A REXX script to produce a XAML bar chart

Table of Contents

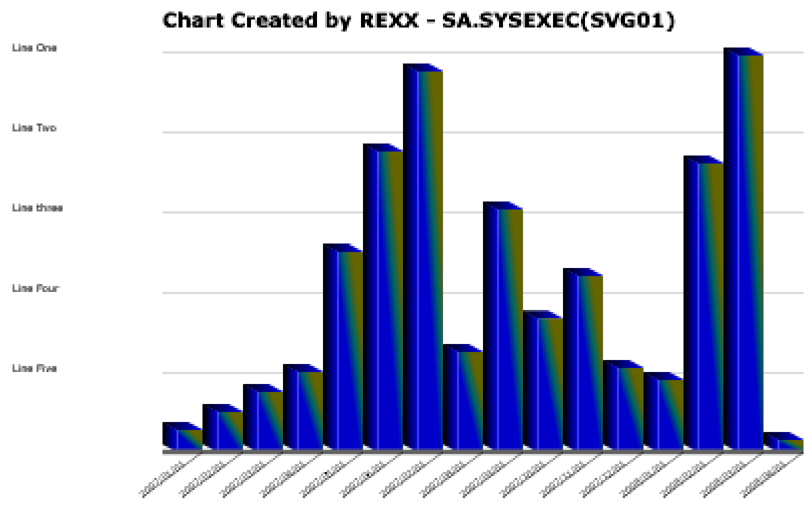
XAML Barchart

You will need an XAML (SilverLight) enabled web browser. XAML is a standard developed by Microsoft. It usage is by several products but for our purposes, the main product that we will be using with XAML is Microsoft's Silverlight®. Silverlight® is a browser plug-in that must be installed prior to viewing the output of this environment. For more information please go to http://silverlight.net/.

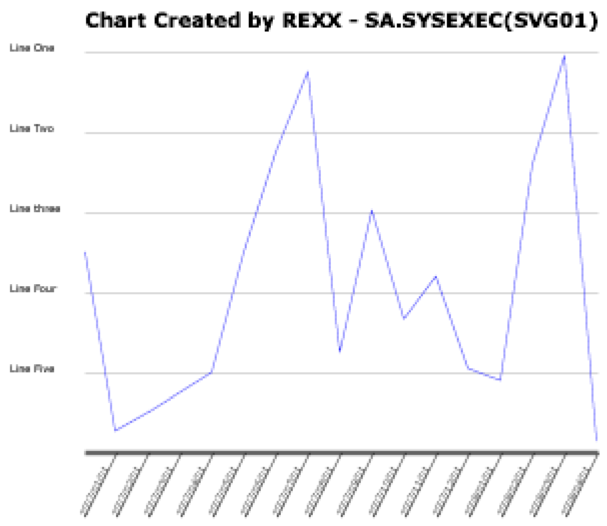
What can be produced

Let's take a look at what our REXX program can produce:

ChartType=”COLUMN”



ChartType=”LINE”

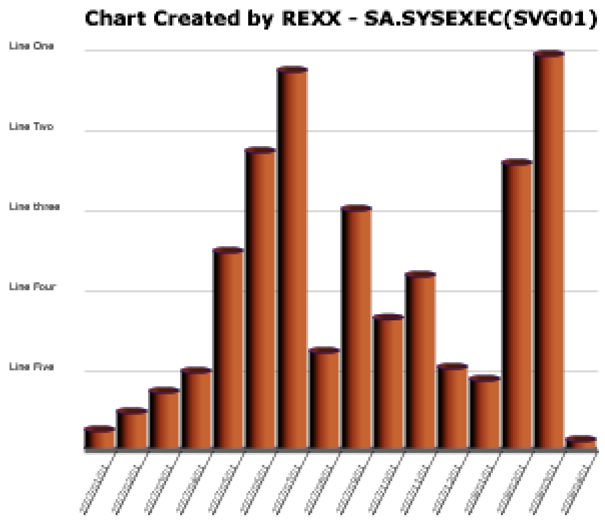


Multiple lines for a ChartType=”LIN” can be produced by supplying all the percentages for all lines in ddname PERCENTS. Then separate each line’s percentages with a forward slash (“/”). And example of how three lines would be represented in ddname //PERCENTS:

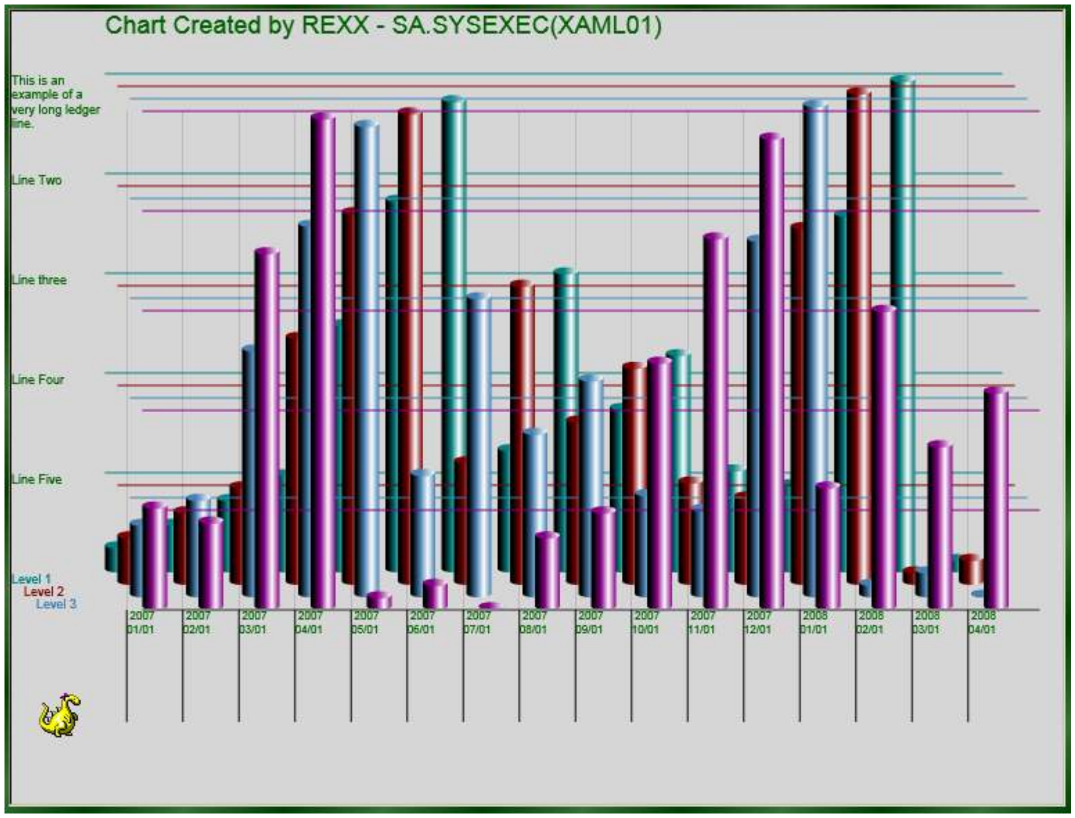
```
//PERCENTS  DD *

10 15 50 45 66

/

66 76 44 69 62

/

5 79 65 43 44
```

It will be assumed that there is equal number of percentages for each line that is graphed.

ChartType="CYLINDER"



Multi-Level Bar-Chart



To achieve this effect, making use of variables SPACING, XSHIFT, and YSHIFT will become necessary.

As for the REXX program is concerned, there is no limit as to how many levels or lines can be produced. The limitation is more with XAML.

How it is produced

```
//CBRXXRUN JOB ,'REXX RUN',CLASS=Q,MSGCLASS=X,MSGLEVEL=(1,1),
//    NOTIFY=TUR0563
//*===============================================================
//STEP10  EXEC PGM=IRXJCL,PARM='XAML01'
//SYSTSIN  DD DUMMY                        /* TERMINAL INPUT
//SYSTSPRT DD SYSOUT=*                     /* TERMINAL OUTPUT
//SYSUDUMP DD SYSOUT=*                     /* TERMINAL OUTPUT
//SYSEXEC DD DISP=SHR,DSN=SA.SYSEXEC
//XAMLIN  DD *
<Canvas
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
<PLACE BARCHART HERE>
</Canvas>
//XAMLOUT DD DISP=(NEW,PASS),
//           DSN=&XAML,
//           UNIT=DASD,
//           STORCLAS=SCUSER,
//           DCB=(LRECL=255,RECFM=VB)
//ENVVAR  DD *   <=== These are default overrides
  Title=”Place title here”
  LedgerLineInterval = 5
  LedgerText.0 = LedgerLineInterval
  LedgerText.1 = 'Line One'
  LedgerText.2 = 'Line Two'
  LedgerText.3 = 'Line three'
  LedgerText.4 = 'Line Four'
  LedgerText.5 = 'Line Five'
  LedgerFontSize=10
  FooterText.0 = 12
  FooterText.1 = '1999'

  FooterText.2 = '2000'

  FooterText.3 = '2001'

  FooterText.4 = '2002'

  FooterText.5 = '2003'

  FooterText.6 = '2004'

  FooterText.7 = '2005'

  FooterText.8 = '2006'

  FooterText.9 = '2007'

  FooterText.10 = '2008'

  FooterText.11 = '2009'

  FooterText.12 = '2010'

//PERCENTS DD *  <== These are your column heights in percentages

5.5 10 15 20 50 75 95 25

60.5 33.3

44

21

//*================================================================

//STEP02  EXEC BATCHTMP

//SYSIN  DD *

 OCOPY INDD(I1) OUTDD(O1)

/*

//I1     DD  DISP=(OLD,DELETE),DSN=&XAML

//O1     DD PATHOPTS=(OWRONLY,OCREAT),PATHMODE=(SIRWXU),

// PATH='/usr/lpp/internet/server_root/saapps/pub/xaml/Xaml.txt'
```

DDname(XAMLIN)

The template in ddname XAMLIN need only contain the XAML text with at least the bar chart location indicator of '**<PLACE BARCHART HERE>**'. This instructs the REXX script to start placing the actual bar chart at this location. You could, if you wanted, have only the location indicator. All the rest is only needed base on what you will eventually do with it. It is not required in any way by the REXX script. The script will still write the resulting bar chart to ddname XAMLOUT even if it isn't a complete XAML document. Just be sure to designate XAMLOUT as a normal type data set (not a Sysout to SMTP).

DDname(ENVAR)

Environment variable overrides can be placed in ddname ENVAR. They are not required but you may not like the size and look of the default bar chart. Try it with out them, but I'm sure you will start using them. **All measurements must be described in Pixels where 100 is approximately 1 inch**.

Variable Name	***Description***
Animation	Set to "Y" if animation is desired. Currently, only animation is available for Type='CYL' charts.
AnimationType	Set to 1, 2, or 3…. Selects a different kind of animation for the chart. They are sort of difficult to describe. Just try them and see which one you like.
BoxHeight	The total height of the bar chart. The height of the bars must be given as percentages. The percentages are then measured against this maximum height. This makes the bar chart scalable.
CapHeight	This regulates the height (or thickness) of the top of the column (or cylinder). The more the height, the steeper the angle of the column appears, because this also affects the look of the bottom of the column as well.
ChartType	"**COL**" for square columns "**CYL**" for circular cylinders "**LIN**" for a line type chart

Color.#	This is another stem variable which can be set to change the default colors of the individual lines and bars that are drawn. To change them, code the following:
	
	Color.0 = 5 (*number of lines or levels that will be drawn*)
	Color.1 = "DarkBlue"
	Color.2 = "Yellow"
	Color.3 = "Chartreuse"
	Color.4 = "DarkRed"
	Color.5 = "Indigo"
	Any of the standard web color names can be used. For a complete list go to:
	http://www.w3schools.com/css/css_colornames.asp

FooterFontSize	**Point** Size (not pixel) of the font used to display the Column Footers.

FooterText.??	The text used as Footers for you bars. You should have the same amount of these as you would have columns. FooterText.0 should be set to the number of expected columns. Example:
	
	FooterText.0 = 2
	FooterText.1 = "2002"
	FooterText.2 = '2003"
	If not provided, these headings default to the percentage value provided in the

Variable Name	*Description*
	//PERCENTS DD statement.
CylinderWidth	If ChartType = "CYL" or "LIN" then this controls the width of the cylinders (*or for line charts, this would control the spacing between the peaks and valleys*). For line charts this can get pretty small (ex: CylinderWidth = 2). This would mean you can get a lot more percentages into a line chart then you would be able to in a cylinder chart.
FooterDepth	By default, the depth of the Footer (or more appropriately the footer), is initialized to the column (or cylinder) width.
FooterSpan	Used only with Type="LIN" charts. A line chart can be a lot more condensed than a cylinder or a column type bar chart. Condensed meaning you can "squeeze" in more peaks and valleys into a smaller area. When this is done (*by use of the PointWidth variable*) you also start "squeezing" in the headings. When the headings start to get too thin, you can "span" you headings across multiple points (Line widths). As an example, you can "span" a heading across, let say, 10 line points (ex: FooterSpan = 10). If you had 100 individual percentages and you specified FooterSpan = 10, then instead of having 100 headings, you would then only have 10 headings (that is 100 divided by 10).
FooterWidth	By default, the width of the Footer is initialized to twice its width.
lci	Line Color Index – controls which starting color of the list in stem variable Color.# that you wish to begin with. Default is 1.
LedgerFontSize	**Point** Size (not pixel) of the font used to display the Ledger text
LedgerLineInterval	This is the number of ledger lines that will be produced. Basically, the BoxHeight is divided by this number to get the spacing between the Ledger Lines that will be created.

LedgerText.??	This the text used to display with each ledger line. These are optional but if used should have as many as specified in variable LedgerLineInterval. Example:
	
	LedgerLineInterval = 3
	LedgerText.0 = 3
	LedgerText.1 = "This is Ledger 1"
	LedgerText.2 = "This is Ledger 2"
	LedgerText.2 = "This is Ledger 3"

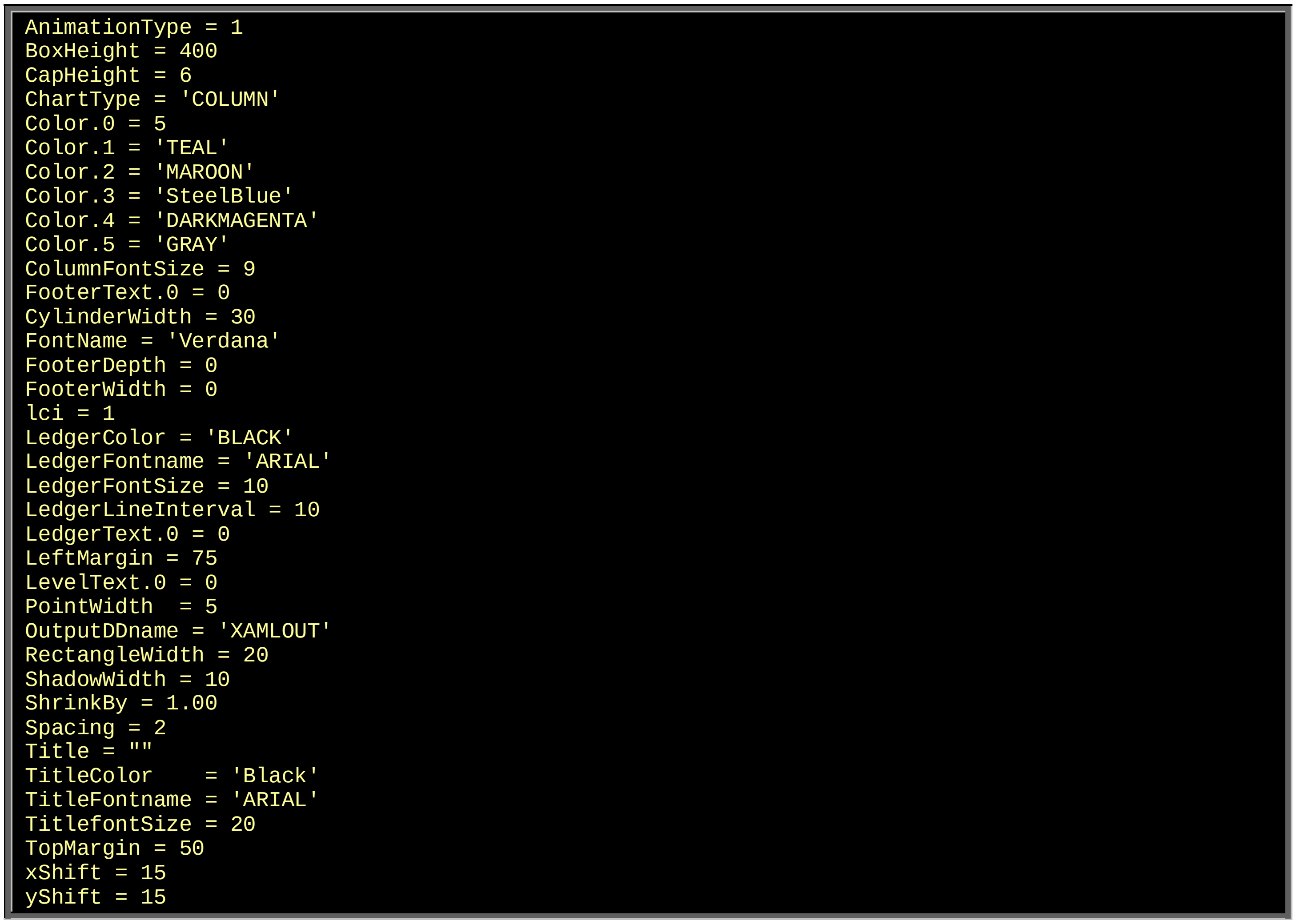
LeftMargin	This is the amount of space to the left of the beginning of the chart. If you specify Ledger line text, you will need some of this space.

LevelText.#	ChartType = "COL" or "CYL"
	
	This contains the text that will be used for multi-level bar charts. If used, it should be set as follows:
	
	LevelText.0 = *number of levels*
	LevelText.1='Text for level 1"

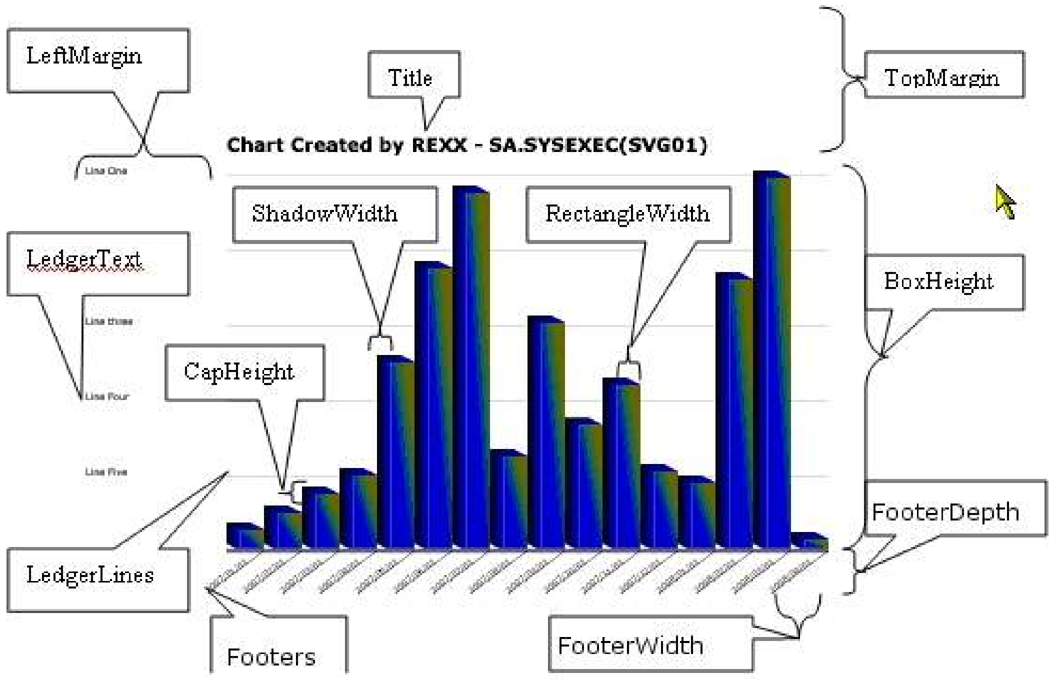
Variable Name	***Description***
	LevelText.n='Text for level n" If more levels are created than text is provided, blanks are the default.
PointWidth	If ChartType = "LIN" Actually controls the spacing between points of a line chart. This is similar to CylinderWidth for the cylinder chart. The smaller the PointWidth, the steeper the angles of the line chart.
RectangleWidth	If ChartType = "COL" then This is the width of the "face" of the individual columns. The entire column width is the total of both the RectangleWidth and the ShadowWidth.
ShadowWidth	If ChartType = "COL" then The width of the shadow for each generated column.
ShrinkBy	Must be > 0.00 and <= 2.00. If ShrinkBy is less than 1.00, the entire bar chart is made smaller by that percentage. If it is greater than 1.00, it will increase the size by that percentage. This is used so that if the overall size of the chart is not acceptable, one can just shrink it (or expand it) by using ShrinkBy. There is no need to calculate every size variable to get the chart within an acceptable size.
Spacing	Sets the spacing between columns (or cylinders). For no spacing just set this to zero.
Title	Text that will be used as the title
TitleFontSize	The size, in points, of your title. The larger the point size, the more TopMargin you may need to provide.
TopMargin	This is the amount of space separating the highest possible bar in the bar chart from the top edge of the page
xShift	ChartType = "COL" or "CYL" For Multi-level bar-charts (see Multi-Level Bar-Chart), this controls the horizontal shifting of subsequent charts. If set to zero, charts could overlay each other completely.
yShift	ChartType = "COL" or "CYL" For Multi-level bar-charts (see Multi-Level Bar-Chart), this controls the vertical shifting of subsequent charts. If set to zero, charts could overlay each other completely.

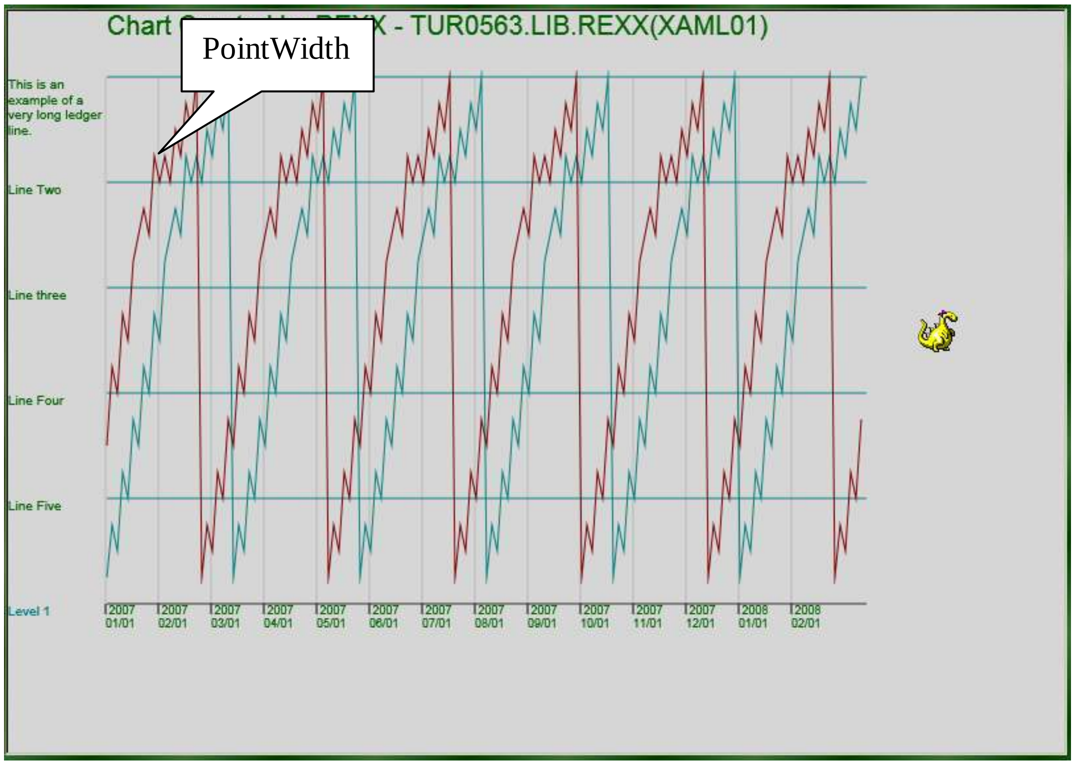
Environment Variable Defaults

```
Animation = "Y"
```



```
AnimationType = 1
BoxHeight = 400
CapHeight = 6
ChartType = 'COLUMN'
Color.0 = 5
Color.1 = 'TEAL'
Color.2 = 'MAROON'
Color.3 = 'SteelBlue'
Color.4 = 'DARKMAGENTA'
Color.5 = 'GRAY'
ColumnFontSize = 9
FooterText.0 = 0
CylinderWidth = 30
FontName = 'Verdana'
FooterDepth = 0
FooterWidth = 0
lci = 1
LedgerColor = 'BLACK'
LedgerFontname = 'ARIAL'
LedgerFontSize = 10
LedgerLineInterval = 10
LedgerText.0 = 0
LeftMargin = 75
LevelText.0 = 0
PointWidth  = 5
OutputDDname = 'XAMLOUT'
RectangleWidth = 20
ShadowWidth = 10
ShrinkBy = 1.00
Spacing = 2
Title = ""
TitleColor    = 'Black'
TitleFontname = 'ARIAL'
TitlefontSize = 20
TopMargin = 50
xShift = 15
yShift = 15
```





DDname(PERCENTS)

This contains the size (and number) of your bars in percentages. Percentages are separated by spaces, and can be all together on one record or scattered amongst many records as in one percentage per record.

The number of these percentages determines the number of bars in the bar chart. Example:

```
33 24.5 88 77
5 10 99
66
37
```

This example would produce a 9 column bar chart. Notice how that not all figures have to be on the same record. You could have even placed each figure on a separate record.

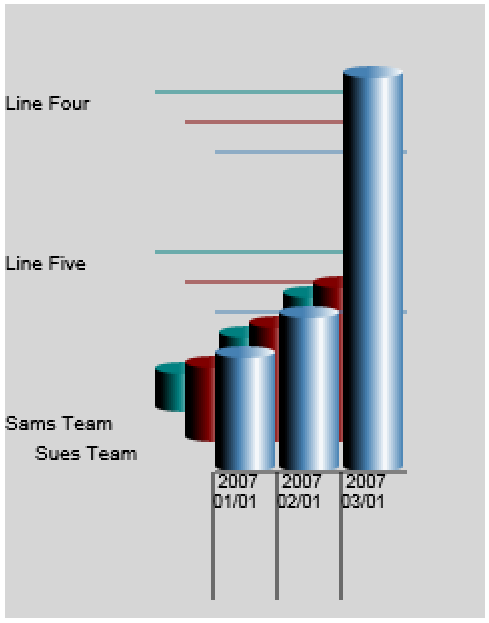
For multi-level (ChartType="COL" or "CYL") or multi-line (ChartType="LIN") charts, an intervening slash ("/") indicates the beginning of a new level/line. Please see the following example:

```
33 24.5 88 77 5 10 99 66 37
/
24.5 88 77 5 10 99 66 37 33
/
88 77 5 10 99 66 37 33 24.5
```

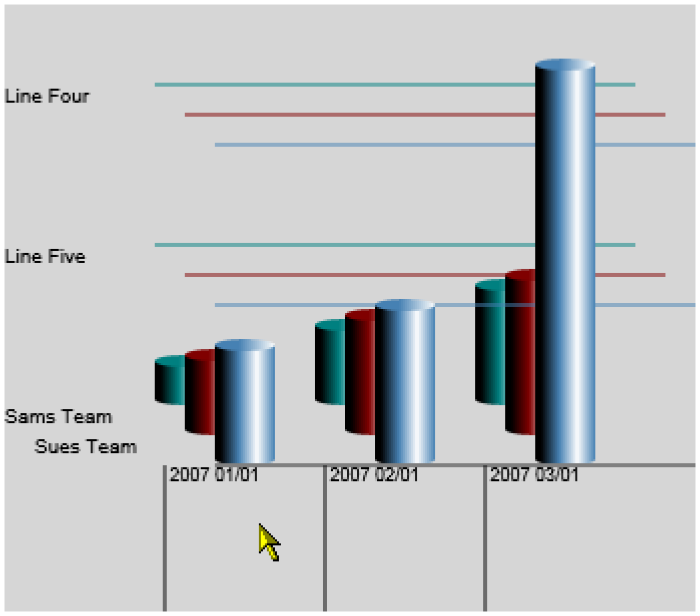
This example would have produced a three level/line chart. When the slash is used, it is presumed that there are an equal number of percentages in each list, else unpredictable results will occur.

xShift, yShift, Spacing

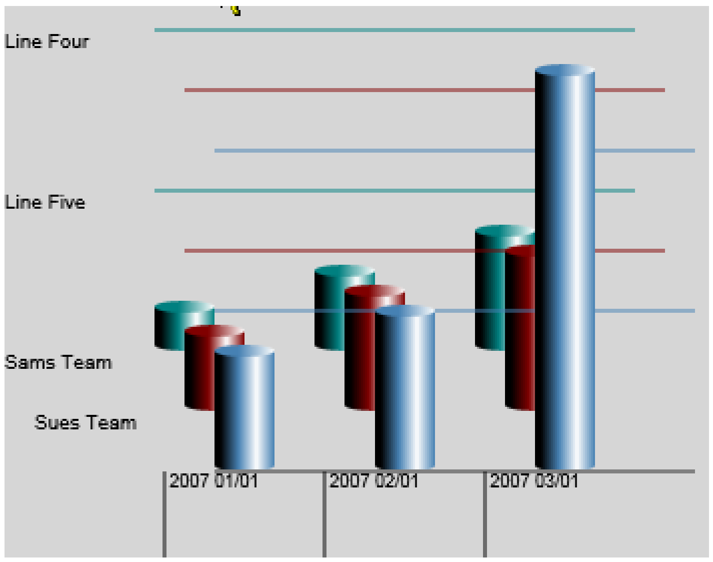
You use the xShift and yShift variable to position subsequent bar charts of a multi-level chart. The following example displays a multi-level bar chart using the supplied defaults.



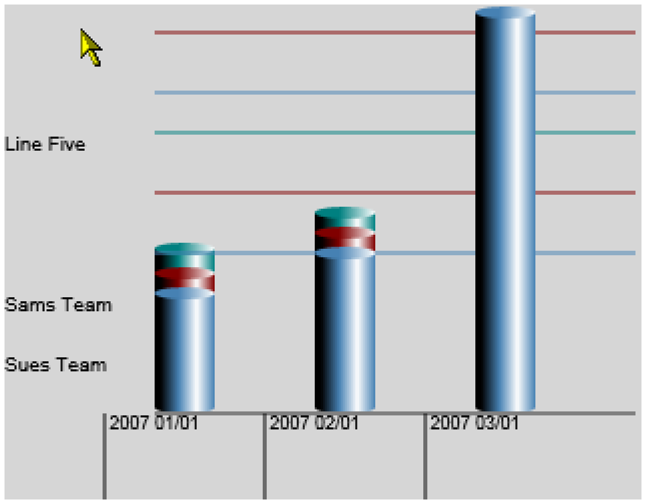
As you can see, things look pretty crowded. By increasing the Spacing, we can spread things out a bit. The next example shows the use of just specifying "*Spacing = 50*" as one of the variable overrides.



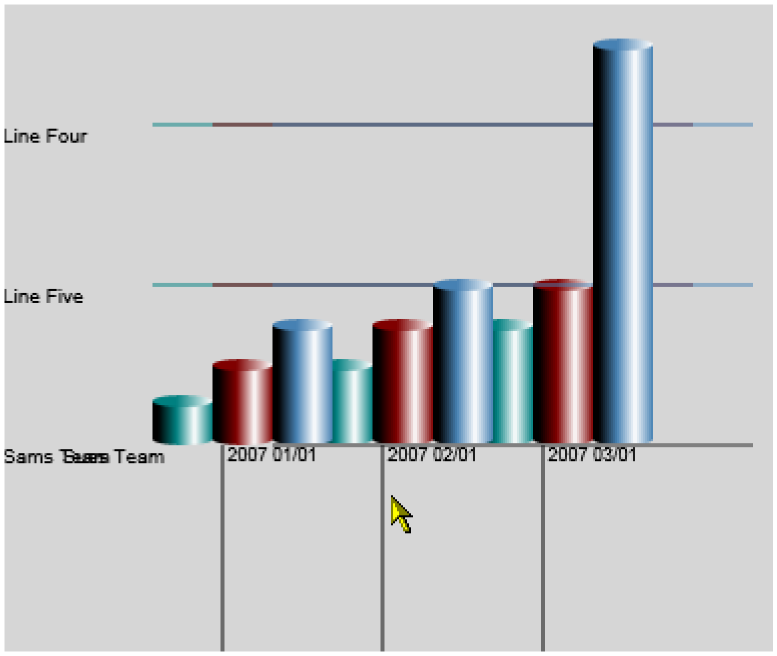
The Spacing variable applies spacing vertically between individual bars (or cylinders). To alter the angle of the multi-levels we use the xShift and yShift variables. To demonstrate the use of the yShift, we will just change it from its default to 30 (*yShift=30*), and let's see what happens.



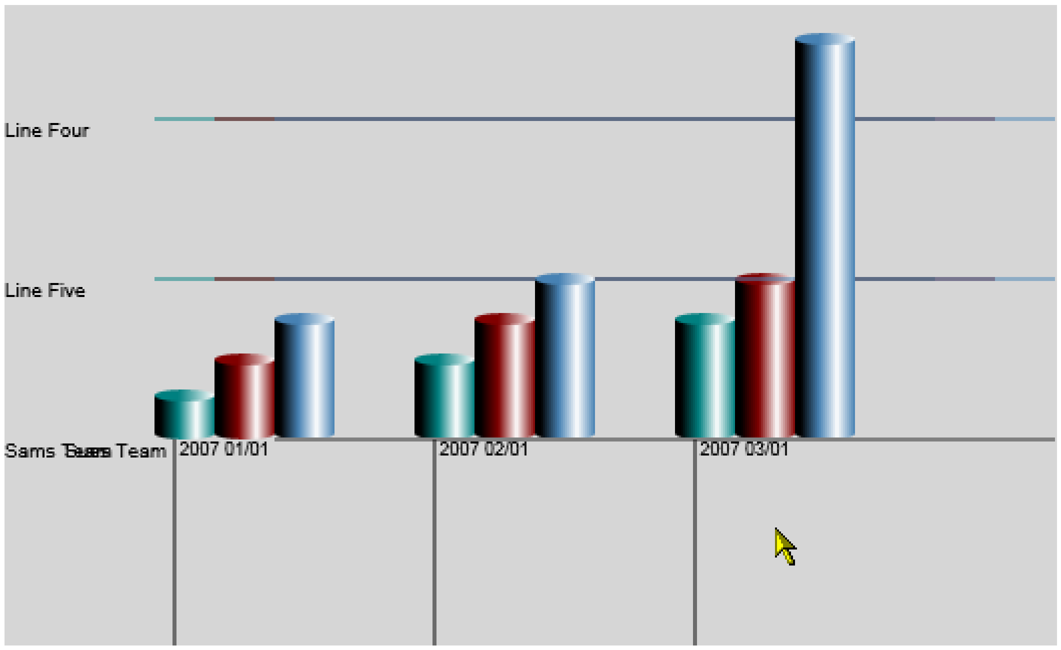
As you might be able to see, the difference is not that much, but you should be able to see that the levels are now spaced a little farther apart. Now let's see what setting the xShift might do. Let's set it to 0 (xShift=0).



Wow, that may not be something we ever want to do. xShift controls the vertical spacing of each subsequent level. Setting xShift to zero forces the level to print on top of them selves. Let's try something else. Let's set yShift to 0 (*yShift=0*) and xShift to CylinderWidth (*xShift=CylinderWidth*).



It appears we could use a little more spacing. Let's make spacing = 100.



That's a little better. Notice, however, our level headers to the left have started overlaying each other. When using this type of configuration, it may be best to just have the one level heading (LevelTest.1 = ????)

Sample HTML Setup

HTML using SilverLight.js

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
```

```
  <title>A Sample HTML page</title>
  <script type="text/javascript" src="../Scripts/Silverlight.js"></script>
  <script type="text/javascript" src="myscript.js"></script>
</head>
<body>
<!-- Where the Silverlight plugin will go-->
<div id="myControl"></div>
<script type="text/javascript">
var parentElement = document.getElementById("myControl");
Silverlight.createObject(
     "../xaml/Xaml3.txt",
     parentElement,
     "myControlID",
     {
            width:'100%',
            height:'90%',
            inplaceInstallPrompt:true,
            background:'#D6D6D6',
            isWindowless:'false',
            framerate:'24',
            version:'1.0'
     },
     {
            onError:null,
            onLoad:null
     },
     null );
</script>
</body>
</html>
```

HTML only (no use of SilverLight.js)

```
<html>
<head>
 <title>A Sample HTML page</title>
 <script type="text/javascript">
        function onSilverlightError(sender, args) {
            var appSource = "";
            if (sender != null && sender != 0) {
                appSource = sender.getHost().Source;
            }
            var errorType = args.ErrorType;
            var iErrorCode = args.ErrorCode;
            var errMsg = "Unhandled Error in Silverlight 2 Application " +
                appSource + "\n";
            errMsg += "Code: " + iErrorCode + "    \n";
            errMsg += "Category: " + errorType + "       \n";
            errMsg += "Message: " + args.ErrorMessage + "     \n";
            if (errorType == "ParserError") {
                errMsg += "File: " + args.xamlFile + "     \n";
                errMsg += "Line: " + args.lineNumber + "     \n";
                errMsg += "Position: " + args.charPosition + "     \n";
            }
            else if (errorType == "RuntimeError") {
                if (args.lineNumber != 0) {
                    errMsg += "Line: " + args.lineNumber + "     \n";
                    errMsg += "Position: " + args.charPosition + "     \n";
                }
                errMsg += "MethodName: " + args.methodName + "     \n";
            }
            throw new Error(errMsg);
        }
    </script>
</head>
<body>
 <div id="silverlightControlHost">
```

```
        <object width="100%" height="100%"
            type="application/x-silverlight-2"
            data="data:application/x-silverlight-2," >
            <param name="source" value="Xaml4.xaml"/>
            <param name="onerror" value="onSilverlightError" />
            <param name="background" value="white" />
            <param name="minRuntimeVersion" value="2.0.31005.0" />
            <param name="autoUpgrade" value="true" />
            <a href="http://go.microsoft.com/fwlink/?LinkID=124807"
                style="text-decoration: none;">
                <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
                    alt="Get Microsoft Silverlight"
                    style="border-style: none"/>
            </a>
        </object>
        <iframe style='visibility:hidden;height:0;width:0;border:0px'></iframe>
    </div>
</body>
</html>
```

Sample Batch Execution JCL

```
//CBRXXRUN JOB ,'REXX RUN',CLASS=Q,MSGCLASS=X,MSGLEVEL=(1,1),
//    NOTIFY=TUR0563
//*===============================================================
//*
//*                    Cylindar Example
//*
//*===============================================================
//STEP01  EXEC PGM=IRXJCL,PARM='XAML01'
//SYSTSIN  DD DUMMY                         /* TERMINAL INPUT
//SYSTSPRT DD SYSOUT=*                      /* TERMINAL OUTPUT
//SYSUDUMP DD SYSOUT=*                      /* TERMINAL OUTPUT
//SYSEXEC DD DISP=SHR,DSN=SA.SYSEXEC
//XAMLIN  DD *
<Canvas
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
<PLACE BARCHART HERE>
</Canvas>
//XAMLOUT DD DISP=(NEW,PASS),
//          DSN=&XAML,
//          UNIT=DASD,
//          STORCLAS=SCUSER,
//          DCB=(LRECL=255,RECFM=VB)
//ENVVAR  DD *
  ChartType='CYL'
  FooterText.1 = " 2007 01/01"
  FooterText.2 = " 2007 02/01"
  FooterText.3 = " 2007 03/01"
  FooterText.4 = " 2007 04/01"
  FooterText.0 = 4
  LedgerLineInterval = 5
  LedgerText.0 = LedgerLineInterval
  LedgerText.1 = 'This is an example of a very long ledger line.'
  LedgerText.2 = 'Line Two'
  LedgerText.3 = 'Line three'
  LedgerText.4 = 'Line Four'
  LedgerText.5 = 'Line Five'
  Title = "Chart Created by REXX - SA.SYSEXEC(XAML01)"
  LevelText.1 = "Sams Team"
  LevelText.2 = "Sues Team"
  LevelText.0 = 2
  Spacing = 100
  yShift = 0
  xShift = CylinderWidth
//PERCENTS DD *
5.5 10 15 0
/
10 15 20 0
/
15 20 50 0
//*=================================================================
//STEP02  EXEC BATCHTMP
//SYSIN  DD *
 OCOPY INDD(I1) OUTDD(O1)
/*
//I1     DD  DISP=(OLD,DELETE),DSN=&XAML
//O1     DD PATHOPTS=(OWRONLY,OCREAT),PATHMODE=(SIRWXU),
// PATH='/usr/lpp/internet/server_root/saapps/pub/xaml/Xaml3.txt'
```

Silverlight.js Script

```
if (!window.Silverlight)
{
    window.Silverlight = { };
}

// Silverlight control instance counter for memory mgt
Silverlight._silverlightCount = 0;
Silverlight.fwlinkRoot='http://go2.microsoft.com/fwlink/?LinkID=';
Silverlight.onGetSilverlight = null;
Silverlight.onSilverlightInstalled = function () {window.location.reload(false);};

//////////////////////////////////////////////////////////////////
// isInstalled, checks to see if the correct version is installed
//////////////////////////////////////////////////////////////////
Silverlight.isInstalled = function(version)
{
    var isVersionSupported=false;
    var container = null;

    try
    {
        var control = null;

        try
        {
            control = new ActiveXObject('AgControl.AgControl');
            if ( version == null )
            {
                isVersionSupported = true;
            }
            else if ( control.IsVersionSupported(version) )
            {
                isVersionSupported = true;
            }
            control = null;
        }
        catch (e)
        {
            var plugin = navigator.plugins["Silverlight Plug-In"] ;
            if ( plugin )
            {
                if ( version === null )
                {
                    isVersionSupported = true;
                }
                else
                {
                    var actualVer = plugin.description;
                    if ( actualVer === "1.0.30226.2")
                        actualVer = "2.0.30226.2";
                    var actualVerArray =actualVer.split(".");
                    while ( actualVerArray.length > 3)
                    {
                        actualVerArray.pop();
                    }
                    while ( actualVerArray.length < 4)
                    {
                        actualVerArray.push(0);
                    }
                    var reqVerArray = version.split(".");
                    while ( reqVerArray.length > 4)
                    {
                        reqVerArray.pop();
                    }
                    var requiredVersionPart ;
                    var actualVersionPart
                    var index = 0;
                    do
                    {
                        requiredVersionPart = parseInt(reqVerArray[index]);
                        actualVersionPart = parseInt(actualVerArray[index]);
                        index++;
                    }
                while (index < reqVerArray.length && requiredVersionPart === actualVersionPart);

                    if ( requiredVersionPart <= actualVersionPart && !isNaN(requiredVersionPart)
)
                    {
                        isVersionSupported = true;
                    }
                }
            }
        }
    }
    catch (e)
    {
        isVersionSupported = false;
    }
    if (container)
    {
        document.body.removeChild(container);
    }

    return isVersionSupported;
}
Silverlight.WaitForInstallCompletion = function()
{
    if ( ! Silverlight.isBrowserRestartRequired && Silverlight.onSilverlightInstalled )
    {
        try
        {
            navigator.plugins.refresh();
        }
        catch(e)
        {
        }
        if ( Silverlight.isInstalled(null) )
        {
            Silverlight.onSilverlightInstalled();
        }
        else
        {
              setTimeout(Silverlight.WaitForInstallCompletion, 3000);
        }
    }
}
Silverlight.__startup = function()
{
    Silverlight.isBrowserRestartRequired = Silverlight.isInstalled(null);//(!window.ActiveXObject
|| Silverlight.isInstalled(null));
    if ( !Silverlight.isBrowserRestartRequired)
    {
        Silverlight.WaitForInstallCompletion();
    }
    if (window.removeEventListener) {
       window.removeEventListener('load', Silverlight.__startup , false);
    }
    else {
        window.detachEvent('onload', Silverlight.__startup );
    }
}

if (window.addEventListener)
{
    window.addEventListener('load', Silverlight.__startup , false);
}
else
{
    window.attachEvent('onload', Silverlight.__startup );
}

///////////////////////////////////////////////////////////////////////////////
// createObject();  Params:
// parentElement of type Element, the parent element of the Silverlight Control
// source of type String
// id of type string
// properties of type String, object literal notation { name:value, name:value, name:value},
//     current properties are: width, height, background, framerate, isWindowless,
enableHtmlAccess, inplaceInstallPrompt:  all are of type string
// events of type String, object literal notation { name:value, name:value, name:value},
//     current events are onLoad onError, both are type string
// initParams of type Object or object literal notation { name:value, name:value, name:value}
// userContext of type Object
/////////////////////////////////////////////////////////////////////////////////

Silverlight.createObject = function(source, parentElement, id, properties, events, initParams,
userContext)
{
    var slPluginHelper = new Object();
    var slProperties = properties;
    var slEvents = events;

    slPluginHelper.version = slProperties.version;
    slProperties.source = source;
    slPluginHelper.alt = slProperties.alt;

    //rename properties to their tag property names
    if ( initParams )
        slProperties.initParams = initParams;
    if ( slProperties.isWindowless && !slProperties.windowless)
        slProperties.windowless = slProperties.isWindowless;
    if ( slProperties.framerate && !slProperties.maxFramerate)
        slProperties.maxFramerate = slProperties.framerate;
    if ( id && !slProperties.id)
        slProperties.id = id;

    // remove elements which are not to be added to the instantiation tag
    delete slProperties.ignoreBrowserVer;
    delete slProperties.inplaceInstallPrompt;
    delete slProperties.version;
    delete slProperties.isWindowless;
    delete slProperties.framerate;
    delete slProperties.data;
    delete slProperties.src;
    delete slProperties.alt;


    // detect that the correct version of Silverlight is installed, else display install
    if (Silverlight.isInstalled(slPluginHelper.version))
    {
        //move unknown events to the slProperties array
        for (var name in slEvents)
        {
            if ( slEvents[name])
            {
     if ( name == "onLoad" && typeof slEvents[name] == "function" && slEvents[name].length != 1 )
                {
                    var onLoadHandler = slEvents[name];
                    slEvents[name]=function (sender){ return
onLoadHandler(document.getElementById(id), userContext, sender)};
                }
                var handlerName = Silverlight.__getHandlerName(slEvents[name]);
                if ( handlerName != null )
                {
                    slProperties[name] = handlerName;
                    slEvents[name] = null;
                }
                else
                {
                    throw "typeof events."+name+" must be 'function' or 'string'";
                }
            }
        }
        slPluginHTML = Silverlight.buildHTML(slProperties);
    }
    //The control could not be instantiated. Show the installation prompt
    else
    {
        slPluginHTML = Silverlight.buildPromptHTML(slPluginHelper);
    }
    // insert or return the HTML
    if(parentElement)
    {
        parentElement.innerHTML = slPluginHTML;
    }
    else
    {
        return slPluginHTML;
    }

}
///////////////////////////////////////////////////////////////////////////////
//
//  create HTML that instantiates the control
//
///////////////////////////////////////////////////////////////////////////////
Silverlight.buildHTML = function( slProperties)
{
    var htmlBuilder = [];

    htmlBuilder.push('<object type=\"application/x-silverlight\" data="data:application/x-
silverlight,"');
    if ( slProperties.id != null )
    {
        htmlBuilder.push(' id="' + slProperties.id + '"');
    }
    if ( slProperties.width != null )
    {
        htmlBuilder.push(' width="' + slProperties.width+ '"');
    }
    if ( slProperties.height != null )
    {
        htmlBuilder.push(' height="' + slProperties.height + '"');
    }
    htmlBuilder.push(' >');

    delete slProperties.id;
    delete slProperties.width;
    delete slProperties.height;

    for (var name in slProperties)
    {
        if (slProperties[name])
        {
            htmlBuilder.push('<param name="'+Silverlight.HtmlAttributeEncode(name)+'"
value="'+Silverlight.HtmlAttributeEncode(slProperties[name])+'" />');
        }
    }
    htmlBuilder.push('<\/object>');
    return htmlBuilder.join('');
}

// createObjectEx, takes a single parameter of all createObject parameters enclosed in {}
Silverlight.createObjectEx = function(params)
{
    var parameters = params;
    var html = Silverlight.createObject(parameters.source, parameters.parentElement,
parameters.id, parameters.properties, parameters.events, parameters.initParams,
parameters.context);
    if (parameters.parentElement == null)
    {
        return html;
    }
}

///////////////////////////////////////////////////////////////////////////////////////////////
// Builds the HTML to prompt the user to download and install Silverlight
///////////////////////////////////////////////////////////////////////////////////////////////
Silverlight.buildPromptHTML = function(slPluginHelper)
{
    var slPluginHTML = "";
    var urlRoot = Silverlight.fwlinkRoot;
    var shortVer = slPluginHelper.version ;
    if ( slPluginHelper.alt )
    {
        slPluginHTML = slPluginHelper.alt;
    }
    else
    {
        if (! shortVer )
        {
            shortVer="";
        }
        slPluginHTML = "<a href='javascript:Silverlight.getSilverlight(\"{1}\");' style='text-
decoration: none;'><img src='{2}' alt='Get Microsoft Silverlight' style='border-style:
none'/></a>";
        slPluginHTML = slPluginHTML.replace('{1}', shortVer );
        slPluginHTML = slPluginHTML.replace('{2}', urlRoot + '108181');
    }

    return slPluginHTML;
}


Silverlight.getSilverlight = function(version)
{
    if (Silverlight.onGetSilverlight )
    {
        Silverlight.onGetSilverlight();
    }

    var shortVer = "";
    var reqVerArray = String(version).split(".");
    if (reqVerArray.length > 1)
    {
        var majorNum = parseInt(reqVerArray[0] );
        if ( isNaN(majorNum) || majorNum < 2 )
        {
            shortVer = "1.0";
        }
        else
        {
            shortVer = reqVerArray[0]+'.'+reqVerArray[1];
        }
    }

    var verArg = "";

    if (shortVer.match(/^\d+\056\d+$/) )
    {
        verArg = "&v="+shortVer;
    }

    Silverlight.followFWLink("114576" + verArg);
}


///////////////////////////////////////////////////////////////////////////////////////////////
/// Navigates to a url based on fwlinkid
///////////////////////////////////////////////////////////////////////////////////////////////
Silverlight.followFWLink = function(linkid)
{
    top.location=Silverlight.fwlinkRoot+String(linkid);
}

///////////////////////////////////////////////////////////////////////////////////////////////
/// Encodes special characters in input strings as charcodes
///////////////////////////////////////////////////////////////////////////////////////////////
Silverlight.HtmlAttributeEncode = function( strInput )
{
      var c;
      var retVal = '';

    if(strInput == null)
      {
          return null;
    }

      for(var cnt = 0; cnt < strInput.length; cnt++)
      {
            c = strInput.charCodeAt(cnt);

            if (( ( c > 96 ) && ( c < 123 ) ) ||
                  ( ( c > 64 ) && ( c < 91 ) ) ||
                  ( ( c > 43 ) && ( c < 58 ) && (c!=47)) ||
                  ( c == 95 ))
            {
                  retVal = retVal + String.fromCharCode(c);
            }
            else
            {
                  retVal = retVal + '&#' + c + ';';
            }
      }

      return retVal;
}
///////////////////////////////////////////////////////////////////////////////
//
//  Default error handling function to be used when a custom error handler is
//  not present
//
///////////////////////////////////////////////////////////////////////////////

Silverlight.default_error_handler = function (sender, args)
{
    var iErrorCode;
    var errorType = args.ErrorType;

    iErrorCode = args.ErrorCode;

    var errMsg = "\nSilverlight error message     \n" ;

    errMsg += "ErrorCode: "+ iErrorCode + "\n";


    errMsg += "ErrorType: " + errorType + "       \n";
    errMsg += "Message: " + args.ErrorMessage + "     \n";

    if (errorType == "ParserError")
    {
        errMsg += "XamlFile: " + args.xamlFile + "     \n";
        errMsg += "Line: " + args.lineNumber + "     \n";
        errMsg += "Position: " + args.charPosition + "     \n";
    }
    else if (errorType == "RuntimeError")
    {
        if (args.lineNumber != 0)
        {
            errMsg += "Line: " + args.lineNumber + "     \n";
            errMsg += "Position: " +  args.charPosition + "     \n";
        }
        errMsg += "MethodName: " + args.methodName + "     \n";
    }
    alert (errMsg);
}

//////////////////////////////////////////////////////////////////////////////////////////////
/// Releases event handler resources when the page is unloaded
//////////////////////////////////////////////////////////////////////////////////////////////
Silverlight.__cleanup = function ()
{
    for (var i = Silverlight._silverlightCount - 1; i >= 0; i--) {
        window['__slEvent' + i] = null;
    }
    Silverlight._silverlightCount = 0;
    if (window.removeEventListener) {
       window.removeEventListener('unload', Silverlight.__cleanup , false);
    }
    else {
        window.detachEvent('onunload', Silverlight.__cleanup );
    }
}
//////////////////////////////////////////////////////////////////////////////////////////////
/// Releases event handler resources when the page is unloaded
//////////////////////////////////////////////////////////////////////////////////////////////
Silverlight.__getHandlerName = function (handler)
{
    var handlerName = "";
    if ( typeof handler == "string")
    {
        handlerName = handler;
    }
    else if ( typeof handler == "function" )
    {
        if (Silverlight._silverlightCount == 0)
        {
            if (window.addEventListener)
            {
                window.addEventListener('onunload', Silverlight.__cleanup , false);
            }
            else
            {
                window.attachEvent('onunload', Silverlight.__cleanup );
            }
        }
        var count = Silverlight._silverlightCount++;
        handlerName = "__slEvent"+count;

        window[handlerName]=handler;
    }
    else
    {
        handlerName = null;
    }
    return handlerName;
}
```