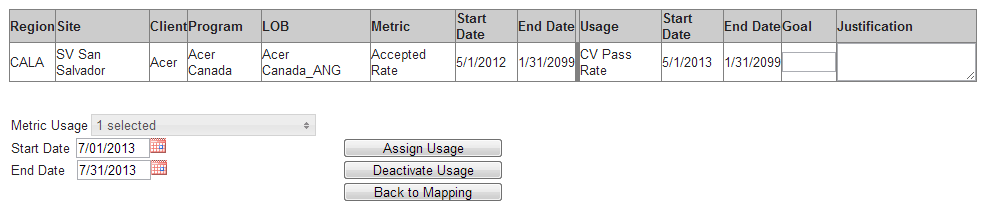


Check one or serveral checkboxes and then click botton “Create Mapping”

Apply Usages Button



Create Mapping: Select Client – Program – LOB – Metric and set start date

<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="ucMetricConfiguration.ascx.cs"

Inherits="DataMaps.ucMetricConfiguration1" %>

<%@ Register Assembly="System.Web" Namespace="System.Web.UI" TagPrefix="asp" %>

<%@ Register Assembly="DataMapsClassLibrary" Namespace="DataMaps" TagPrefix="cc1" %>

<%@ Register Src="~/usercontrols/ucMetricInsert.ascx" TagName="ucMetricInsert" TagPrefix="uc1" %>

<script src="../scripts/jquery-1.6.2.min.js" type="text/javascript"></script>

<script src="../scripts/jquery-ui-1.8.16.custom.min.js" type="text/javascript"></script>

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

<link href="../css/jquery-ui-1.8.17.custom.css" rel="stylesheet" type="text/css" />

<script type="text/javascript">

function IsDateGreater(DateValue1, DateValue2) {

var DaysDiff;

Date1 = new Date(DateValue1);

Date2 = new Date(DateValue2);

DaysDiff = Math.floor((Date1.getTime() - Date2.getTime()) / (1000 \* 60 \* 60 \* 24));

if (DaysDiff > 0)

return false;

else

return true;

}

function enables(checkeo) {

if (checkeo.checked) {

if (window.location.hostname == "referencemapping.webstg.stream.com" || window.location.hostname == "localhost")

{

var submitObj = document.getElementById('<% =BtnDeactive.ClientID %>');

if (submitObj != null) {

submitObj.disabled = false;

}

}

var submitObj = document.getElementById('<% =BtnApplyUsages.ClientID %>');

submitObj.disabled = false;

}

else {

/\* var chequear = false;

$("#<% =GVMetrics.ClientID %> tr").each(function () {

var ss = $(this).find("td:first").html;

var ss2 = $(this).find("Check1").html;

if (ss2.checked)

alert("si");

else

alert("no");

});\*/

}

}

function Verificar3() {

var Date2 = document.getElementById('<%=Date33.ClientID%>').value;

if (!isDate(Date2)) {

alert('Type a valid end date');

return false;

};

return confirm('Are you sure you want to deactivate mapping on ' + Date2+' ?');

}

function Verificar() {

var retorno = true;

var Region = $("#<%=ddlRegion.ClientID%> option:selected").val();

var Site = $("#<%=ddlSite.ClientID%> option:selected").val();

var Client = $("#<%=ddlClient.ClientID%> option:selected").val();

var Program = $("#<%=ddlProgram.ClientID%> option:selected").val();

var LOB = $("#<%=ddlLOB.ClientID%> option:selected").val();

var Metric = $("#<%=ddlMetric.ClientID%> option:selected").val();

var errores = "Select ";

if (Client == 0 || Program == 0 || LOB == 0) errores = errores + " Client, Program, LOB, ";

if (Site == 0) errores = errores + " Site, ";

if (Metric == 0) errores = errores + " Metric ";

if (errores == "Select ") errores = "";

var Date1 = document.getElementById('<%=Date1.ClientID%>').value;

//var Date2 = document.getElementById('<%=Date2.ClientID%>').value;

if (!isDate(Date1)) {

errores = errores + "Not a valid starting date ";

}

//if (!isDate(Date2)) {

// errores = errores + " Not a valid end date";

//}

// if (errores == "") {

// retorno = IsDateGreater(Date1, Date2);

// if (!retorno) errores = "Start date must be before end date";

// }

if (errores != "") {

document.getElementById('<%=Message1.ClientID%>').innerText = errores;

retorno = false;

}

if (retorno)

document.getElementById('<%=Message1.ClientID%>').innerText = "Processing ...";

Blink("<%=Message1.ClientID%>");

return retorno;

}

function Verificar1() {

var Date1 = "";

var Existe = $('#<%=Date11.ClientID%>');

if (Existe.length != 0) {

Date1 = document.getElementById('<%=Date11.ClientID%>').value;

}

var errores = "";

if (!isDate(Date1)) errores = "Not a valid start date";

document.getElementById('<%=Message2.ClientID%>').innerText = errores;

if (errores != "") {

Blink("<%=Message2.ClientID%>")

return false;

}

return true;

}

function Verificar2() {

var Existe = $('#<%=Date22.ClientID%>');

if (Existe.length == 0) return true;

Date2 = document.getElementById('<%=Date22.ClientID%>').value;

var errores = "";

if (!isDate(Date2)) errores = "Not a valid end date";

document.getElementById('<%=Message2.ClientID%>').innerText = errores;

if (errores != "") {

Blink("<%=Message2.ClientID%>")

return false;

}

return confirm('Are you sure you want to deactivate usages on ' + Date2 + ' ?');

}

/\* function GetDay(txDate2){

var day, month;

var ss = txDate2;

var posi1 = ss.indexOf("/", 0);

var posi2 = ss.indexOf("/", (posi1 + 1));

month = ss.substring(0, posi1);

posi1++;

day = ss.substring(posi1, posi2);

return day;

}

function isLastDay(dt2) {

var dt = new Date(dt2);

var test = new Date(dt.getTime());

test.setDate(test.getDate() + 1);

//return test.getDate() === 1;

return true;

}

\*/

</script>

<asp:ScriptManager ID="ScriptManager1"

runat="server"

EnablePageMethods="True">

</asp:ScriptManager>

<asp:Panel ID="Panel0" runat="server">

<table style="width: 100%">

<tr id="Region\_TR" runat="server">

<td id="largo" runat="server" style="height: 35px; text-align: right; width: 11%">

<asp:Label ID="Label1" runat="server" CssClass="label" Text="Region" Font-Size="Small" />

</td>

<td style="width: 20%">

<asp:DropDownList ID="ddlRegion" runat="server" OnSelectedIndexChanged="ddlRegion\_SelectedIndexChanged"

AutoPostBack="True" Font-Size="Small" />

</td>

<td style="text-align: right; width: 10%;">

<asp:Label ID="Label2" runat="server" CssClass="label" Text="Site" Font-Size="Small" />

</td>

<td style="width: 18%">

<asp:DropDownList ID="ddlSite" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ddlSite\_SelectedIndexChanged"

Font-Size="Small" />

</td>

<td style="text-align: right; width: 10%;">

</td>

</tr>

<tr id="Client\_TR" runat="server">

<td align="right" style="text-align: right; vertical-align: middle">

<asp:Label ID="Label4" runat="server" CssClass="label" Text="Client" Font-Size="Small" />

</td>

<td>

<asp:DropDownList ID="ddlClient" runat="server" OnSelectedIndexChanged="ddlClient\_SelectedIndexChanged"

AutoPostBack="True" Font-Size="Small" />

</td>

<td align="right">

<asp:Label ID="Label5" runat="server" CssClass="label" Text="Program" Font-Size="Small" />

</td>

<td>

<asp:DropDownList ID="ddlProgram" runat="server" OnSelectedIndexChanged="ddlProgram\_SelectedIndexChanged"

AutoPostBack="True" Font-Size="Small" />

</td>

<td style="width: 5%" align="right">

<asp:Label ID="Label6" runat="server" CssClass="label" Text=" LOB" Font-Size="Small" />

</td>

<td>

<asp:DropDownList ID="ddlLOB" runat="server" OnSelectedIndexChanged="ddlLOB\_SelectedIndexChanged"

AutoPostBack="True" Font-Size="Small" />

</td>

</tr>

<tr id="DeptoLOB" runat="server">

<td align="right" class="style1" style="text-align: right; vertical-align: middle">

<asp:Label ID="Label7" runat="server" CssClass="label" Text="Metric Category" Font-Size="Small" />

</td>

<td class="style1">

<asp:DropDownList ID="ddlCategory" runat="server" AutoPostBack="True" Font-Size="Small"

OnSelectedIndexChanged="ddlCategory\_SelectedIndexChanged" />

</td>

<td align="right" class="style1">

<asp:Label ID="Label8" runat="server" CssClass="label" Text="Metric Name" Font-Size="Small" />

</td>

<td class="style1">

<asp:DropDownList ID="ddlMetric" runat="server" AutoPostBack="True" Style="vertical-align: middle"

Font-Size="Small" OnSelectedIndexChanged="ddlMetric\_SelectedIndexChanged">

</asp:DropDownList>

</td>

<td align="right" class="style1">

<asp:Label ID="Label11" runat="server" CssClass="label" Font-Size="Small" Text="Symbol" />

</td>

<td class="style1">

<asp:DropDownList ID="ddlSymbol" runat="server" AutoPostBack="True" Style="vertical-align: middle"

Font-Size="Small">

<asp:ListItem Value="cur"></asp:ListItem>

<asp:ListItem Value="%"></asp:ListItem>

<asp:ListItem Value="min"></asp:ListItem>

<asp:ListItem Value="sec"></asp:ListItem>

<asp:ListItem Value="#"></asp:ListItem>

</asp:DropDownList>

</td>

<td class="style1">

<asp:ImageButton ID="ImageButton1" runat="server" Height="25px" ImageUrl="~/images/excel\_icon.png"

OnClick="ImageButton1\_Click" Visible="False" Width="29px" OnPreRender="ImageButton1\_PreRender"

ToolTip="Export Data into Excel" />

</td>

</tr>

</table>

<hr />

</asp:Panel>

<asp:Panel ID="Panel1" Visible = "false" runat="server">

<asp:Table ID="Table1" runat="server">

<asp:TableRow ID="TableRow2" runat="server">

<asp:TableCell ID="TableCell7" runat="server">

</asp:TableCell>

<asp:TableCell ID="TableCell5" runat="server">

&nbsp;&nbsp;&nbsp;

<asp:Button ID="BtnInsert" Enabled="false" Width="165px" runat="server" Font-Size="Small"

OnClick="BtnInsert\_Click" Text="Create Mapping" OnClientClick="return Verificar();" />

</asp:TableCell>

<asp:TableCell ID="TableCell9" runat="server">

&nbsp;&nbsp;&nbsp;

<asp:Button ID="BtnDeactive" Width="165px" runat="server" Enabled="false" Font-Size="Small"

OnClick="BtnDeactive\_Click" Text="Deactivate Mapping" />

</asp:TableCell>

<asp:TableCell ID="TableCell6" runat="server">

&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;

</asp:TableCell>

<asp:TableCell ID="TableCell8" Style="text-align: right;" Width="600px" runat="server">

<asp:Button ID="BtnApplyUsages" Width="145px" runat="server" BackColor="#FFCC66"

Enabled="false" Font-Size="Small" OnClick="BtnApplyUsages\_Click" Text="Apply Usages" />

</asp:TableCell>

</asp:TableRow>

<asp:TableRow ID="TableRow1" runat="server">

<asp:TableCell ID="TableCell10" runat="server">

</asp:TableCell>

<asp:TableCell ID="TableCell1" runat="server">

&nbsp;&nbsp;

<asp:Label ID="lDate1" runat="server" Font-Size="Small" Text="Start Date"></asp:Label>&nbsp;

&nbsp;

<asp:DropDownList ID="ddlMonth" runat="server" Font-Size="Small" Visible="false">

<asp:ListItem Value="Jan"></asp:ListItem>

<asp:ListItem Value="Feb"></asp:ListItem>

<asp:ListItem Value="Mar"></asp:ListItem>

<asp:ListItem Value="Apr"></asp:ListItem>

<asp:ListItem Value="May"></asp:ListItem>

<asp:ListItem Value="Jun"></asp:ListItem>

<asp:ListItem Value="Jul"></asp:ListItem>

<asp:ListItem Value="Aug"></asp:ListItem>

<asp:ListItem Value="Sep"></asp:ListItem>

<asp:ListItem Value="Oct"></asp:ListItem>

<asp:ListItem Value="Nov"></asp:ListItem>

<asp:ListItem Value="Dec"></asp:ListItem>

</asp:DropDownList>

&nbsp;

<asp:DropDownList ID="ddlYear" runat="server" Font-Size="Small" Visible="false">

</asp:DropDownList>

<asp:TextBox ID="Date1" CssClass="datepicker" Visible="true" runat="server" Font-Size="Small"

onblur='this.style.backgroundColor="FFFFFF";' onfocus='this.style.backgroundColor="CCCCCC"; this.innerText=""; '

Width="70px"></asp:TextBox>

</asp:TableCell>

<asp:TableCell ID="TableCell3" HorizontalAlign="Right" runat="server">

&nbsp;

<asp:Label ID="lDate2" runat="server" Visible="false" Font-Size="Small" Text="End Date"></asp:Label>

&nbsp;

<asp:TextBox ID="Date2" CssClass="datepicker" runat="server" Visible="false" Font-Size="Small"

onblur='this.style.backgroundColor="FFFFFF";' onfocus='this.style.backgroundColor="CCCCCC"; this.innerText=""; '

Width="70px"></asp:TextBox>

</asp:TableCell>

</asp:TableRow>

</asp:Table>

</asp:Panel>

<%--<div style="margin-top: 20px; color: Red">--%>

<asp:Label ID="Message1" runat="server" Font-Size="Small" ForeColor="Red"

Visible="True"></asp:Label>

<%--</div>--%>

<div id="DivDialog1" title="My Dialog Title">

<p> </p>

<p> </p>

</div>

<asp:UpdatePanel ID="UpdatePanel2" runat="server">

<ContentTemplate>

<asp:UpdateProgress ID="UpdateProgress1" runat="server" AssociatedUpdatePanelID="UpdatePanel2">

<ProgressTemplate>

<div style="margin-top: 20px; color: Green">

Processing, please wait...

</div>

</ProgressTemplate>

</asp:UpdateProgress>

<asp:GridView ID="GVMetrics" runat="server" AllowPaging="True" AutoGenerateColumns="False"

EmptyDataText="No records found." AllowSorting="True" Width="100%" PageSize="15"

CssClass="GridView" OnPageIndexChanging="GVMetrics\_PageIndexChanging" DataKeyNames="MetricConfigurationID"

OnSorting="GVMetrics\_Sorting" OnRowCommand="GVMetrics\_RowCommand" OnRowDataBound="GVMetrics\_RowDataBound"

OnRowCreated="GVMetrics\_RowCreated"

onselectedindexchanged="GVMetrics\_SelectedIndexChanged">

<Columns>

<asp:TemplateField>

<ItemTemplate>

<asp:CheckBox ID="Check1" onClick="enables(this);" runat="server" />

</ItemTemplate>

<ItemStyle HorizontalAlign="Center"></ItemStyle>

</asp:TemplateField>

<asp:BoundField DataField="Region\_Name" HeaderText="Region" SortExpression="Region\_Name">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="SubRegion\_Name" HeaderText="Sub Region" Visible = "false" SortExpression="SubRegion\_Name">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Site" HeaderText="Site" SortExpression="Site">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Client" HeaderText="Client" SortExpression="Client">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Left"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Program" HeaderText="Program" SortExpression="Program">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="LOB" HeaderText="LOB" SortExpression="LOB">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Category" HeaderText="Category" SortExpression="Category">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="MetricName" HeaderText="Metric" SortExpression="MetricName">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Symbol\_Code" HeaderText="Symbol">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Center"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="EffectiveStart\_DT" DataFormatString="{0:M-dd-yyyy}" HeaderText="Start Date">

<HeaderStyle Font-Size="Small" Wrap="False"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Center" Width="60px"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="EffectiveEnd\_DT" DataFormatString="{0:M-dd-yyyy}" HeaderText="End Date">

<HeaderStyle Font-Size="Small" Wrap="False"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Center" Width="60px"></ItemStyle>

</asp:BoundField>

<asp:ButtonField CommandName="Views" HeaderText="Metric Usages" Text="View"

DataTextField="NumUsage">

<HeaderStyle Font-Size="Small" HorizontalAlign="Center" />

<ItemStyle Font-Size="Small" HorizontalAlign="Center" />

</asp:ButtonField>

<asp:BoundField DataField="MetricConfigurationID" Visible="false" HeaderText="MetricConfigurationID">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Center" Width="60px"></ItemStyle>

</asp:BoundField>

</Columns>

<HeaderStyle CssClass="HeaderStyle" BackColor="#CCCCCC" />

<RowStyle CssClass="RowStyle" />

<SelectedRowStyle CssClass="SelectedRowStyle" />

<AlternatingRowStyle CssClass="AlternatingRowStyle" />

</asp:GridView>

</ContentTemplate>

</asp:UpdatePanel>

<asp:Panel ID="Panel3" runat="server" Visible="false">

<asp:GridView ID="GVLista" runat="server" DataKeyNames="MetricID" AutoGenerateColumns="False"

CellPadding="2" CellSpacing="2">

<Columns>

<asp:BoundField DataField="ID" HeaderText="ID" Visible="False">

<HeaderStyle Font-Size="Small" Width="0px"></HeaderStyle>

<ItemStyle Font-Size="Small" Width="0px"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Region" HeaderText="Region">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="SubRegion" visible = "false" HeaderText="Sub Region">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Site" HeaderText="Site">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Client" HeaderText="Client">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small" HorizontalAlign="Left"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Program" HeaderText="Program">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="LOB" HeaderText="LOB">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Metric" HeaderText="Metric">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="StartDT" DataFormatString="{0:MMMM d, yyyy}" HeaderText="Start Date">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="EndDT" DataFormatString="{0:MMMM d, yyyy}" HeaderText="End Date">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField>

<ItemStyle BackColor="Gray" Width="0.5px" />

</asp:BoundField>

<asp:BoundField DataField="MetricUsageName" HeaderText="Usage">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="UsageStart\_DT" DataFormatString="{0:MMMM d, yyyy}" HeaderText="Start Date">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="UsageEnd\_DT" DataFormatString="{0:MMMM d, yyyy}" HeaderText="End Date">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:TemplateField HeaderText="Goal">

<ItemTemplate>

<asp:TextBox ID="txtGoalTmp" runat="server" Font-Size="Small" MaxLength="10"

Width="50px"></asp:TextBox>

</ItemTemplate>

<HeaderStyle Font-Size="Small" />

</asp:TemplateField>

<asp:TemplateField HeaderText="Justification">

<ItemTemplate>

<asp:TextBox ID="txtJustificationTmp" runat="server" Font-Size="X-Small"

MaxLength="2000" Rows="3" TextMode="MultiLine"></asp:TextBox>

</ItemTemplate>

<HeaderStyle Font-Size="Small" />

</asp:TemplateField>

</Columns>

<HeaderStyle CssClass="HeaderStyle" BackColor="#CCCCCC" />

<RowStyle CssClass="RowStyle" />

<SelectedRowStyle CssClass="SelectedRowStyle" />

<AlternatingRowStyle CssClass="AlternatingRowStyle" />

</asp:GridView>

<br />

<asp:Panel ID="Panel7" runat="server" Visible="false">

&nbsp;&nbsp;&nbsp;

<asp:Table ID="Table2" runat="server">

<asp:TableRow ID="TableRow4" runat="server">

<asp:TableCell ID="TableCell14" width="330px" runat="server">

<uc1:ucMetricInsert ID="ucMetricInsert1" runat="server" />

</asp:TableCell>

<asp:TableCell ID="TableCell15" runat="server">

</asp:TableCell>

</asp:TableRow>

<asp:TableRow ID="TableRow5" runat="server">

<asp:TableCell ID="TableCell2" width="330px" runat="server">

<asp:Label ID="Label12" runat="server" Font-Size="Small" Text="Start Date"></asp:Label>&nbsp;

<asp:TextBox ID="Date11" Visible="true" CssClass="datepicker" runat="server" Font-Size="Small"

onblur='this.style.backgroundColor="FFFFFF";' onfocus='this.style.backgroundColor="CCCCCC"; this.innerText=""; '

Width="70px"></asp:TextBox>

&nbsp;&nbsp;

</asp:TableCell>

<asp:TableCell ID="TableCell11" width="330px" runat="server">

<asp:Button ID="BtnAssignUsage" runat="server" width = "160px" OnClick="BtnAssignUsage\_Click" OnClientClick="return Verificar1();"

Text="Assign Usage" />

</asp:TableCell>

</asp:TableRow>

<asp:TableRow ID="TableRow6" runat="server">

<asp:TableCell ID="TableCell12" width="330px" runat="server">

<asp:Label ID="Label10" runat="server" Font-Size="Small" Text="End Date"></asp:Label>

&nbsp;&nbsp;

<asp:TextBox ID="Date22" runat="server" CssClass="datepicker" Font-Size="Small"

onblur="this.style.backgroundColor=&quot;FFFFFF&quot;;"

onfocus="this.style.backgroundColor=&quot;CCCCCC&quot;; this.innerText=&quot;&quot;; "

Width="70px"></asp:TextBox>

&nbsp;&nbsp;

</asp:TableCell>

<asp:TableCell ID="TableCell13" width="330px" runat="server">

<asp:Button ID="BtnDeactivate" runat="server" OnClick="BtnDeactivate\_Click"

OnClientClick="return Verificar2();" Text="Deactivate Usage" width="160px" />

</asp:TableCell>

</asp:TableRow>

<asp:TableRow ID="TableRow7" runat="server">

<asp:TableCell ID="TableCell16" Width="330px" runat="server">

</asp:TableCell>

<asp:TableCell ID="TableCell17" Width="330px" runat="server">

<asp:Button ID="BtnBack2Mapping" runat="server" width="160px" OnClick="BtnBack2Mapping\_Click" Text="Back to Mapping" />

</asp:TableCell>

</asp:TableRow>

</asp:Table>

</asp:Panel>

<br />

<asp:Panel ID="Panel6" runat="server" Visible="false">

<asp:Table ID="Table3" runat="server">

<asp:TableRow ID="TableRow3" runat="server">

<asp:TableCell ID="TableCell4" runat="server">

&nbsp;&nbsp;

<asp:Label ID="Label9" runat="server" Font-Size="Small" Text="End Date"></asp:Label>

&nbsp;

<asp:TextBox ID="Date33" CssClass="datepicker" runat="server" Font-Size="Small" onblur='this.style.backgroundColor="FFFFFF";'

onfocus='this.style.backgroundColor="CCCCCC"; this.innerText=""; ' Width="70px"></asp:TextBox>

&nbsp;

&nbsp;

<asp:Button ID="BtnDeactiveMapping" runat="server" width = "160px" OnClientClick="return Verificar3();"

Text="Deactivate Mapping" onclick="BtnDeactiveMapping\_Click" />

&nbsp;

&nbsp;

<asp:Button ID="Button4" runat="server" OnClick="BtnBack2Mapping\_Click" Text="Back to Mapping" />

</asp:TableCell>

</asp:TableRow>

</asp:Table>

</asp:Panel>

<asp:Label ID="Message2" runat="server" Font-Size="Small" ForeColor="Red"></asp:Label>

</asp:Panel>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:Panel ID="Panel4" runat="server">

<br>

<asp:GridView ID="GridView3" runat="server" AutoGenerateColumns="False" Width="412px">

<Columns>

<asp:BoundField DataField="MetricUsageName" HeaderText="Metric Usage">

<HeaderStyle Font-Size="Small" Width="0px" Wrap="False" />

<ItemStyle Font-Size="Small" Width="0px" Wrap="False" HorizontalAlign="Center" />

</asp:BoundField>

<asp:BoundField DataField="Start" DataFormatString="{0:MMMM d, yyyy}" HeaderText="Start Date">

<HeaderStyle Font-Size="Small" Width="0px" HorizontalAlign="Center" Wrap="False" />

<ItemStyle Font-Size="Small" Width="0px" Wrap="False" HorizontalAlign="Center" />

</asp:BoundField>

<asp:BoundField DataField="End" DataFormatString="{0:MMMM d, yyyy}" HeaderText="End Date">

<HeaderStyle Font-Size="Small" Width="0px" Wrap="False" />

<ItemStyle Font-Size="Small" Width="0px" Wrap="False" HorizontalAlign="Center" />

</asp:BoundField>

</Columns>

<HeaderStyle BackColor="#CCCCCC" CssClass="HeaderStyle" />

<RowStyle CssClass="RowStyle" />

<SelectedRowStyle CssClass="SelectedRowStyle" />

<AlternatingRowStyle CssClass="AlternatingRowStyle" />

</asp:GridView>

</asp:Panel>

</ContentTemplate>

</asp:UpdatePanel>

<asp:HiddenField ID="Outset" runat="server" Value="True" />

<asp:HiddenField ID="AllowUpdate" runat="server" Value="False" />

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Script.Services;

using System.Web.Security;

using System.Web.Services;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.ApplicationBlocks.Data;

using Microsoft.SqlServer.Server;

using System.Text;

using System.Net.Mail;

namespace DataMaps

{

public partial class ucMetricConfiguration1 : System.Web.UI.UserControl

{

bool InsertMapping = false;

static int CacheTime = 30;

static int ColUsage = 12;

static string \_connStr = AppSettings.sqlConnDWReference;

bcCVMetricConfig bcMC = null; //ram

public CUser MCUser

{

get { return (CUser)Session["User"]; }

set { Session["User"] = value; }

}

#region Blink

public void Blinks(string controlID)

{

string script = "<script type=\"text/javascript\">";

if (controlID == "Message1") script = script + "Blink('" + Message1.ClientID + "');";

if (controlID == "Message2") script = script + "Blink('" + Message2.ClientID + "');";

script = script + " </script>";

Page page = HttpContext.Current.CurrentHandler as Page;

if (page != null && !page.ClientScript.IsClientScriptBlockRegistered("alert2"))

{

page.ClientScript.RegisterStartupScript(page.GetType(), "alert2", script);

}

}

#endregion

#region JQuery Dialog

private string getjQueryCode(string jsCodetoRun)

{

StringBuilder sb = new StringBuilder();

sb.AppendLine("$(document).ready(function() {");

sb.AppendLine(jsCodetoRun);

sb.AppendLine(" });");

return sb.ToString();

}

private void runjQueryCode(string jsCodetoRun)

{

ScriptManager requestSM = ScriptManager.GetCurrent(this.Page);

if (requestSM != null && requestSM.IsInAsyncPostBack)

{

ScriptManager.RegisterClientScriptBlock(this.Page,

typeof(Page),

Guid.NewGuid().ToString(),

getjQueryCode(jsCodetoRun),

true);

}

else

{

this.Page.ClientScript.RegisterClientScriptBlock(typeof(Page),

Guid.NewGuid().ToString(),

getjQueryCode(jsCodetoRun),

true);

}

}

public void ShowMessage(string Message, string Title)

{

StringBuilder sb = new StringBuilder();

sb.Append("dlg = $('#DivDialog1').dialog({autoOpen: false, modal: true, buttons: {Ok: function() {$( this ).dialog( 'close' );} }, show: 'fold', hide: 'explode', title: '" + Title + "'}); ");

//sb.Append("dlg = $('#DivDialog1').dialog({autoOpen: false, show: 'fold', hide: 'explode', title: '" + Title + "'}); ");

sb.Append("$('#DivDialog1').dialog('open'); ");

sb.Append("$('#DivDialog1').html('" + Message + "'); ");

runjQueryCode(sb.ToString());

}

#endregion

//130520 130521 - submit

//130523 - Level 1

//130524 & 130525 - Level 2

//130526 & 130578 - Level 3

#region Load

protected string GetUser()

{

string userName = "130578";

try

{

#if (!DEBUG)

{

MembershipUser u;

//userName = System.Web.Security.Membership.GetUser(HttpContext.Current.User.Identity.Name).UserName.ToString();

u = System.Web.Security.Membership.GetUser();

if (u != null)

{

//userId = u.ProviderUserKey.ToString();

userName = u.UserName;

}

else

{

userName = Page.User.Identity.Name;

if (userName.Contains("\\"))

userName = userName.Substring(userName.LastIndexOf("\\") + 1);

}

}

#else

// SDM = 102538, 130085

// DG = 232523

// VP = 211975

// BD = 130144

userName = "130578";

#endif

}

catch

{

//userName = "E04922";

userName = Page.User.Identity.Name;

}

return userName;

}

protected void Page\_Load(object sender, EventArgs e)

{

bcMC = new bcCVMetricConfig();

//modify check\_authorization--insertmapping variable;

Check\_Authorization();

//ram

MCUser = bcMC.GetUser("Metric Name Usage", GetUser());

Session["User"] = MCUser;

BtnInsert.Visible = InsertMapping;

BtnDeactive.Visible = (InsertMapping) && (MCUser.Position == "Data Governance");

BtnApplyUsages.Visible = InsertMapping;

//BtnDeactive.Attributes.Add("onclick", "return Verif();");

if (IsPostBack) return;

//Date1.Text = DateTime.Today.ToShortDateString();

Date1.Text = DateTime.Today.Month.ToString() + "/01/" + DateTime.Today.Year;

//Date2.Text = "01/31/2099";

Date2.Text = DateTime.Now.Month.ToString() + "/" + DateTime.DaysInMonth(DateTime.Now.Year, DateTime.Now.Month).ToString() + "/" + DateTime.Now.Year.ToString();

int mes1 = System.DateTime.Today.Month;

int year1 = System.DateTime.Today.Year;

ddlYear.Items.Add(year1.ToString());

if (mes1 > 10)

{

year1++;

ddlYear.Items.Add(year1.ToString());

}

ddlMonth.SelectedIndex = mes1 - 1;

LoadRegion();

//LoadSubRegion();

LoadSite();

LoadClient();

LoadCategories("CVPR");

}

[WebMethod]

[ScriptMethod(UseHttpGet = true)]

public static string GetListUsages(string ConfigID)

{

return " CV Pass Rate ";

}

protected void Check\_Authorization()

{

bool inicio = System.Convert.ToBoolean(Outset.Value);

if (!inicio)

{

InsertMapping = System.Convert.ToBoolean(AllowUpdate.Value);

return;

}

InsertMapping = true;

Outset.Value = "False";

if (Roles.Enabled)

{

string[] roles = Roles.GetRolesForUser(HttpContext.Current.User.Identity.Name);

InsertMapping = roles.Contains("RefMap.MetricConf.Change");

}

AllowUpdate.Value = System.Convert.ToString(InsertMapping);

}

protected DataSet RetrieveData()

{

SqlParameter[] parameters = new SqlParameter[9];

parameters[0] = new SqlParameter("@RegionID", SqlDbType.Int);

parameters[0].Value = ddlRegion.SelectedValue;

parameters[1] = new SqlParameter("@SubRegionID", SqlDbType.Int);

parameters[1].Value = 0; // ddlSubRegion.SelectedValue;

parameters[2] = new SqlParameter("@SiteID", SqlDbType.Int);

parameters[2].Value = ddlSite.SelectedValue;

parameters[3] = new SqlParameter("@ClientID", SqlDbType.Int);

parameters[3].Value = ddlClient.SelectedValue;

parameters[4] = new SqlParameter("@ProgramID", SqlDbType.Int);

parameters[4].Value = ddlProgram.SelectedValue;

parameters[5] = new SqlParameter("@LOB", SqlDbType.Int);

parameters[5].Value = ddlLOB.SelectedValue;

parameters[6] = new SqlParameter("@CAT", SqlDbType.Int);

parameters[6].Value = ddlCategory.SelectedValue;

parameters[7] = new SqlParameter("@MetricName\_ID", SqlDbType.Int);

parameters[7].Value = ddlMetric.SelectedValue;

parameters[8] = new SqlParameter("@Usage", SqlDbType.Int);

parameters[8].Value = 0;

DataSet ds;

ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_MetricConfig\_View", parameters);

//ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_MetricConfig\_View2", parameters);

return ds;

}

protected void RetrieveDataUsage()

{

SqlParameter[] parameters = new SqlParameter[8];

parameters[0] = new SqlParameter("@RegionID", SqlDbType.Int);

parameters[0].Value = ddlRegion.SelectedValue;

parameters[1] = new SqlParameter("@SubRegionID", SqlDbType.Int);

parameters[1].Value = 0; // ddlSubRegion.SelectedValue;

parameters[2] = new SqlParameter("@SiteID", SqlDbType.Int);

parameters[2].Value = ddlSite.SelectedValue;

parameters[3] = new SqlParameter("@ClientID", SqlDbType.Int);

parameters[3].Value = ddlClient.SelectedValue;

parameters[4] = new SqlParameter("@ProgramID", SqlDbType.Int);

parameters[4].Value = ddlProgram.SelectedValue;

parameters[5] = new SqlParameter("@LOB", SqlDbType.Int);

parameters[5].Value = ddlLOB.SelectedValue;

parameters[6] = new SqlParameter("@CAT", SqlDbType.Int);

parameters[6].Value = ddlCategory.SelectedValue;

parameters[7] = new SqlParameter("@MetricName\_ID", SqlDbType.Int);

parameters[7].Value = ddlMetric.SelectedValue;

DataSet ds;

ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_MetricConfiguration\_View\_Usage", parameters);

Session["Usage"] = ds;

}

protected void Load\_Data()

{

//Limpiar.CleanMessage();

Message1.Text = "";

Session["Usage"] = null;

GridView3.DataSource = null;

GridView3.DataBind();

DataSet ds = RetrieveData();

//if (ds.Tables[0].Rows.Count > 0)

// RetrieveDataUsage();

GVMetrics.DataSource = ds;

GVMetrics.DataBind();

Session["Data"] = ds;

BtnInsert.Enabled = true;

if (GVMetrics.Rows.Count > 0)

{

ImageButton1.Visible = true;

BtnDeactive.Enabled = false;

BtnApplyUsages.Enabled = false;

}

else

{

ImageButton1.Visible = false;

BtnDeactive.Enabled = false;

BtnApplyUsages.Enabled = false;

}

Panel1.Visible = InsertMapping;

GVMetrics.Columns[0].Visible = InsertMapping;

}

#endregion

#region Select Combo

protected void ddlRegion\_SelectedIndexChanged(object sender, EventArgs e)

{

//LoadSubRegion();

LoadSite();

Load\_Data();

}

protected void ddlSubRegion\_SelectedIndexChanged1(object sender, EventArgs e)

{

LoadSite();

Load\_Data();

}

protected void ddlSite\_SelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void ddlClient\_SelectedIndexChanged(object sender, EventArgs e)

{

LoadProgram();

Load\_Data();

}

protected void ddlProgram\_SelectedIndexChanged(object sender, EventArgs e)

{

LoadLOB();

Load\_Data();

}

protected void ddlLOB\_SelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void ddlCategory\_SelectedIndexChanged(object sender, EventArgs e)

{

LoadMetric();

Load\_Data();

}

protected void ddlMetric\_SelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

#endregion

#region Export Excel

protected void ImageButton1\_PreRender(object sender, EventArgs e)

{

if (sender is ImageButton)

{

ImageButton imgBtn = (ImageButton)sender;

ScriptManager ScriptMgr = (ScriptManager)this.FindControl("ScriptManager1");

ScriptMgr.RegisterPostBackControl(imgBtn);

}

}

protected DataSet LoadAllData()

{

SqlParameter[] parameters = new SqlParameter[9];

parameters[0] = new SqlParameter("@RegionID", SqlDbType.Int);

parameters[0].Value = ddlRegion.SelectedValue;

parameters[1] = new SqlParameter("@SubRegionID", SqlDbType.Int);

parameters[1].Value = 0; // ddlSubRegion.SelectedValue;

parameters[2] = new SqlParameter("@SiteID", SqlDbType.Int);

parameters[2].Value = ddlSite.SelectedValue;

parameters[3] = new SqlParameter("@ClientID", SqlDbType.Int);

parameters[3].Value = ddlClient.SelectedValue;

parameters[4] = new SqlParameter("@ProgramID", SqlDbType.Int);

parameters[4].Value = ddlProgram.SelectedValue;

parameters[5] = new SqlParameter("@LOB", SqlDbType.Int);

parameters[5].Value = ddlLOB.SelectedValue;

parameters[6] = new SqlParameter("@CAT", SqlDbType.Int);

parameters[6].Value = ddlCategory.SelectedValue;

parameters[7] = new SqlParameter("@MetricName\_ID", SqlDbType.Int);

parameters[7].Value = ddlMetric.SelectedValue;

return SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_MetricUsage\_View", parameters);

}

protected void ImageButton1\_Click(object sender, ImageClickEventArgs e)

{

Response.Clear();

Response.Charset = "";

Response.ContentType = "application/vnd.ms-excel";

Response.AddHeader("Content-Disposition", "attachment;filename=Data.xls");

StringWriter sw = new System.IO.StringWriter();

HtmlTextWriter HtmlText = new HtmlTextWriter(sw);

DataGrid dg = new DataGrid();

dg.DataSource = LoadAllData();

dg.DataBind();

dg.RenderControl(HtmlText);

try

{

Response.Write(sw.ToString());

Response.End();

}

catch (Exception)

{

}

}

#endregion

#region Load Filters

private void LoadRegion()

{

if (Cache["Regions"] == null)

{

DataSet ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetRegion");

ddlRegion.DataSource = ds;

Cache.Insert("Regions", ds, null, DateTime.Now.AddMinutes(CacheTime), System.Web.Caching.Cache.NoSlidingExpiration);

}

else

{

ddlRegion.DataSource = (DataSet)Cache["Regions"];

}

ddlRegion.DataTextField = "Region\_Name";

ddlRegion.DataValueField = "Region\_ID";

ddlRegion.DataBind();

ListItem li = new ListItem("--All--", "0");

ddlRegion.Items.Insert(0, li);

}

private void LoadAllSites()

{

SqlParameter[] parameters = new SqlParameter[2];

parameters[0] = new SqlParameter("@RegionId", SqlDbType.Int);

parameters[0].Value = 0;

parameters[1] = new SqlParameter("@SubRegionId", SqlDbType.Int);

parameters[1].Value = 0;

DataSet ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetSites", parameters);

Cache.Insert("SitesAll", ds, null, DateTime.Now.AddMinutes(CacheTime), System.Web.Caching.Cache.NoSlidingExpiration);

}

private void LoadSite()

{

ddlSite.Items.Clear();

int RegionID = System.Convert.ToInt32(ddlRegion.SelectedValue);

int SubRegionID = 0; // System.Convert.ToInt32(ddlSubRegion.SelectedValue);

if (Cache["SitesAll"] == null) LoadAllSites();

DataSet ds2 = (DataSet)Cache["SitesAll"];

DataTable dt = ds2.Tables[0];

var query = from o in dt.AsEnumerable()

where (o.Field<int>("Region\_ID") == RegionID || RegionID == 0)

&& (o.Field<int>("SubRegion\_ID") == SubRegionID || SubRegionID == 0)

select new

{

Site\_Name = o.Field<string>("Site\_Name"),

Site\_ID = o.Field<int>("Site\_ID")

};

ddlSite.DataSource = query;

ddlSite.DataTextField = "Site\_Name";

ddlSite.DataValueField = "Site\_ID";

ddlSite.DataBind();

ListItem li = new ListItem("--All--", "0");

ddlSite.Items.Insert(0, li);

}

private void LoadClient()

{

ListItem li = new ListItem("--All--", "0");

ddlProgram.Items.Clear();

ddlProgram.Items.Insert(0, li);

ddlLOB.Items.Clear();

ddlLOB.Items.Insert(0, li);

ddlMetric.Items.Clear();

ddlMetric.Items.Insert(0, li);

if (Cache["Clients"] == null)

{

DataSet ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetClients");

ddlClient.DataSource = ds;

Cache.Insert("Clients", ds, null, DateTime.Now.AddMinutes(CacheTime), System.Web.Caching.Cache.NoSlidingExpiration);

}

else

{

ddlClient.DataSource = (DataSet)Cache["Clients"];

}

ddlClient.DataTextField = "Client\_Name";

ddlClient.DataValueField = "ClientId";

ddlClient.DataBind();

ddlClient.Items.Insert(0, li);

}

private void LoadProgram()

{

ddlProgram.Items.Clear();

int SiteID = 0;

if (Region\_TR.Visible) SiteID = System.Convert.ToInt32(ddlSite.SelectedValue);

if (ddlClient.SelectedValue != "0")

{

SqlParameter[] parameters = new SqlParameter[1];

parameters[0] = new SqlParameter("@Client\_ID", SqlDbType.Int);

parameters[0].Value = ddlClient.SelectedValue;

ddlProgram.DataSource = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetProgram", parameters);

ddlProgram.DataTextField = "Program\_Name";

ddlProgram.DataValueField = "ProgramId";

ddlProgram.DataBind();

}

ListItem li = new ListItem("--All--", "0");

ddlProgram.Items.Insert(0, li);

ddlLOB.Items.Clear();

ddlLOB.Items.Insert(0, li);

}

private void LoadLOB()

{

ddlLOB.Items.Clear();

if (ddlClient.SelectedValue != "0")

{

SqlParameter[] parameters = new SqlParameter[2];

parameters[0] = new SqlParameter("@Client\_ID", SqlDbType.Int);

parameters[0].Value = ddlClient.SelectedValue;

parameters[1] = new SqlParameter("@Program\_ID", SqlDbType.Int);

parameters[1].Value = ddlProgram.SelectedValue;

ddlLOB.DataSource = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetLOB", parameters);

ddlLOB.DataTextField = "LineofBusiness\_Name";

ddlLOB.DataValueField = "LineofBusiness\_ID";

ddlLOB.DataBind();

}

ListItem li = new ListItem("--All--", "0");

ddlLOB.Items.Insert(0, li);

}

private void LoadCategories(string uType)

{

DataSet ds = null;

if (uType == "CVPR")

ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetMetricCategory2");

else

ds = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetMetricCategory");

ddlCategory.DataSource = ds;

ddlCategory.DataTextField = "MetricCategory\_Name";

ddlCategory.DataValueField = "MetricCategory\_ID";

ddlCategory.DataBind();

ListItem li = new ListItem("--All--", "0");

ddlCategory.Items.Insert(0, li);

}

private void LoadMetric()

{

SqlParameter[] parameters = new SqlParameter[2];

parameters[0] = new SqlParameter("@Category", ddlCategory.SelectedValue);

parameters[1] = new SqlParameter("@Metric", "");

ddlMetric.DataSource = SqlHelper.ExecuteDataset(\_connStr, CommandType.StoredProcedure, "sp\_RefMap\_GetMetrics", parameters);

ddlMetric.DataTextField = "Metric\_Name";

ddlMetric.DataValueField = "MetricName\_ID";

ddlMetric.DataBind();

ListItem li = new ListItem("--All--", "0");

ddlMetric.Items.Insert(0, li);

}

#endregion

#region Sort

private string ConvertSortDirectionToSql(SortDirection sortDirection)

{

string newSortDirection = String.Empty;

switch (sortDirection)

{

case SortDirection.Ascending:

newSortDirection = "ASC";

break;

case SortDirection.Descending:

newSortDirection = "DESC";

break;

}

return newSortDirection;

}

protected void GVMetrics\_Sorting(object sender, GridViewSortEventArgs e)

{

DataSet ds = RetrieveData();

if (ds.Tables[0].Rows.Count > 0)

{

DataView dataView = new DataView(ds.Tables[0]);

dataView.Sort = e.SortExpression + " " + ConvertSortDirectionToSql(e.SortDirection);

Session["Data"] = dataView;

GVMetrics.DataSource = dataView;

GVMetrics.DataBind();

}

//}

}

#endregion

#region Grid GVMetrics

protected void GVMetrics\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

GVMetrics.PageIndex = e.NewPageIndex;

GVMetrics.DataSource = Session["Data"];

GVMetrics.DataBind();

BtnDeactive.Enabled = false;

BtnApplyUsages.Enabled = false;

}

protected void GVMetrics\_RowCreated(object sender, GridViewRowEventArgs e)

{

//string usages = "";

//if (e.Row.RowType == DataControlRowType.DataRow)

//{

// // e.Row.Attributes.Add("onmouseover", "this.previous\_color=this.style.backgroundColor;this.style.backgroundColor='#C1DAD7'");

// // e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor=this.previous\_color;");

// object objTemp = GVMetrics.DataKeys[e.Row.RowIndex].Value;

// int keys2 = System.Convert.ToInt32(objTemp);

// if (Session["Usage"] == null) return;

// DataSet ds2 = (DataSet)Session["Usage"];

// DataTable dt = ds2.Tables[0];

// var query = from o in dt.AsEnumerable()

// where o.Field<int>("MetricConfigurationID") == keys2

// select o;

// usages = "Usages:\n\n";

// int conteo = 0;

// if (query.Count() > 0)

// {

// foreach (var emp in query)

// {

// usages = usages + emp[1] + "\n";

// conteo++;

// }

// if (conteo == 1)

// e.Row.Cells[ColUsage].ToolTip = conteo.ToString() + " Usage";

// else

// e.Row.Cells[ColUsage].ToolTip = conteo.ToString() + " Usages";

// e.Row.Cells[ColUsage].Font.Underline = true;

// e.Row.Cells[ColUsage].ForeColor = System.Drawing.Color.Blue;

// e.Row.Cells[ColUsage].Text = conteo.ToString();

// }

// else

// e.Row.Cells[ColUsage].Text = "";

//}

}

protected void GVMetrics\_RowCommand(object sender, GridViewCommandEventArgs e)

{

string opcion = e.CommandName;

if (opcion == "Views")

{

int index = Convert.ToInt32(e.CommandArgument);

string MetricID2 = GVMetrics.DataKeys[index].Values["MetricConfigurationID"].ToString();

int MetricID = System.Convert.ToInt32(MetricID2);

if (Session["Usage"] == null) return;

DataSet ds2 = (DataSet)Session["Usage"];

DataTable dt = ds2.Tables[0];

var query = from o in dt.AsEnumerable()

where o.Field<int>("MetricConfigurationID") == MetricID

select new

{

MetricUsageName = o.Field<string>("MetricUsageName"),

Start = o.Field<System.DateTime>("effectivestart\_DT"),

End = o.Field<System.DateTime>("effectiveend\_DT")

};

GridView3.DataSource = query;

GridView3.DataBind();

}

}

protected void GVMetrics\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

int cual = e.Row.RowIndex;

//if (cual > 0)

//{

string fecha = e.Row.Cells[11].Text;

//string fecha2 = e.Row.Cells[14].Text; //ram - Status\_ID

if ( DateTime.Parse(fecha) < DateTime.Today ) //ram

{

e.Row.Cells[0].Enabled = false;

e.Row.Cells[11].ForeColor = System.Drawing.Color.Red;

}

//else if (fecha2.Trim() != "Active")

//{

// e.Row.Cells[0].Enabled = false;

// e.Row.Cells[14].ForeColor = System.Drawing.Color.Red; //ram

//}

else

{

e.Row.Cells[0].Enabled = true;

e.Row.Cells[11].ForeColor = System.Drawing.Color.Black;

//e.Row.Cells[14].ForeColor = System.Drawing.Color.Black;

}

//}

}

}

#endregion

#region First Buttons

protected void BtnInsert\_Click(object sender, EventArgs e)

{

string userName = GetUser();

int RegionID = System.Convert.ToInt32(ddlRegion.SelectedValue);

int SubRegionID = 0; // System.Convert.ToInt32(ddlSubRegion.SelectedValue);

int SiteID = System.Convert.ToInt32(ddlSite.SelectedValue);

int ClientID = System.Convert.ToInt32(ddlClient.SelectedValue);

int ProgramID = System.Convert.ToInt32(ddlProgram.SelectedValue);

int LOB = System.Convert.ToInt32(ddlLOB.SelectedValue);

int Metric = System.Convert.ToInt32(ddlMetric.SelectedValue);

string Symbol = ddlSymbol.Text;

try

{

if (Convert.ToDateTime(Date1.Text).Day != 1)

Date1.Text = Convert.ToDateTime(Date1.Text).Month.ToString() + "/01/" + Convert.ToDateTime(Date1.Text).Year.ToString();

}

catch

{

Date1.Text = DateTime.Now.Month.ToString() + "/01/" + DateTime.Now.Year.ToString();

}

string D1 = Date1.Text;

string D2 = DateTime.Now.Month.ToString() + "/" + DateTime.DaysInMonth(DateTime.Now.Year, DateTime.Now.Month).ToString() + "/" + DateTime.Now.Year.ToString();

string strCheck1 = CheckStartEndDate(D1, D2);

if (strCheck1 != "")

{

Message1.Text = strCheck1;

Blinks("Message1");

}

else

{

if (RegionID == 0 && SubRegionID == 0 && SiteID == 0)

{

String strSQL = "sp\_MetricConfig\_Insert\_CPL";

SqlParameter[] parameters = new SqlParameter[9];

parameters[0] = new SqlParameter("@ClientID", ClientID);

parameters[1] = new SqlParameter("@ProgramID", ProgramID);

parameters[2] = new SqlParameter("@LOB", LOB);

parameters[3] = new SqlParameter("@user1", userName);

parameters[4] = new SqlParameter("@Metric", Metric);

parameters[5] = new SqlParameter("@Symbolic", Symbol);

parameters[6] = new SqlParameter("@Date1", D1);

parameters[7] = new SqlParameter("@Date2", D2);

parameters[8] = new SqlParameter("@ReturnValue", SqlDbType.Int);

parameters[8].Direction = ParameterDirection.ReturnValue;

SqlHelper.ExecuteNonQuery(AppSettings.sqlConnDWReference, CommandType.StoredProcedure, strSQL, parameters);

string result = parameters[8].Value.ToString();

//string strMsg = "";

if (result == "1")

Message1.Text = "Metric Configuration already exists, dates updated.";

//strMsg = "Metric Configuration already exists, dates updated";

else

{

Message1.Text = "Metric configuration successfully created";

//strMsg = "Request for metric configuration mapping has been successfully created.";

Cache["ClientsCV"] = null;

}

Load\_Data();

Blinks("Message1");

//ShowMessage(strMsg, "xStream Datamaps");

}

else

{

String strSQL = "sp\_MetricConfig\_Insert";

SqlParameter[] parameters = new SqlParameter[11];

parameters[0] = new SqlParameter("@RegionID", RegionID);

parameters[1] = new SqlParameter("@SubRegionID", SubRegionID);

parameters[2] = new SqlParameter("@SiteID", SiteID);

parameters[3] = new SqlParameter("@ClientID", ClientID);

parameters[4] = new SqlParameter("@ProgramID", ProgramID);

parameters[5] = new SqlParameter("@LOB", LOB);

parameters[6] = new SqlParameter("@user1", userName);

parameters[7] = new SqlParameter("@Metric", Metric);

parameters[8] = new SqlParameter("@Symbolic", Symbol);

parameters[9] = new SqlParameter("@Date1", D1);

parameters[10] = new SqlParameter("@ReturnValue", SqlDbType.Int);

parameters[10].Direction = ParameterDirection.ReturnValue;

SqlHelper.ExecuteNonQuery(AppSettings.sqlConnDWReference, CommandType.StoredProcedure, strSQL, parameters);

string result = parameters[10].Value.ToString();

Load\_Data();

if (result == "1")

{

Message1.Text = "Metric Configuration already exists, dates and symbol updated.";

}

else

{

Message1.Text = "Metric configuration successfully created.";

Cache.Insert("ClientsCV", "1", null, DateTime.Now.AddSeconds(2), System.Web.Caching.Cache.NoSlidingExpiration);

}

Blinks("Message1");

}

}

}

protected void BtnDeactive\_Click(object sender, EventArgs e)

{

ucMetricInsert1.Visible = true;

SelectedRows();

Date33.Text = DateTime.Now.ToShortDateString();

Panel6.Visible = true;

Panel7.Visible = false;

BtnBack2Mapping.Visible = false;

Button4.Visible = true;

}

protected void BtnApplyUsages\_Click(object sender, EventArgs e)

{

Panel6.Visible = false;

Panel7.Visible = true;

BtnBack2Mapping.Visible = true;

Button4.Visible = false;

try

{

if (Convert.ToDateTime(Date1.Text).Day != 1)

Date1.Text = Convert.ToDateTime(Date1.Text).Month.ToString() + "/01/" + Convert.ToDateTime(Date1.Text).Year.ToString();

}

catch

{

Date1.Text = DateTime.Now.Month.ToString() + "/01/" + DateTime.Now.Year.ToString();

}

ucMetricInsert1.Visible = true;

SelectedRows();

//Date11.Text = DateTime.Now.ToShortDateString();

Date11.Text = Date1.Text;

//Date22.Text = "01/31/2099";

Date22.Text = DateTime.Now.Month.ToString() + "/" + DateTime.DaysInMonth(DateTime.Now.Year, DateTime.Now.Month).ToString() + "/" + DateTime.Now.Year.ToString();

ucMetricInsert1.SelectFromList("CV Pass Rate");

}

protected void BtnCancel\_Click(object sender, EventArgs e)

{

Visibles(false);

}

protected void Visibles(bool show)

{

Message1.Text = "";

BtnInsert.Visible = !show;

BtnDeactive.Visible = show;

lDate1.Visible = !show;

Date1.Visible = !show;

}

protected string ConvertFormat(string dt)

{

try

{

DateTime dt2 = DateTime.Parse(dt);

return dt2.ToShortDateString();

}

catch

{

return "";

}

}

protected void SelectedRows()

{

if (!GVMetrics.Visible) return;

ucMetricInsert1.CleanList();

DataSet Listado = new DataSet("Listado");

DataTable Lista = Listado.Tables.Add("Lista");

Lista.Columns.Add("Region", typeof(string));

Lista.Columns.Add("SubRegion", typeof(string));

Lista.Columns.Add("Site", typeof(string));

Lista.Columns.Add("Client", typeof(string));

Lista.Columns.Add("Program", typeof(string));

Lista.Columns.Add("LOB", typeof(string));

Lista.Columns.Add("Metric", typeof(string));

Lista.Columns.Add("MetricID", typeof(string));

Lista.Columns.Add("StartDT", typeof(string));

Lista.Columns.Add("EndDT", typeof(string));

Lista.Columns.Add("MetricUsageName", typeof(string));

Lista.Columns.Add("UsageStart\_DT", typeof(string));

Lista.Columns.Add("UsageEnd\_DT", typeof(string));

bool selected = false;

DataSet ds = new DataSet();

ds = LoadAllData();

DataTable dt = new DataTable();

dt = ds.Tables[0];

int row = 0;

foreach (GridViewRow dr in GVMetrics.Rows)

{

CheckBox chk = (CheckBox)dr.Cells[0].FindControl("Check1");

if (chk.Checked)

{

string ConfigID2 = GVMetrics.DataKeys[row].Values["MetricConfigurationID"].ToString();

int ConfigID3 = System.Convert.ToInt32(ConfigID2);

var query = from o in dt.AsEnumerable()

where (o.Field<int>("MetricConfigurationID") == ConfigID3)

select o;

string Site2 = "";

string Client2 = "";

string Program2 = "";

string LOB2 = "";

string Metric2 = "";

foreach (DataRow dr2 in query)

{

string Region = dr2["Region\_Name"].ToString();

string SubRegion = dr2["SubRegion\_Name"].ToString();

string Site = dr2["Site"].ToString();

string Client = dr2["Client"].ToString();

string Program = dr2["Program"].ToString();

string LOB = dr2["LOB"].ToString();

string Metric = dr2["MetricName"].ToString();

string ConfigID = dr2["MetricConfigurationID"].ToString();

string DT1 = ConvertFormat(dr2["EffectiveStart\_DT"].ToString());

string DT2 = ConvertFormat(dr2["EffectiveEnd\_DT"].ToString());

string Usage = dr2["MetricUsageName"].ToString();

string DT11 = ConvertFormat(dr2["UsageStart\_DT"].ToString());

string DT22 = ConvertFormat(dr2["UsageEnd\_DT"].ToString());

if (Site2 == Site && Client2 == Client && LOB2 == LOB && Program2 == Program && Metric2 == Metric)

{

Lista.Rows.Add("", "", "", "", "", "", "", 0, "", "", Usage, DT11, DT22);

}

else

{

Lista.Rows.Add(Region, SubRegion, Site, Client, Program, LOB, Metric, ConfigID, DT1, DT2, Usage, DT11, DT22);

}

Site2 = Site;

Client2 = Client;

LOB2 = LOB;

Program2 = Program;

Metric2 = Metric;

}

selected = true;

}

row++;

}

if (!selected)

{

Message1.Text = "No rows selected.";

Blinks("Message1");

//ShowMessage("No rows selected", "xStream Datamaps");

return;

}

Message1.Text = "";

Hide\_Show(true);

GVLista.DataSource = Listado;

GVLista.DataBind();

GVLista.Visible = true;

string compare = "";

string compare1 = "";

for (int row2 = 0; row2 < GVLista.Rows.Count; row2++)

{

compare = GVLista.Rows[row2].Cells[8].Text;

if (compare != "&nbsp;")

{

if (compare1 == "") compare1 = compare;

if (DateTime.Parse(compare1) < DateTime.Parse(compare)) compare1 = compare;

}

}

compare1 = compare1.Replace("-", "/");

compare1 = DateTime.Parse(compare1).ToShortDateString();

//Date11.Text = compare1;

}

protected void LoadBack()

{

ucMetricInsert1.CleanList();

DataSet Listado = new DataSet("Listado");

DataTable Lista = Listado.Tables.Add("Lista");

Lista.Columns.Add("Region", typeof(string));

Lista.Columns.Add("SubRegion", typeof(string));

Lista.Columns.Add("Site", typeof(string));

Lista.Columns.Add("Client", typeof(string));

Lista.Columns.Add("Program", typeof(string));

Lista.Columns.Add("LOB", typeof(string));

Lista.Columns.Add("Metric", typeof(string));

Lista.Columns.Add("MetricID", typeof(string));

Lista.Columns.Add("StartDT", typeof(string));

Lista.Columns.Add("EndDT", typeof(string));

Lista.Columns.Add("MetricUsageName", typeof(string));

Lista.Columns.Add("UsageStart\_DT", typeof(string));

Lista.Columns.Add("UsageEnd\_DT", typeof(string));

DataSet ds = new DataSet();

ds = LoadAllData();

DataTable dt = new DataTable();

dt = ds.Tables[0];

for (int row = 0; row < GVLista.Rows.Count; row++)

{

string ConfigID2 = GVLista.DataKeys[row].Values["MetricID"].ToString();

if (ConfigID2 != "0")

{

int ConfigID3 = System.Convert.ToInt32(ConfigID2);

var query = from o in dt.AsEnumerable()

where (o.Field<int>("MetricConfigurationID") == ConfigID3)

select o;

string Site2 = "";

string Client2 = "";

string Program2 = "";

string LOB2 = "";

string Metric2 = "";

foreach (DataRow dr2 in query)

{

string Region = dr2["Region\_Name"].ToString();

string SubRegion = dr2["SubRegion\_Name"].ToString();

string Site = dr2["Site"].ToString();

string Client = dr2["Client"].ToString();

string Program = dr2["Program"].ToString();

string LOB = dr2["LOB"].ToString();

string Metric = dr2["MetricName"].ToString();

string ConfigID = dr2["MetricConfigurationID"].ToString();

string DT1 = ConvertFormat(dr2["EffectiveStart\_DT"].ToString());

string DT2 = ConvertFormat(dr2["EffectiveEnd\_DT"].ToString());

string Usage = dr2["MetricUsageName"].ToString();

string DT11 = ConvertFormat(dr2["UsageStart\_DT"].ToString());

string DT22 = ConvertFormat(dr2["UsageEnd\_DT"].ToString());

if (Site2 == Site && Client2 == Client && LOB2 == LOB && Program2 == Program && Metric2 == Metric)

{

Lista.Rows.Add("", "", "", "", "", "", "", 0, "", "", Usage, DT11, DT22);

}

else

{

Lista.Rows.Add(Region, SubRegion, Site, Client, Program, LOB, Metric, ConfigID, DT1, DT2, Usage, DT11, DT22);

}

Site2 = Site;

Client2 = Client;

LOB2 = LOB;

Program2 = Program;

Metric2 = Metric;

}

}

}

GVLista.DataSource = Listado;

GVLista.DataBind();

GVLista.Visible = true;

string compare = "";

string compare1 = "";

for (int row2 = 0; row2 < GVLista.Rows.Count; row2++)

{

compare = GVLista.Rows[row2].Cells[8].Text;

if (compare != "&nbsp;")

{

if (compare1 == "") compare1 = compare;

if (DateTime.Parse(compare1) < DateTime.Parse(compare)) compare1 = compare;

}

}

compare1 = compare1.Replace("-", "/");

compare1 = DateTime.Parse(compare1).ToShortDateString();

}

#endregion

#region Second Buttons

private string CheckStartEndDate(string startDate, string endDate)

{

string str1 = "";

if (Convert.ToDateTime(startDate).Day != 1)

str1 = "Start date should be the 1st of the month.";

else

{

if (DateTime.Now.Month - Convert.ToDateTime(startDate).Month > 1)

str1 = "Start date not allowed. Earliest start date allowed is the first of the previous month.";

}

if (str1 == "")

str1 = CheckEndDate(endDate);

//HERE: Check if start and end dates are both in the same month?

return str1;

}

private string CheckStartDate(string startDate)

{

string str1 = "";

if (Convert.ToDateTime(startDate).Day != 1)

str1 = "Start date should be the 1st of the month.";

else

{

if (DateTime.Now.Month - Convert.ToDateTime(startDate).Month > 1)

str1 = "Start date not allowed. Earliest start date allowed is the first of the previous month.";

}

return str1;

}

private string CheckEndDate(string endDate)

{

if (Convert.ToDateTime(endDate).Day != 1 &&

Convert.ToDateTime(endDate).Day != DateTime.DaysInMonth(Convert.ToDateTime(endDate).Year, Convert.ToDateTime(endDate).Month))

return "End date should be the 1st of the month or the last day of the month.";

else

return "";

}

//New: For inserting change request (ram)

// ACTIVATE USAGE

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

protected void BtnAssignUsage\_Click(object sender, EventArgs e)

{

bcMC = new bcCVMetricConfig();

Message2.Text = "";

string userName = GetUser();

string tmpR = bcMC.GetUserDetails(userName, "Metric Name Usage");

string D1 = Date11.Text;

if (D1 == "")

D1 = DateTime.Today.Month.ToString() + "/01/" + DateTime.Today.Year.ToString();

//string D2 = Date22.Text;

string D2 = "01/31/2099";

string strCheck1 = CheckStartDate(D1);

if (strCheck1 != "")

Message2.Text = strCheck1;

else

{

string D3 = DateTime.Parse(D1).Month.ToString() + "/01/" + DateTime.Parse(D1).Year.ToString(); ;

//DateTime D4 = DateTime.Parse(D3);

//D3 = DateTime.Parse(D1).Month.ToString() + "/01/" + DateTime.Parse(D1).Year.ToString();

string tmpR2 = bcMC.CheckSubmitAuthority();

if (tmpR2 == "")

{

bool validStartDate = true;

bool validJustif = true;

bool validGoal = true;

string compareSD = "";

string newGoal = "";

string justif = "";

for (int row = 0; row < GVLista.Rows.Count; row++)

{

//Get mapping start date

compareSD = GVLista.Rows[row].Cells[8].Text;

newGoal = ((TextBox)GVLista.Rows[row].FindControl("txtGoalTmp")).Text;

justif = ((TextBox)GVLista.Rows[row].FindControl("txtJustificationTmp")).Text;

if (compareSD != "&nbsp;")

{

if (DateTime.Parse(compareSD) > DateTime.Parse(D1))

{

validStartDate = false;

Message2.Text = "Start Date is older than at least one mapping start date. change start date.";

break;

}

}

if (justif.Trim() == "")

{

validJustif = false;

Message2.Text = "Justification is required. Please check justification column value(s).";

break;

}

double newG;

validGoal = Double.TryParse(newGoal, out newG);

if (!validGoal)

{

Message2.Text = "Invalid goal value found. Please check the goal column.";

break;

}

else

if (newG < bcMC.GetMinValidGoal)

{

Message2.Text = "Invalid goal value found. Minimum allowed value is " + bcMC.GetMinValidGoal.ToString() + ".";

validGoal = false;

break;

}

}

if (validJustif && validGoal && validStartDate)

{

try

{

int[] lista = (int[])ucMetricInsert1.listado2();

double goal = 0;

bool Applied = false;

for (int row = 0; row < GVLista.Rows.Count; row++)

{

string ConfigID = GVLista.DataKeys[row].Values["MetricID"].ToString();

goal = Convert.ToDouble(((TextBox)GVLista.Rows[row].FindControl("txtGoalTmp")).Text);

if (ConfigID != "0")

{

int row2 = 1;

while (lista[row2] != 0)

{

bcMC.AddNewMetricUsageRequest(lista[row2], System.Convert.ToInt32(ConfigID), userName, D1, D2, D3, justif, goal);

Applied = true;

ProcessEmail1("Add Usage", row, D1, D2, justif, goal);

row2++;

}

}

}

if (Applied)

{

Message2.Text = "Assign usage request has been created successfully.";

LoadBack();

}

else

Message2.Text = "No metric usage was selected, select at least one from the list.";

//ShowMessage("No metric usage was selected, select at least one from the list", "Metric Usage");

}

catch (Exception ex)

{

if (ex.Message.Contains("UNIQUE KEY"))

Message2.Text = "Error. Duplicate assign usage request found.";

else

Message2.Text = "Error: " + ex.Message;

}

}

}

else //ram

{

Message2.Text = tmpR2;

//ShowMessage(tmpR2, "xStream Datamaps");

}

} //strCheck

if (Message2.Text != "")

Blinks("Message2");

}

private void ProcessEmail1(string changeType, int iRow, string startDate, string endDate, string justif, double goal)

{

string strBody = "";

if (changeType == "Add Usage")

strBody = "There is a request to ADD USAGE for" + "\n\n";

else if (changeType == "Remove Usage")

strBody = "There is a request to REMOVE USAGE for" + "\n\n";

strBody += "Site: " + HttpUtility.HtmlDecode(GVLista.Rows[iRow].Cells[3].Text.Trim().ToString()) + "\n";

strBody += "Client: " + HttpUtility.HtmlDecode(GVLista.Rows[iRow].Cells[4].Text.Trim().ToString()) + "\n";

strBody += "Program: " + HttpUtility.HtmlDecode(GVLista.Rows[iRow].Cells[5].Text.Trim().ToString()) + "\n";

strBody += "LOB: " + HttpUtility.HtmlDecode(GVLista.Rows[iRow].Cells[6].Text.Trim().ToString()) + "\n";

strBody += "Metric: " + HttpUtility.HtmlDecode(GVLista.Rows[iRow].Cells[7].Text.Trim().ToString()) + "\n";

if (changeType == "Add Usage")

{

strBody += "Start Date: " + startDate + "\n";

strBody += "Goal: " + goal.ToString() + "\n";

}

else if (changeType == "Remove Usage")

{

strBody += "End Date: " + endDate + "\n";

}

strBody += "Justification: " + justif + "\n\n";

SendEmail(strBody, "Data Governance");

}

protected void SendEmail(string strBody, string sendToGroup)

{

string machine = System.Environment.MachineName.ToString();

//if (machine == "WWS10-001-STAGE" || machine == "LAPTOP\_RMARTIN") return;

int steps = 0;

string SendTo = "";

try

{

MailMessage mail = new MailMessage();

SmtpClient SmtpServer = new SmtpClient("mailedc01");

steps = 1;

string userName = GetUser();

string From = "CVMetric\_ChangeRequest\_No\_Reply@stream.com";

string BCC = "requester@stream.com";

try

{

//From = System.Web.Security.Membership.GetUser(userName).Email.ToString();

BCC = System.Web.Security.Membership.GetUser(userName).Email.ToString();

steps = 2;

}

catch

{

//From = "CVMetric\_ChangeRequest\_No\_Reply@stream.com";

//BCC = "robert.martin@stream.com";

}

mail.From = new MailAddress(From);

mail.Bcc.Add(new MailAddress(BCC));

steps = 3;

SendTo = GetUsers(sendToGroup);

if (SendTo == "")

{

Message2.Text = "Error sending the email. The list for approvers is empty";

Blinks("Message2");

return;

}

steps = 4;

// take the last ','

int largo = SendTo.Length;

SendTo = SendTo.Substring(0, largo - 1);

steps = 5;

mail.To.Add(SendTo);

steps = 6;

//sp\_RefMap\_GetEmployeeName

string Nombre = GetUserName();

steps = 7;

string place = "";

try

{

int pos1 = strBody.IndexOf("Site");

int pos2 = strBody.IndexOf("Client");

string sitio = strBody.Substring(pos1, (pos2 - pos1 - 1));

place = " for " + sitio;

}

catch

{

}

string strSubject = "xStream Data Maps, Request to Approve ";

if (machine == "WWS10-001-STAGE" || machine == "LAPTOP\_RMARTIN")

strSubject = "Test Email: " + strSubject;

if (SendTo.Length > 40)

mail.Subject = strSubject + place;

else

mail.Subject = strSubject;

mail.Body = strBody + "\nRequestor: " + Nombre + "\n\nClick this link to access the tool: http://referencesystem.stream.com/ ";

SmtpServer.Port = 25;

//SmtpServer.Credentials = new System.Net.NetworkCredential("90980", "SAL@Si0920");

//SmtpServer.EnableSsl = true;

steps = 10;

SmtpServer.Send(mail);

}

catch

{

Message2.Text = "Error sending the email in step " + steps.ToString() + " , report it to Robert.Martin@Stream.com " + SendTo;

Blinks("Message2");

}

}

protected string GetUserName()

{

if (Session["Employee"] != null)

{

return Session["Employee"].ToString();

}

string From = GetUser();

String strSQL = "sp\_RefMap\_GetEmployeeName2";

SqlParameter[] parameters = new SqlParameter[1];

parameters[0] = new SqlParameter("@LoginID", SqlDbType.NVarChar, 100);

//parameters[0].Value = System.Convert.ToInt32(From);

parameters[0].Value = From;

object names = SqlHelper.ExecuteScalar(\_connStr, CommandType.StoredProcedure, strSQL, parameters);

Session["Employee"] = names.ToString();

return names.ToString();

}

protected string GetUsers(string strRole)

{

bcCVMetricConfig bcMC = new bcCVMetricConfig();

if (Cache["UserList"] != null)

{

return Cache["UserList"].ToString();

}

string listado = "";

try

{

string correo1, user1;

string[] users = bcMC.GetUsersInRole(strRole);

for (int i = 0; i < users.Length; i++)

{

user1 = users[i];

correo1 = System.Web.Security.Membership.GetUser(user1).Email.ToString();

listado = listado + correo1 + ",";

}

}

catch

{

listado = "Robert.Martin@stream.com,";

}

if (listado.Length > 40)

{

Cache.Insert("UserList", listado, null, System.Web.Caching.Cache.NoAbsoluteExpiration, new TimeSpan(0, 15, 0));

}

return listado;

}

protected void BtnDeactivate\_Click(object sender, EventArgs e)

{

bcMC = new bcCVMetricConfig();

string userName = GetUser();

string tmpR = bcMC.GetUserDetails(userName, "Metric Name Usage");

Message2.Text = "";

string D2 = Date22.Text;

if (D2 == "")

//D2 = "01/31/2099";

D2 = DateTime.Now.Month.ToString() + "/" + DateTime.DaysInMonth(DateTime.Now.Year, DateTime.Now.Month).ToString() + "/" + DateTime.Now.Year.ToString();

string tmpR2 = bcMC.CheckSubmitAuthority();

if (tmpR2 == "")

{

bool validJustif = true;

bool validEndDate = true;

bool validUsage = true;

string justif = "";

string usageStartDate = "";

string usageEndDate = "";

for (int row = 0; row < GVLista.Rows.Count; row++)

{

usageStartDate = GVLista.Rows[row].Cells[12].Text.ToString().Trim();

usageEndDate = GVLista.Rows[row].Cells[13].Text.ToString().Trim();

justif = ((TextBox)GVLista.Rows[row].FindControl("txtJustificationTmp")).Text;

if (usageEndDate == "" || usageEndDate == "&nbsp;")

{

Message2.Text = "Unable to deactivate usage. Metric must be active to deactivate.";

validUsage = false;

break;

}

if (DateTime.Parse(usageEndDate) <= DateTime.Now)

{

Message2.Text = "Usage is already end dated. Metric must be active to deactivate.";

validUsage = false;

break;

}

if (DateTime.Parse(D2) < DateTime.Parse(usageStartDate))

{

Message2.Text = "End date is earlier than the metric usage start date. Check end date value.";

validEndDate = false;

break;

}

string strCheckEndDate = CheckEndDate(D2);

if (strCheckEndDate != "")

{

Message2.Text = strCheckEndDate;

validEndDate = false;

break;

}

if (justif.Trim() == "")

{

Message2.Text = "Justification is required. Please check justification column value(s).";

validJustif = false;

break;

}

} //end for

if (validUsage && validJustif && validEndDate)

{

try

{

int[] lista = (int[])ucMetricInsert1.listado2();

bool Applied = false;

for (int row = 0; row < GVLista.Rows.Count; row++)

{

string ConfigID = GVLista.DataKeys[row].Values["MetricID"].ToString();

if (ConfigID != "0")

{

int row2 = 1;

while (lista[row2] != 0)

{

bcMC.DeactivateMetricUsageRequest(lista[row2], System.Convert.ToInt32(ConfigID), userName, justif, D2, D2, D2);

Applied = true;

ProcessEmail1("Remove Usage", row, D2, D2, justif, 0.00);

row2++;

}

}

}

if (Applied)

{

Message2.Text = "Remove usage request has been created successfully.";

LoadBack();

}

else

Message2.Text = "No metric usage selected. Select at least one from the list.";

}

catch (Exception ex)

{

if (ex.Message.Contains("UNIQUE KEY"))

Message2.Text = "Error. Duplicate assign/remove usage request found.";

else

Message2.Text = "Error: " + ex.Message;

}

}

} //if tmpR2 == ""

else

{

Message2.Text = tmpR2;

}

if (Message2.Text != "")

Blinks("Message2");

}

protected void BtnBack2Mapping\_Click(object sender, EventArgs e)

{

Hide\_Show(false);

Load\_Data();

}

protected void Hide\_Show(bool option)

{

Panel0.Visible = !option;

Panel1.Visible = !option;

GVMetrics.Visible = !option;

Panel3.Visible = option;

Panel4.Visible = !option;

Message2.Text = "";

}

//++++++++++++++++

protected void BtnDeactiveMapping\_Click(object sender, EventArgs e)

{

List<SqlDataRecord> ID\_list = new List<SqlDataRecord>();

SqlMetaData[] tvp\_definition = { new SqlMetaData("n", SqlDbType.Int) };

SqlDataRecord rec;

bool Applies = false;

bool Applies2 = true;

for (int row = 0; row < GVLista.Rows.Count; row++)

{

string ConfigID = GVLista.DataKeys[row].Values["MetricID"].ToString();

if (ConfigID != "0")

{

string start\_fecha = GVLista.Rows[row].Cells[8].Text.ToString();

if (DateTime.Parse(start\_fecha) > DateTime.Parse(Date33.Text) )

{

Message2.Text = "End date is earlier than the metric creation date for "+ GVLista.Rows[row].Cells[6].Text.ToString()+". Change end date";

//ShowMessage("End date is earlier than the metric creation date for " + GVLista.Rows[row].Cells[6].Text.ToString() + ". Change end date", "xStream Datamaps");

Blinks("Message2");

return;

}

// mapping ya esta desactivado

string end\_fecha = GVLista.Rows[row].Cells[9].Text.ToString();

if (DateTime.Parse(end\_fecha) > DateTime.Today)

{

Applies = true;

rec = new SqlDataRecord(tvp\_definition);

rec.SetInt32(0, Int32.Parse(ConfigID));

ID\_list.Add(rec);

}

}

}

if (!Applies)

{

Message2.Text = "All mappings are already deactivated.";

//ShowMessage("All mappings are already deactivated.", "Metric Config Mapping");

return;

}

string D2 = Date33.Text;

string D1 = D2;

//string userName = HttpContext.Current.User.Identity.Name;

string userName = GetUser();

String strSQL = "sp\_MetricConfiguration\_Deactivation";

SqlParameter param1 = new SqlParameter("@Config\_List", ID\_list);

SqlParameter param2 = new SqlParameter("@Date1", D1);

SqlParameter param3 = new SqlParameter("@Date2", D2);

SqlParameter param4 = new SqlParameter("@user1", userName);

param1.SqlDbType = SqlDbType.Structured;

param2.SqlDbType = SqlDbType.DateTime;

param3.SqlDbType = SqlDbType.DateTime;

param4.SqlDbType = SqlDbType.NVarChar;

SqlParameter[] parms = new SqlParameter[5];

parms[0] = param1;

parms[1] = param2;

parms[2] = param3;

parms[3] = param4;

parms[4] = new SqlParameter("@ReturnValue", SqlDbType.Int);

parms[4].Direction = ParameterDirection.ReturnValue;

SqlHelper.ExecuteNonQuery(\_connStr, CommandType.StoredProcedure, strSQL, parms);

int result = System.Convert.ToInt32(parms[4].Value.ToString());

result = ID\_list.Count() - result;

if (result != 0)

{

Message2.Text = "End date is earlier than the metric creation date for "+ result.ToString() + " mapping";

//ShowMessage("End date is earlier than the metric creation date for " + result.ToString() + " mapping", "Metric Config Mapping");

Applies2 = false;

}

if (Applies2)

{

Hide\_Show(false);

Load\_Data();

Message2.Text = "The mappings and usages selected have been deactivated ";

Blinks("Message2");

//ShowMessage("The Mappings and Usages selected have been deactivated ", "xStream Datamaps");

}

}

protected void GVMetrics\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

#endregion

}