Imports System.ComponentModel

Public Class MyForm

Inherits DevExpress.XtraEditors.XtraForm

Public Structure FormAttributesData

Public CanAdd As Boolean

Public CanEdit As Boolean

Public CanDelete As Boolean

End Structure

Public Enum ViewModeEnum

ReadMode = 1

EditMode = 0

End Enum

<Browsable(False), DefaultValue(True)> \_

Public Property IsNew As Boolean

Public Property ID As Integer

<Browsable(False), DefaultValue(False)> \_

Public Property IsSaved As Boolean

<Browsable(False), DefaultValue(False)> \_

Public Property CanClose As Boolean

Public FormAttribute As FormAttributesData

Public Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

StartPosition = FormStartPosition.CenterScreen

End Sub

Public Sub SetFormAttribute()

If Me.Tag Is Nothing OrElse Me.Tag.ToString.Length = 0 Then Exit Sub

Dim strFilter As String = "IDModule=" & Me.Tag

Dim aRow() As DataRow = oClassActiveUser.UserAccessLevel.Select(strFilter)

If aRow.Count = 0 Then

With FormAttribute

.CanAdd = False

.CanDelete = False

.CanEdit = False

End With

Else

With FormAttribute

.CanAdd = Convert.ToBoolean(aRow(0)("CanAdd"))

.CanEdit = Convert.ToBoolean(aRow(0)("CanEdit"))

.CanDelete = Convert.ToBoolean(aRow(0)("CanDelete"))

End With

End If

AppliedAC(Me, True)

End Sub

Private Sub AppliedAC(ByVal \_ControlContainer As Object, Optional ByVal ViewMode As ViewModeEnum = MyForm.ViewModeEnum.ReadMode)

For Each oCtrl As Control In \_ControlContainer.Controls

If oCtrl.Tag Is Nothing OrElse oCtrl.Tag.ToString.Length = 0 Then

If oCtrl.HasChildren Then

AppliedAC(oCtrl)

End If

Continue For

End If

Select Case oCtrl.Tag.ToString.ToLower

Case "view"

oCtrl.Enabled = CBool(ViewMode)

Case "input"

oCtrl.Enabled = Not CBool(ViewMode)

Case "add"

oCtrl.Enabled = FormAttribute.CanAdd

Case "edit"

oCtrl.Enabled = FormAttribute.CanEdit

Case "save"

oCtrl.Enabled = Not CBool(ViewMode)

Case "cancel"

oCtrl.Enabled = True

End Select

If TypeOf oCtrl Is LabelControl OrElse TypeOf oCtrl Is GroupControl Then

oCtrl.Enabled = True

End If

If oCtrl.HasChildren Then

AppliedAC(oCtrl)

End If

Next

End Sub

Public Sub SwitchMode(ByVal \_ControlContainer As Object, Optional ByVal IsEditMode As Boolean = False)

For Each oCtrl As Control In \_ControlContainer.Controls

If oCtrl.Tag Is Nothing OrElse oCtrl.Tag.ToString.Length = 0 Then

If oCtrl.HasChildren Then

SwitchMode(oCtrl, IsEditMode)

End If

Continue For

End If

Select Case oCtrl.Tag.ToString.ToLower

Case "view"

oCtrl.Enabled = Not IsEditMode

Case "input"

oCtrl.Enabled = IsEditMode

Case "add"

oCtrl.Enabled = Not IsEditMode

Case "edit"

oCtrl.Enabled = Not IsEditMode

Case "delete"

oCtrl.Enabled = Not IsEditMode

Case "save"

oCtrl.Enabled = IsEditMode

Case "cancel"

oCtrl.Enabled = True

End Select

If TypeOf oCtrl Is LabelControl OrElse TypeOf oCtrl Is GroupControl Then

oCtrl.Enabled = True

End If

If oCtrl.HasChildren Then

SwitchMode(oCtrl, IsEditMode)

End If

Next

If Not IsEditMode Then

AppliedAC(Me, ViewModeEnum.ReadMode)

End If

End Sub

Private Sub MyForm\_Activated(sender As Object, e As EventArgs) Handles Me.Activated

SuspendLayout()

MdiParent = MainForm

ResumeLayout(True)

End Sub

Private Sub MyForm\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If Not CanClose Then e.Cancel = True

End Sub

Private Sub MyForm\_FormClosed(sender As Object, e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Try

'toostrip

Catch ex As Exception

End Try

End Sub

Private Sub MyForm\_Load(sender As Object, e As System.EventArgs) Handles Me.Load

Try

CanClose = False

If Not Me.Tag Is Nothing Or Me.Tag.ToString.Length > 0 Then

SetFormAttribute()

End If

Catch ex As Exception

End Try

End Sub

End Class

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> \_

Partial Class myLine

Inherits DevExpress.XtraEditors.XtraUserControl

'UserControl overrides dispose to clean up the component list.

<System.Diagnostics.DebuggerNonUserCode()> \_

Protected Overrides Sub Dispose(ByVal disposing As Boolean)

If disposing AndAlso components IsNot Nothing Then

components.Dispose()

End If

MyBase.Dispose(disposing)

End Sub

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

<System.Diagnostics.DebuggerStepThrough()> \_

Private Sub InitializeComponent()

Me.LabelControl1 = New DevExpress.XtraEditors.LabelControl()

Me.SuspendLayout()

'

'LabelControl1

'

Me.LabelControl1.AutoSizeMode = DevExpress.XtraEditors.LabelAutoSizeMode.None

Me.LabelControl1.LineVisible = True

Me.LabelControl1.Location = New System.Drawing.Point(3, -7)

Me.LabelControl1.Name = "LabelControl1"

Me.LabelControl1.Size = New System.Drawing.Size(351, 30)

Me.LabelControl1.TabIndex = 0

Me.LabelControl1.Text = "LabelControl1"

'

'myLine

'

Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)

Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font

Me.Controls.Add(Me.LabelControl1)

Me.Name = "myLine"

Me.Size = New System.Drawing.Size(376, 18)

Me.ResumeLayout(False)

End Sub

Friend WithEvents LabelControl1 As DevExpress.XtraEditors.LabelControl

End Class

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> \_

Partial Class MyForm

Inherits DevExpress.XtraEditors.XtraForm

'Form overrides dispose to clean up the component list.

<System.Diagnostics.DebuggerNonUserCode()> \_

Protected Overrides Sub Dispose(ByVal disposing As Boolean)

If disposing AndAlso components IsNot Nothing Then

components.Dispose()

End If

MyBase.Dispose(disposing)

End Sub

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

<System.Diagnostics.DebuggerStepThrough()> \_

Private Sub InitializeComponent()

Me.SuspendLayout()

'

'MyForm

'

Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)

Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font

Me.ClientSize = New System.Drawing.Size(284, 261)

Me.Name = "MyForm"

Me.Text = "MyForm"

Me.ResumeLayout(False)

End Sub

End Class

<?xml version="1.0" encoding="utf-8"?>

<root>

<!--

Microsoft ResX Schema

Version 2.0

The primary goals of this format is to allow a simple XML format

that is mostly human readable. The generation and parsing of the

various data types are done through the TypeConverter classes

associated with the data types.

Example:

... ado.net/XML headers & schema ...

<resheader name="resmimetype">text/microsoft-resx</resheader>

<resheader name="version">2.0</resheader>

<resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms, ...</resheader>

<resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms, ...</resheader>

<data name="Name1"><value>this is my long string</value><comment>this is a comment</comment></data>

<data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>

<data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">

<value>[base64 mime encoded serialized .NET Framework object]</value>

</data>

<data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-microsoft.net.object.bytearray.base64">

<value>[base64 mime encoded string representing a byte array form of the .NET Framework object]</value>

<comment>This is a comment</comment>

</data>

There are any number of "resheader" rows that contain simple

name/value pairs.

Each data row contains a name, and value. The row also contains a

type or mimetype. Type corresponds to a .NET class that support

text/value conversion through the TypeConverter architecture.

Classes that don't support this are serialized and stored with the

mimetype set.

The mimetype is used for serialized objects, and tells the

ResXResourceReader how to depersist the object. This is currently not

extensible. For a given mimetype the value must be set accordingly:

Note - application/x-microsoft.net.object.binary.base64 is the format

that the ResXResourceWriter will generate, however the reader can

read any of the formats listed below.

mimetype: application/x-microsoft.net.object.binary.base64

value : The object must be serialized with

: System.Runtime.Serialization.Formatters.Binary.BinaryFormatter

: and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.soap.base64

value : The object must be serialized with

: System.Runtime.Serialization.Formatters.Soap.SoapFormatter

: and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.bytearray.base64

value : The object must be serialized into a byte array

: using a System.ComponentModel.TypeConverter

: and then encoded with base64 encoding.

-->

<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xsd:import namespace="http://www.w3.org/XML/1998/namespace" />

<xsd:element name="root" msdata:IsDataSet="true">

<xsd:complexType>

<xsd:choice maxOccurs="unbounded">

<xsd:element name="metadata">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" />

</xsd:sequence>

<xsd:attribute name="name" use="required" type="xsd:string" />

<xsd:attribute name="type" type="xsd:string" />

<xsd:attribute name="mimetype" type="xsd:string" />

<xsd:attribute ref="xml:space" />

</xsd:complexType>

</xsd:element>

<xsd:element name="assembly">

<xsd:complexType>

<xsd:attribute name="alias" type="xsd:string" />

<xsd:attribute name="name" type="xsd:string" />

</xsd:complexType>

</xsd:element>

<xsd:element name="data">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />

<xsd:element name="comment" type="xsd:string" minOccurs="0" msdata:Ordinal="2" />

</xsd:sequence>

<xsd:attribute name="name" type="xsd:string" use="required" msdata:Ordinal="1" />

<xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />

<xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />

<xsd:attribute ref="xml:space" />

</xsd:complexType>

</xsd:element>

<xsd:element name="resheader">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />

</xsd:sequence>

<xsd:attribute name="name" type="xsd:string" use="required" />

</xsd:complexType>

</xsd:element>

</xsd:choice>

</xsd:complexType>

</xsd:element>

</xsd:schema>

<resheader name="resmimetype">

<value>text/microsoft-resx</value>

</resheader>

<resheader name="version">

<value>2.0</value>

</resheader>

<resheader name="reader">

<value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>

</resheader>

<resheader name="writer">

<value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>

</resheader>

</root>

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> \_

Partial Class RibbonMainForm

Inherits DevExpress.XtraBars.Ribbon.RibbonForm

'UserControl overrides dispose to clean up the component list.

<System.Diagnostics.DebuggerNonUserCode()> \_

Protected Overrides Sub Dispose(ByVal disposing As Boolean)

If disposing AndAlso components IsNot Nothing Then

components.Dispose()

End If

MyBase.Dispose(disposing)

End Sub

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

<System.Diagnostics.DebuggerStepThrough()> \_

Private Sub InitializeComponent()

Me.RibbonStatusBar1 = New DevExpress.XtraBars.Ribbon.RibbonStatusBar()

Me.RibbonControl1 = New DevExpress.XtraBars.Ribbon.RibbonControl()

Me.RibbonPage1 = New DevExpress.XtraBars.Ribbon.RibbonPage()

Me.RibbonPageGroup1 = New DevExpress.XtraBars.Ribbon.RibbonPageGroup()

CType(Me.RibbonControl1, System.ComponentModel.ISupportInitialize).BeginInit()

Me.SuspendLayout()

'

'RibbonStatusBar1

'

Me.RibbonStatusBar1.Location = New System.Drawing.Point(0, 403)

Me.RibbonStatusBar1.Name = "RibbonStatusBar1"

Me.RibbonStatusBar1.Ribbon = Me.RibbonControl1

Me.RibbonStatusBar1.Size = New System.Drawing.Size(633, 31)

'

'RibbonControl1

'

Me.RibbonControl1.ExpandCollapseItem.Id = 0

Me.RibbonControl1.Items.AddRange(New DevExpress.XtraBars.BarItem() {Me.RibbonControl1.ExpandCollapseItem})

Me.RibbonControl1.Location = New System.Drawing.Point(0, 0)

Me.RibbonControl1.MaxItemId = 1

Me.RibbonControl1.Name = "RibbonControl1"

Me.RibbonControl1.Pages.AddRange(New DevExpress.XtraBars.Ribbon.RibbonPage() {Me.RibbonPage1})

Me.RibbonControl1.RibbonStyle = DevExpress.XtraBars.Ribbon.RibbonControlStyle.Office2013

Me.RibbonControl1.Size = New System.Drawing.Size(633, 144)

Me.RibbonControl1.StatusBar = Me.RibbonStatusBar1

'

'RibbonPage1

'

Me.RibbonPage1.Groups.AddRange(New DevExpress.XtraBars.Ribbon.RibbonPageGroup() {Me.RibbonPageGroup1})

Me.RibbonPage1.Name = "RibbonPage1"

Me.RibbonPage1.Text = "RibbonPage1"

'

'RibbonPageGroup1

'

Me.RibbonPageGroup1.Name = "RibbonPageGroup1"

Me.RibbonPageGroup1.Text = "RibbonPageGroup1"

'

'RibbonMainForm

'

Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)

Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font

Me.ClientSize = New System.Drawing.Size(633, 434)

Me.Controls.Add(Me.RibbonStatusBar1)

Me.Controls.Add(Me.RibbonControl1)

Me.Name = "RibbonMainForm"

Me.Ribbon = Me.RibbonControl1

Me.StatusBar = Me.RibbonStatusBar1

CType(Me.RibbonControl1, System.ComponentModel.ISupportInitialize).EndInit()

Me.ResumeLayout(False)

Me.PerformLayout()

End Sub

Friend WithEvents RibbonStatusBar1 As DevExpress.XtraBars.Ribbon.RibbonStatusBar

Friend WithEvents RibbonControl1 As DevExpress.XtraBars.Ribbon.RibbonControl

Friend WithEvents RibbonPage1 As DevExpress.XtraBars.Ribbon.RibbonPage

Friend WithEvents RibbonPageGroup1 As DevExpress.XtraBars.Ribbon.RibbonPageGroup

End Class

Imports System.ComponentModel

Public Class RibbonMainForm

Public Structure FormAttributesData

Public CanAdd As Boolean

Public CanEdit As Boolean

Public CanDelete As Boolean

End Structure

Public Enum ViewModeEnum

ReadMode = 1

EditMode = 0

End Enum

<Browsable(False), DefaultValue(True)> \_

Public Property IsNew As Boolean

<Browsable(False), DefaultValue(False)> \_

Public Property IsSaved As Boolean

<Browsable(False), DefaultValue(False)> \_

Public Property CanClose As Boolean

Public FormAttribute As FormAttributesData

Public Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

StartPosition = FormStartPosition.CenterScreen

End Sub

Public Sub SetFormAttribute()

If Me.Tag Is Nothing OrElse Me.Tag.ToString.Length = 0 Then Exit Sub

Dim strFilter As String = "IDModule=" & Me.Tag

Dim aRow() As DataRow = oClassActiveUser.UserAccessLevel.Select(strFilter)

If aRow.Count = 0 Then

With FormAttribute

.CanAdd = False

.CanDelete = False

.CanEdit = False

End With

Else

With FormAttribute

.CanAdd = Convert.ToBoolean(aRow(0)("CanAdd"))

.CanEdit = Convert.ToBoolean(aRow(0)("CanEdit"))

.CanDelete = Convert.ToBoolean(aRow(0)("CanDelete"))

End With

End If

AppliedAC(Me, True)

End Sub

Private Sub AppliedAC(ByVal \_ControlContainer As Object, Optional ByVal ViewMode As ViewModeEnum = MyForm.ViewModeEnum.ReadMode)

For Each oCtrl As Control In \_ControlContainer.Controls

If oCtrl.Tag Is Nothing OrElse oCtrl.Tag.ToString.Length = 0 Then

If oCtrl.HasChildren Then

AppliedAC(oCtrl)

End If

Continue For

End If

Select Case oCtrl.Tag.ToString.ToLower

Case "view"

oCtrl.Enabled = CBool(ViewMode)

Case "input"

oCtrl.Enabled = Not CBool(ViewMode)

Case "add"

oCtrl.Enabled = FormAttribute.CanAdd

Case "edit"

oCtrl.Enabled = FormAttribute.CanEdit

Case "save"

oCtrl.Enabled = Not CBool(ViewMode)

Case "cancel"

oCtrl.Enabled = True

End Select

If TypeOf oCtrl Is LabelControl OrElse TypeOf oCtrl Is GroupControl Then

oCtrl.Enabled = True

End If

If oCtrl.HasChildren Then

AppliedAC(oCtrl)

End If

Next

End Sub

Public Sub SwitchMode(ByVal \_ControlContainer As Object, Optional ByVal IsEditMode As Boolean = False)

For Each oCtrl As Control In \_ControlContainer.Controls

If oCtrl.Tag Is Nothing OrElse oCtrl.Tag.ToString.Length = 0 Then

If oCtrl.HasChildren Then

SwitchMode(oCtrl, IsEditMode)

End If

Continue For

End If

Select Case oCtrl.Tag.ToString.ToLower

Case "view"

oCtrl.Enabled = Not IsEditMode

Case "input"

oCtrl.Enabled = IsEditMode

Case "add"

oCtrl.Enabled = Not IsEditMode

Case "edit"

oCtrl.Enabled = Not IsEditMode

Case "delete"

oCtrl.Enabled = Not IsEditMode

Case "save"

oCtrl.Enabled = IsEditMode

Case "cancel"

oCtrl.Enabled = True

End Select

If TypeOf oCtrl Is LabelControl OrElse TypeOf oCtrl Is GroupControl Then

oCtrl.Enabled = True

End If

If oCtrl.HasChildren Then

SwitchMode(oCtrl, IsEditMode)

End If

Next

If Not IsEditMode Then

AppliedAC(Me, ViewModeEnum.ReadMode)

End If

End Sub

Private Sub MyForm\_Activated(sender As Object, e As EventArgs) Handles Me.Activated

SuspendLayout()

MdiParent = MainForm

ResumeLayout(True)

End Sub

Private Sub MyForm\_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

If Not CanClose Then e.Cancel = True

End Sub

Private Sub MyForm\_FormClosed(sender As Object, e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Try

'toostrip

Catch ex As Exception

End Try

End Sub

Private Sub MyForm\_Load(sender As Object, e As System.EventArgs) Handles Me.Load

Try

CanClose = False

If Not Me.Tag Is Nothing Or Me.Tag.ToString.Length > 0 Then

SetFormAttribute()

End If

Catch ex As Exception

End Try

End Sub

End Class

USE SROI

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*MODIFY THE VALUES BELOW TO SUIT YOUR NEEDS\*/

DECLARE @DBName nvarchar(100)=N'<Your Database>';

DECLARE @ProcName nvarchar(100)=N'<Your Proc Name>';

DECLARE @DBRoleName nvarchar(100)=N'<Role that should have exec Rights>';

DECLARE @TableName nvarchar(100)=N'<Your Table Name>';

DECLARE @ViewName nvarchar(100)=N'<Your View Name>';

DECLARE @OrderBy nvarchar(100)=N'<Your Field Name>';

DECLARE @OrderByDir nvarchar(4)=N'ASC';

DECLARE @AUTHOR nvarchar(50) ='<Your Name & Company>';

DECLARE @DESC nvarchar(100) ='<Proc Information>'; -- Ex. 'User Data' will return 'Description : Upsert User Data'

DECLARE @ISACTIVE\_SEL bit =0; --Set to 1 if your table has a Bit field named IsActive

/\*DO NOT MODIFY BELOW THIS LINE!!!\*/

DECLARE @NNND char(23) ='NOT\_NULLABLE\_NO\_DEFAULT';

DECLARE @NNWD char(22) ='NOT\_NULLABLE\_W\_DEFAULT';

DECLARE @NBLE char(8) ='NULLABLE';

DECLARE @LEGEND nvarchar(max);

DECLARE @PRIMARY\_KEY nvarchar(100);

--Set up Legend

SET @LEGEND = N'USE [' + @DBName + N'];' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'SET ANSI\_NULLS ON' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'SET QUOTED\_IDENTIFIER ON' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'-- ===================================================================' + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'-- Author : ' + @AUTHOR + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'-- Create date : ' + CONVERT(nvarchar(30),GETDATE(),101) + CHAR(13) + CHAR(10)

SET @LEGEND = @LEGEND + N'-- Revised date: ' + CHAR(13) + CHAR(10)

--Get Primary Key Field

SELECT TOP 1 @PRIMARY\_KEY = COLUMN\_NAME

FROM INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE

WHERE OBJECTPROPERTY(OBJECT\_ID(constraint\_name), 'IsPrimaryKey') = 1 AND TABLE\_NAME = @TableName AND TABLE\_CATALOG = @DBName;

DECLARE TableCol Cursor FOR

SELECT c.TABLE\_SCHEMA, c.TABLE\_NAME, c.COLUMN\_NAME, c.DATA\_TYPE, c.CHARACTER\_MAXIMUM\_LENGTH

, IIF(c.COLUMN\_NAME='RowVersion',@NBLE,IIF(c.COLUMN\_NAME=@PRIMARY\_KEY,@NBLE,IIF(c.IS\_NULLABLE = 'NO' AND c.COLUMN\_DEFAULT IS NULL,@NNND,IIF(c.IS\_NULLABLE = 'NO' AND c.COLUMN\_DEFAULT IS NOT NULL,@NNWD,@NBLE)))) AS [NULLABLE\_TYPE]

FROM INFORMATION\_SCHEMA.Columns c INNER JOIN

INFORMATION\_SCHEMA.Tables t ON c.TABLE\_NAME = t.TABLE\_NAME

WHERE t.Table\_Catalog = @DBName

AND t.TABLE\_TYPE = 'BASE TABLE'

AND t.TABLE\_NAME = @TableName

ORDER BY [NULLABLE\_TYPE], c.ORDINAL\_POSITION;

DECLARE @TableSchema varchar(100), @cTableName varchar(100), @ColumnName varchar(100);

DECLARE @DataType varchar(30), @CharLength int, @NullableType varchar(30);

DECLARE @PARAMETERS nvarchar(max);

DECLARE @INSERT\_FIELDS nvarchar(max),@INSERT\_VALUES nvarchar(max);

DECLARE @UPDATE\_VALUES nvarchar(max);

SET @PARAMETERS ='@MyID int,';

SET @INSERT\_FIELDS ='';

SET @INSERT\_VALUES ='';

SET @UPDATE\_VALUES ='';

-- open the cursor

OPEN TableCol

-- get the first row of cursor into variables

FETCH NEXT FROM TableCol INTO @TableSchema, @cTableName, @ColumnName, @DataType, @CharLength, @NullableType

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @ColumnName NOT IN('Created','CreatedBy','Modified','ModifiedBy')

BEGIN

SET @PARAMETERS=@PARAMETERS + '@' + IIF(@ColumnName=@PRIMARY\_KEY,'ID',@ColumnName) + ' ' + iif(@CharLength IS NULL,@DataType,@DataType + '(' +

CAST(@CharLength AS nvarchar(10)) + ')') + IIF(@NullableType=@NNND OR @NullableType=@NNWD,',','=NULL,');

IF @ColumnName <> @PRIMARY\_KEY AND @ColumnName <> N'RowVersion'

BEGIN

SET @INSERT\_FIELDS=@INSERT\_FIELDS + '[' + @ColumnName + '],';

SET @INSERT\_VALUES=@INSERT\_VALUES + '@' + IIF(@ColumnName=@PRIMARY\_KEY,'ID',@ColumnName) + ',';

SET @UPDATE\_VALUES=@UPDATE\_VALUES + '[' + @ColumnName + ']=@' + IIF(@ColumnName=@PRIMARY\_KEY,'ID',@ColumnName) + ',';

END

END

FETCH NEXT FROM TableCol INTO @TableSchema, @cTableName, @ColumnName, @DataType, @CharLength, @NullableType

END;

SET @PARAMETERS=LEFT(@PARAMETERS,LEN(@PARAMETERS)-1)

SET @INSERT\_FIELDS=LEFT(@INSERT\_FIELDS,LEN(@INSERT\_FIELDS)-1)

SET @INSERT\_VALUES=LEFT(@INSERT\_VALUES,LEN(@INSERT\_VALUES)-1)

SET @UPDATE\_VALUES=LEFT(@UPDATE\_VALUES,LEN(@UPDATE\_VALUES)-1)

-- ----------------

-- clean up cursor

-- ----------------

CLOSE TableCol;

DEALLOCATE TableCol;

--Print Upsert Statement

PRINT N'/\*\*\*\*\*\* Object: StoredProcedure [dbo].[' + @ProcName + '\_ups] Script Date: ' + CAST(GETDATE() AS nvarchar(30)) + ' \*\*\*\*\*\*/' + CHAR(13) + CHAR(10)

PRINT @LEGEND;

PRINT N'-- Description : Upsert ' + @DESC + CHAR(13) + CHAR(10)

PRINT N'-- ===================================================================' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName + '\_ups]' + CHAR(13) + CHAR(10);

PRINT N' (' + @PARAMETERS + N')' + CHAR(13) + CHAR(10);

PRINT N'AS' + CHAR(13) + CHAR(10)

PRINT N'BEGIN' + CHAR(13) + CHAR(10)

PRINT N' SET CONTEXT\_INFO @MyID;' + CHAR(13) + CHAR(10)

PRINT N' IF @ID IS NULL OR @ID = 0' + CHAR(13) + CHAR(10)

PRINT N' BEGIN' + CHAR(13) + CHAR(10)

PRINT N' INSERT INTO [dbo].[' + @TableName + ']' + CHAR(13) + CHAR(10)

PRINT N' (' + @INSERT\_FIELDS + N')' + CHAR(13) + CHAR(10)

PRINT N' VALUES' + CHAR(13) + CHAR(10)

PRINT N' (' + @INSERT\_VALUES + N');' + CHAR(13) + CHAR(10)

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] WHERE [ID] = SCOPE\_IDENTITY();' + CHAR(13) + CHAR(10)

PRINT N' END' + CHAR(13) + CHAR(10)

PRINT N' ELSE' + CHAR(13) + CHAR(10)

PRINT N' BEGIN' + CHAR(13) + CHAR(10)

PRINT N' UPDATE [dbo].[' + @TableName + ']' + CHAR(13) + CHAR(10)

PRINT N' SET ' + @UPDATE\_VALUES + CHAR(13) + CHAR(10)

PRINT N' WHERE ([' + @PRIMARY\_KEY + '] = @ID) AND ([RowVersion] = @RowVersion);' + CHAR(13) + CHAR(10)

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] WHERE [ID] = @ID;' + CHAR(13) + CHAR(10)

PRINT N' END' + CHAR(13) + CHAR(10)

PRINT N'END' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role

PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '\_ups] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName + '\_ups] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

--Print Select Statement

PRINT N'/\*\*\*\*\*\* Object: StoredProcedure [dbo].[' + @ProcName + '\_sel] Script Date: ' + CAST(GETDATE() AS nvarchar(30)) + ' \*\*\*\*\*\*/' + CHAR(13) + CHAR(10)

PRINT @LEGEND;

PRINT N'-- Description : Select ' + @DESC + CHAR(13) + CHAR(10)

PRINT N'-- ===================================================================' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName + '\_sel]' + CHAR(13) + CHAR(10);

PRINT N' (@MyID int, @ID int=NULL' + IIF(@ISACTIVE\_SEL = 1,', @IsActive bit=NULL','') + ')' + CHAR(13) + CHAR(10);

PRINT N'AS' + CHAR(13) + CHAR(10)

PRINT N'BEGIN' + CHAR(13) + CHAR(10)

PRINT N' SET CONTEXT\_INFO @MyID;' + CHAR(13) + CHAR(10)

PRINT N' IF @ID IS NULL OR @ID = 0' + CHAR(13) + CHAR(10)

IF @ISACTIVE\_SEL = 1

BEGIN

PRINT N' BEGIN' + CHAR(13) + CHAR(10)

PRINT N' IF @IsActive IS NULL' + CHAR(13) + CHAR(10)

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] ORDER BY [' + @OrderBy + '] ' + @OrderByDir + ';' + CHAR(13) + CHAR(10)

PRINT N' ELSE' + CHAR(13) + CHAR(10)

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] WHERE [isActive] = @IsActive ORDER BY [' + @OrderBy + '] ' + @OrderByDir + ';' + CHAR(13) + CHAR(10)

PRINT N' END' + CHAR(13) + CHAR(10)

END

ELSE

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] ORDER BY [' + @OrderBy + '] ' + @OrderByDir + ';' + CHAR(13) + CHAR(10)

PRINT N' ELSE' + CHAR(13) + CHAR(10)

PRINT N' SELECT \* FROM [dbo].[' + @ViewName + '] WHERE [ID] = @ID;' + CHAR(13) + CHAR(10)

PRINT N'END' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role

PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '\_sel] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName +'\_sel] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

--Print Delete Statement

PRINT N'/\*\*\*\*\*\* Object: StoredProcedure [dbo].[' + @ProcName + '\_del] Script Date: ' + CAST(GETDATE() AS nvarchar(30)) + ' \*\*\*\*\*\*/' + CHAR(13) + CHAR(10)

PRINT @LEGEND;

PRINT N'-- Description : Delete ' + @DESC + CHAR(13) + CHAR(10)

PRINT N'-- ===================================================================' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName + '\_del]' + CHAR(13) + CHAR(10);

PRINT N' (@MyID int, @ID int, @RowVersion int)' + CHAR(13) + CHAR(10);

PRINT N'AS' + CHAR(13) + CHAR(10)

PRINT N'BEGIN' + CHAR(13) + CHAR(10)

PRINT N' SET CONTEXT\_INFO @MyID;' + CHAR(13) + CHAR(10)

PRINT N' SET NOCOUNT ON;' + CHAR(13) + CHAR(10)

PRINT N' DELETE FROM [dbo].[' + @TableName + '] WHERE [' + @PRIMARY\_KEY + ']=@ID AND [RowVersion]=@RowVersion;' + CHAR(13) + CHAR(10)

PRINT N' SELECT @@ROWCOUNT as [Rows Affected];' + CHAR(13) + CHAR(10)

PRINT N'END' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role

PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '\_del] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName +'\_del] TO [' + @DBRoleName + ']' + CHAR(13) + CHAR(10)

PRINT N'GO' + CHAR(13) + CHAR(10)

USE [SROI]

GO

/\*\*\*\*\*\* Object: StoredProcedure [sys].[sp\_add\_log\_file\_recover\_suspect\_db] Script Date: 2/20/2016 4:08:06 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

-- Name: sp\_add\_log\_file\_recover\_suspect\_db

-- Purpose: Adds a log file to a suspect database and runs

-- recovery on the database. This SP should only be used

-- on databases that have been marked suspect due to

-- insufficient data (error 1105) or log (error 9002) space.

--

ALTER procedure [sys].[sp\_add\_log\_file\_recover\_suspect\_db]

@dbName sysname -- database name

,@name nvarchar(260) -- logical file name

,@filename nvarchar(260) -- OS file name

,@size nvarchar(20) = NULL -- initial file size

,@maxsize nvarchar(20) = NULL -- maximum file size

,@filegrowth nvarchar(20) = NULL -- growth increment

as

if (SERVERPROPERTY('IsMatrix') = 1)

begin

raiserror (28401, -1, -1, N'sys.sp\_add\_log\_file\_recover\_suspect\_db')

return (1)

end

EXEC sys.sp\_add\_file\_recover\_suspect\_db @dbName, 'LOG', NULL, @name, @filename, @size, @maxsize, @filegrowth

USE [SROI]

GO

/\*\*\*\*\*\* Object: StoredProcedure [sys].[sp\_add\_data\_file\_recover\_suspect\_db] Script Date: 2/20/2016 4:08:04 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

-- Name: sp\_add\_data\_file\_recover\_suspect\_db

-- Purpose: Adds a data file to a suspect database and runs

-- recovery on the database. This SP should only be used

-- on databases that have been marked suspect due to

-- insufficient data (error 1105) or log (error 9002) space.

--

ALTER procedure [sys].[sp\_add\_data\_file\_recover\_suspect\_db]

@dbName sysname -- database name

,@filegroup nvarchar(260) -- file group for new file

,@name nvarchar(260) -- logical file name

,@filename nvarchar(260) -- OS file name

,@size nvarchar(20) = NULL -- initial file size

,@maxsize nvarchar(20) = NULL -- maximum file size

,@filegrowth nvarchar(20) = NULL -- growth increment

as

if (SERVERPROPERTY('IsMatrix') = 1)

begin

raiserror (28401, -1, -1, N'sys.sp\_add\_data\_file\_recover\_suspect\_db')

return (1)

end

EXEC sys.sp\_add\_file\_recover\_suspect\_db @dbName, 'DATA', @filegroup, @name, @filename, @size, @maxsize, @filegrowth

USE [SROI]

GO

/\*\*\*\*\*\* Object: StoredProcedure [sys].[sp\_add\_agent\_profile] Script Date: 2/20/2016 4:08:02 PM \*\*\*\*\*\*/

SET ANSI\_NULLS OFF

GO

SET QUOTED\_IDENTIFIER OFF

GO

/\*

\*\* The system profile of the same type of agent will be used as a template for

\*\* the parameters in this new user profile.

\*/

ALTER procedure [sys].[sp\_add\_agent\_profile] (

@profile\_id int = NULL OUTPUT,

@profile\_name sysname,

@agent\_type int, -- 1-Snapshot, 2-Logreader,

-- 3-Distribution, 4-Merge,

-- 9-Qreader

@profile\_type int = 1, -- 0-System, 1-Custom

@description nvarchar(3000) = NULL,

@default bit = 0 -- 0-Not Default, 1-Default

)

AS

SET NOCOUNT ON

declare @sys\_profile int

declare @default\_id int

declare @sysdefault\_id int

declare @retcode int

/\*

\*\* Security Check: require sysadmin

\*/

IF (ISNULL(IS\_SRVROLEMEMBER('sysadmin'),0) = 0)

BEGIN

RAISERROR(21089,16,-1)

RETURN (1)

END

SELECT @profile\_name = RTRIM(@profile\_name)

exec @retcode = sys.sp\_MSreplcheck\_name @profile\_name, '@profile\_name', 'sp\_add\_agent\_profile'

if @@ERROR <> 0 or @retcode <> 0

return(1)

/\* The profile name is unique across a particular agent type \*/

IF EXISTS ( SELECT \* FROM msdb..MSagent\_profiles

WHERE profile\_name = @profile\_name collate database\_default

AND agent\_type = @agent\_type )

BEGIN

RAISERROR(20057, 16, -1, @profile\_name)

RETURN (1)

END

IF @agent\_type NOT IN (1, 2, 3, 4, 9)

BEGIN

RAISERROR(20058, 16, -1)

return (1)

END

IF @profile\_type NOT IN (0, 1)

BEGIN

RAISERROR(20059, 16, -1)

return (1)

END

BEGIN TRAN

/\*

\*\* First find out the default profile of the same agent type

\*/

select @default\_id = profile\_id from msdb..MSagent\_profiles WHERE agent\_type = @agent\_type AND def\_profile = 1

INSERT INTO msdb..MSagent\_profiles

VALUES (@profile\_name, @agent\_type, @profile\_type, @description, 0)

IF @@ERROR <> 0

GOTO UNDO

SELECT @profile\_id = profile\_id

FROM msdb..MSagent\_profiles

WHERE profile\_name = @profile\_name collate database\_default AND agent\_type = @agent\_type

/\*

\*\* If there is system profiles of the same agent type, use the primary system one as template of parameter set

\*/

IF EXISTS (select \* from msdb..MSagent\_profiles where agent\_type = @agent\_type AND type = 0)

select @sysdefault\_id = min(profile\_id) from msdb..MSagent\_profiles where agent\_type = @agent\_type AND type = 0

if @sysdefault\_id IS NULL

select @sysdefault\_id = @default\_id

if @profile\_type=1

BEGIN

INSERT INTO msdb..MSagent\_parameters

select @profile\_id, parameter\_name, value from msdb..MSagent\_parameters

where profile\_id = @sysdefault\_id

if @@ERROR<>0

GOTO UNDO

END

/\* Only one profile for an agent type must be default \*/

IF @default = 1

BEGIN

UPDATE msdb..MSagent\_profiles

SET def\_profile = 0 WHERE profile\_id = @default\_id

IF @@ERROR <> 0

GOTO UNDO

UPDATE msdb..MSagent\_profiles

SET def\_profile = 1 WHERE profile\_id = @profile\_id

IF @@ERROR <> 0

GOTO UNDO

END

COMMIT TRAN

RETURN 0

UNDO:

IF @@TRANCOUNT = 1

ROLLBACK TRAN

ELSE

COMMIT TRAN

RETURN 1

USE [SROI]

GO

/\*\*\*\*\*\* Object: StoredProcedure [sys].[sp\_add\_agent\_parameter] Script Date: 2/20/2016 4:07:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS OFF

GO

SET QUOTED\_IDENTIFIER OFF

GO

-- Add a row into the "MSagent\_parameters" table

ALTER procedure [sys].[sp\_add\_agent\_parameter] (

@profile\_id int,

@parameter\_name sysname,

@parameter\_value nvarchar(255)

)

as

declare @slash\_parameter\_name sysname

declare @dash\_parameter\_name sysname

declare @retcode int

set nocount on

/\*

\*\* Security Check: require sysadmin

\*/

IF (ISNULL(IS\_SRVROLEMEMBER('sysadmin'),0) = 0)

BEGIN

RAISERROR(21089,16,-1)

RETURN (1)

END

select @parameter\_name = RTRIM(@parameter\_name)

-- Append leading '-' when not given

if (substring(@parameter\_name, 1, 1) <> '/' and

substring(@parameter\_name, 1, 1) <> '-')

BEGIN

select @parameter\_name = N'-' + @parameter\_name

END

-- Call proc to validate parameter value

exec @retcode = sys.sp\_MSvalidate\_agent\_parameter

@profile\_id = @profile\_id,

@parameter\_name = @parameter\_name,

@parameter\_value = @parameter\_value

if @retcode <> 0

RETURN(1)

select @slash\_parameter\_name = lower(stuff(@parameter\_name, 1, 1, N'/') collate SQL\_Latin1\_General\_CP1\_CS\_AS)

select @dash\_parameter\_name = lower(stuff(@parameter\_name, 1, 1, N'-') collate SQL\_Latin1\_General\_CP1\_CS\_AS)

/\* A parameter may be defined only once per profile \*/

if exists ( select \* from msdb..MSagent\_parameters

where profile\_id = @profile\_id

and (lower(parameter\_name collate SQL\_Latin1\_General\_CP1\_CS\_AS) = @slash\_parameter\_name collate SQL\_Latin1\_General\_CP1\_CS\_AS

or lower(parameter\_name collate SQL\_Latin1\_General\_CP1\_CS\_AS) = @dash\_parameter\_name collate SQL\_Latin1\_General\_CP1\_CS\_AS))

BEGIN

RAISERROR (20067, 16, -1, @parameter\_name) -- The parameter name ''%s'' already exists for the specified profile

RETURN (1)

END

insert into msdb..MSagent\_parameters

values (@profile\_id, @parameter\_name, @parameter\_value) ;

if @@error <> 0

return(1)