Chapter 1 - Introduction

1.1. The DTS Model

1.2. General Guidelines for Customizations

1.3. Web View Types

Chapter 2 - Dts Fields

2.1. Data Types

2.2. Container Types

2.3. Relationship Types

2.4. Choice Tables

Chapter 3 - Web Browser API

3.1. Guidelines when making customizations to Web Views

3.2. CustomSettings.js

3.3. TmplCensusMain.htm

3.4. tmplRecord.htm
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
Chapter 9 - Data Model (Database Schema) ................................................................. 68

9.1. Licenses Database ......................................................................................... 68

  TblMdbSystem .................................................................................................. 68

  TblProject Access .......................................................................................... 68

  TblProjects ...................................................................................................... 69

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
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
    TblEmptyTableName ......................................................................................................... 76
    TblFeatures ...................................................................................................................... 77
    TblFeaturesDependent ................................................................................................. 77
    TblFieldAccess ............................................................................................................... 77
    TblMacros ....................................................................................................................... 78
    tblMailContents ............................................................................................................. 78
    tblNextObjectID ............................................................................................................ 79
    tblNotification ............................................................................................................... 79
    TblOperators .................................................................................................................. 80
    TblSystemConditions .................................................................................................... 80
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. WebView 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 VNFormTmpl.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 VNFormTmpl.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.
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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>ID</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Memo</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>Single Choice</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>Multi Choice</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>SCC</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Related Record</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>DateTime</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>SubRecordLink</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>HTML Rich Text</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>SQLRowVersion</td>
<td>121</td>
<td></td>
</tr>
</tbody>
</table>

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 known container types used by VIZOR*:

*Please note that others may have been created by customers or resellers.
<table>
<thead>
<tr>
<th>DataType</th>
<th>Container Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(all)</td>
<td>1</td>
<td>Default container type. It produces the default type of field from the Web View Editor.</td>
</tr>
<tr>
<td></td>
<td>2 Attachments</td>
<td>Attachments button that shows the number of attachments and it opens a dialog with the complete list of attachments.</td>
</tr>
<tr>
<td></td>
<td>3 VSS</td>
<td>VSS grid and controls.</td>
</tr>
<tr>
<td></td>
<td>4 Report Only</td>
<td>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.</td>
</tr>
</tbody>
</table>
|          | 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) |
<p>|          | 6 Hyperlink    | E.g. Parent Issue |
|          | 7 Alternate Choice List | |
|          | 8 Single-choice Rating | Display a single-choice as a rating with images (e.g. stars) |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Tasks</td>
<td>Tasks local to the project and shown in a list/grid table.</td>
</tr>
<tr>
<td>16</td>
<td>Append Memo</td>
<td>Memo field where new text is appended on save, always added. Usually used in the new Log Memo fields like Activity Log.</td>
</tr>
<tr>
<td>30</td>
<td>Total Time with Stopwatch</td>
<td>Stopwatch/timer control. Includes timer icons, buttons and functionality. E.g. Total Effort Time</td>
</tr>
</tbody>
</table>
| 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.

<table>
<thead>
<tr>
<th></th>
<th>Main</th>
<th>tAliasSourceTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>tblDts</td>
<td>nID; nRelRec</td>
</tr>
<tr>
<td>1</td>
<td>One to One, joined as defined by Main.nID = TABLE.nRelRec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where the joined fields are specified in tAliasSourceTable = nID; nRelRec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separated by ; if one field is absent, “nID” is assumed</td>
</tr>
<tr>
<td>2</td>
<td>One to Many, joined as defined by Main.nID = TABLE.nID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where the joined fields are specified in tAliasSourceTable = nID; nID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separated by ; if one field is absent, “nID” is assumed</td>
</tr>
<tr>
<td>3</td>
<td>Many to One</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>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</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The join is defined by Main.nID = TABLE1.nRelatedFieldID TABLE1.nID = TABLE2.nID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where the joined fields are specified in tAliasSourceTable = nID; nRelatedFieldID</td>
</tr>
</tbody>
</table>
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):

<table>
<thead>
<tr>
<th>Field in tblDtsFields</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>tFieldValuesTable</td>
<td>tblUser</td>
<td>Name of the database table where the choices come from.</td>
</tr>
<tr>
<td>nValueTableDSType</td>
<td>64</td>
<td>Numeric ID that defines the database where the table is.</td>
</tr>
<tr>
<td>tWhere</td>
<td>WHERE Security_Type=8</td>
<td>Where condition appended to the query of the specified table to get the value for the choices.</td>
</tr>
</tbody>
</table>
| 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. tmplRecord.htm

tmplRecord.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 tmplRecord.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 tmplRecord.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.tblTabs, 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.

Types based on the naming usually have names like:

fld_103_1_0.htmpl

which follows the naming convention of:

fld_data type_container type_sub type/attributes.htmpl

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

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ID%</td>
<td>ID of the field</td>
</tr>
<tr>
<td>%CtlName%</td>
<td>Control name of the field</td>
</tr>
<tr>
<td></td>
<td>(tblDtsFields)</td>
</tr>
<tr>
<td>%AssociatedFieldID%</td>
<td></td>
</tr>
<tr>
<td>%Caption%</td>
<td></td>
</tr>
<tr>
<td>%CaptionClass%</td>
<td></td>
</tr>
<tr>
<td>%TabID%</td>
<td></td>
</tr>
<tr>
<td>%DataViewID%</td>
<td></td>
</tr>
<tr>
<td>%ControlType%</td>
<td></td>
</tr>
<tr>
<td>%FieldTableName%</td>
<td>Table name of choices</td>
</tr>
<tr>
<td>%DataSize%</td>
<td></td>
</tr>
<tr>
<td>%FIdAttrs%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read only=0?</td>
</tr>
<tr>
<td>%TDClass_Caption%</td>
<td></td>
</tr>
<tr>
<td>%TDClass_Control%</td>
<td></td>
</tr>
<tr>
<td>%ColSpan%</td>
<td></td>
</tr>
<tr>
<td>%DefColSpan%</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>%BrowseAlternateChoiceListPostfix%</td>
<td></td>
</tr>
<tr>
<td>%ValueTableDSType%</td>
<td></td>
</tr>
<tr>
<td>%SearchURL%</td>
<td></td>
</tr>
<tr>
<td>%TableName%</td>
<td></td>
</tr>
<tr>
<td>%Tooltip%</td>
<td></td>
</tr>
<tr>
<td>%ProjectName%</td>
<td></td>
</tr>
<tr>
<td>%ReadOnly%</td>
<td></td>
</tr>
<tr>
<td>%Visible%</td>
<td></td>
</tr>
<tr>
<td>%Hidden%</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Template</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fld_Asset-Type_Model.htmpl</code></td>
<td>For displaying the asset type and the device model together, in a better-looking UI</td>
</tr>
<tr>
<td><code>fld_User_Image_Assets.htmpl</code></td>
<td>For displaying images of a person and an asset in readonly mode and with optional arrows. Used in the Check In/Out view.</td>
</tr>
<tr>
<td><code>fld_VoteQuestionFixed.htmpl</code></td>
<td>For displaying the question in approvals but using a fixed text</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fld_Image_RO_Users.htmpl</td>
<td>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.</td>
</tr>
<tr>
<td>fld_101_12_SpanTwoWidthOne.htmpl</td>
<td>Time field with Span Two Columns and Width One.</td>
</tr>
<tr>
<td>fld_HiddenField.htmpl</td>
<td>To force a field to be hidden regardless of the workflow/business rules.</td>
</tr>
<tr>
<td>fld_Image_RO.htmpl</td>
<td>For displaying an image of in readonly mode.</td>
</tr>
<tr>
<td>fld_NoField.htmpl</td>
<td>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. 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.  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. 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. When doing so, the field needs to use this template to the Web View Editor does not generate a duplicate html control.</td>
</tr>
</tbody>
</table>

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
  
  ```javascript
g_VWS.HTTPRequest()
  ```

- Get the fields collection
- Get the current record (VNRecord)

```java
get_VWS.RecordMgr().CurrentRecord()
```

- Get the current location

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

```java
get_VWS.MainWindow().BaseUrl()
```

which returns a path like:

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

- Get the current session URL object

```java
get_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.

```java
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:

```java
if(get_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.

```javascript
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.

```javascript
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
```
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.:

```javascript
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:

```html
<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:

```js
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:

```javascript
const vnRec = g_VWS.RecordMgr().CurrentRecord();
vnRec.IsNewRecord();
```

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:

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

or

```javascript
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:

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

Has:

| VNFormFields                  | Collection of form fields displayed in the main |
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 have 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 form element 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:

<table>
<thead>
<tr>
<th>Caption Control</th>
<th>The caption as specified in the Field Caption (Field Editor) or Web View Editor (Field Attribute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Control</td>
<td>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.</td>
</tr>
<tr>
<td>Value Control</td>
<td>The HTML element that has the value, like an INPUT, TEXT or SELECT, etc.</td>
</tr>
<tr>
<td>Extra Control</td>
<td>The container HTML with the extra UI elements that support the field. It could anything, from buttons, to maps, to images, etc.</td>
</tr>
<tr>
<td>GetHTMLDoc( )</td>
<td>Returns a reference to the HTML document where the DTS field control is.</td>
</tr>
<tr>
<td>GetFieldDef( )</td>
<td>Return a reference to the DtsField definition.</td>
</tr>
</tbody>
</table>

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:

```javascript
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.

```javascript
OpenModalWin(lngWidth,lngHeight,blnNearFullscreen,intMode);
```

3.21. Debugging

Add code for logging to facilitate future debugging.

Use WriteConsole

Suggested Format:

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

Example:

```javascript
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( )
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:

```javascript
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:
   ```javascript
   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":
   ```javascript
   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.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnBeforeLoadRec</td>
</tr>
<tr>
<td>OnAfterLoadRec</td>
</tr>
<tr>
<td>SaveBefore</td>
</tr>
<tr>
<td>SaveAfter</td>
</tr>
<tr>
<td>SubmitBefore</td>
</tr>
<tr>
<td>SubmitAfter</td>
</tr>
<tr>
<td>BatchSubmitBefore</td>
</tr>
<tr>
<td>BatchSubmitAfter</td>
</tr>
<tr>
<td>CancelBefore</td>
</tr>
<tr>
<td>CancelAfter;</td>
</tr>
<tr>
<td>AddNewBefore</td>
</tr>
<tr>
<td>AddNewAfter</td>
</tr>
<tr>
<td>CopyBefore</td>
</tr>
<tr>
<td>CopyAfter</td>
</tr>
<tr>
<td>DeleteBefore</td>
</tr>
<tr>
<td>DeleteAfter</td>
</tr>
</tbody>
</table>

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.

```javascript
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.

Eg.

```
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.asmx

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.

5.3. PowerShell consumer

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

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

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

```csharp
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:

```javascript
ret = ClientResStrings.DUE_DATE_MAX_REQUESTS_REACHED;
```
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:**

```vbnet
// 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
String. A string containing the field name and attachment info. This string must be in the format Name=File;fCopyFile=[1/0].

fCopyFile=1 means that the file is uploaded.

fCopyFile=0 means that the file is linked.

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()**

```vba
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()**

```vba
Dim objSession As IMqCenApp
Set objSession = New CMqCensusSession

'call OpenSession
```
IMqRecordsMgr.Initialize()

Dim objRecMgr As IMqRecordsMgr

Set objRecMgr = New CMqRecordsMgr

'call OpenProject
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%26EmployeeAssets&')`

### 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:

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


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.

<table>
<thead>
<tr>
<th>nID</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[nRuleID]</td>
<td>1</td>
</tr>
<tr>
<td>tName</td>
<td>&lt;User in Group&gt;</td>
</tr>
<tr>
<td>tTest</td>
<td>=</td>
</tr>
<tr>
<td>tValue</td>
<td></td>
</tr>
<tr>
<td>tAndOr</td>
<td>AND</td>
</tr>
</tbody>
</table>
### TblCondExpr (model)

New table for Condition Expressions:

<table>
<thead>
<tr>
<th>nID</th>
<th>10</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRow</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Type</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>nFieldType</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>tField</td>
<td>18</td>
<td>&lt;Owner&gt;</td>
<td>1</td>
</tr>
<tr>
<td>nTestType</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### Format Types for SQL Expression Components

For fields, operators, values and parameters

<table>
<thead>
<tr>
<th>Type(number)</th>
<th>Value</th>
<th>Text (displayed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro</td>
<td><code>&lt;Owner&gt;</code> or 23</td>
<td><code>&lt;Owner&gt;</code></td>
</tr>
<tr>
<td>DtsField</td>
<td>20</td>
<td>State</td>
</tr>
<tr>
<td>Parameter</td>
<td><code>&lt;ValOfField&gt;</code> 56&lt;/ValOfField&gt;</td>
<td>ComputerName</td>
</tr>
<tr>
<td>Simple Value</td>
<td>&quot;time%&quot;</td>
<td>&quot;time%&quot;</td>
</tr>
</tbody>
</table>

A single choice field has:

<table>
<thead>
<tr>
<th>Value</th>
<th>Text</th>
</tr>
</thead>
</table>
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:

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

// Select issue details
g_VWS.ReportMgr().Report("5002.10").QueryType=3;

// Set viewer type
g_VWS.ReportMgr().Report("5002.10").SetViewerType("pdf");

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

7.2. Summary list / Drill down

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>var m_blnDDInSL=true;</code></td>
<td>false will not enable drill-down for any query in that view. true will enable drill-down for all queries in that view.</td>
</tr>
<tr>
<td><code>var m_blnAllowDisableDD=true;</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

7.3. Details Form - Tabs

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| `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.
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):

<table>
<thead>
<tr>
<th>Macro</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDepartment</td>
<td>GetDepartment</td>
<td>Get the Department of an employee. If the person has multiple jobs, the department of the primary job is returned.</td>
</tr>
<tr>
<td>GetMacroUser</td>
<td>GetMacroUser</td>
<td>Get the full name of the current user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return=string</td>
</tr>
<tr>
<td>GetMacroIsUserInGroup</td>
<td>GetMacroIsUserInGroup</td>
<td>Used in a WHERE condition like &lt;User in Group&gt; = 'Admins'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check if the user who last updated the record specified by lngDtsRecordID is a member of the group specified in the oExpression.Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return=modified Expression object to give true or false</td>
</tr>
<tr>
<td>GetMacroMyWorkTeam</td>
<td>GetMacroMyWorkTeam</td>
<td>Get the name of the work team of the current user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return=string</td>
</tr>
<tr>
<td>GetFullName</td>
<td>GetFullName</td>
<td>Get the full name of a specified strLogonName</td>
</tr>
<tr>
<td>GetMacroProjectName</td>
<td>GetMacroProjectName</td>
<td>Get the name of the current project, e.g. &quot;HelpDesk&quot;, &quot;Resources&quot;, etc.</td>
</tr>
<tr>
<td>GetMacroStakeholders</td>
<td>GetMacroStakeholders</td>
<td>Get the list of users saved in the field Stakeholders of the record specified by lngDtsRecordID</td>
</tr>
<tr>
<td>GetDtsFieldValue</td>
<td>GetDtsFieldValue</td>
<td>Get the value of a DTS Field for the record specified by lngDtsRecordID</td>
</tr>
</tbody>
</table>
The field can be specified either by name (strFieldName) or by ID (lngFldID).

For choice value fields, the caller can also specify whether to get the value as IDs or as Names (bReturnValueIDs).

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFullNameByID</td>
<td>Get the full name of a user given the ID (lngUserID)</td>
</tr>
<tr>
<td>GetFieldValueFromUsers</td>
<td>Get the value of a field name specified by sFldName for a user given the ID (lngUserID). sFldName is the name of the physical field in tblUser.</td>
</tr>
<tr>
<td>GetIDByLogonName</td>
<td>Get the ID of the user matching the specified logon name (strLogonName)</td>
</tr>
<tr>
<td>GetIDByFullName</td>
<td>Get the ID of the user matching the specified full name (strFullName)</td>
</tr>
<tr>
<td>GetUserIDByFieldValue</td>
<td>Get the ID of the user matching the value strFieldValue in the field strField</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
<th>RETURN VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnAddNewRecord</td>
<td>This function will set the default values for fields in a new record.</td>
<td>not used</td>
</tr>
<tr>
<td>OnCopyRecord</td>
<td>This function will set the default values for fields in a copied record</td>
<td>not used</td>
</tr>
</tbody>
</table>
| 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 |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnAfterSave</td>
<td>This function will allow user to add custom code after saving the record.</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>Here you can add code specific to your needs.</td>
<td></td>
</tr>
<tr>
<td>OnAfterCommitSave</td>
<td>This function will allow user to add custom code after completely saving the record.</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>Here you can add code specific to your needs.</td>
<td></td>
</tr>
<tr>
<td>OnSubstateChange</td>
<td>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.</td>
<td>Returned not used.</td>
</tr>
<tr>
<td></td>
<td><code>varSubState = new value of Substate/Status/Progress.</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>It is in the process of being saved.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>rs = recordset with the latest values of fields for the current record in the main table (e.g. tblDts).</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>objField = definition of the field being saved, which in this case is the Substate/Status/Progress.</code></td>
<td></td>
</tr>
<tr>
<td>OnUpdateSLA</td>
<td>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.</td>
<td>Not used</td>
</tr>
</tbody>
</table>

**Arguments used in MqCustomCode**
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.
9.1. Licenses Database

**TblMdbSystem**

<table>
<thead>
<tr>
<th>tMdbName</th>
<th>nVersion</th>
<th>dTime</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VizorMdbVersion</td>
<td>18</td>
<td>4/28/2018</td>
<td>Major Version of Census.mdb</td>
</tr>
<tr>
<td>VizorLibVersion</td>
<td>0</td>
<td></td>
<td>Major version of CensLib.mde</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>nID</th>
<th>TGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Admins</td>
</tr>
<tr>
<td>0</td>
<td>Developers</td>
</tr>
<tr>
<td>0</td>
<td>Guests</td>
</tr>
<tr>
<td>0</td>
<td>QA</td>
</tr>
<tr>
<td>0</td>
<td>Users</td>
</tr>
<tr>
<td>0</td>
<td>Managers</td>
</tr>
<tr>
<td>1</td>
<td>Admins</td>
</tr>
<tr>
<td>1</td>
<td>Developers</td>
</tr>
<tr>
<td>1</td>
<td>Managers</td>
</tr>
<tr>
<td>1</td>
<td>QA</td>
</tr>
<tr>
<td>1</td>
<td>Users</td>
</tr>
</tbody>
</table>

nID - ID of the project
Tgroup – group name

TblProjects

<table>
<thead>
<tr>
<th>NID</th>
<th>tProjectNa</th>
<th>dCreateDa</th>
<th>tDatName</th>
<th>tDefName</th>
<th>tUsrName</th>
<th>tCenName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Demo</td>
<td></td>
<td>Demo01.dat</td>
<td>d:\Vizor97\demo0</td>
<td>Demo03.usr</td>
<td>Demo.cen</td>
</tr>
<tr>
<td>1</td>
<td>Project</td>
<td>4/17/2018</td>
<td>Project01.dat</td>
<td>D:\Temp\Project0</td>
<td>Project03.usr</td>
<td>Project.cen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nProjectVersion</th>
<th>fFieldsChanged</th>
<th>tProjectFilesLocation</th>
<th>nWebTimeOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>d:\Vizor97</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>D:\Temp</td>
<td>10</td>
</tr>
</tbody>
</table>

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 : 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

<table>
<thead>
<tr>
<th>nID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Used to assign an ID to a new user
### TblUser

<table>
<thead>
<tr>
<th>User ID</th>
<th>User Name</th>
<th>Personal</th>
<th>Email Address</th>
<th>Company</th>
<th>Address</th>
<th>Phone</th>
<th>fDeleted</th>
<th>Flocked</th>
<th>Foriginator</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>&lt;Originator&gt;</td>
<td>&lt;Originator&gt;</td>
<td>&lt;Originator&gt;</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>-3</td>
<td>&lt;User&gt;</td>
<td>&lt;User&gt;</td>
<td>&lt;User&gt;</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>-2</td>
<td>&lt;Previous&gt;</td>
<td>&lt;Previous&gt;</td>
<td>&lt;Previous&gt;</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>-1</td>
<td>&lt;Owner&gt;</td>
<td>&lt;Owner&gt;</td>
<td>&lt;Owner&gt;</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>&lt;None&gt;</td>
<td>&lt;None&gt;</td>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Admin</td>
<td>Admin</td>
<td>admin</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>demo</td>
<td>demo</td>
<td>demo</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>guest</td>
<td>guest</td>
<td>guest</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>nID</th>
<th>TAttachments</th>
<th>fModified</th>
</tr>
</thead>
</table>

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.

---

**TblDts**
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 tShowInsSummary = 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

<table>
<thead>
<tr>
<th>nID</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
</tr>
</tbody>
</table>

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.
Contains the revision all the modified fields of all records.

**TblSystemMessagesDat**

<table>
<thead>
<tr>
<th>fShutDown</th>
<th>TMessage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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           | nDataTyp | nDataSiz | tCaption   | fChoiceOrder | fRequire | f editable | f visible | nRelatedField | tLabelCaption | tAliasSourceTable | tBoundControlName | tDestinationTable | tWhere |
|----|------------|-----------------|----------|----------|------------|--------------|------------|-----------|------------|------------|---------------|----------------|-------------------|-------------------|------------------|--------|
| 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            | 0          |           |            |            |               |                |                   |                   |                |        |
| 15 | tblDts     | tVersion        | 104      | 255      | Version    | 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 form "frmDts"

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

### TblDtsFieldValueTables

<table>
<thead>
<tr>
<th>Dts Field Value Tables</th>
<th>tSourceTableName</th>
<th>fDelete</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblFunctionalArea</td>
<td>tblFunctionalArea</td>
<td>No</td>
</tr>
<tr>
<td>tblHowFound</td>
<td>tblHowFound</td>
<td>No</td>
</tr>
<tr>
<td>tblNetWork</td>
<td>tblNetWork</td>
<td>No</td>
</tr>
<tr>
<td>tblPlatform</td>
<td>tblPlatform</td>
<td>No</td>
</tr>
<tr>
<td>tblPriority</td>
<td>tblPriority</td>
<td>No</td>
</tr>
<tr>
<td>tblProduct</td>
<td>tblProduct</td>
<td>No</td>
</tr>
<tr>
<td>tblSeverity</td>
<td>tblSeverity</td>
<td>No</td>
</tr>
<tr>
<td>tblState</td>
<td>tblState</td>
<td>No</td>
</tr>
<tr>
<td>tblSubstate</td>
<td>tblSubState</td>
<td>No</td>
</tr>
<tr>
<td>tblType</td>
<td>tblType</td>
<td>No</td>
</tr>
<tr>
<td>tblUser</td>
<td>tblUser</td>
<td>No</td>
</tr>
<tr>
<td>tblVersion</td>
<td>tblVersion</td>
<td>No</td>
</tr>
<tr>
<td>tblWorkaround</td>
<td>tblWorkaround</td>
<td>No</td>
</tr>
</tbody>
</table>

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.

### TblDtsTables

<table>
<thead>
<tr>
<th>NID</th>
<th>tTableName</th>
<th>tTableNameTemp</th>
<th>tDefaultTableName</th>
<th>tBeforeSubmitFunc</th>
<th>tSubmitFunc</th>
<th>tCopyRecordsFunc</th>
<th>tRevisionFunc</th>
</tr>
</thead>
</table>
TTable: TableName: table Name
TTableTemp: 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)
TbeforeSubmitFunc: 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

<table>
<thead>
<tr>
<th>nFeatureID</th>
<th>NRelatedFeature</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>14</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ID</th>
<th>tName</th>
<th>tFunctionName</th>
<th>tTableName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;This Year&quot;</td>
<td>GetMacroThisYear</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&quot;Today&quot;</td>
<td>GetMacroToday</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&quot;Owner&quot;</td>
<td>GetMacroOwner</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&quot;Previous&quot;</td>
<td>GetMacroPreviousOwner</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&quot;User&quot;</td>
<td>GetMacroUser</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>&quot;This&quot;</td>
<td>GetMacroThisMonth</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>&quot;This&quot;</td>
<td>GetMacroThisQuarter</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>&quot;This Year&quot;</td>
<td>GetMacroThisYear</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>&quot;User in&quot;</td>
<td>GetMacroThisYear</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>&quot;Originator&quot;</td>
<td>GetMacroOriginator</td>
<td></td>
</tr>
</tbody>
</table>

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.

**tTable**Name is not currently used.

### tblMailContents

<table>
<thead>
<tr>
<th>nID</th>
<th>tName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Notification Description</td>
</tr>
<tr>
<td>2</td>
<td>Detailed</td>
</tr>
<tr>
<td>3</td>
<td>Revision Record</td>
</tr>
<tr>
<td>4</td>
<td>Summary</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>nNewQueryID</th>
<th>NNewSortID</th>
<th>nNewLayoutID</th>
<th>nNewReportID</th>
<th>nNewPageID</th>
<th>nNewNotificationID</th>
<th>nNewWhenID</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>23</td>
<td>8</td>
<td>25</td>
<td>22</td>
<td>24</td>
<td>22</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ID</th>
<th>nParentTableID</th>
<th>Name</th>
<th>Mail Content</th>
<th>Description</th>
<th>Fields State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>OwnerWas</td>
<td>Notification Description</td>
<td>The defect was re-assigned.</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>ProductWas</td>
<td>Notification Description</td>
<td>The Product was updated.</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>Description$</td>
<td>Notification Description</td>
<td>The Description was updated</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>Detailed$Des</td>
<td>Notification Description</td>
<td>The Detailed Description</td>
<td>2</td>
</tr>
</tbody>
</table>

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).

NParenTableID: 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

<table>
<thead>
<tr>
<th>nI</th>
<th>tName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>=</td>
</tr>
<tr>
<td>1</td>
<td>&lt;&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;</td>
</tr>
<tr>
<td>3</td>
<td>&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;=</td>
</tr>
<tr>
<td>5</td>
<td>&gt;=</td>
</tr>
<tr>
<td>6</td>
<td>Like</td>
</tr>
<tr>
<td>7</td>
<td>Not Like</td>
</tr>
<tr>
<td>8</td>
<td>Update</td>
</tr>
<tr>
<td>9</td>
<td>Contains</td>
</tr>
</tbody>
</table>

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

### TblSystemConditions

<table>
<thead>
<tr>
<th>nID</th>
<th>tTotal</th>
<th>tName</th>
<th>tTest</th>
<th>tValue</th>
<th>tAndOr</th>
<th>nLevel</th>
<th>nParent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Owner</td>
<td>Update</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Priority</td>
<td>Update</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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.

TTotal: 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

Nhbrother: 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

<table>
<thead>
<tr>
<th>nID</th>
<th>tName</th>
<th>tCaption</th>
<th>nParentTableI</th>
<th>fFieldsChanged</th>
<th>fHasTotalVariables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Defect Is Re-</td>
<td>Defect Is Re-</td>
<td>15</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Priority Is Up</td>
<td>Priority Is</td>
<td>15</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Product Is Updated</td>
<td>Product Is Updated</td>
<td>15</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Description Is Updated</td>
<td>Description Is Updated</td>
<td>15</td>
<td>0</td>
<td>No</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>nID</th>
<th>tAlign</th>
<th>nWidth</th>
<th>tTitle</th>
<th>nRow</th>
<th>nDtsFieldID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Right</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Right</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Right</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Center</td>
<td>8</td>
<td></td>
<td>6</td>
<td>20</td>
</tr>
</tbody>
</table>

This defines the layout in the summary list.

**TblSystemMessagesDef**

<table>
<thead>
<tr>
<th>fShutDown</th>
<th>Tmessage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

**TblSystemQueries**

<table>
<thead>
<tr>
<th>nID</th>
<th>tName</th>
<th>tTest</th>
<th>tValue</th>
<th>tAndOr</th>
<th>nLevel</th>
<th>nParent</th>
<th>nChild</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>State</td>
<td>=</td>
<td>Open</td>
<td>And</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Target Date</td>
<td>&lt;</td>
<td>&lt;Today&gt;</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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

Nbrotger: this is the row number of the immediate brother

NRow: this is the row of the query

NDtsfieldID: ID of the field

### TblSystemReportListing

<table>
<thead>
<tr>
<th>nID</th>
<th>tFieldAlignme</th>
<th>tValueAlignm</th>
<th>tTitle</th>
<th>nValueWidth</th>
<th>nRow</th>
<th>nDtsFieldID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Right</td>
<td>Right</td>
<td>Record</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Left</td>
<td>Left</td>
<td>Record</td>
<td>25</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Right</td>
<td>Right</td>
<td>Record</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Left</td>
<td>Left</td>
<td>Record</td>
<td>25</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

All the fields that define a Listing report.

### TblSystemReportPageLayouts

<table>
<thead>
<tr>
<th>nID</th>
<th>nItemsAcross</th>
<th>nRowSpacing</th>
<th>nColumnSpac</th>
<th>fSameAsDetail</th>
<th>nWidth</th>
<th>nHeight</th>
<th>nItemL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
<td>6.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
<td>6.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
<td>6.5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
This table is no longer used.

### tblSystemReportPageSections

<table>
<thead>
<tr>
<th>nID</th>
<th>nSection</th>
<th>nIndent</th>
<th>tFontName</th>
<th>nFontSize</th>
<th>nFontWe</th>
<th>fFont</th>
<th>fFontUnd</th>
<th>tText</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>MS Sans Serif</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>[Page]</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>MS Sans Serif</td>
<td>8</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>Arial</td>
<td>14</td>
<td>700</td>
<td>No</td>
<td>No</td>
<td>Vizor Listing</td>
</tr>
</tbody>
</table>

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

Nindent: left, right, center

### tblSystemReportSummaries

<table>
<thead>
<tr>
<th>nID</th>
<th>nRow</th>
<th>tName</th>
<th>tValue</th>
<th>tTitle</th>
<th>fShow</th>
<th>fXY</th>
<th>nColumnsWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td>Total1</td>
<td>&lt;All&gt;</td>
<td>Yes</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Total1</td>
<td>&lt;All&gt;</td>
<td>Yes</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>nID</th>
<th>TcustomTable</th>
<th>tDataTable</th>
<th>tDisplayStri</th>
<th>nTyp</th>
<th>nLoc</th>
<th>tTableReportPrefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TblCustomLayouts</td>
<td>tblLayouts</td>
<td></td>
<td>3</td>
<td>1</td>
<td>LayoutL</td>
</tr>
<tr>
<td>2</td>
<td>TblSystemCustomLayouts</td>
<td>tblSystemLayouts</td>
<td></td>
<td>3</td>
<td>0</td>
<td>LayoutG</td>
</tr>
<tr>
<td>3</td>
<td>TblCustomSorts</td>
<td>tblSorts</td>
<td></td>
<td>4</td>
<td>1</td>
<td>SortL</td>
</tr>
<tr>
<td>4</td>
<td>TblSystemCustomSorts</td>
<td>tblSystemSorts</td>
<td></td>
<td>4</td>
<td>0</td>
<td>SortG</td>
</tr>
<tr>
<td>5</td>
<td>TblCustomQueries</td>
<td>tblQueries</td>
<td></td>
<td>5</td>
<td>1</td>
<td>QueryL</td>
</tr>
<tr>
<td>6</td>
<td>TblSystemCustomQueries</td>
<td>tblSystemQueries</td>
<td></td>
<td>5</td>
<td>0</td>
<td>QueryG</td>
</tr>
</tbody>
</table>
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).

TDisplayString: 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.

<table>
<thead>
<tr>
<th>Tbl</th>
<th>Table</th>
<th>Description</th>
<th>ID</th>
<th>Used</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TblCustomReports</td>
<td>TblReportSummaries</td>
<td>0</td>
<td>1</td>
<td>SummaryL</td>
<td></td>
</tr>
<tr>
<td>TblSystemCustomReports</td>
<td>TblSystemReportSummaries</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>SummaryG</td>
</tr>
<tr>
<td>TblCustomReports</td>
<td>TblReportListings</td>
<td>1</td>
<td>1</td>
<td>ListingL</td>
<td></td>
</tr>
<tr>
<td>TblSystemCustomReports</td>
<td>TblSystemReportListings</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>ListingG</td>
</tr>
<tr>
<td>TblCustomReports</td>
<td>TblReportTimes</td>
<td>2</td>
<td>1</td>
<td>TimeL</td>
<td></td>
</tr>
<tr>
<td>TblSystemCustomReports</td>
<td>TblSystemReportTimes</td>
<td>*</td>
<td>2</td>
<td>0</td>
<td>TimeG</td>
</tr>
<tr>
<td>TblCustomReportPages</td>
<td>TblReportPageLayouts</td>
<td>7</td>
<td>1</td>
<td>PageL</td>
<td></td>
</tr>
<tr>
<td>TblCustomReportPages</td>
<td>TblReportPageSections</td>
<td>7</td>
<td>1</td>
<td>PageL</td>
<td></td>
</tr>
<tr>
<td>TblSystemCustomReportPages</td>
<td>TblSystemReportPageLayouts</td>
<td>*</td>
<td>7</td>
<td>0</td>
<td>PageG</td>
</tr>
<tr>
<td>TblSystemCustomReportPages</td>
<td>TblSystemReportPageSections</td>
<td>*</td>
<td>7</td>
<td>0</td>
<td>PageG</td>
</tr>
<tr>
<td>TblSystemCustomConditions</td>
<td>TblSystemConditions</td>
<td>*</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TblDts</td>
<td>TblDts</td>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TblNotification</td>
<td>TblNotification</td>
<td>*</td>
<td>20</td>
<td>0</td>
<td>NotificationG</td>
</tr>
<tr>
<td>TblDtsFields</td>
<td>TblDtsFields</td>
<td>19</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TblWho

<table>
<thead>
<tr>
<th>Notification</th>
<th>User ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used by the notification editor to determine who to send the notification to.</td>
<td></td>
</tr>
</tbody>
</table>

NotificationID: ID from tblNotification
UserID: ID of user from tblUser.

```
tblWorkaround

<table>
<thead>
<tr>
<th>nID</th>
<th>TName</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
</tr>
</tbody>
</table>
```

```
TblTabs

<table>
<thead>
<tr>
<th>nID</th>
<th>tTabName</th>
<th>tSubFormName</th>
<th>tFormName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Overview</td>
<td>fsubDtsOvervie</td>
<td>fsubDtsOverview</td>
</tr>
<tr>
<td>1</td>
<td>Description</td>
<td>fsubDtsDescrip</td>
<td>fsubDtsDescription</td>
</tr>
<tr>
<td>2</td>
<td>Detail</td>
<td>fsubDtsDetail</td>
<td>fsubDtsDetail</td>
</tr>
<tr>
<td>3</td>
<td>Fix</td>
<td>fsubDtsFix</td>
<td>fsubDtsFix</td>
</tr>
<tr>
<td>4</td>
<td>Origin</td>
<td>fsubDtsSubmitter</td>
<td>fsubDtsSubmitterInfo</td>
</tr>
</tbody>
</table>
```

```
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:

```csharp
m_objWorkFlow.AddBizRule(
{
    GlobalID:"<%=GlobalID%>",
```
10.2. Optional Built-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
Add the optional files from

CustomizedFiles\#Project\#Resources\Optional\bizrls

  br-ValidateMinimumDateTime.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:

```vbnet
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:

   ```
   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:

   ```
   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

\[ g_{\text{MqRef}}.\text{RESX} \] is the global object available via javascript for browser-side localized/translated resources.

\[ \text{RESX} \] 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: \( \text{MqSysUtils.js} \) via the \( g_{\text{MqRef}} \) object/namespace. \( \text{SysUtils} \) loads the culture–specific via /localization/Resources.js.asp.

\( \text{MqSysutils} \) 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:

\[
g_{\text{MqRef}}.\text{RESX}[ \text{ResourceKey} ]
\]

To use the localized resources in JS:

\[
\text{e.g.}
\]

\[
\text{alert}(g_{\text{MqRef}}.\text{RESX}["Hello"]);
\]

- You can also use it in an HTML tag in a declarative way. For this case, add the resource via the following attributes:
the framework (via mqsysutils) will add the resource as text inside the tag. An example of the html will therefore be:

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

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

```html
<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.

```javascript
  g_MqRef.ResMgr.ProcessAll(document);
```
There's a similar object for the server side, currently available in the ASP session.

**In ASP Files**
- Add the lines:
  ```vbscript
  ' 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>`
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:

<table>
<thead>
<tr>
<th>VIZOR Field</th>
<th>Database Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Assets</td>
<td>Total_Assets</td>
</tr>
<tr>
<td>Total Allocated</td>
<td>Total_Allocated (same as for Bulk Assets)</td>
</tr>
<tr>
<td>Total Available</td>
<td>Total_Available</td>
</tr>
</tbody>
</table>

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.
CHAPTER 14 - EMAILS

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.
15.1. Department, Location and Company Fields

Location Fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Caption</th>
<th>GUID</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of Request Asset Owner Requested_Asset_Owner (Requested_Asset_Owner)</td>
</tr>
<tr>
<td>BugTrk (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of Contact nOriginatorID</td>
</tr>
<tr>
<td>ChangeRequests (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of Requestor Name nOriginatorID</td>
</tr>
<tr>
<td>Configuration (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of current User</td>
</tr>
<tr>
<td>HelpDesk (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of Contact nOriginatorID</td>
</tr>
<tr>
<td>Installations (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of current User</td>
</tr>
<tr>
<td>InventoriedSoftware (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of current User</td>
</tr>
<tr>
<td>KnowledgeBase (new in u305)</td>
<td>Location 714</td>
<td>Location</td>
<td>Location</td>
<td>Location of current User</td>
</tr>
<tr>
<td>LicenseMgr</td>
<td>Allocated to Location 714</td>
<td>Allocated to Location</td>
<td>Location</td>
<td>Location of current User</td>
</tr>
<tr>
<td>Purchases &amp; Agreements</td>
<td>Purchased For Location</td>
<td>944</td>
<td>Location of current User</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------</td>
<td>-----</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>Release Mgmt (new in u305)</td>
<td>Location</td>
<td>714</td>
<td>Location of current User</td>
<td></td>
</tr>
<tr>
<td>Resources</td>
<td>Asset Location</td>
<td>882 1196</td>
<td>Location (Overview Tab)</td>
<td></td>
</tr>
<tr>
<td>Roles</td>
<td>Location</td>
<td>799</td>
<td>Location of current User</td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td>Location</td>
<td>259</td>
<td>Location of current User</td>
<td></td>
</tr>
</tbody>
</table>
Department Fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Caption</th>
<th>GUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions (new in u304)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>BugTrk (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>ChangeRequests (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>Configuration (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>HelpDesk (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>Installations (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>InventoriedSoftware (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>KnowledgeBase (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>LicenseMgr</td>
<td>713</td>
<td>Allocated to Department</td>
<td>Department</td>
</tr>
<tr>
<td>Purchases&amp;Agreements</td>
<td>943</td>
<td>Purchased For Department</td>
<td>Department</td>
</tr>
<tr>
<td>ReleaseMgmt (new in u305)</td>
<td>713</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>Resources</td>
<td>1019</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>Roles</td>
<td>834</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td>Users</td>
<td>255</td>
<td>Department</td>
<td>Department</td>
</tr>
<tr>
<td></td>
<td>271</td>
<td>Department</td>
<td>Department</td>
</tr>
</tbody>
</table>
## Company Fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Caption</th>
<th>GUID</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>BugTrk</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><em>(new in u305)</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ChangeRequests</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>548</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td></td>
<td>552</td>
<td>Company Name <em>(Overview Tab)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company Name <em>(Ticket Tab)</em></td>
<td></td>
</tr>
<tr>
<td><strong>Configuration</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>HelpDesk</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name <em>(Ticket Tab)</em></td>
<td>Organization</td>
</tr>
<tr>
<td></td>
<td>604</td>
<td>Company Name <em>(Overview Tab)</em></td>
<td></td>
</tr>
<tr>
<td><strong>Installations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>InventoriedSoftware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><em>(new in u305)</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LicenseManager</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>513</td>
<td>Company Name <em>(Overview Tab)</em></td>
<td>Organization</td>
</tr>
<tr>
<td><strong>Purchases&amp;Agreements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>ReleaseMgmt</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>358</td>
<td>Company Name <em>(Planning Tab)</em></td>
<td>Organization</td>
</tr>
<tr>
<td></td>
<td>497</td>
<td>Company Name <em>(Overview Tab)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>501</td>
<td>Company Name <em>(Contact Tab)</em></td>
<td></td>
</tr>
<tr>
<td><strong>Resources</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>(new in u305)</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>Roles</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>600</td>
<td>Company Name</td>
<td>Organization</td>
</tr>
<tr>
<td><strong>Users</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nCompany_Name</td>
<td>56</td>
<td>Company Name <em>(Contact Tab)</em></td>
<td>Organization</td>
</tr>
<tr>
<td></td>
<td>161</td>
<td>Company Name <em>(Person Tab)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>Company Name <em>(Organization Name)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company Name <em>(Organization Name)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company Name <em>(Organization Name)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company Name <em>(Organization Name)</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company Name <em>(Organization Name)</em></td>
<td></td>
</tr>
</tbody>
</table>