListData?]

/HTML/SummaryList.ASP?WCI=SummaryListTmpl&ATO=1&data=1&VT=2

Onload()

Parent. LoadSummaryList()

[tmplSumLstMon.htm]

Onload

SetSLRefreshTimer()

RefreshSummaryList()

```
parent.parent.LoadData.RegenerateSumList( true, GetSQEFldID(), true );  
  
ToolBar.submitinfo  
  
SetSLRefreshTimer( )
```

3.23. Examples of Customizations

Change the height of a frame for a given view

Overwrite the default height of a 'frame' by setting the appropriate constant in *viewconstants.js*, which will overwrite the default value stored in *constant.js*.

For example, to set the tabs strip or Tabs frame to 5 px, add:

```
var DEFAULT_TABS=5;
```

Open the New Email dialog with a specific Email Template

When working with the details form of an asset, tickets, or any other VIZOR entity, users can compose a new email by clicking on the top toolbar button called "New Email".

By default, the email that opens is blank, however, administrators can configure VIZOR to open the new email using a specific email template.

To do that, you need to set the default value of *CURRENT_ISSUE_TMPL_NAME* for all web views. stored in *constant.js*.

1. Look for the file *Constants.js* in the *Templates/view/js* folder or, if it is already customized, in the equivalent folder for the project/view in question.

For example, the file may be in *SPTemplates*:

1. *#AllWebViews#/js*
2. *#WebView#HelpDesk_HelpDesk/js*

If not already there, copy the file from *Templates/view/js* and into the *SPTemplates* folder.

2. Open it in a text editor and search for CURRENT_ISSUE_TMPL_NAME.
3. You will find a line like this:
`var CURRENT_ISSUE_TMPL_NAME=" ";`

Enter the name of the email template to use by default inside the "".

For example, for the email template called "Current Issue":

```
var CURRENT_ISSUE_TMPL_NAME="Current Issue";
```

4. Save and close the file.
5. Generate the web view

CHAPTER 4 - WEB VIEW EXTENSIONS

Web View Extensions allow to extend the default functionality of web views in cases where the framework capabilities, such as Business Rules, is not enough. Web view extensions typically involve more than one or two fields so it is not practical to use field html templates. Another reason to use WV Extensions is because they encapsulate several pieces of custom functionality that belong together, like the CheckInOut extension.

Use Web View Extensions for adding behaviours or functionality that is reusable and can be encapsulated.

Once created, extensions are pluggable and can be easily turned on for other web views.

4.1. Interface

Web View Extensions work via an interface VIZOR calls. Every method is optional, but Name is strongly suggested.

A Web View Extension has stubs for executing custom code at events such as: On Load (of a Dts Record into the tabs), On New (Dts Record), Save, Cancel, etc.

OnBeforeLoadRec
OnAfterLoadRec
SaveBefore
SaveAfter
SubmitBefore
SubmitAfter
BatchSubmitBefore
BatchSubmitAfter
CancelBefore
CancelAfter;
AddNewBefore
AddNewAfter
CopyBefore
CopyAfter
DeleteBefore
DeleteAfter

4.2. Give it a Name

Always return the name of the extension. This is useful for debugging and for keeping things in order.

E.g.

```
this.Name="MqCheckInOutExt";
```

4.3. Cancelling the Event

If you want your Web View extension to cancel the current event, that is, the current Save, Submit, Copy, Delete, etc. use the ExtensionHandler object.

The main methods to use are SetCancelProcess and SetErrorMsg.

E.g.

```
function SaveBefore(oExtensionHandler)
{
    // Cancel the Save
    var sMsg = "Cannot save bla bla";
    oExtensionHandler.SetCancelProcess(true);
    oExtensionHandler.SetErrorMsg(sMsg);
    return oExtensionHandler;
```

You can check the existing messages in the oExtensionHandler passed as argument.

4.4. Adding a WV Extension

Web View extensions should normally be added or turned on in CustomSettings.js or via custom HTML fields.

To add an extension (MqWVExtensions) from inside a JS or HTML file:

```
var oMyCustomExt= (create here your javascript extension object)

// and add it to the active extensions

objCustomCode.GetWVExtensions().AddExtension(oMyCustomExt);
```

CHAPTER 5 - WEB SERVER API

This chapter details the usage of the Web API for VIZOR.

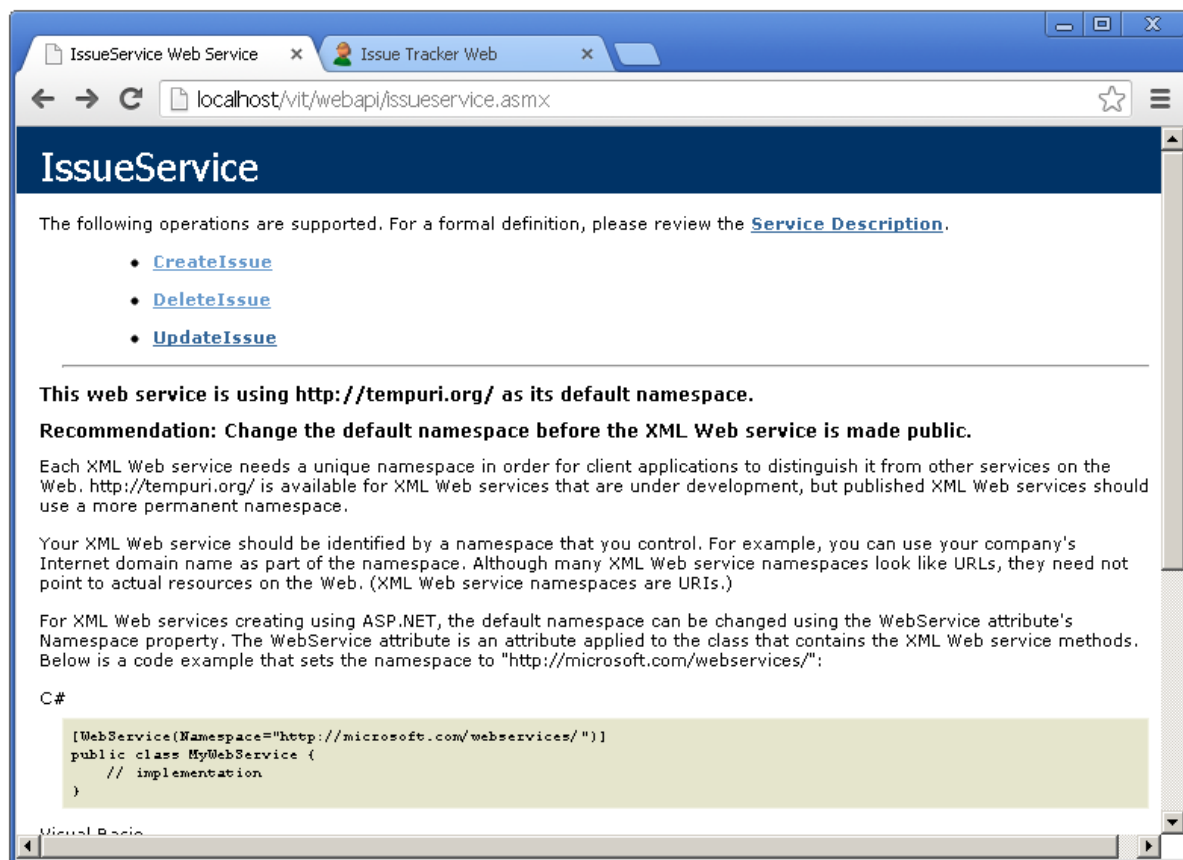
See also the topic REST API.

5.1. Issue Service

The issue service allows Create, Update and Delete operations against all Vizor projects including Helpdesk, Change Management, Defect Tracking as well as Roles, Actions and Resources.

The service is located at :-

<http://localhost/connect/webapi/issueservice.aspx>



5.2. Interactive test

The issue service can be tested in a non production environment against demo data by browsing directly to the service URL on the actual VIZOR application server. A page showing all the available operations will be displayed (as above). Clicking a operation will allow interactive testing of the service. All operations

require a username and password (this is a normal VIZOR user account) and project name as shown in VIZOR Admin.

here for a complete list of operations.' The main section is titled 'DeleteIssue' and contains a 'Test' section. The 'Test' section says 'To test the operation using the HTTP POST protocol, click the 'Invoke' button.' Below this is a form with four input fields: 'Username:' with 'admin', 'Password:' with 'admin', 'Project:' with 'helpdesk', and 'ID:' with '2'. An 'Invoke' button is at the bottom right of the form." data-bbox="127 143 741 545"/>

IssueService Web Service x HelpDesk & Problem Tickets x

localhost/vit/webapi/issueservice.asmx?op=DeleteIssue

IssueService

Click [here](#) for a complete list of operations.

DeleteIssue

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
Username:	admin
Password:	admin
Project:	helpdesk
ID:	2

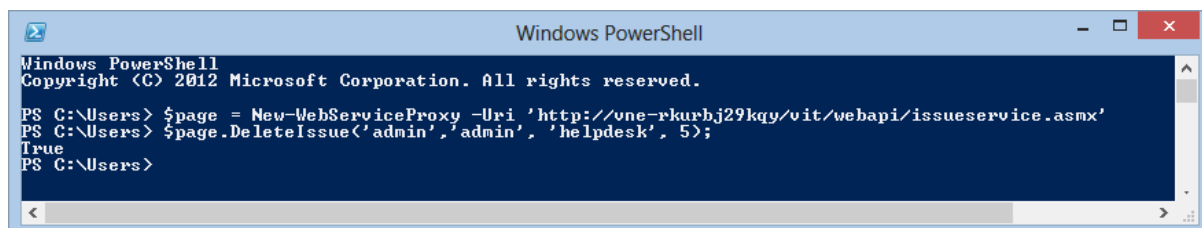
Invoke

5.3. PowerShell consumer

The Web API can be consumed from PowerShell and PowerShell scripts using the following syntax :-

```
$page = New-WebServiceProxy -Uri 'http://vne-rkurbj29kqy/connect/webapi/issueservice.asmx'  
$page.DeleteIssue('admin','admin','helpdesk', 1);
```

The operation will return a Boolean if the operation is successful or not.



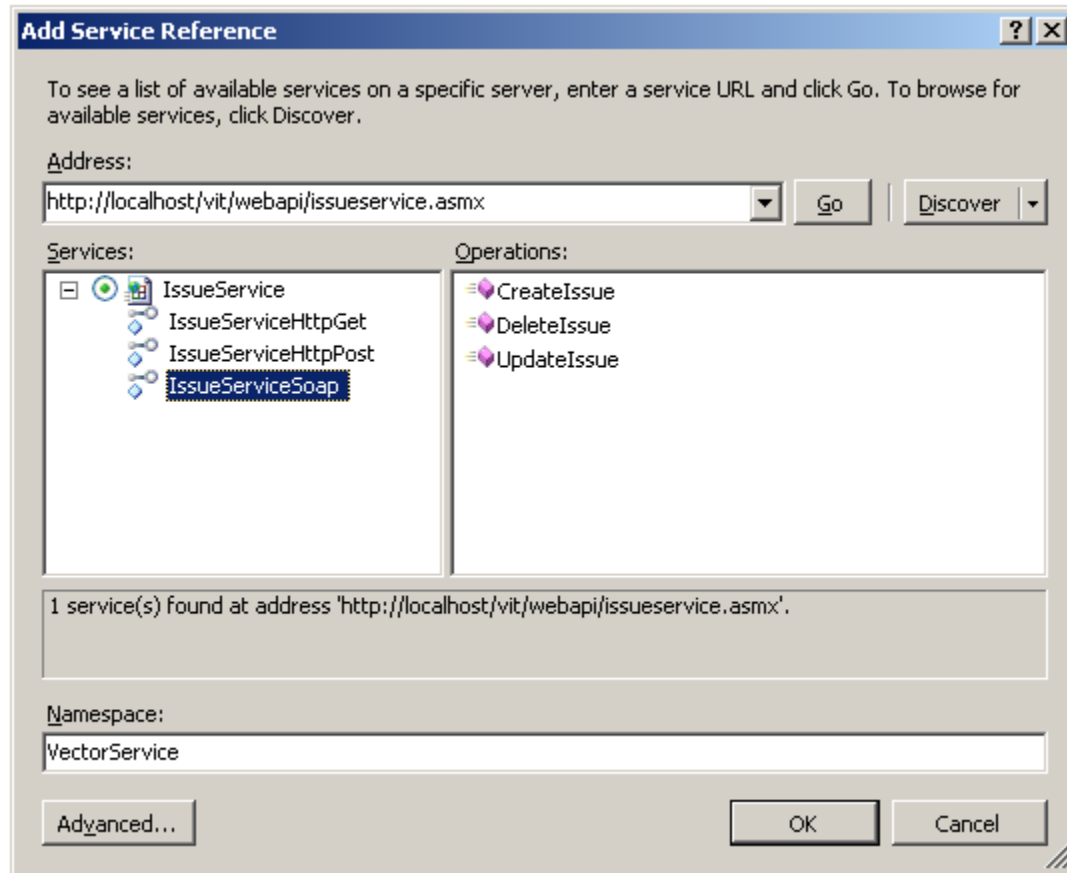
```
Windows PowerShell  
Copyright (C) 2012 Microsoft Corporation. All rights reserved.  
  
PS C:\Users> $page = New-WebServiceProxy -Uri 'http://vne-rkurbj29kqy/vit/webapi/issueservice.asmx'  
PS C:\Users> $page.DeleteIssue('admin','admin','helpdesk', 5);  
True  
PS C:\Users>
```

.net consumer

Source VectorWebAPI_Example.zip

The Web API can be consumed from any .net application such as Windows Forms, Console applications and ASP.NET applications.

Add service / web reference to your Visual Studio project.



Once the service / web reference is added simply create a instance of the proxy class generated by Visual Studio.

```
VectorService.IssueServiceSoapClient vectorServiceClient = new VectorService.IssueServiceSoapClient();  
bool bStatus = vectorServiceClient.DeleteIssue(sUsername, sPassword, sProject, iIssueID);
```

Best practices

Create a specific Vizor user account for each consuming application. This makes it easy to determine the changes each application makes using the standard issue history functionality.

5.4. UI Localized messages sent from Server API

Messages to be displayed in the UI and sent from the server after a specific event must be added to **ClientResStrings** in MqCenCore in order for them to be localized properly.

Since the language varies per user session, the server component should return the message ID that corresponds to an entry in the Resources.js file, such as Resources.en-US.js. The ID needs to be an exact match in the Resources.js string.

Example:

```
ret = ClientResStrings.DUE_DATE_MAX_REQUESTS_REACHED;
```

CHAPTER 6 - APPLICATION SERVER COM API

The main object is **IMqCenApp**.

Usually when the code runs in the Application server (Web server), but it could also run from a computer with direct LAN access (UNC) to the Application/ConnectServer folder share.

6.1. Concepts

Use this API for controlling the VIZOR application from compiled code or server side scripts.

You can create VIZOR records (DTS Records), such as tickets, assets, purchases, etc.

References

The following are the main references needed in your project.

MetaQuest Census External Library (MqCenX20.dll)

Vector IE Interfaces (MqCenInt.tlb)

MqCenCore (MqCenCore.tlb)

NOTES:

Field definitions can be found in the tblDtsFields table in the <projectname>02_def database. The nID field will be needed for updating the fields. The tName field will be needed for adding attachments.

When updating single choice lists, the value to be saved will correspond the nID value of the choice value in the field's choice list. In all other cases, the value to be saved will be the actual value. For multi-choice lists, this value should be semi-colon (;) delimited.

6.2. The CenApp object

The CenApp object is the root object of a VIZOR session. A developer works with this object via the IMqCenApp interface, either from a web application or windows application.

Below there are some examples of how to use the CenApp object.

6.3. Open a Session

Description: Open a session to VIZOR.

Uses: IMqCenApp / CMqCensusSession

Syntax:

IMqCenApp.OpenSession (UserName, Password, SessionInfo, RegistryInfo)

UserName	String. A HelpDesk user account that has permissions to access the project.
Password	String. The password for the HelpDesk user.
SessionInfo	String. Path to CenSession.xml file in HelpDesk Server folder.
RegistryInfo	Internal use. Pass "" (empty string).

[See Example](#)

6.4. Get a Project object

Description: Get the object that represent a VIZOR project in order to get other attributes such as the folder, the name, the ID, or open the databases for querying.

Uses: IMqCenApp / CMqCensusSession

Syntax:
Projects (ProjectID)

ProjectID
Variant. The ID of the project (Long).

Return the CMqProject object.

Example:

```
// this gets project with ID 3
```

```
Set objProject = oCenApp.Projects(3)
```

```
// Print the name of the project, like "HelpDesk"
```

```
Print objProject.ProjectName
```

6.5. Open a Project

Description: Open a project so that the database connections and fields are accessible.

Uses: IMqCenApp / CMqCensusSession

Syntax:

OpenProject (Project, [ServerFiles])

Project

Variant. Either the ID of the project (Long) or the name of the project (String).

ServerFiles

Optional, boolean. Pass True.

See Example

Initialize Record Manager

DESCRIPTION: INITIALIZE THE OBJECT THAT WILL BE USED TO UPDATE AN ISSUE.

Uses: IMqRecordsMgr / CMqRecordsMgr

Syntax:

Initialize (SessionObject, Project)

SessionObject

IMqCenApp. The session that was opened.

Project

CMqProject. The project that was opened. Accessible as the Session object's CurrentProject.

[See Example](#)

6.6. Update Issue

Description: Update the field values of an issue.

Uses: IMqRecordsMgr / CMqRecordsMgr

Syntax:

Update (FieldInfoToUpdate, IssueID, [ValuesEncoded])

FieldInfoToUpdate	String. A string containing the field IDs and values to set for the issue. This string must be in the format ID=Value;ID=Value...
IssueID	Long. Number of the issue to update.
ValuesEncoded	Boolean. Pass True to specify that the values in the FieldInfoToUpdate were encoded using IMqRecordsMgr.URLEncode().

[See Example](#)

6.7. Add Attachment

Description: Add an attachment to an issue

Uses: IMqRecordsMgr / CMqRecordsMgr

Syntax:

AddAttachment(IssueID, AttachmentFieldName, AttachmentInfo, [ValuesEncoded])

IssueID	Long. Number of the issue to add the attachment to.
AttachmentFieldName	String. tName value from tblDtsFields for the attachment field.
AttachmentInfo	<p>String. A string containing the field name and attachment info. This string must be in the format Name=File;fCopyFile=[1/0].</p> <p>fCopyFile=1 means that the file is uploaded.</p> <p>fCopyFile=0 means that the file is linked.</p>
ValuesEncoded	Boolean. Pass True to specify that the values in the FieldInfoToUpdate were encoded using IMqRecordsMgr.URLEncode().

[See Example](#)

Note: In addition to calling AddAttachment(), if the file is an uploaded file, the file must also be copied to the correct location. This location can be created as follows:

CMqProject.DataFilesLocation & "\" & <Attachment Field Caption> & "\" & <Issue #>

Example:

C:\Program Files\PC-Duo Enterprise\HelpDeskServer\HelpDesk\Data\Attachments\10

Examples

IMqCenApp.OpenSession()

```
Dim objSession As IMqCenApp  
Set objSession = New CMqCensusSession  
  
objSession.OpenSession "admin", "admin", "C:\Program Files\PC-Duo  
Enterprise\HelpDeskServer\CenSession.xml", ""  
  
Set objSession = Nothing
```

IMqCenApp.OpenProject()

```
Dim objSession As IMqCenApp  
Set objSession = New CMqCensusSession  
  
'call OpenSession
```

IMqRecordsMgr.Initialize()

```
Dim objRecMgr As IMqRecordsMgr
```

```
Set objRecMgr = New CMqRecordsMgr
```

```
'call OpenProject
```

IMqRecordsMgr. Update ()

```
'Initialize Record Manager
```

IMqRecordsMgr.AddAttachment()

```
'Initialize Record Manager
```

```
'Add info.txt as an attachment
```

```
objRecMgr.AddAttachment 10, "tAttachments", "tAttachments=" &
```

6.8. Add, Delete or Update (save) DTS records

use the CenX.CMqRecordsMgr.

6.9. CenApp Variables

CenApp variables hold temporary values, only valid in memory while the object exists.

They are usually used as generic parameters.

The typical use is as parameters to queries during a Web session. This is done via a query that uses a Macro which itself gets the value of the CenApp variable. The variable is often set initially via a query string in CensusMain or a separate call to \View\ASP\ SetSessionProp.asp.

The query string must have this format:

apvid- <id of CenApp variable>

Example sequence:

From the browser, open a web view into the main window and set the variable

```
LoadView('View=Users%26Contacts%2FEmployeeAssets&')
```

6.10. MultiChoice Fields

Use CMqMultiChoiceField.

Validate Values

Convert ID-Names, to a valid value

6.11. SingleChoice Fields

Use CMqMultiChoiceField.

Validate Values

Convert ID-Names, to a valid value

6.12. SQL Engine

The SQL Engine is used to create valid SELECT SQL queries that use DtsFields. Conditions that modify the WHERE clause use SQL Expressions.

Notes on usage:

- After initialization, call the method GetFrom first, so the field names in other parts of the query statement, such as the SELECT and WHERE clauses, properly use the final field names which may have aliases. The field and table aliases are generated in the FROM usually from JOINS.
- The generated SQL statement is usually used in a dataset or recordset. To get the column names for the fields in the resulting dataset or recordset:
 - The SQLEngine has a Field property collection of CMqSQLFields.
 - Each one of these CMqSQLFields represents a DtsField in a query.
 - Each CMqSQLFields has the property SqlFieldAlias which is the name to use when getting the value of that DtsField from the resulting dataset or recordset.

6.13. SQL Expressions

SQL Expressions are used to represent query conditions.

Expressions supported:

Type	UI - Display	UI Formula	SQL	SQL Formula	Input
1	Priority is High	Priority is Vd	[tblDts].[nPriority] = 1	F O V	F, O, V
2	Issue Age is more than 4 hours	Issue Age is more than V P0d	SQL Server: DateDiff(d, [tblDts].[dSubmittedDate] , getdate()) > 4 Jet/Access: DateDiff('d', [tblDts].[dSubmittedDate] , Now()) > 4	SQL Server: DateDiff(P0, F, getdate()) > V Jet/Access: DateDiff('P0', F, Now()) > V	F, O, V, P0

F:= Field

O:= Operator

V:= Value

SQL Server:

`DateDiff(d, [tblDts].[dSubmittedDate] , getdate()) > 4`

Jet/Access:

`DateDiff("d", [tblDts].[dSubmittedDate] , Now()) > 4`

TblRuleCondition, tblSystemQueries

Rows for the sql expressions used in the conditions per rule.

Each row is an expression.

nID	1
[nRuleID]	1
tName	<User in Group>
tTest	=
tValue	
tAndOr	AND

nLevel	0
nParent	0
nChild	0
nBrother	0
nRow	1
nFieldID	-55
nValueType	0

TblCondExpr (model)

New table for Condition Expressions:

nID	10	10	10
nRow	1	2	3
Type	0	0	2
nFieldType	3	2	
tField	18	<Owner>	1
nTestType	0	0	

tTest	=	>	
nValueType	4	100	
tValue	4	4	
tAndOr			
nLevel	0	0	0
nParent	0	0	0
nChild	0	0	0
nBrother	0	0	0

Format Types for SQL Expression Components

For fields, operators, values and parameters

Type(number)	Value	Text (displayed)
Macro	<Owner> or 23	<Owner>
DtsField	20	State
Parameter	<ValOfField>56</ValOfField>	ComputerName
Simple Value	"time%"	"time%"

A single choice field has:

Value Text

So, the Value should contain enough info. for determining the Type and Value from the table above.

CHAPTER 7 - WEB VIEW SETTINGS & CONFIGURATIONS

7.1. Reports

In CustomSettings.js, you can set attributes for each report like this:

```
g_VWS.ReportMgr().Report("173.10").Description="Activity effort report shows the issues and how much effort has been expended to resolve the issue.;"
```

```
g_VWS.ReportMgr().Report("5002.10").QueryType=3;
```

```
g_VWS.ReportMgr().Report("5002.10").Description="Details of the currently selected issue.;"
```

```
g_VWS.ReportMgr().Report("5002.10").SetViewerType("pdf");
```

```
g_VWS.ReportMgr().Report("5002.10").Orientation="portrait";
```

7.2. Summary list / Drill down

In ViewConstants.js, set the following attributes as needed:

Setting	Description
var m_blnDDInSL=true;	false will not enable drill-down for any query in that view. true will enable drill-down for all queries in that view.
var m_blnAllowDisableDD=true;	false will hide the checkbox in the query settings pop-up for enabling/disabling drill-down. true will show the checkbox in the query settings pop-up for enabling/disabling drill-down

7.3. Details Form - Tabs

In ViewConstants.js, set the following attributes as needed:

Setting	Description
g_VWS.CommandBar().EnableApplyInNewRec=false;	false will not show the Apply button when

creating/submitting a new item.

true will show the Apply button when
creating/submitting a new item.

CHAPTER 8 - MACROS

8.1. Macros

You can extend VIZOR using your own scripts and with minimum development efforts. Another advantage of using these extensions is that they are often compatible with future versions.

Extend VIZOR by creating your own macros and including the cde in CustomMacro.bas.

The file MacroXX.bas is reserved for Vector use.

Useful functions in macros.bas (in progress):

Macro	Function	Description
	GetDepartment	Get the Department of an employee. If the person has multiple jobs, the department of the primary job is returned.
	GetMacroUser	Get the full name of the current user. Return=string
	GetMacroIsUserInGroup	Used in a WHERE condition like <User in Group> = 'Admins' Check if the user who last updated the record specified by lngDtsRecordID is a member of the group specified in the oExpression.Value Return=modified Expression object to give true or false
	GetMacroMyWorkTeam	Get the name of the work team of the current user Return=string
	GetFullName	Get the full name of a specified strLogonName
	GetMacroProjectName	Get the name of the current project, e.g. "HelpDesk", "Resources", etc.
	GetMacroStakeholders	Get the list of users saved in the field Stakeholders of the record specified by lngDtsRecordID
	GetDtsFieldValue	Get the value of a DTS Field for the record specified by lngDtsRecordID

		<p>The field can be specified either by name (strFieldName) or by ID (lngFldID)</p> <p>For choice value fields, the caller can also specify whether to get the value as IDs or as Names (bReturnValueIDs)</p>
	GetFullNameByID	Get the full name of a user given the ID (lngUserID)
	GetFieldValueFromUsers	<p>Get the value of a field name specified by sFldName for a user given the ID (lngUserID).</p> <p>sFldName is the name of the physical field in tblUser.</p>
	GetIDByLogonName	Get the ID of the user matching the specified logon name (strLogonName)
	GetIDByFullName	Get the ID of the user matching the specified full name (strFullName)
	GetUserIDByFieldValue	Get the ID of the user matching the value strFieldValue in the field strField

8.2. Evaluating a macro

Use a call like:

```
strMacroVal = objCurrentProject.Macros.Item("<MacroName>").Expand([lngIssueID],
[oExpressionReturned], [rsRevHistoryRec])
```

Note: in the MqCenX20.CMqDefValsGenerator, we are passing a recordset of the current values being saved instead of the revision history recordset, since we need those values in these macros, and don't want to break compatibility on MqSQL (CMqMacro).

8.3. Working with CustomCode.bas

You can add scripting code into CustomCode.bas and MqCustomCode.bas.

These files are located in the HTML folder of every Web View.

These files have programming stubs ready for scripts to be plugged in.

Available event stubs are:

FUNCTION	DESCRIPTION	RETURN VALUE
OnAddNewRecord	This function will set the default values for fields in a new record.	not used
OnCopyRecord	This function will set the default values for fields in a copied record	not used
OnValidate	This function will allow user to add custom validation before saving the record. Here you can do validation specific to your needs.	Type: string Info: if the returned string is not empty, the save action is canceled and the string is displayed to the user
OnBeforeSave	This function will allow user to add custom code before saving the record. Here you can add code specific to your needs.	Type: string Info: You must return an empty string
OnFieldSave	This function will allow user to add custom code to further modify the record based on another field value. For example, in the Mq_OnFieldSave function, if we determine that the record was closed, we set the fields close date and close time. Argument: strChangedFields: a comma separated list of changed fields. Add: the related fields that you've changed	Type: boolean Info: return true if you changed the strChangedFields list

OnAfterSave	This function will allow user to add custom code after saving the record. Here you can add code specific to your needs.	Not used
OnAfterCommitSave	This function will allow user to add custom code after completely saving the record. Here you can add code specific to your needs.	Not used
OnSubstateChange	<p>This function allows customizing actions triggered after the Substate/Status/Progress field has been modified. The event occurs server side during the saving of the record from a Web view. In particular, when saving the new value of this field to the database.</p> <p>varSubState = new value of Substate/Status/Progress.</p> <p>It is in the process of being saved.</p> <p>rs = recordset with the latest values of fields for the current record in the main table (e.g. tblDts).</p> <p>objField = definition of the field being saved, which in this case is the Substate/Status/Progress.</p>	Returned not used. Argument strChangedFields: a comma seperated list of changed fields
OnUpdateSLA	This function will allow user to add custom code for updating the SLA from the view. Here you can add code specific to your needs.	Not used

dctFldVals

Contains a dictionary of the fields that have been modified and will be saved in the Save event.

To check if a given field has been updated:

```
dctFldVals.Exists(strFldID)
```

To Get the value of a given field:

```
lngVal = dctFldVals.Item(strFldID)
```

To Set the value of a given field:

```
dctFldVals.Item(strFldID)=""
```

Debugging

How can I log messages or see the values of variables?

Use the LogMsg function. E.g.

```
LogMsg "This is a message"
```

And change the variable DEBUG_ON to True, at the top of the MqCustomCode.bas

```
Private Const DEBUG_ON = False
```

The message will be recorded in the Windows Application Event log.

CHAPTER 9 - DATA MODEL (DATABASE SCHEMA)

9.1. Licenses Database

TlbMdbSystem

tMdbName	nVersion	dTime	Description
VizorMdbVersion	18	4/28/2018	Major Version of Census.mdb
SystemMdaVersion	1	4/17/2018	Major Version of Censys.mdw
VizorMdbMinorVersion	23	4/28/2018	Minor Version of Census mdb
VizorLibVersion	0		Major version of CensLib.mde

After modifications to the template editor, the table is updated based on the type of modifications.

Every time a user logs on, we verify the user's version (stored in the registry) against the version stored in this table. If the user's version is out-of-date, then we copy the latest version from the server.

TblProject Access

nID	TGroup
0	Admins
0	Developers
0	Guests
0	QA
0	Users
0	Managers
1	Admins
1	Developers
1	Managers
1	QA
1	Users

nID - ID of the project

Tgroup – group name

TblProjects

NID	tProjectNa	dCreateDa	tDatName	tDefName	tUsrName	tCenName
0	Demo		Demo01.dat	d:\Vizor97\demo0	Demo03.usr	Demo.cen
1	Project	4/17/2018	Project01.dat	D:\Temp\Project0	Project03.usr	Project.cen

nProjectVersion	fFieldsChanged	tProjectFilesLocation	nWebTimeOut
8	0	d:\Vizor97	10
1	0	D:\Temp	10

NProjectVersion:

When the admin modifies the template editor for a given project, the nProjectVersion is updated. The nProjectVersion is used to synchronise the .usr database. If a user is out-of-date on the nProjectVersion, we open tblDtsFields and update the following tables on the user's local machine: tblDtsTemporary, tblDtsDefault and tblFixInformationTemporary, tblFixInformationDefault.

Note that tblDtsTables has a list of all the tables that need to be updated when a project is modified.

Note also that the nProjectVersion is stored on the user's system in the .CEN file.

FFieldsChanged:

Flag to indicate that another user is editing the project. Only one user can edit a project at a given time.

NwebTimeOut:

Time in minutes to timeout a VizorWeb session.

tblSystemMessagesLicenses

fShutDown	tMessage
0	

FShutdown : 1= Template editor is modified. 2 = SysAdmin request to shutdown. 3 = Choice values were modified.

TMessage: Not used

When we need to shutdown users, we update this table with the appropriate fShutdown value.

Each client application constantly monitors this table and if a Shutdown message is received, the client application shuts down gracefully.

The time to wait for the machine to shutdown is currently hard coded to 2 minutes in ShutdownAllWorkstations() in basTemplateEditor1

9.2. Users Database

TblgroupPrivileges

Defines which group has access to which feature

This table is updated by the Security editor when adding or removing features for a given group.

tblNextUserID

nID
4

Used to assign an ID to a new user

TblUser

<i>User ID</i>	<i>User Name</i>	<i>Personal</i>	<i>Email Address</i>	<i>Company</i>	<i>Address</i>	<i>Phone</i>	<i>fDeleted</i>
-4	<Originator>	<Originator>	<Originator>				No
-3	<User>	<User>	<User>				No
-2	<Previous	<Previous	<Previous				No
-1	<Owner>	<Owner>	<Owner>				No
0	<None>	<None>	<None>				No
1	Admin	Admin	admin				No
2	demo	demo	demo				No
3	quest	quest	quest				No

Fdeleted: Flag to indicate that the user is deleted.

Flocked: Flag to indicate that someone is editing the data for this user

Foriginator: Flag to indicate that this user is an originator

9.3. DAT Database

TblAttachments

nID	TAttachments	fModified
-----	--------------	-----------

This table maintains all the attachments references found in tblDts.

FModified indicates that a field was modified. FModified is not really used in this table.

It is used in the temporary version of this table which is found in the .usr database.

TlbDts

This is the main table that tracks the defects. Only some fields are mandatory and cannot be removed by the user. All the other ones are customizable through the template editor. To see a list of the mandatory field, check the Vizor Beyond the Basics manual.

TShowInSummary: When a new defect is created, this field will contain the current user's name. In the summary we force the following SQL condition: `tShowInSummary = -1` or `tShowInSummary = CurrentUser()`

This guarantees that user A will not see new defects created by user B that are not saved yet.

When user B saves a new defect, the `tShowInSummary` is reset to `-1`.

NRevisionNumber: When a user saves a defect, we compare the `nRevisionNumber` in `tblDtsTemporary` with the one in `tblDts`. If the one in `tblDts` is more recent, then someone else has updated this defect in the meantime. In that case we popup a dialog box warning the user.

TblFixInformation

Contains the fix portion of the defect. The `nID` corresponds to the `nID` of `tblDts`.

This table also has a corresponding temporary table.

TblNextDefect

nID
51

Contains the next available defect number. If a user creates a new defect and then deletes it, the defect number is returned to `tblNextDefect`. `TblNextDefect` may contain more than one defect.

TblRevisionHistory

Contains the revision all the modified fields of all records.

TblSystemMessagesDat

fShutDown	TMessage
0	

Same as the table in Licenses Database.

By setting the Shutdown flag in this table, only the users of the same project will be affected.

When setting the Shutdown flag in the Licenses table, all Vizor users are affected.

9.4. DEF Database

TblDtsFields

nI	tTableName	tName	nDataType	nDataSiz	tCaption	fChoiceOrder	fRequire	
1	tblDts	nID	102	4	Record	0	0	
2	tblDts	dSubmittedDate	8		Submitted	0	0	
3	tblDts	dSubmittedTime	101		Submitted	0	0	
4	tblDts	dTargetDate	8		Target Date	0	0	
5	tblDts	dClosedDate	8		Closed Date	0	0	
6	tblDts	dClosedTime	101		Closed Time	0	0	
7	tblDts	dAssignedDate	8		Assigned	0	0	
8	tblDts	dAssignedTime	101		Assigned	0	0	
9	tblDts	dLastUpdateDate	8		Updated	0	0	
10	tblDts	dUpdateTime	101		Updated	0	0	
11	tblDts	nUserID	103	3	Owner	0	0	
12	tblDts	nSubmitterID	103	3	Submitter	0	0	
13	tblDts	nOriginatorID	103	3	Originator	0	0	
14	tblDts	tProduct	104	255	Product	0	-1	
15	tblDts	tVersion	104	255	Version	0	0	

16	tblDts	tTargetVersion	104	255	Target	0	0
17	tblDts	nFunctionalAreaID	103	3	Functional	0	0
18	tblDts	nPriorityID	103	3	Priority	-1	0
19	tblDts	nSeverityID	103	3	Severity	-1	-1
20	tblDts	nStateID	103	3	State	0	0
21	tblDts	nSubstateID	103	3	Substate	0	0
22	tblDts	nExternalNumber	102	3	External	0	0
23	tblDts	nTypeID	103	3	Type	0	-1
24	tblDts	tPlatform	104	255	Platform	0	0
25	tblDts	tVerifiedPlatform	104	255	Verified	0	0
26	tblDts	nHowFoundID	103	3	How Found	0	0
27	tblDts	nWorkaroundID	103	3	Workaround	0	0
28	tblDts	mWorkaroundDescriptio	12		Workaround	0	0
29	tblDts	mDetailedDescription	12		Detailed	0	0
30	tblDts	tBriefDescription	10	255	Brief	0	-1
31	tblDts	tRelatedDefect	102	3	Related	0	0
32	tblAttachments	tAttachments	10		Attachment	0	0
33	tblDts	tDuplicateDefect	10	255	Duplicate	0	0
34	tblDts	tNetwork	104	255	Network	0	0
35	tblDts	tTestSuite	10	255	Test Suite	0	0
36	tblDts	nRevisionNumber	102	4	Revision	0	0
37	tblFixInformatio	tFixedonPlatform	104	255	Fixed On	0	0
38	tblFixInformatio	dEstimatedFixTime	102	4	Estimated	0	0
39	tblFixInformatio	dActualFixTime	102	4	Actual Fix	0	0
40	tblFixInformatio	tFixedInVersion	104	255	Fixed In	0	0
41	tblFixInformatio	mFixInformation	12		Fix	0	0
42	tblUser_1	tEmailName	10	50	Submitter	0	0
43	tblUser	tEmailName	10	50	Originator	0	0
44	tblUser	tCompanyName	10	50	Originator	0	0
45	tblUser	tWorkPhone	10	50	Originator	0	0
46	tblUser	tAddress	10	50	Originator	0	0
47	tblUser	tFax	10	50	Originator	0	0
48	tblDts	nSubmitterID	103	3	Submitter	0	0
49	tblDts	nOriginatorID	103	3	Originator	0	0
50	tblFixInformatio	nFixedByID	103	3	Fixed By	0	0

This table is used to create/modify/manage all the fields in the current template. When the admin modifies the template using the template editor, he is actually editing tblDtsFields. If new fields are added, removed, etc. when the template editor is applied we add/remove fields in tblDts, tblFixInformation, etc.

Note that the temporary version of these tables is only updated when the user starts Vizor by verifying the fProjectVersion flag in tblProjects.

Note that all users have to be logged off when making these changes.

TTableName: Name of table where the field belongs

TName: Field name

FEditable: If false, then this field cannot be edited by the template editor

NrelatedField: Possibly no longer used

TAliasSourceTable: This defines the fields to be used in the data link or join when the data field is not in the primary table.

TBoundControlName: Name of the bound control at detail section of the from "frmDts"

TWhere: Where condition applied to the Choice list used for the single or multi-choice field.

TbIDtsFieldValueTables

Dts Field Value Tables	tSourceTableName	fDelete
tblFunctionalArea	tblFunctionalArea	No
tblHowFound	tblHowFound	No
tblNetWork	tblNetWork	No
tblPlatform	tblPlatform	No
tblPriority	tblPriority	No
tblProduct	tblProduct	No
tblSeverity	tblSeverity	No
tblState	tblState	No
tblSubstate	tblSubState	No
tblType	tblType	No
tblUser	tblUser	No
tblVersion	tblVersion	No
tblWorkaround	tblWorkaround	No

This table lists all the choice tables.

Note that DtsFieldValueTables and tSourceTable are the same. They should always have the same value.

Fdelete: flag to indicate that the choice table was deleted. Possibly not used. Needs more looking into.

TbIDtsTables

NID	tTableName	tTableNameTemp	tDefaultTableName	tBeforeSubmitFunc	tSub
-----	------------	----------------	-------------------	-------------------	------

8	tblDts	tblDtsTemporary	tblDtsDefault	DtsBeforeSubmitFunc	DtsS
9	TblFixInforma	tblFixInformationsTemp	tblFixInformationsDef		FixSu
10	tblAttachmen	tblAttachmentsTempora			

TTableName: table Name

TTableNameTemp: Name of corresponding temporary table (used when modifying or creating new defects)

TdefaultTableName: Name of default table used when creating a new defect (sets the default initial values)

TbeforeSumitFunc: function called before creating a new defect

TSubmitFunc: function called before saving an existing defect

TCopyRecordsFunc: function called after the save operation is completed whether it is successful or not. Typically just copies from the original table back to the temp table (undo) or vice versa to commit the changes.

TRevisionFunc: function called while looping though the fields of a defect but before saving the defect. This function typically adds records to the revision history table.

This table is used during the following operations:

create and save a new defect

update an existing defect

used when selecting records in the summary list. When a record is selected, we loop though tblDtsTables, and we check if this record is in the temporary tables. If it is not, then we get the name of the corresponding non-temporary table and copy the record to the temp table.

TblEmptyTableNames

Used by the Project Editor. When creating a new project if the user selects to copy the definitions and the data, then we do nothing. If the user selects to copy the definitions only, then we empty all the tables that are marked fSystemData = Yes. If the user selects not to copy anything, then we delete all the tables listed here.

Tname: table name

FSystemData: does it contain data or definitions?

TblFeatures

Contains a list of all the Vizor features.

TFeatureName; Name of feature

FVisible: this flag indicates that the feature is not visible in the Security editor. Note that some features are related. For example, the Add Report Editor and Add Report Page Editor depend on the Report Editor feature. If Report Editor is disabled, then all dependent features will be disabled (see tblFeaturesDependent)

TFunctionName: If a function is defined, then this function is called to determine if this feature should be enabled or not. For example, SendMail will call OnEmailSystemExist to determine if the user has a MAPI compliant email system. If he does, then Send Mail is enabled.

TblFeaturesDependent

nFeatureID	NRelatedFeatur
24	4
25	14

Lists all the features that are dependent on other ones. See note above (tblFeatures)

TblFieldAccess

NID: ID of the field in tblDtsFields

TGroup: group name

This table is used to determine who has privileges to access a given field. If a group does not have access to a field, then this field is disabled and is read-only.

This table is edited via the template editor.

TblMacros

ID	tName	tFunctionName	tTableName
0	<This Year>	GetMacroThisYear	
1	<Today>	GetMacroToday	
2	<Owner>	GetMacroOwner	
3	<Previous	GetMacroPreviousOwner	
6	<User>	GetMacroUser	
7	<This	GetMacroThisMonth	
8	<This	GetMacroThisQuarter	
9	<This Year>	GetMacroThisYear	
10	<User in		
11	<Originator>	GetMacroOriginator	

Macros are used in the following editor: Query, Notification, Report, When.

When a macro is selected in one the combo-boxes, we lookup the corresponding function and call it to expand the macro to its real value.

TtableName is not currently used.

tblMailContents

nID	tName
1	Notification Description
2	Detailed
3	Revision Record
4	Summary

Used by the notification editor to determine the contents of the notification message sent to the user.

Notification description: this is a user defined description.

Detailed: all the fields in the record are sent

Revision: only the modified fields are sent.

Summary: only the fields appearing in the current layout of the summary list are sent

tblNextObjectID

nNewQueryID	NNewSortID	nNewLayoutI	nNewReportI	nNewPageID	nNewNotificationID	nNewWhe
25	23	8	25	22	24	22

This is used by all Vizor editors to determine the next available ID for the give object.

We do not reuse deleted Ids.

Note that nNewUserID is not currently used.

tblNotification

ID	nParentTableI	Name	Mail Content	Description	Fields State
1	17	Owner\$Was\$	Notification Description.	The defect was re-	2
2	17	Product\$Was	Notification Description.	The Product was updated.	2
3	17	Description\$	Notification Description.	The Description was	2
4	17	Detailed\$Des	Notification Description.	The Detailed Description	2

This table maintains all the user defined notifications.

This table is edited via the Notification editor.

TName: this is an internal name based on the user typed caption (with \$ replacing the space).

NParentTableID: this is usually used to distinguish between local and global objects. In the case of notifications, there are only global objects so this field is not really useful. However for consistency with the other features, we need it.

FFieldsChanged: flag to indicate that the notification is currently being edited. This prevents other users from editing the same notification. Also indicates that the notification is dirty.

NWhenID: ID of the when condition

NWhenParentTableID: see NParentTableID

TblOperators

nI	tName
0	=
1	<>
2	<
3	>
4	<=
5	>=
6	Like
7	Not Like
8	Update
9	Contains

Used by the query editor and the simple query editor and the when editor.

TblSystemConditions

nID	tTotal	tName	tTest	tValue	tAndOr	nLevel	nParen
0		Owner	Update			0	0
1		Priority	Update			0	0

2		Product	Update			0	0
3		Brief	Update			0	0

This is the When condition defined by the When editor.

The when editor is made up of 2 parts: (a) the when condition and (b) the when definition

NID: this is the ID of the condition. Note that since a condition can consist of more than one record, NID is not unique. NID and nRow are used to uniquely identify a record.

TTotat: this defines the name of a new variable (total1, total2, etc.)

TName: Field name

TTest: the possible values are fetched from tblOperators.

NLevel: this is a hidden field that defines the indentation level of this condition. This is used to properly place the parenthesis around the expression.

NPARENT: the parent of the condition (based on the row number nRow)

Nchild: number of children

Nbrother: this is the row number of the immediate brother

NRow: this is the row of the condition

FWhenCondition: defines whether this is a definition or a condition.

NDtsfieldID: ID of the field

TblSystemCustomConditions

nID	tName	tCaption	nParentTableI	fFieldsChanged	fHasTotalVariables
0	Defect\$Is\$Re-	Defect Is Re-	15	2	No
1	Priority\$Is\$Up	Priority Is	15	0	No
2	Product\$Is\$U	Product Is	15	2	No
3	Description\$I	Description Is	15	0	No

NID- ID of the When condition as defined in tblSystemConditions

Tname : name of the condition

NParentTableID: see comment above.

Ffieldchanged: Used to lock multiple editing of the same condition and flag a dirty state.

TblSystemLayouts

nID	tAlign	nWidth	tTitle	nRow	nDtsFieldID
2	Right	7		2	1
3	Right	7		2	1
4	Right	7		2	1
2	Center	8		6	20

This defines the layout in the summary list.

TblSystemMessagesDef

fShutDown	Tmessage
0	

This is used to shutdown Vizor applications when 2 projects share the same definitions.

TblSystemQueries

nID	tName	tTest	tValue	tAndOr	nLevel	nParent	nChild
2	State	=	Open	And	0	0	0
2	Target Date	<	<Today>		0	0	0

7	Sub-state	=	Fix To Be		0	0	0
8	Sub-state	=	To Be Verified		0	0	0
9	Record	=	-1		0	0	0

This contains user-defined queries

TName: Field name

TTest: the possible values are fetched from tblOperators.

NLevel: this is a hidden field that defines the indentation level of this query. This is used to properly place the parenthesis around the expression.

NParent: the parent of the query (based on the row number nRow)

Nchild: number of children

Nbrother: this is the row number of the immediate brother

NRow: this is the row of the query

NDtsfieldID: ID of the field

TblSystemReportListing

nID	tFieldAlignme	tValueAlignm	tTitle	nValueWidth	nRow	nDtsFieldID
3	Right	Right	Record	6	1	1
2	Left	Left	Record	25	1	1
1	Right	Right	Record	6	1	1
4	Left	Left	Record	25	1	1

All the fields that define a Listing report.

tblSystemReportPageLayouts

nID	nItemsAcross	nRowSpacing	nColumnSpac	fSameAsDetai	nWidth	nHeight	nItemL
3	1	0	0	Yes	6.5	1	1
4	1	0	0	Yes	6.5	1	1
5	1	0	0	Yes	6.5	1	1

This table is no longer used.

tblSystemReportPageSections

nID	nSection	nIndent	tFontName	nFontSize	nFontWe	fFon	fFontUnd	tText
3	1	1	MS Sans Serif	8	0	No	No	
3	2	3	MS Sans Serif	8	0	No	No	= [Page]
3	3	2	Arial	14	700	No	No	Vizor Lis

Nsection: 1 = Header, 2 = footer , 3 = title

Nindent: left, right, center

tblSystemReportSummaries

nID	nRow	tName	tValue	tTitle	fShow	fXY	nColor
7	2	Total1	<All>		Yes	-1	
8	2	Total1	<All>		Yes	-1	

Summary report definition

FXY: If fXY = -1, then the data defined in the record is displayed in the Horizontal section. Otherwise, it is displayed in the Vertical section.

TblTableTypes

nID	TcustomTable	tDataTable	tDisplayStri	nTyp	nLoc	tTableReportPrefix
1	TblCustomLayouts	tblLayouts		3	1	LayoutL
2	TblSystemCustomLayouts	tblSystemLayouts	*	3	0	LayoutG
3	TblCustomSorts	tblSorts		4	1	SortL
4	TblSystemCustomSorts	tblSystemSorts	*	4	0	SortG
5	TblCustomQueries	tblQueries		5	1	QueryL
6	TblSystemCustomQueries	tblSystemQueries	*	5	0	QueryG

7	TblCustomReports	tblReportSummaries		0	1	SummaryL
8	TblSystemCustomReports	tblSystemReportSummaries	*	0	0	SummaryG
9	TblCustomReports	tblReportListings		1	1	ListingL
10	TblSystemCustomReports	tblSystemReportListings	*	1	0	ListingG
11	TblCustomReports	tblReportTimes		2	1	TimeL
12	TblSystemCustomReports	tblSystemReportTimes	*	2	0	TimeG
13	TblCustomReportPages	tblReportPageLayouts		7	1	PageL
13	TblCustomReportPages	tblReportPageSections		7	1	PageL
14	TblSystemCustomReportPa	tblSystemReportPageLayouts	*	7	0	PageG
14	TblSystemCustomReportPa	tblSystemReportPageSections	*	7	0	PageG
15	TblSystemCustomConditio	tblSystemConditions	*	6	0	
16		tblDts		8	0	
17	TblNotification	tblNotification	*	20	0	NotificationG
18	TblDtsFields	tblDtsFields		19	0	

This table is used in the following cases:

All the Vizor editors.

At startup to attach all the summary list tables (this is determined using the TABLE_QUERY_TYPE constant to determine which tables are used by the summary view. We should add a field here to identify if a table is used by the summary view and not depend on the current ID values in this table).

TDisplayStyle: string to distinguish between local and global tables

NType: these values are global constants that define the type of the object (query, layout, report, etc).

TTableReportPrefix: Prefix used when generating a report.

TPrefix: label used when displaying error/warning messages.

TblWho

Notification	User ID
--------------	---------

Used by the notification editor to determine who to send the notification to.

NotificationID: ID from tblNotification

UserID: ID of user from tblUser.

tblWorkaround

nID	TName
2	Yes
3	No

TblTabs

nID	tTabName	tSubFormName	tFormName
0	Overview	fsubDtsOvervie	fsubDtsOverview
1	Description	fsubDtsDescript	fsubDtsDescription
2	Detail	fsubDtsDetail	fsubDtsDetail
3	Fix	fsubDtsFix	fsubDtsFix
4	Origin	fsubDtsSubmitt	fsubDtsSubmitterInfo

tblRule

Table of main rules for Workflow and Business Rules.

tblRuleCondition

FK nRuleID

tblSetValue

Used to define actions that set field values, also known as Dependent Value rules.

FK nRuleID

tblPossibleValue

(for all actions of rules except SetValue)

FK nRuleID

CHAPTER 10 - BUSINESS RULES

This section refers to configuring and creating business rules in VIZOR.

10.1. File Business Rules

Each File Business Rule is stored in a separate file, which usually named using the convention:

br-[brief reference to what it does].asp

At runtime, these files are located in the folder:

CensusWebVD/<Web View>/**bizrls/**

Any file added to the folder will be loaded at runtime, it is not necessary to register the rule.

Like Business Rules defined in the Workflow Editor, they can be of the following types:

- PossibleValues: 0,
- SetValues: 1,
- HiddenFields: 2,
- VisibleFields: 7,
- Hidden_Tabs: 8,
- Required_Fields: 9,
- Disabled_Fields: 10,
- SetURLDynamicValues: 11,
- Disabled_Ticket: 22,
- CustomAction: 99

The most common type is PossibleValues, where the list of values of a single-choice field is filtered depending on the values of one or more fields.

Typical example of the definition of a business rule in the file would be:

```
m_objWorkflow.AddBizRule(  
    {  
        GlobalID:"<%=GlobalID%>",
```

```

        Conditions:[
            {
                FieldID: <F.ID:Category>, // Category
                ParamReplace: "%1",
                ValueAsText:true
            }
        ],

        Actions:[
            {
                ActionType:ActionType.PossibleValues,
                TargetFieldID: <F.ID:Owner>, // Owner
                ValueSource: "nID"
            }
        ],
        TriggerOnChange:true
    }
);

```

10.2. Optional Bultin-in File Business Rules

By default, when requesting an asset and using the minimum lead time (or provisioning time), if the desired requested time is less than the minimum, VIZOR will display a message and prevent the user from requesting the asset.

This behavior can be changed to a warning that still allows the user to submit the request.

In order to do that, you need to:

Export these fields to the Request Portal web view:

[Asset Type](#)

[Minimum Provisioning Time](#)

[Request Required By Date](#)

[Request Required By Time](#)

Add the optional files from

CustomizedFiles\#Project#Resources\Optional\bizrls

br-ValidateMinimunDateTime.asp

br-SetMinimumProvisioningTime.asp

To the Request Portal view in question:

CustomizedFiles\#Project#Resources\#WebView#Resources_Request_Assets\bizrls

Then, open CustomCode.Bas

Add this line inside OnValidate, before calling Mq_OnValidate:

```
ASSET_REQUEST_VALIDATE_MIN_ALLOC_TIME= False  
strValidate = Mq_OnValidate(objApp, lngRecord, dctFldVals)
```

and copy it to the appropriate customized files folder.

CHAPTER 11 - CONFIGURING LOCALIZATIONS AND TRANSLATIONS

This section refers to enabling and configuring languages and localizations in VIZOR.

11.1. How to change to a non-English language

This process makes the desired language the default one for all users.

1. Check what the ID of the desired language is.

Languages / Locales / Cultures

- en-US
- fr-CA
- es-MX
- de-DE

2. Edit **TmplGlobal.asa**

Look for this line:

```
' Load Active Localizations/Languages
Server.Execute("/%BaseVDName%/Localization/en-US/Resources.en-US.asp")
```

Add a line for enabling the desired language, for example:

```
Server.Execute("/%BaseVDName%/Localization/fr-CA/Resources.fr-CA.asp")
```

Then look for the line:

```
' Define the default Language/Culture
Session(SESS_VAR_LOCAL_CULTURE)="en-US"
```

And change it to the desired locale.

```
Session(SESS_VAR_LOCAL_CULTURE)="fr-CA"
```

3. Generate a Web view

Any Web view in that virtual directory. This will generate a new global.asa file based on the new **TmplGlobal.asa**.

4. Restart IIS

Either restart the services or execute the iisreset command.

You can also control the locale for a particular session from the URL in the Web browser by adding this text:

```
lc=fr-CA
```

So the logon URL could look like this:

```
https://try.vizor.cloud/lc=fr-CA
```

11.2. Folders

Localization and languages are stored in these folders:

Templates:

<CensusWebVD>/localization/

<CensusWebVD>/<Web view>/localization/

Runtime:

<CensusWebVD>/localization/

<CensusWebVD>/<Web view>/localization/

CHAPTER 12 - PROGRAMMING WITH LOCALIZATIONS, TRANSLATIONS AND RESOURCE STRINGS

This section refers to the modifications that are usually made to a VIZOR system in order to support non-English languages. The translated resource strings covered in this section are displayed in the VIZOR Web UI (Web Views).

12.1. Web Browser Side

`g_MqRef.RESX` is the global object available via javascript for browser-side localized/translated resources.

RESX is the object responsible of localizing strings according to the culture of the Web session (currently defined in the ASP session, when the ASP session starts). It's loaded automatically and dynamically once per Web session.

Library: `MqSysUtils.js` via the `g_MqRef` object/namespace. `SysUtils` loads the culture –specific via `/localization/Resources.js.asp`.

`MqSysutils` loads the resources using:

`LoadGlobalRes` (loaded automatically within the `MqSysUtils` itself)

`LoadViewRes` (called from `CensusMain` or specific windows opened, like `Revision History`)

How to Use it

- Add the key of the string/resource to the `/localization/en-US/ Resources.en-US.js`
- Get the string using javascript code like this:

```
g_MqRef.RESX[ ResourceKey ]
```

To use the localized resources in JS:

e.g.

```
alert(g_MqRef.RESX["Hello"]);
```

- You can also use it in an HTML tag in a declarative way. For this case, add the resource via the following attributes:

`translate="yes" data-res="reskey"`

the framework (via `mqsysutils`) will add the resource as text inside the tag. An example of the html will therefore be:

`<label translate="yes" data-res="CAPTION_USERNAME"> <label>`

which is also equivalent to the following using javascript code (non-declarative):

`<label ><script> document.write(g_MqRef.RESX["CAPTION_USERNAME"]); </script> <label>`

Requirements

- Add a reference to `MqSysUtils.js`.
- If you want to use the declarative way, you must also:
 - + Add a reference to `jquery` before `mqsysutils.js`
 - + Call this line in the onload of the html document.

`g_MqRef.ResMgr.ProcessAll(document);`

12.2. Server Side

There's a similar object for the server side, currently available in the ASP session.

In ASP Files

- Add the lines:

```
' Load Resources
Dim RESX
Set RESX=Application("Resources." & Session("Localization.Culture"))
' End of Load Resources
```

- Call it RESX(reskey)

The resources in the asp are loaded at the global.asa using a safe-threaded dictionary (global/asp application) object.

In Web Classes

You can get any ASP session variable to the html page in censusmain.htm via [WC@SES <name of variable>](#)

CHAPTER 13 - ASSET INTELLIGENCE AND CALCULATIONS

13.1. Asset Totals

VIZOR calculates automatically the total assets per asset type.

The totals that are automatically calculated and saved in the database per asset type are:

VIZOR Field	Database Field
Total Assets	Total_Assets
Total Allocated	Total_Allocated (same as for Bulk Assets)
Total Available	Total_Available

The fields are updated whenever an asset is created in VIZOR and regularly checked via the SWDisc.exe, a process launched by the Mq Issue Agent service.

You can view this information in the details of an asset type in the Manage Asset Types view. By default, the fields are in the Allocation tab.

14.1. Outgoing Emails

Emails to be sent from VIZOR are first added to tblMailQueue. This table is per project and its contents should change all the time if the Mq Issue Agent is running. The ideal state of this table is to have no records or very few, since it is the outgoing queue. Any records that stay in this table for long (depending on the defined interval for sending emails) usually indicate a problem with the email server or the specific email.

14.2. Incoming Emails

Incoming emails are taken by VIZOR per Email Integration rule. Each rule specifies a mailbox to monitor. This mailbox is regularly monitored by the service Mq Mail Integration and new emails processed to determine their workflow. Most emails are linked to tickets but some may be added to the Inbox queue or ignored. Ignored emails are not saved in VIZOR but all the other emails are in a format that is database friendly and searchable.

Each email integration has a timestamp that indicates the time after which emails will be processed by VIZOR in the next cycle.

An administrator can set back that timestamp and emails will be re-evaluated from that new time. This can be useful if for some reason some emails were failed to be processed. If the timestamp is set back, emails that were not processed before will be processed and emails that were processed will not be processed again. So, if an email already created a ticket, it will not create a new ticket again. If an email was added to the ticket conversation, it will not be added twice.

CHAPTER 15 - APENDICES

15.1. Department, Location and Company Fields

Location Fields:

	Name	ID	Caption	GUID	Default Value
Actions (new in u305)	Location	714	Location	Location	Location of Request Asset Owner (Requested_Ass et_Owner)
BugTrk (new in u305)	Location	714	Location	Location	Location of Contact (nOriginatorID)
ChangeRequests (new in u305)	Location	714	Location	Location	Location of Requestor Name (nOriginatorID)
Configuration (new in u305)	Location	714	Location	Location	Location of current User
HelpDesk (new in u305)	Location	714	Location	Location	Location of Contact (nOriginatorID)
Installations (new in u305)	Location	714	Location	Location	Location of current User
InventoriedSoftware (new in u305)	Location	714	Location	Location	Location of current User
KnowledgeBase (new in u305)	Location	714	Location	Location	Location of current User
LicenseMgr	Allocated_to_Loca tion	714	Allocated to Location	Location	Location of current User

Purchases&Agreements	Purchased_For_Location	944	Purchased For Location	Location	Location of current User
ReleaseMgmt (new in u305)	Location	714	Location	Location	Location of current User
Resources	Asset_Location	882 1196	Location Location (Overview Tab)	Location	Location of current User
Roles	Location	799	Location	Location	Location of current User
Users	Location	259	Location (Employment Tab)	Location	Location of current User

Department Fields:

	Name	ID	Caption	GUID
Actions (new in u304)	Department	713	Department	Department
BugTrk (new in u305)	Department	713	Department	Department
ChangeRequests (new in u305)	Department	713	Department	Department
Configuration (new in u305)	Department	713	Department	Department
HelpDesk (new in u305)	Department	713	Department	Department
Installations (new in u305)	Department	713	Department	Department
InventoriedSoftware (new in u305)	Department	713	Department	Department
KnowledgeBase (new in u305)	Department	713	Department	Department
LicenseMgr	Allocated_to_Department	713	Allocated to Department	Department
Purchases&Agreements	Purchased_For_Department	943	Purchased For Department	Department
ReleaseMgmt (new in u305)	Department	713	Department	Department
Resources	Allocated_to_Department	1019	Department (Allocation tab)	Department
Roles	Department	834	Department	Department
Users	Department	255 271	Department (Employment Tab) Employee Department (Person Tab)	Department

Company Fields:

	Name	ID	Caption	GUID
Actions	Company_Name	600	Company Name	Organization
BugTrk (new in u305)	Company_Name	600	Company Name	Organization
ChangeRequests	Company_Name	548 552	Company Name (Overview Tab) Company Name (Ticket Tab)	Organization
Configuration	Company_Name	600	Company Name	Organization
HelpDesk	Company_Name	600 604	Company Name Company Name (Ticket Tab)	Organization
Installations	Company_Name	600	Company Name	Organization
InventoriedSoftware	Company_Name	600	Company Name	Organization
KnowledgeBase (new in u305)	Company_Name	600	Company Name	Organization
LicenseManager	Company_Name	513	Company Name (Overview Tab)	Organization
Purchases&Agreements	Company_Name	600	Company Name	Organization
ReleaseMgmt	Company_Name	358 497 501	Company Name Company Name (Planning Tab) Company Name (Overview Tab)	Organization
Resources (new in u305)	Company_Name	600		Organization
Roles	Company_Name	600	Company Name	Organization
Users	nCompanyName	56 161 24	Company Name (Contact Tab) Organization Name Organization Name (Person Tab)	Organization

END OF DOCUMENT