ADISRA

SmartView

C# Script Reference

Scope of this document

This document will help the SmartView users to understand the C# Scripts that are present in many different places in the application. It will also explain the build process performed by the SmartView Engineering Tool and the outputs of this process. This will be the ground preparation for talking about the scripts.
It is not the intent of this document to teach nothing more than necessary to allow the user to write scripts and understand what can be done in different scripts.
Some images and diagrams were added to make the learning more didactic, the suggestion is to read it from top to bottom the first time. It can also be used as a future reference using the table of contents above.

About C#

C# is a programming language developed by Microsoft that runs on the .NET Framework. It is widely-used and known by developers,.it is a high-level language, it abstracts away most of the complex tasks and since it is a popular language, it will keep evolving and bringing more functionalities and improvements.

Why C# and not any other language?

The SmartView was created to be a flexible and easy-to-use tool. With this prerogative, the SmartView was developed using C# since it is a language with such a good performance (vital for a SCADA software) and so many great features. After taking that decision, it was easy to choose C# as the scripting language. The users would be able to implement powerful code using a language that is greatly documented and supported by so many other developers.

How can I learn C#?

There are many ways to learn it, depending on the type of student you are. You can read documentations, take online or presential classes, follow tutorials or get your hands dirty prior to any studies. But actually for writing scripts to your application you won't need too much knowledge, so relax because no matter your learning method, you will be able to create very good scripts with minimum effort.
Feel free to check some of the references below in case you want more information and material to learn:

Reference Description	URL
Microsoft C# HomePage with Tutorials and	https://docs.microsoft.com/en-us/dotnet/csharp/

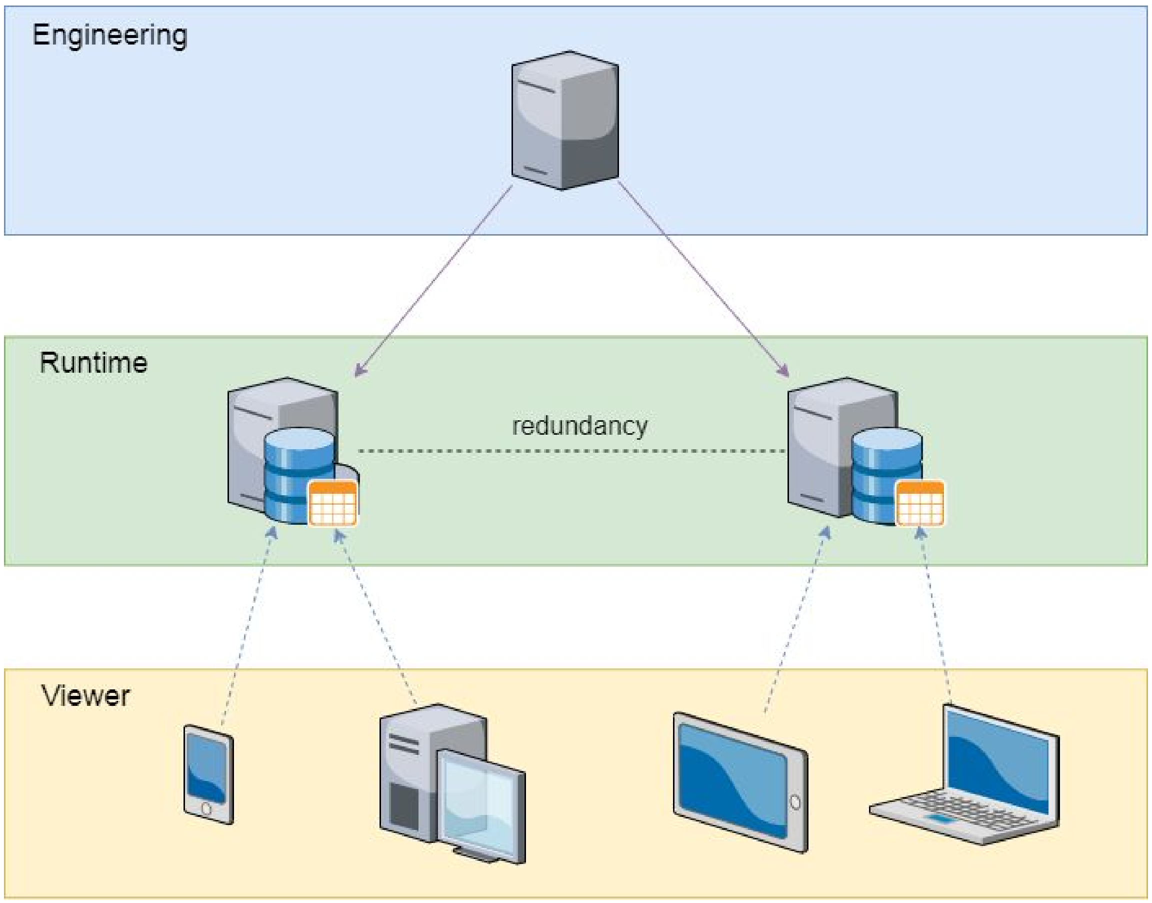
Documentation	
Microsoft C# Programming Guide	https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
Great Documentation and Tutorial page with working examples	https://www.w3schools.com/cs/
Another great page which includes a Tutorial, Tips and detailed explanation	https://csharp.net-tutorials.com/
Company leader in online education	https://www.codecademy.com/learn/learn-c-sharp

SmartView Build

The SmartView can be divided into 3 main components.

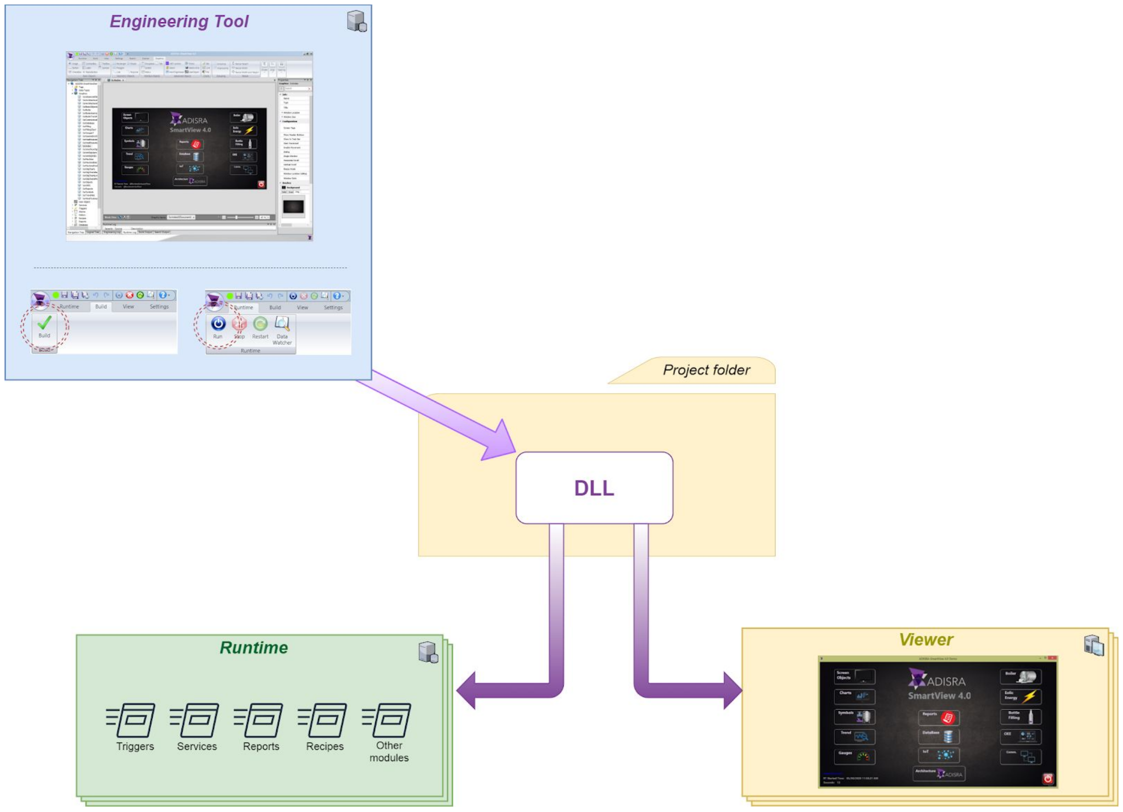
- Engineering Tool
- Runtime
- Viewer

It is not the purpose of this document to give a detailed explanation of those components, but it is important to mention them since it will help you create a general understanding and later assimilate more information.



The SmartView Engineering Tool allows you to create the entire application. It can be composed by Graphics (Screens), Services, Triggers, Communication Drivers, OPC Clients, etc. At any point this application can be built or run. Both of these actions require an entire build of the application since the output of this build will be used by the Runtime and the Viewer.
But what actually happens when you build an application?

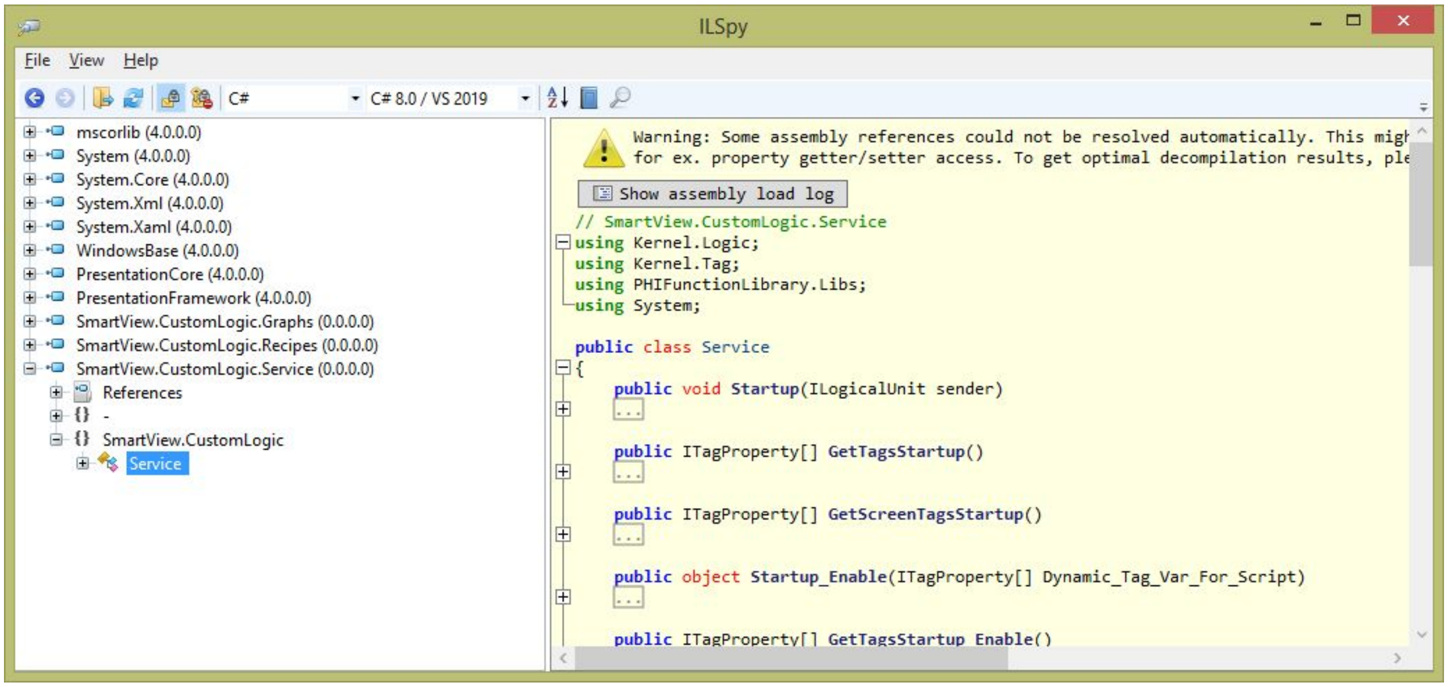
The output of the build are several DLL files which will be generated in the project folder. The dlls must be valid dlls or else the build will fail and an error will appear in the engineering logs. If there are errors in a button for example, you will still be able to run the application and start the RunTime and the Viewer, but that button will not work. The engineering tool has different ways to help you validate your script, it contains validate buttons with texts showing if the script is ok or not, it colors the borders of the script in case errors appear, it generates log messages in the engineering logs window, it tells you in case a tag was not created and allows you to create it.
As soon as the DLLs are generated, they will be used by the Runtime and the Viewer. Those auto generated DLLs are basically C# classes that represent maybe a Graphics screen, a trigger or maybe a service. If you are a developer you will understand that those classes have different namespaces and in order to import different classes you need to add using statements to them. But let's avoid technical detailing, what you need to know is that each of those classes generated inside those DLLs will have a pre-defined scope, they will be able to reach some other classes but won't reach others.
Stay together to take a look at a generated class in the following sections, it will make the current explanation more clear. Now let's take a look at the diagram that represents the 3 components and the DLLs.



** Engineering tool creates the DLLs that will be used by the Runtime and the Viewer.*

Auto generated DLLs

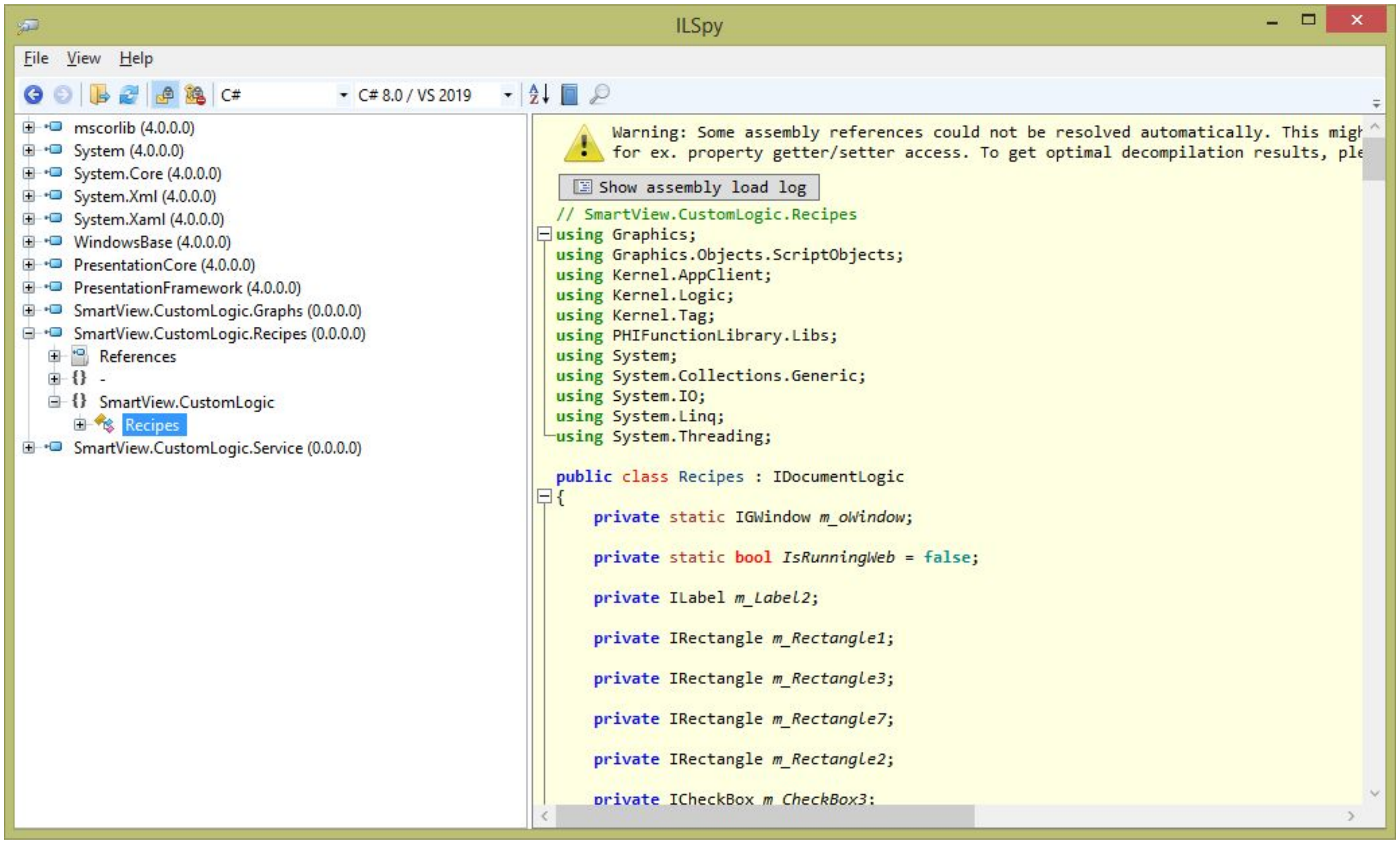
The DLLs will be generated inside the projects folder. In case you want to check a Services' DLL, go to the Services folder and you will find the dll there. If you want to take a look at them, there are some tools that allow you to check the content of DLLs. I used ILSpy for the sake of this documentation, I hope it gives you a better understanding.
The first image below shows you a Service class. Pay attention to the using statements, more specifically the Kernel.Tag which indicates that Tags can be used in your Services. It also adds System, which is a .net framework namespace that contains many commonly-used types and any other sub-namespaces. This is an important information, because it tells you that you can use any other c# code that you would use in case you were developing on Visual Studio a new software.



```
Warning: Some assembly references could not be resolved automatically. This migh
for ex. property getter/setter access. To get optimal decompilation results, ple

// SmartView.CustomLogic.Service
using Kernel.Logic;
using Kernel.Tag;
using PHIFunctionLibrary.Libs;
using System;

public class Service
{
    public void Startup(ILogicalUnit sender)
    ...

    public ITagProperty[] GetTagsStartup()
    ...

    public ITagProperty[] GetScreenTagsStartup()
    ...

    public object Startup_Enable(ITagProperty[] Dynamic_Tag_Var_For_Script)
    ...

    public ITagProperty[] GetTagsStartup_Enable()
```

** Close view of SmartView.CustomLogic.Service.dll using ILSpy*

On the next image you will see a Graphic screen. There will be one DLL for each screen you created. And there is no relationship between one and the other. (you won't see any using statements for other screen classes). The name of the screen is the name of the class generated (i.e. below is Recipes screen).
Now there are more using statements like the System Functions Library or System.IO and again the Kernel.Tag.
All the objects in the screen are private properties of this class, which explains why a Screen Script can change an object property or maybe a click in Rectangle1 can change the width of Rectangle2. They are all in the same class and they know each other.



```
mscorlib (4.0.0.0)
System (4.0.0.0)
System.Core (4.0.0.0)
System.Xml (4.0.0.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
SmartView.CustomLogic.Graphs (0.0.0.0)
SmartView.CustomLogic.Recipes (0.0.0.0)
    References
    {} -
    {} SmartView.CustomLogic
        Recipes
SmartView.CustomLogic.Service (0.0.0.0)
```

```
Warning: Some assembly references could not be resolved automatically. This migh
for ex. property getter/setter access. To get optimal decompilation results, ple
```

```
// SmartView.CustomLogic.Recipes
using Graphics;
using Graphics.Objects.ScriptObjects;
using Kernel.AppClient;
using Kernel.Logic;
using Kernel.Tag;
using PHIFunctionLibrary.Libs;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;

public class Recipes : IDocumentLogic
{
    private static IGWindow m_oWindow;

    private static bool IsRunningWeb = false;

    private ILabel m_Label2;

    private IRectangle m_Rectangle1;

    private IRectangle m_Rectangle3;

    private IRectangle m_Rectangle7;

    private IRectangle m_Rectangle2;

    private ICheckBox m_CheckBox3;
```

** Close view of SmartView.CustomLogic.Service.dll using ILSpy*

Libraries

All the DLLs have access to mscorlib.dll which stands for Microsoft's Common Object Runtime Library. They are at the heart of everything.
When Microsoft first started working on the .NET Framework, MSCorLib.dll was an acronym for Microsoft Common Object Runtime Library. Once ECMA started to standardize the CLR and parts of the FCL, MSCorLib.dll officially became the acronym for Multilanguage Standard Common Object Runtime Library

Please find below some of the namespaces that are part of the mscorlib.dll

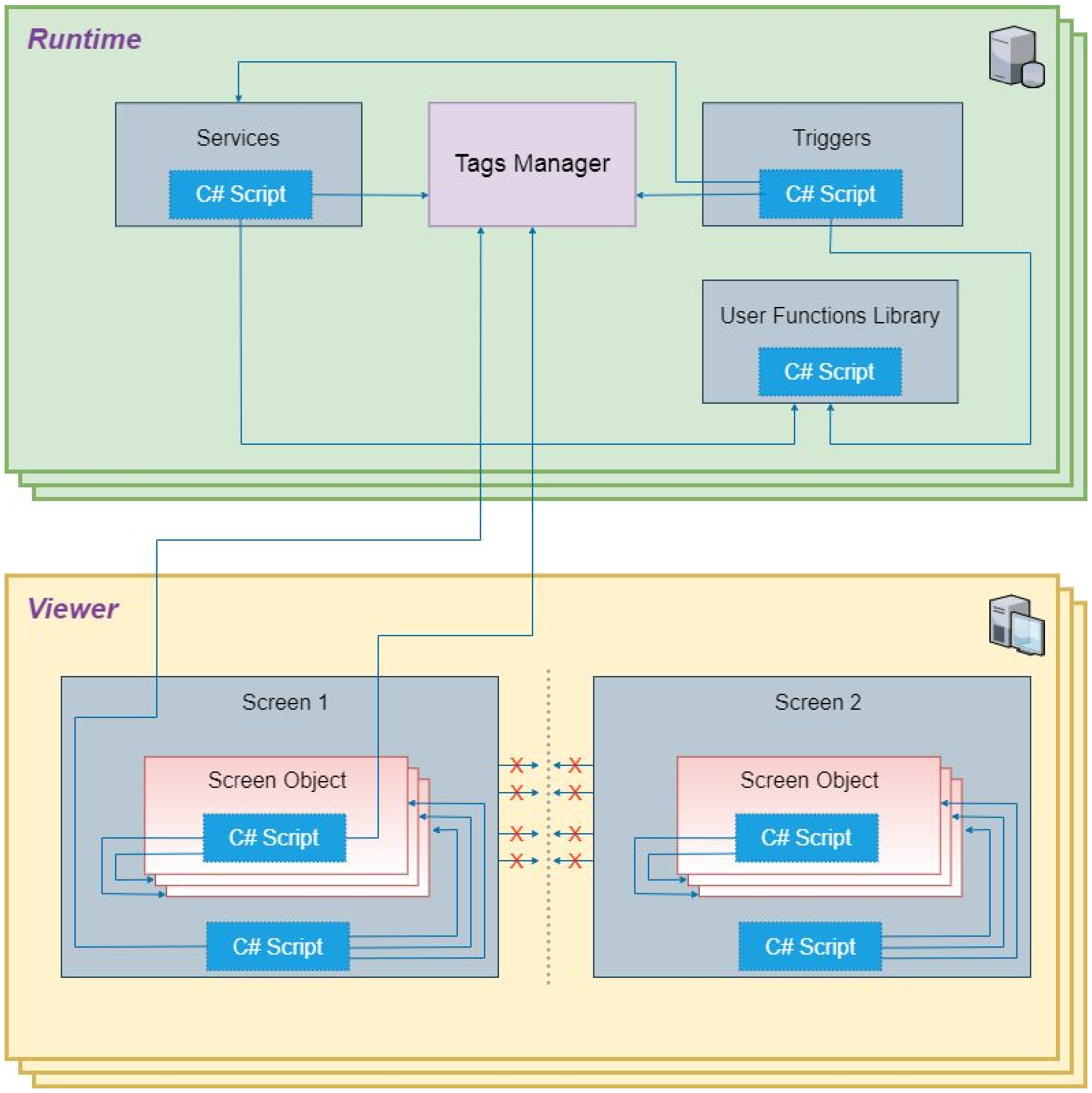
```
System
System.Collections
System.Configuration.Assemblies
System.Diagnostics
System.Diagnostics.SymbolStore
System.Globalization
System.IO System.IO.IsolatedStorage
System.Reflection
System.Reflection.Emit
System.Resources
System.Runtime.CompilerServices
System.Runtime.InteropServices
System.Runtime.InteropServices.Expando
System.Runtime.Remoting
System.Runtime.Remoting.Activation
System.Runtime.Remoting.Channels
System.Runtime.Remoting.Contexts
System.Runtime.Remoting.Lifetime
System.Runtime.Remoting.Messaging
System.Runtime.Remoting.Metadata
System.Runtime.Remoting.Metadata.W3cXsd2001
System.Runtime.Remoting.Proxies
System.Runtime.Remoting.Services
System.Runtime.Serialization
System.Runtime.Serialization.Formatters
System.Runtime.Serialization.Formatters.Binary
System.Security
System.Security.Cryptography
System.Security.Cryptography.X509Certificates
System.Security.Permissions
System.Security.Policy
System.Security.Principal
System.Text
System.Threading
Microsoft.Win32
```

SmartView Scripts

Overview

Let’s get to know more about the SmartView Scripts now. They are not only C# code but they are something else, which makes them very powerful to connect the entire application.
The SmartView is a SCADA software, as you may know (Supervisory Control and Data Acquisition), and it is vital that the SmartView will allow their users to manipulate all the data that is being collected or sent to different locations whether its a Database, a PLC (Programmable Logic Controller) or any other device.
All that data needs to be stored in variables on the SmartView and those variables are called Tags which are available to be used on Scripts.
SmartView was not only designed to connect devices, but also to create amazing interfaces to display all that data, so it has a large collection of Screen Objects from the most basic like Labels, TextBoxes, ComboBoxes, CheckBoxes to the most complex such as MatrixGrids, MultiTagViewers, Charts, Trends, Alarms, etc… And the Scripts also have access to those objects. It means that in one single script you can add rotation to a Geometric Object using any Tags value, or maybe adding new items to a ComboBox or adding new Pens to a Trend object. The possibilities are endless.
It is important to understand the architecture of the SmartView and the scope of different modules before designing your application and writing the scripts that may be necessary. The next section will show you the scripts that can be found in different modules and their reach, all represented in a nice diagram.

Scope and Relationship



The above diagram shows the scope of the scripts and their relationship to other modules. All of the Graphics Script (Screen, Objects and Templates) are executed on the Viewer. The Services, Triggers and User Functions are executed on the Runtime. All of them have access to the Tag Manager, which means that if a tag changes values, they will all be notified and will share the same tag value. The diagram also shows there is no relationship between 2 different screens. So a script from Screen 1 cannot be used to change a screen object's property on Screen 2.

There are some particular cases where the script can be executed on the runtime or on the viewer, it depends on the location of the Script. For example the User Functions Library, if they are called from a Graphics Script, they will be executed in the Viewer but if they are called from a Service, they will be executed in the Runtime. It is also worth mentioning the System Functions Library. Some of them are always executed in the Runtime (i.e. SVTags and SVRecipe). As you can see, the Tags Manager is located in the Runtime, so whenever a Script needs the value of a Tag or needs to update a Tag, there will be a communication between Viewer and Runtime.

Places to be used

There are many different places a script can be written in the SmartView. Please check the list below and pay attention to the comments

Location	Execution	Allow Tags	Can change screen objects	Comments
Screen Object Scripts	Viewer	Yes	Yes	The screen object scripts are executed in the Viewer. They can change other screen object's properties on their scripts too, but they need to be in the same screen. If they call a System/User Function for example, it will be executed in the Viewer, unless it calls a System Function that changes a Tag (i.e. SVTags) or maybe a Recipe (i.e. SVRecipe).
Screen Scripts	Viewer	Yes	Yes	Can only change screen objects located in the same screen.
Services	Runtime	Yes	No	The Services are always executed in the Runtime
Triggers	Runtime	Yes	No	The triggers are always executed in the Runtime. They can also execute Services.
User Functions Library	Viewer/Runtime	No	No	Cannot change tags values or use them directly. But you can pass them as parameters and use its return values to change a tag. It can be executed on the Viewer or on the Runtime, it depends which module is calling it. If a graphics is calling it (screen script or object script) it will execute in the Viewer. If a Service calls it, it will be executed in the Runtime.
Template Object	Viewer	Yes	Yes	Any object added to the user object can change other object's public properties inside the same

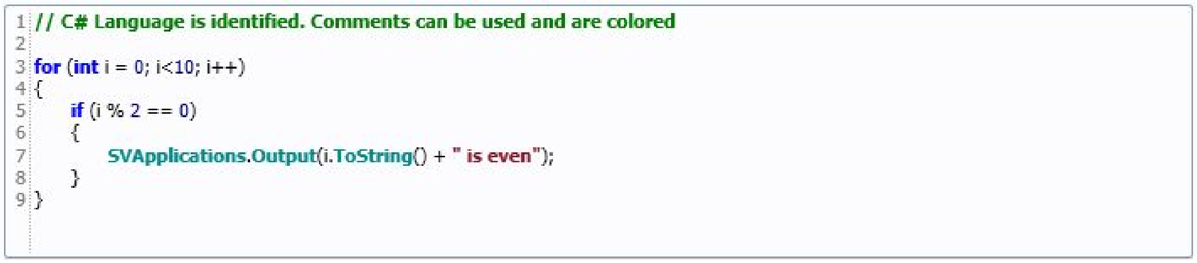
				template. The objects part of a Template cannot change objects outside this Template directly, but they can change tags that are configured to external objects. They can also have internal tags that can be used in the scripts.

**For more information please check the following sections*

In the following sections you will be introduced to the features mentioned above and you will learn the scope of the Scripts while understanding the architecture.

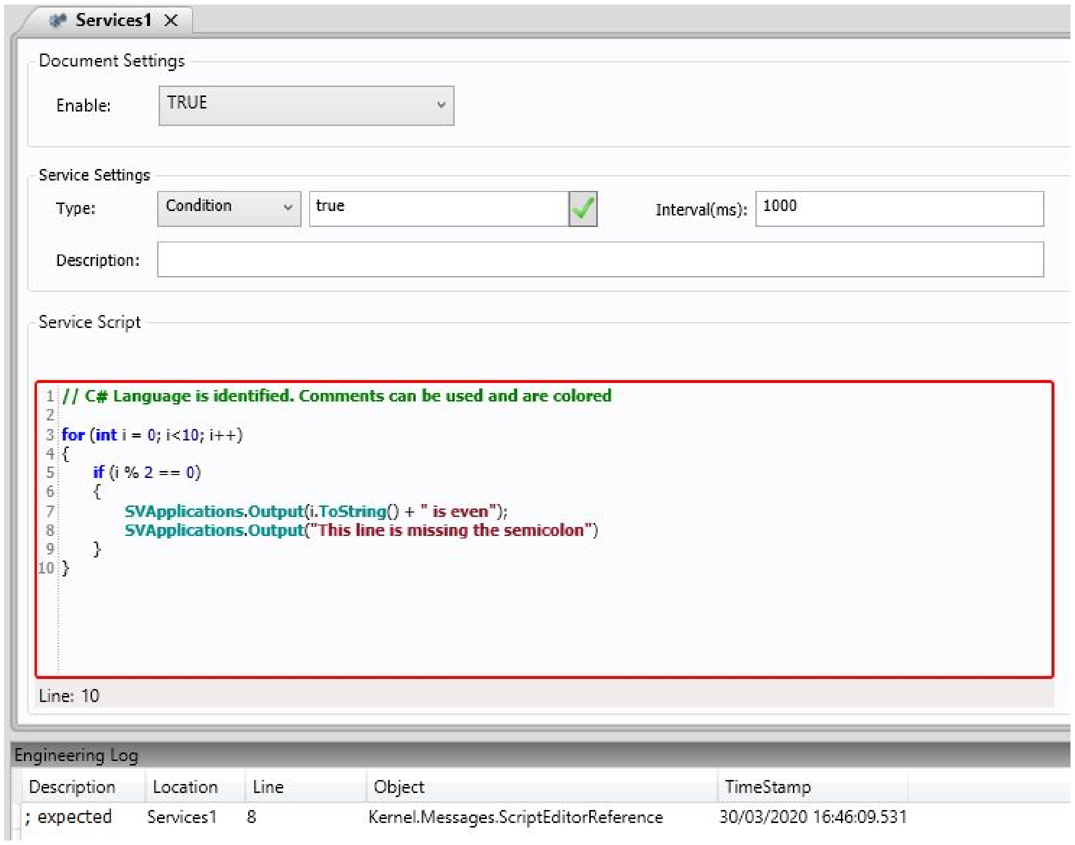
Scripts and C#

The script identifies the C# language and it makes it clear while changing the keywords or reserved words font color.



```
// C# Language is identified. Comments can be used and are colored

for (int i = 0; i<10; i++)
{
    if (i % 2 == 0)
    {
        SVApplications.Output(i.ToString() + " is even");
    }
}
```

It also compiles the code written everytime it is saved (i.e. Services)



Services1

Document Settings

Enable: TRUE

Service Settings

Type: Condition true Interval(ms): 1000

Description:

Service Script

```
// C# Language is identified. Comments can be used and are colored

for (int i = 0; i<10; i++)
{
    if (i % 2 == 0)
    {
        SVApplications.Output(i.ToString() + " is even");
        SVApplications.Output("This line is missing the semicolon")
    }
}
```

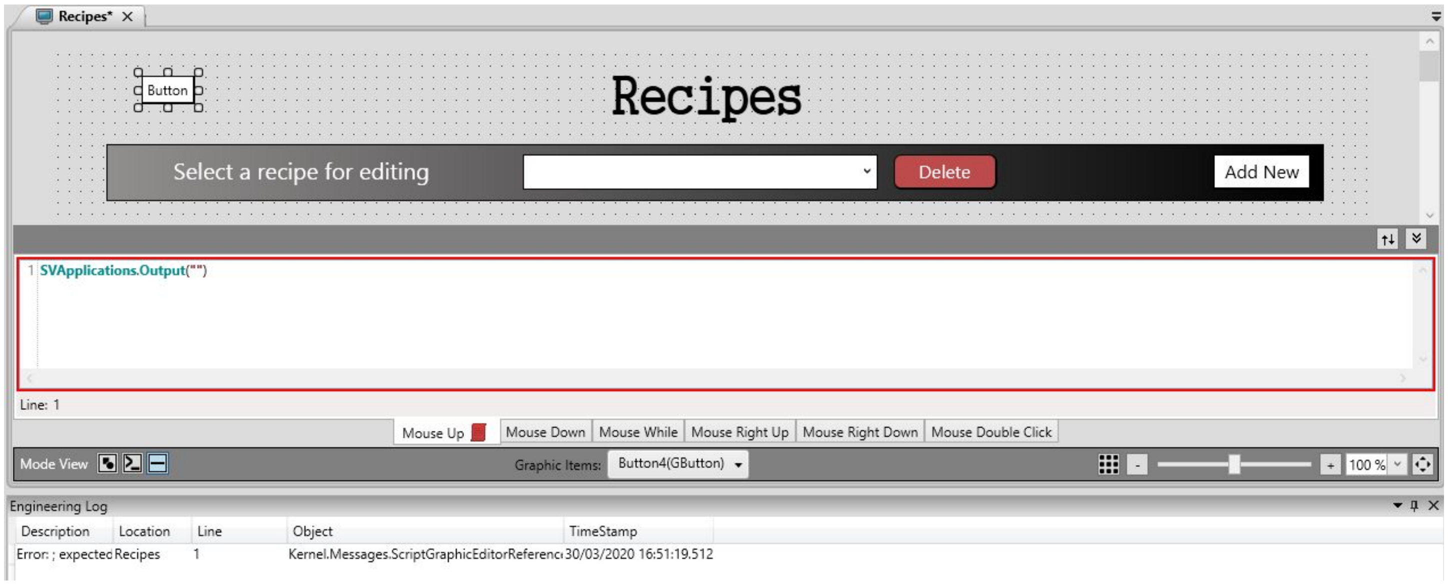
Line: 10

Engineering Log

Description	Location	Line	Object	TimeStamp
; expected	Services1	8	Kernel.Messages.ScriptEditorReference	30/03/2020 16:46:09.531

The red rectangle shows there is an error on the script. At the bottom of the image you can see the error in the engineering log.

In the next image it is an example in a Screen Object Script. The same error was added to a button. This time the error is raised as soon as you remove the focus from the script (even before saving it).



```
SVApplications.Output("")
```

Line: 1

```
Description      Location  Line  Object                                   TimeStamp
Error: ; expected Recipes  1     Kernel.Messages.ScriptGraphicEditorReferenc 30/03/2020 16:51:19.512
```

Constants

The SmartView scripts contain some constants that can also be used by the developers. Take a look at the table below with examples and descriptions.

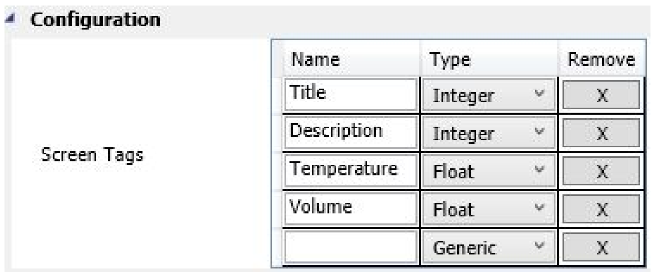
Constant	Example	Description

IsRunningWeb	`if (IsWebRunning)`
	`{`
	`   // Do something for the web`
	`}`
		Checks if the current viewer is on the web or not. This constant is a bool type that returns true in case it is on the web viewer.

Scripts and ScreenTags

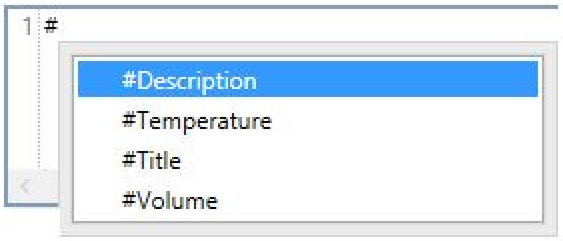
The SmartView allows you to create screen tags. The scope of those tags are only local to that screen and they can be used directly on the objects of the screen or in the Scripts. Since the scope of this document is the scripts, let's take a look at them. The screen tags allow you to create a generic screen that can be used with different inputs, not restricting you to always use the same global tags.

They can be created in the Screen's property grid as you can see in the image below.



Name	Type	Remove
Title	Integer	X
Description	Integer	X
Temperature	Float	X
Volume	Float	X
	Generic	X

And they can be used using the # token on the scripts. As soon as you type the # sign, all the screen tags will be listed to help you find them.



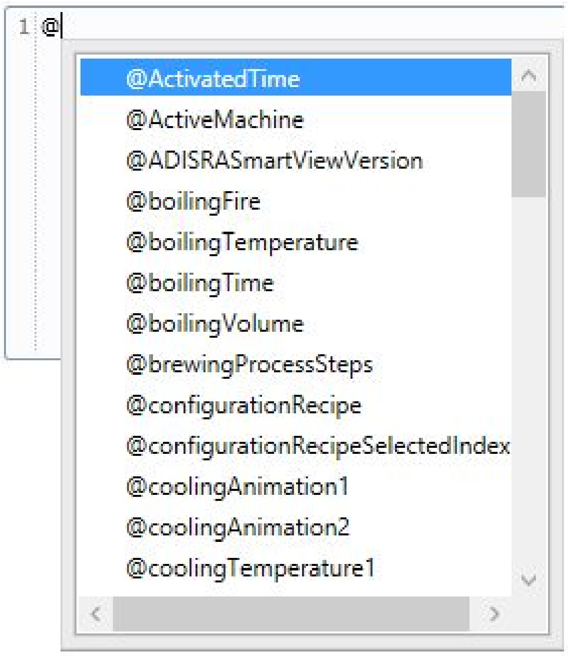
```
#
```

Scripts and Template Tags

The template objects work the same way. As soon as you create a new template object, you will have the option to create Object Tags on the property grid that can also be accessed using the # sign. This makes the template objects incredibly flexible and generic, allowing them to be used with different inputs.

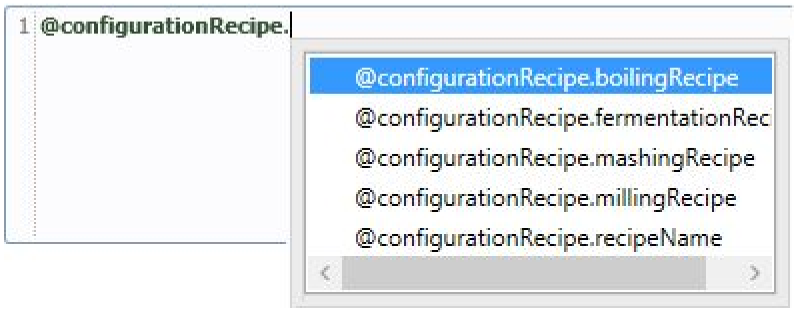
Scripts and Tags

In the SmartView, all the Tags created in the application can be used in the Scripts, you just need to add the token “@” and the Scripts will display all the Tags available.



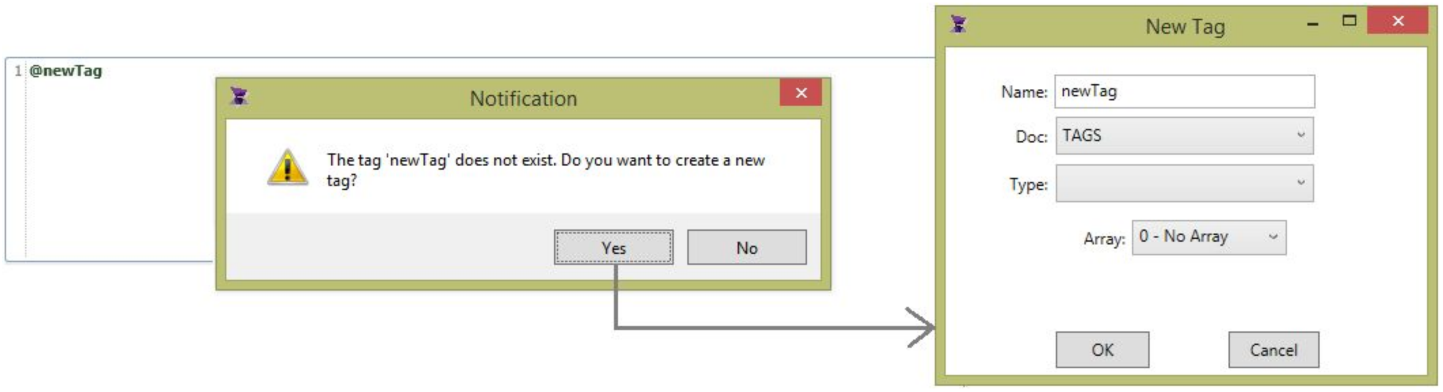
```
@
@ActivatedTime
@ActiveMachine
@ADISRASmartViewVersion
@boilingFire
@boilingTemperature
@boilingTime
@boilingVolume
@brewingProcessSteps
@configurationRecipe
@configurationRecipeSelectedIndex
@coolingAnimation1
@coolingAnimation2
@coolingTemperature1
```

In case the tag is a data type, the autocomplete will also help you find the inner tag like in the example below.



```
@configurationRecipe.
@configurationRecipe.boilingRecipe
@configurationRecipe.fermentationRec
@configurationRecipe.mashingRecipe
@configurationRecipe.millingRecipe
@configurationRecipe.recipeName
```

If the tag doesn't exist, the SmartView will let you create it, by selecting its type and the document location..



```
@newTag
```

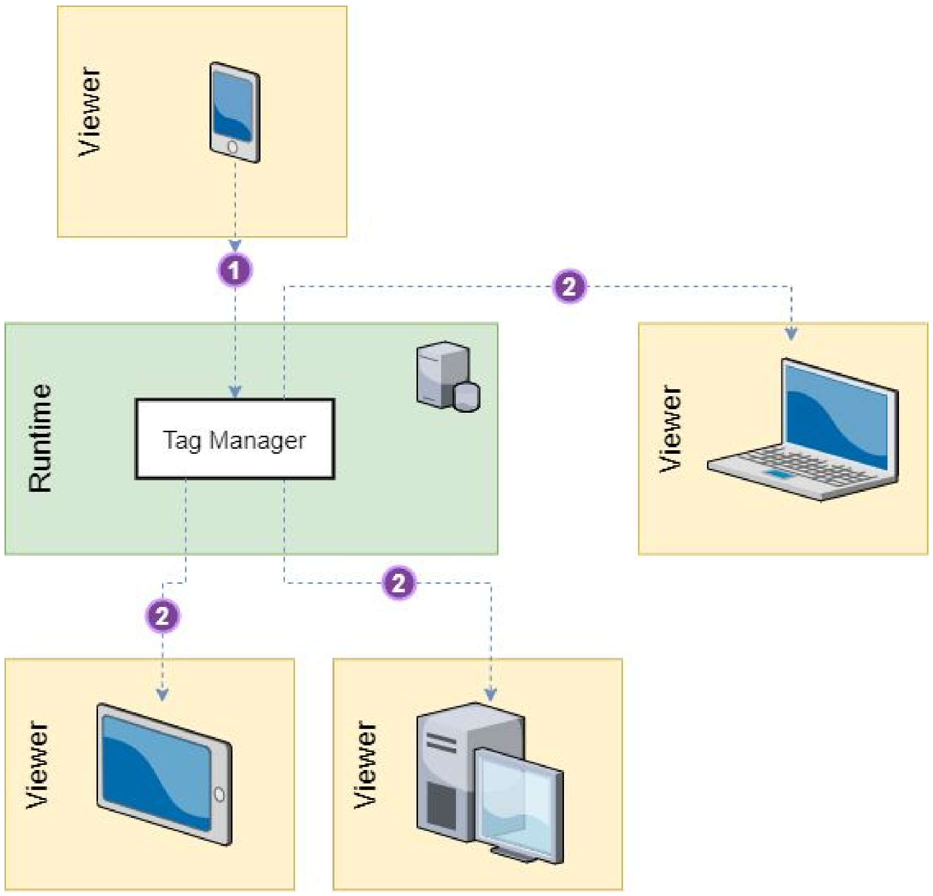
Notification

The tag 'newTag' does not exist. Do you want to create a new tag?

New Tag

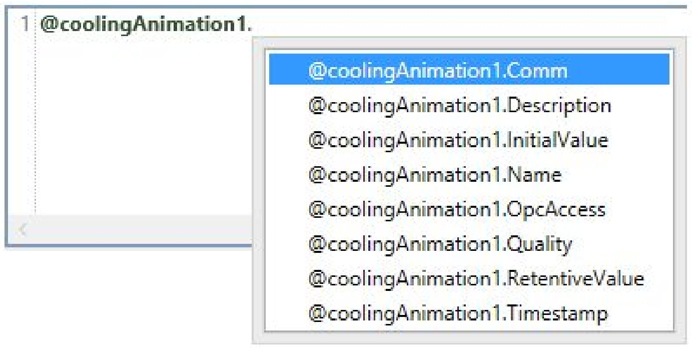
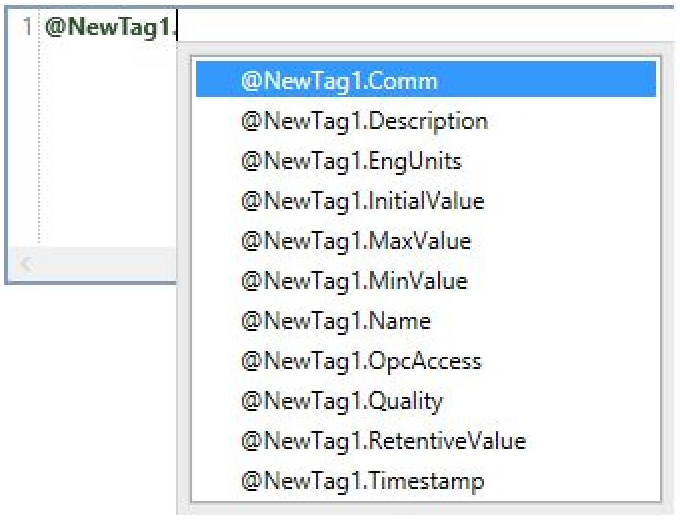
Name:	newTag
Doc:	TAGS
Type:	
Array:	0 - No Array

Whenever a tag's value is set in a script, this tag change is sent to the Runtime's Tag Manager and it is responsible for notifying all the Viewers.



Tag Properties

It is possible not only to change a Tag's value but also to read some of its properties in the scripts. The properties available to be used by the scripts depends on the type of the tag. As an example, you can find a boolean and an integer tag's properties on the images below.

Boolean Tag	Integer Tag
	

```
@coolingAnimation1.
  @coolingAnimation1.Comm
  @coolingAnimation1.Description
  @coolingAnimation1.InitialValue
  @coolingAnimation1.Name
  @coolingAnimation1.OpcAccess
  @coolingAnimation1.Quality
  @coolingAnimation1.RetentiveValue
  @coolingAnimation1.Timestamp
```

```
@NewTag1.
  @NewTag1.Comm
  @NewTag1.Description
  @NewTag1.EngUnits
  @NewTag1.InitialValue
  @NewTag1.MaxValue
  @NewTag1.MinValue
  @NewTag1.Name
  @NewTag1.OpcAccess
  @NewTag1.Quality
  @NewTag1.RetentiveValue
  @NewTag1.Timestamp
```

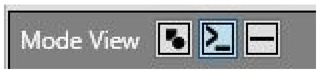
On the following table you will see all the properties available categorized by the tag type, its type of access (read or write) and a description.

Property	Type	Read/Write	Description
Comm	All	Read	Indicates if tag is set to any communication. It can be none, simulation or maybe a driver, opc client, etc… A tag can only be associated with one type of communication.

Description	All	Read	Get tag's description
EngUnits	Integer, Float	Read	
InitialValue	All	Read	Get tag's initial value
MaxValue	Integer, Float	Read	
MinValue	Integer, Float	Read	
Name	All	Read	Get tag's name
OpcAccess	All	Read	Get tag's opc access (Read, Write or Read/Write)
Quality	All	Read	Get tag's Quality. 192 is a good quality. Anything different is in error.
RetentiveValue	All	Read	Check if the tag is set to true/false for retentive value. In case it is true, value will be kept saved and restored in case the application restarts
TimeStamp	All	Read	Last time tag changed value. Its format is Date and Time

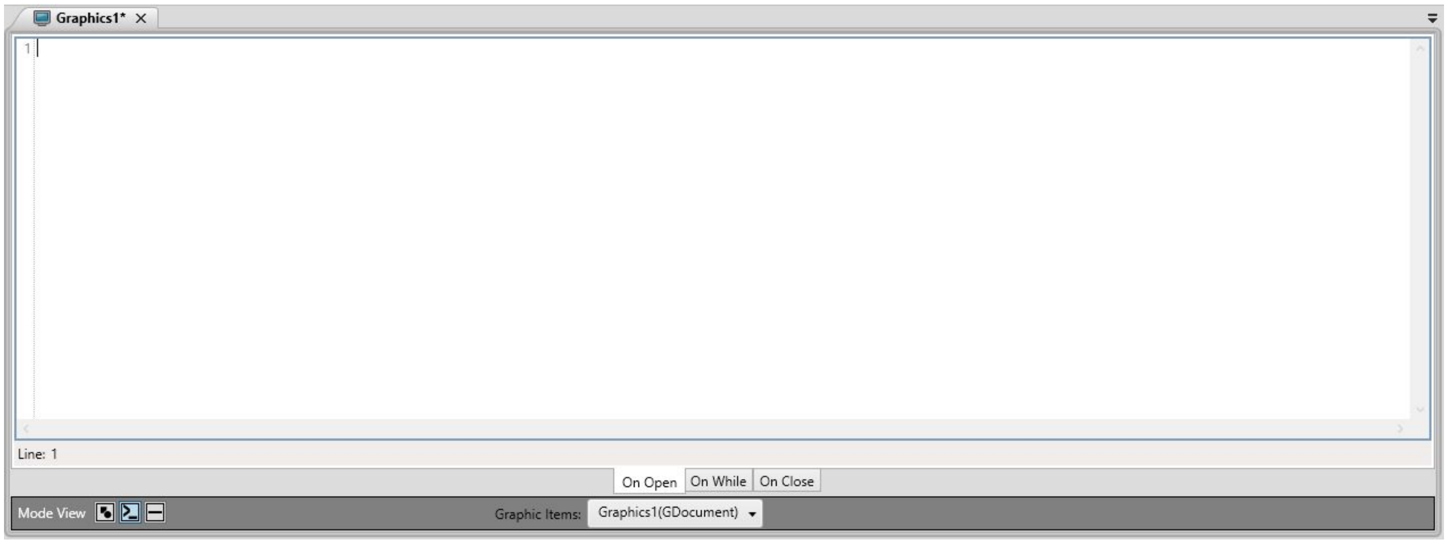
Scripts and Screens

The screens have 3 events where a script can be written. To have access to those script windows you can either select the Script View or the Split View and the screen must be selected. Just look for the Mode View component and make your choice.



**Mode View selection*

The following image shows the script window in the Script View mode.



As previously mentioned, it accepts any C# compliant code and it also accepts tags with the @ sign, User Functions, System Functions and it can also manipulate screen object's properties. The screen object properties will be explained in the next section in detail. *These scripts accept multiple lines of code, but you cannot write functions nor classes*. Think about it as a function already declared with no parameters or return value.

Script Events

There are 3 types of events in the Screen Scripts. OnOpen, OnWhile and OnClose

OnOpen

Executes every time the current screen opens. It will be executed only once. It doesn't matter if the screen is being opened by the Screen Object (part of the interface objects), or by the SVGraphics system function or any other way, this code will be executed once.

OnWhile

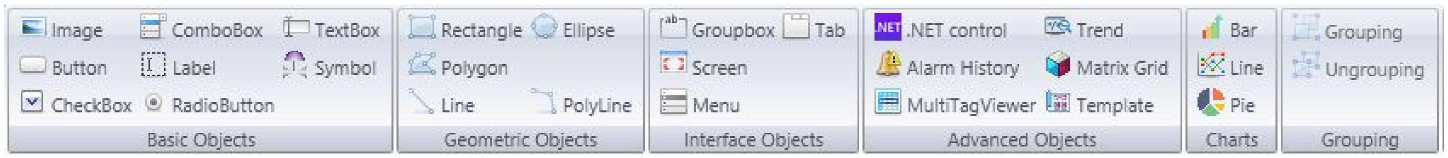
Executes multiple times while the screen is still opened. The frequency of its execution can be very low (around 300ms) and it depends on how much processing is happening on the Viewer. This is commonly used for animating the screen or checking for a specific condition (although you can use a trigger in case you expect certain conditions over a tag's value).

OnClose

Executes every time the current screen is closed. It will be executed only once.

Scripts and Screen Objects

There are multiple Objects available in the SmartView to be added to the screens. They fall under different categories like Geometric objects, Basic objects, Interface objects, Advanced objects and Charts. The scope of this document is to show all the properties and functions that can be used to manipulate those objects.
It is important to be very clear that those object properties and functions can only be used by Screen Scripts (must be in the screen script where the object was added) and Object Scripts that are in the same screen.
To have access to the properties and functions on the scripts, you will need the object name. By default, the SmartView already assigns a name to the objects when they are created, but you can also change the name for better organization and maintenance..



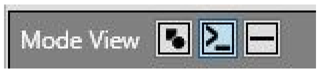
**Graphics object bar*

Script Events

There are 6 common types of events in the Object Scripts. MouseUp, MouseDown, MouseWhile, MouseRightUp, MouseRightDown and MouseDoubleClick. But for the ComboBox object there is one more event called SelectionChanged. Let's take a look at each event in further detail.

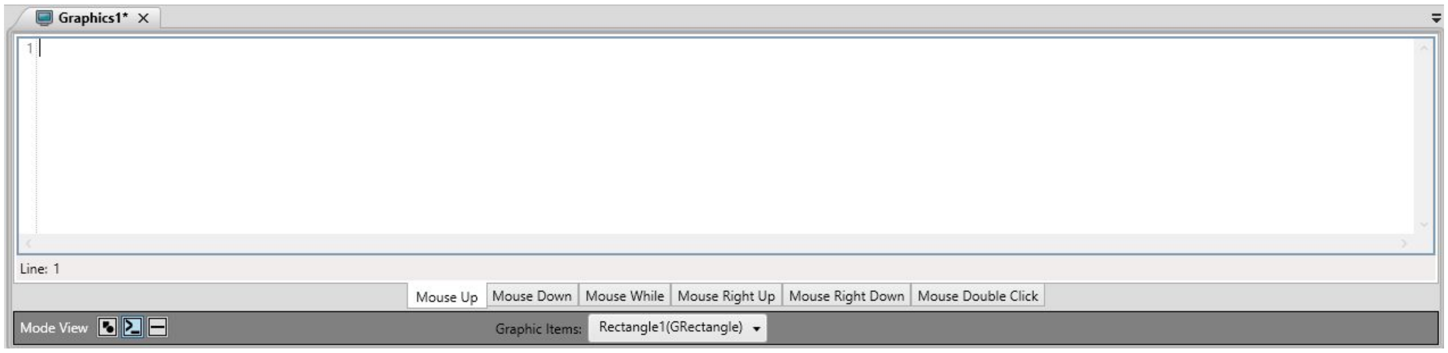
There are 2 ways to have access to the Object's Script Events.

1) You can either select the object and change the Mode View to Script View (you can also double click the object to open that mode view) or select the Split View.

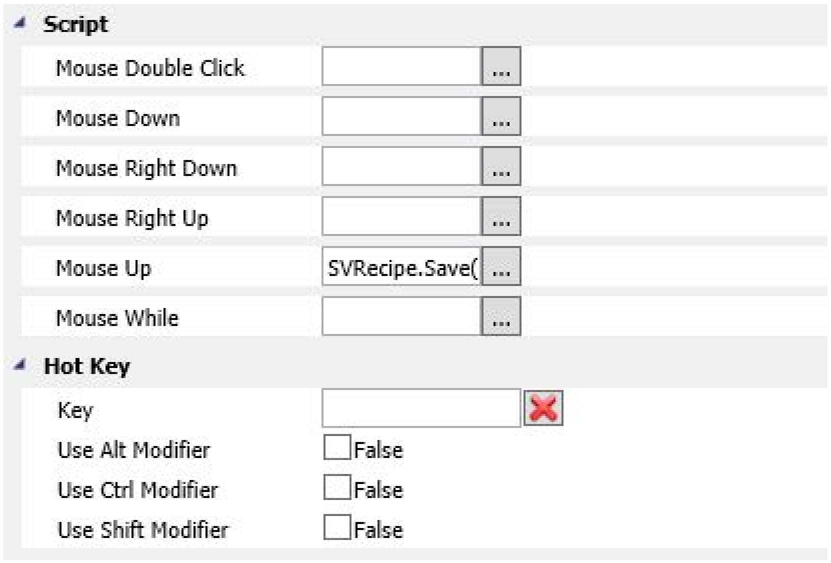


**Mode View selection*

The following image shows the Rectangle's Script Events in the Script View mode. Those same events are common to most of the objects



2) The second way is getting access to the script events using the Property Grid after selecting the object as you can see in the image below.



Script	
Mouse Double Click	
Mouse Down	
Mouse Right Down	
Mouse Right Up	
Mouse Up	SVRecipe.Save(
Mouse While	
Hot Key	
Key	
Use Alt Modifier	False
Use Ctrl Modifier	False
Use Shift Modifier	False

**Access to the object's scripts using the Property Grid*

MouseUp

Is executed whenever someone releases the **left** button of the mouse over the selected object. This event is commonly used on actions that can be more critical and you give a chance to the user to quit. If the user clicks down a button for example but while it is clicked he decides not to complete the action, he can drag the mouse away from the button and then release it. This way the event won't be executed. It can also be executed when configuring a HotKey.

MouseDown

Is executed whenever someone clicks down the selected object with the **left** button of the mouse. It can also be executed when configuring a HotKey.

MouseWhile

Is executed while the selected object is still clicked, between MouseDown and MouseUp events of the **left** button of the mouse..

MouseRightUp

Is executed whenever someone releases the **right** button of the mouse over the selected object.

MouseRightDown

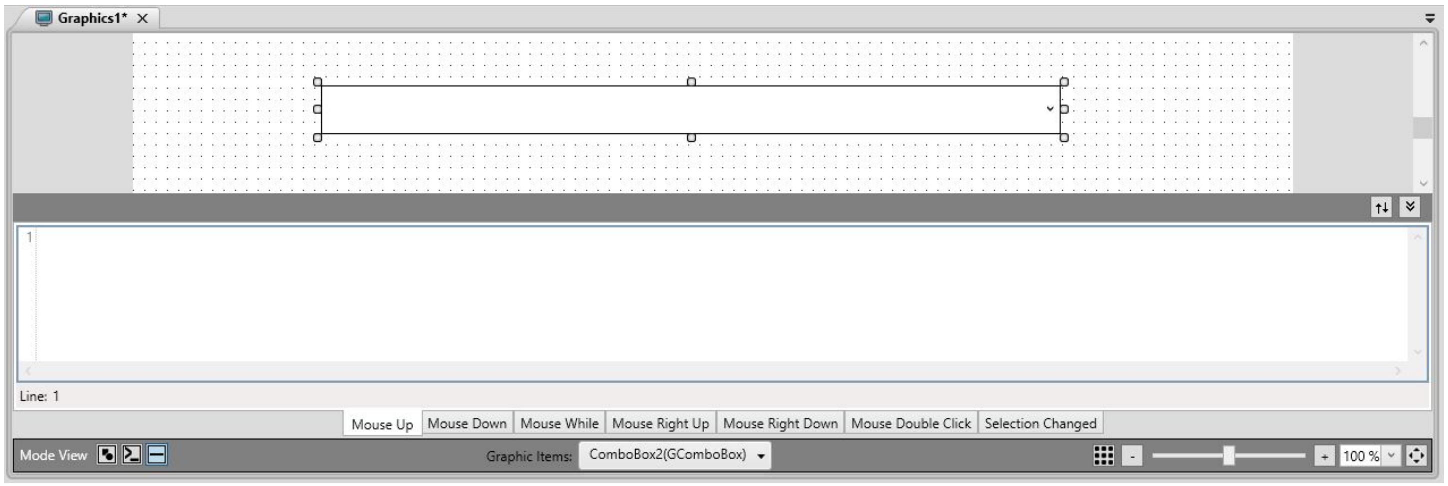
Is executed whenever someone clicks down the selected object using the **right** button of the mouse.

MouseDoubleClick

Is executed when a double click is identified using the mouse **left** button..

SelectionChanged

Is executed whenever the selected item of a ComboBox changes. Let's see an image of the ComboBox scripts below.



Let's take a look at the list of objects, their category and if they have click events or not.

Name	Category	Allow Scripts
Image	Basic Object	Yes
ComboBox	Basic Object	Yes
TextBox	Basic Object	Yes
Button	Basic Object	Yes
Label	Basic Object	Yes
Symbol	Basic Object	Yes
CheckBox	Basic Object	Yes
RadioButton	Basic Object	Yes
Rectangle	Geometric Object	Yes
Ellipse	Geometric Object	Yes
Polygon	Geometric Object	Yes
Line	Geometric Object	Yes
PolyLine	Geometric Object	Yes
GroupBox	Interface Object	Yes
Tab	Interface Object	No
Screen	Interface Object	No
Menu	Interface Object	Yes
.NET Control	Advanced Object	No
Trend	Advanced Object	No
Alarm/Events	Advanced Object	No
Matrix Grid	Advanced Object	Yes
MultiTagViewer	Advanced Object	Yes
Template	Advanced Object	Yes

Bar	Charts	No
Line	Charts	No
Pie	Charts	No
Grouping	Grouping	Yes

Object Properties and Functions

In this section you will learn not only which objects contain scripts but also all the properties and functions available for each object.
As mentioned before, the Object properties and functions can only be used inside the graphic screen it is created. They can be used by the Screen Scripts or by any other object located in the same script.

Some properties are inherited and common to all the objects (or at least most of the objects) and those properties will be listed here to avoid replicating it to all objects.

1. Common Properties

Name	Type	Read/Write	Description
isEnabled	bool	Read/Write	Enables or Disables the object
isVisible	bool	Read/Write	Shows or hides the object
Width	double	Read/Write	Gets or Sets the width of the object
Height	double	Read/Write	Gets or Sets the height of the object
Left	double	Read/Write	Gets or Sets the left of the object
Top	double	Read/Write	Gets or Sets the top of the object

2. Rotation Properties

Name	Type	Access (Read/Write)	Description
Angle	double	Read/Write	Gets or Sets the angle of the object

Basic Objects

Image (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Source	string	Read/Write	Gets or Sets the source of the image. The source image must be located in the Images folder of the project. Just need to pass the file name of the image. (i.e. Image1.Source = “ADISRA.png”;)

2. Functions

No functions available

ComboBox (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Items	List<string>	Read/Write	Gets or Sets the list of items in the ComboBox.
SelectedIndex	int	Read/Write	Gets or Sets the selected item of a combobox using the index.

SelectedText	string	Read/Write	Gets or Sets the selected item of a combobox using the text.
FontColor	string	Read/Write	Gets or Sets the font color of the items in the combobox.

2. Functions

Name	Return	Description
InsertItem(string item)	void	Insets a new item to the combobox. It must be a string passed as parameter
InsertItemsFromTag(string tag)	void	Insert items from a tag. It can be a simple tag or an array of tags. The values will be converted into strings.
InsertItemsFromTag(string tag, bool isScreenTag)	void	Insert items from a tag similar to the previous function but it indicates if the tag is a screen tag or not. If it is, this function must be used instead
InsertItemsFromTag(string tag, string member)	void	Insert items from a data type tag. In this case you will need to give the name of the data type member
InsertItemsFromTag(string tag, string member, bool isScreenTag)	void	Insert items from a data type tag similar to the previous function but it indicates if the data type tag is a screen tag or not. If it is, this function must be used instead
RemoveSelectedItem()	void	Removes selected item from the combobox
RemoveItem(int index)	void	Removes item of the combobox for the given index. The index starts with 0 as the first item.

RemoveAllItems()	void	Removes all the items in the combobox

TextBox (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text on the TextBox
FontColor	string	Read/Write	Gets or Sets the font color of the TextBox. (i.e. TextBox1.FontColor = “red”;)

2. Functions

Name	Return	Description
ResetPasswordBox()	void	Clear the text in case the TextBox has the property Password set to true in the property grid
GetFocus()	void	Set focus on the TextBox
SelectAll()	void	Select the entire text of the TextBox

Button (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text of the Button

			(i.e. Button1.Text = “Welcome”;)
FontColor	string	Read/Write	Gets or Sets the font color of the Button. (i.e. Button1.FontColor = “red”;)
IsChecked	bool	Read/Write	Gets or Sets the button as checked or unchecked. It only works if toggle button is set to true in the property grid (i.e. btnlabel.Text = btn.IsChecked.ToString();)

2. Functions

Name	Return	Description
changeBorderThickness(int left, int top, int right, int bottom)	void	Changes the border thickness of the button
changeBorderThicknessPressed(int left, int top, int right, int bottom)	void	Changes the border thickness of the button while pressed.

Label (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text of the Label. (i.e. Label1.Text = “Welcome”;)
FontColor	string	Read/Write	Gets or Sets the font color of the Label. (i.e. Label1.FontColor = “red”;)

2. Functions

No functions available

Symbol (Common Properties,Rotation Properties)

3. Properties

No properties available apart from the ones inherited

4. Functions

No functions available

CheckBox (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text of the Checkbox. (i.e. CheckBox1.Text = “Checkbox 1”;)
FontColor	string	Read/Write	Gets or Sets the font color of the Checkbox. (i.e. CheckBox1.FontColor = “red”;)
IsChecked	bool	Read/Write	Gets or Sets the checkbox (as checked or unchecked). If the checkbox is Three States, the true will only be assigned to the checked state (not to unchecked or the third unexpected state)
IsCheckedInt	int	Read/Write	Gets or Sets the checkbox using an integer (0 as unchecked or 1 as checked). If it is a Three State checkbox, it will also have -1 to represent the third unexpected state.

2. Functions

No functions available

RadioButton (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text of the RadioButton. (i.e. RadioButton1.Text = “Option 1”;)
FontColor	string	Read/Write	Gets or Sets the font color of the Checkbox. (i.e. RadioButton1.FontColor = “red”;)
IsChecked	bool	Read/Write	Gets or Sets the RadioButton as selected.

2. Functions

No functions available

Geometric Objects

Rectangle (Common Properties,Rotation Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Ellipse (Common Properties,Rotation Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Polygon (Common Properties,Rotation Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Line (Common Properties,Rotation Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
X1	double	Read/Write	Gets or Sets the first point in the X axis
Y1	double	Read/Write	Gets or Sets the first point in the Y axis
X2	double	Read/Write	Gets or Sets the second point in the X axis

Y2	double	Read/Write	Gets or Sets the second point in the Y axis

2. Functions

No functions available

PolyLine (Common Properties,Rotation Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Interface Objects

GroupBox (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
Text	string	Read/Write	Gets or Sets the text of the GroupBox. (i.e. GroupBox1.Text = “GroupBox 1”;)
FontColor	string	Read/Write	Gets or Sets the font color of the GroupBox. (i.e. GroupBox1.FontColor = “red”;)

2. Functions

No functions available

Tab (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
FontColor	string	Read/Write	Gets or Sets the font color of the Tabs. (i.e. Button1.FontColor = “red”;)

2. Functions

Name	Return	Description
SetVisibility(int index, bool IsVisible)	void	Change a tab visibility for a given index, starting from 0.
SetIsEnabled(int index, bool IsEnabled)	void	Enable or Disable a tab for a given index, starting from 0.
ChangeSelectedIndex(int index)	void	Change the selected tab for a given index, starting from 0.
IsTabVisible(int index)	bool	Check if tab is visible or hidden for a given index, starting from 0

Screen (Common Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

Name	Return	Description
GetCurrentScreen()	string	Gets current opened graphic screen opened by the Object screen

Menu (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
FontColor	string	Read/Write	Gets or Sets the font color of the Menu. (i.e. Menu1.FontColor = “red”;)

2. Functions

No functions available

Advanced Objects

.NET Control (Common Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Trend (Common Properties)

The trend object has many functions that can be used in the Scripts and some of them make reference to SPC which stands for Statistical Process Control which is an industry-standard methodology for measuring and controlling quality during the manufacturing process. Data that falls within the control limits indicates that everything is operating as expected. Any variation within the control limits is likely due to a common cause—the natural variation that is expected as part of the process. If data falls outside of the control limits, this indicates that an assignable cause is likely the source of the product variation, and something within the process should be changed to fix the issue before defects occur.
It is also important to detail the deviation equation.

1. Properties

No properties available apart from the ones inherited

2. Functions

Name	Return	Description
Play()	void	Plays the trend object
Pause()	void	Pauses the trend object
AddPen(string tagName, string description)	void	Adds a new pen to the trend object adding an existing tag and a description for the pen.
AddPen(string tagName, string description, bool showSpc)	void	Adds a new pen to the trend object as the previous function and also indicates if all the SPC information should be visible or not.
AddPen(string tagName, string description, bool showSpcMin, bool showSpcMax, bool showSpcAvg, bool showSpcDev, int spcDevSigma)	void	Adds a new pen to the Trend object and allows the user to specify which SPC information should be visible.
AddPen(string tagName, string description, string penColor)	void	Adds a new pen to the Trend object and allows the user to specify the pen color.
AddPen(string tagName, string description,	void	Adds a new pen to the Trend object and

string penColor, bool showSpc)		allows the user to specify the pen color and if all the SPC information should be visible or not
AddPen(string tagName, string description, string penColor, bool showSpcMin, bool showSpcMax, bool showSpcAvg, bool showSpcDev, int spcDevSigma)	void	Adds a new pen to the Trend object and allows the user to specify the pen color and which SPC information should be visible.
AddPen(string tagName, string description, int lineStyle, double thickness, string penColor, bool interpol, int marker, bool showLegend)	void	Adds a new pen to the Trend object and allows the user to specify the line style (starts from zero and uses the same line style that you can find in the trend's property grid), the pen thickness, pen color, if it should interpolate, the type of marker (the marker starts with 0 for None, 1 for Circle and 2 for Triangle) and if legend should be visible or not.
AddPen(string tagName, string description, int lineStyle, double thickness, string penColor, bool interpol, int marker, bool showLegend, bool showSpc)	void	Adds a new pen to the Trend object just like the previous version and it also allows the all the SPC to be visible.
AddPen(string tagName, string description, int lineStyle, double thickness, string penColor, bool interpol, int marker, bool showLegend, bool showSpcMin, bool showSpcMax, bool showSpcAvg, bool showSpcDev, int spcDevSigma)	void	Adds a new pen to the Trend object just like the previous version but it allows adding the SPC information individually.
SetPenDescription(string tagName, string description)	void	Sets a pen description by the name of the tag.
SetPenLineStyle(string tagName, int lineStyle)	void	Sets a pen line style by the name of the tag (line style starts from zero and uses the same line style that you can find in the trend's property grid)
SetPenThickness(string tagName, double thickness)	void	Sets a pen thickness by the name of the tag
SetPenColor(string tagName, string penColor)	void	Sets a pen color by the name of the tag

SetPenInterpolated(string tagName, bool interpol)	void	Sets a pen to be interpolated by the name of the tag
SetPenMin(string tagName, double min)	void	Sets a pen min value by the name of the tag
SetPenMax(string tagName, double max)	void	Sets a pen max value by the name of the tag
SetPenMarker(string tagName, int marker)	void	Sets a pen marker by the name of the tag (the marker starts with 0 for None, 1 for Circle and 2 for Triangle)
SetPenShowLegend(string tagName, bool showLegend)	void	Sets the visibility of the pen legend.
HasPenSPCMin(string tagName)	bool	Checks if SPCMin is visible or not.
HasPenSPCMax(string tagName)	bool	Checks if SPCMax is visible or not.
HasPenSPCAvg(string tagName)	bool	Checks if SPCAvg is visible or not.
HasPenSPCDev(string tagName)	bool	Checks if SPCDev is visible or not.
GetPenSPCDevSigma(string tagName)	int	Gets the SPCDevSignal used in the deviation calculation of the pen
GetPenSPCMin(string tagName)	double	Gets current PenSPCMin value of the points visible
GetPenSPCMax(string tagName)	double	Gets current PenSPCMax value of the points visible
GetPenSPCAvg(string tagName)	double	Gets current PenSPCAvg value of the points visible
GetPenSPCDev(string tagName)	double	Gets current PenSPCDev value of the points visible
GetPenSPCDevPositive(string tagName)	double	Gets the value of the positive deviation (upper deviation line)
GetPenSPCDevNegative(string tagName)	double	Gets the value of the negative deviation (lower deviation line)
SetSPC(bool showSPC)	void	Sets all the SPC visible for all pens in the

		trend
SetSPC(bool showSpcMin, bool showSpcMax, bool showSpcAvg, bool showSpcDev, int spcDevSigma)	void	Sets individually the SPC information visible for all pens in the trend
SetPenSPC(string tagName, bool showSPC)	void	Sets all the SPC information visible for a specific tag.
SetPenSPC(string tagName, bool showSpcMin, bool showSpcMax, bool showSpcAvg, bool showSpcDev, int spcDevSigma)	void	Sets individually the SPC information visible for a specific tag.
HidePen(string tagName, bool isHide)	void	This function hides or show a desired pen in the trend object
RemovePen(string tagName)	void	Remove the pen from the Trend object.
UpdateStartDate(DateTime date)	void	Updates the start date of the Trend object.
UpdateEndDate(DateTime date)	void	Updates the end date of the Trend object.
UpdateDuration(int duration)	void	Updates the duration of the Trend object. The parameter unity is seconds.
UpdateDuration(TimeSpan duration)	void	Updates the duration of the Trend object passing a timespan as parameter
UpdateDurationEnd(int duration, DateTime date)	void	Updates the EndDuration of the Trend object. The StartDuration is calculated by the EndDuration - Duration. The duration expected is in seconds.
UpdateDurationEnd(TimeSpan duration, DateTime date)	void	Updates the EndDuration of the Trend object. The duration parameter is a TimeSpan.

Alarm/Events (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
SelectedLine	int	Read/Write	Gets or Sets the selected line in the Alarm Object.

2. Functions

Name	Return	Description
Ack()	void	Acknowledge selected alarm.
AckAll()	void	Acknowledge all the alarms
DeselectLines()	void	Remove selection from all the lines
Filter(string type, object filter)	bool	Configures a filter for the Alarm object. (i.e Alarm1.Filter("Name.Value", NameTextBox.Text);)
SetAbsolutStartDate(DateTime date)	void	Sets the Absolute Start Date of the Alarm Object
SetAbsolutStartDate(string date)	void	Sets the Absolute Start Date of the Alarm Object
SetAbsolutEndDate(DateTime date)	void	Sets the Absolute End Date of the Alarm Object
SetAbsolutEndDate(string date)	void	Sets the Absolute End Date of the Alarm Object
SetAbsolutStartEndDate(DateTime startDate, DateTime endDate)	void	Sets the Absolute Start and End Date of the Alarm Object
SetAbsolutStartEndDate(string startDate, string endDate)	void	Sets the Absolute Start and End Date of the Alarm Object
SetAbsolutStartEndDate(DateTime startDate, string endDate)	void	Sets the Absolute Start and End Date of the Alarm Object
SetAbsolutStartEndDate(string startDate, DateTime endDate)	void	Sets the Absolute Start and End Date of the Alarm Object
SetRelativeDate(DateTime date)	void	Sets the Relative Start Date of the Alarm Object

SetRelativeDate(string date)	void	Sets the Relative Start Date of the Alarm Object
SetRelativePeriod(TimeSpan period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativePeriod(string period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativePeriod(double period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativePeriod(TimeSpan period, bool isBackward, bool isForward)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativePeriod(string period, bool isBackward, bool isForward)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativePeriod(double period, bool isBackward, bool isForward)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(DateTime date, TimeSpan period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(DateTime date, string period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(DateTime date, double period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, TimeSpan period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, string period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, double period)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(DateTime date, TimeSpan period, bool isBackward, bool isForward)	void	Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(DateTime date, string period, bool isBackward, bool isForward)	void	Sets the period that the alarms will be tracked in the Alarm Object.

SetRelativeDatePeriod(DateTime date, double period, bool isBackward, bool isForward)		Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, TimeSpan period, bool isBackward, bool isForward)		Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, string period, bool isBackward, bool isForward)		Sets the period that the alarms will be tracked in the Alarm Object.
SetRelativeDatePeriod(string date, double period, bool isBackward, bool isForward)		Sets the period that the alarms will be tracked in the Alarm Object.
AddFilter(string newFilter)		Adds a new filter for the alarm object
RemoveFilter(string filterToRemove)		Removes a filter from the alarm object
CleanFilter()		Removes all filters from the alarm object
ClearEvents()		Removes all the events from the alarm object

Matrix Grid (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
ColumnIndex	int	Read/Write	Returns the column index of the selected cell. (index starts with 0 and increments)
RowIndex	int	Read/Write	Returns the row index of the selected cell (index starts with 0 and increments)
DepthIndex	int	Read/Write	Returns the depth index of the selected cell. In case there are 3 dimensions and you selected a cell in the 3rd dimension, it will return 2 (index starts with 0 for 1st dimension and increments)

2. Functions

Name	Return	Description
ChangeDataSource(string tagName)	void	Change the MatrixGrid data source by the tag's name passed as parameter

MultiTagViewer (Common Properties)

1. Properties

Name	Type	Access (Read/Write)	Description
SelectedGridIndex	int	Read/Write	Gets or Sets the selected grid line of the given index. The first index is 0
SelectedArrayIndex	int	Read/Write	Gets or Sets the tag's array index. The first index is 0. This property is very useful when the user sorts the grid lines, so the index of the grid line does not correspond to the index of the array.
Count	int	Read	Gets the number of lines in the MatrixGrid.

2. Functions

Name	Return	Description
ChangeImageSource(int line, string key, string newSource)	void	This function configures the image that will be shown in the specific property (key) in the MultiTagViewer
ChangeBackground(int line, string key, string color)	void	This function configures the background color that will be used in the specific property (key) in the MultiTagViewer
Remove()	void	This function will remove the selected line

		in the MultiTagViewer object
Add(string property, object value)	void	
Add(Dictionary<string, object> properties)	void	
Filter(string property, object value)	void	Configures a filter for the MultiTagViewer object. (i.e MTV1.Filter("Name.Value", NameTextBox.Text);)
Filter(Dictionary<string, object> properties)	void	Configures multiple filters to the MultiTagViewer object.
RefreshIndex()	void	This function will refresh the index of the MultiTagViewer object
HideColumn(int index)	void	This function hides a column in the MultiTagViewer object.
ShowColumn(int index)	void	This function shows a column in the MultiTagViewer object.
IsColumnVisible(int index)	bool	Checks if the column index received as parameter is Visible (true) or hidden (false)

Template (Common Properties)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available

Charts

For the charts I will add a few more properties and functions that are common to them as they have the same ancestor. Let's name it as Chart Ancestor

1. Chart Ancestor

Properties

Name	Type	Read/Write	Description
Title	string	Read/Write	Gets or Sets the chart title
TitleHorizontalAlignment	string	Read/Write	Gets or Sets the title alignment of the graphic object. (“Center”, “Left” or “Right”)
BackgroundColor	string	Read/Write	Gets or Sets the chart’s background color.
PlotAreaColor	string	Read/Write	Gets or Sets the plot area color of the graphic object. (i.e. Chart1.PlotAreaColor = “blue”)
ShowGridBorder	bool	Read/Write	Shows or Hides the grid border
ShowVerticalLines	bool	Read/Write	Shows or Hides the vertical lines
ShowHorizontalLines	bool	Read/Write	Shows or Hides the horizontal lines

Functions

Name	Return	Description
UpdateTitleFormat(string family, int size, bool isBold, bool isItalic)	void	Updates the Chart’s title font format (i.e. Chart1.UpdateTitleFormat("Arial",10,true,true);)
UpdateLegendFormat(string family, int size, bool isBold, bool isItalic)	void	Updates the Chart’s legend font format (i.e Chart1.UpdateLegendFormat("Arial",10,true,false);
UpdateLegendSize(double width,	void	Updates the Chart’s legend size

double height)		

Bar (Common Properties,Chart Ancestor)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available apart from the ones inherited

Line (Common Properties,Chart Ancestor)

1. Properties

No properties available apart from the ones inherited

2. Functions

Name	Return	Description
InsertItem(int serieNumber, object x, object y)	void	Inserts an item to the Scatter Chart Object (i.e. LineChart.InsertItem(serieNumber, x, y);) The object type can be either string or numeric.
InsertItems(int serieNumber, List<object> x, List<object> y)	void	Inserts a list of items to the Line Chart Object (i.e. LineChart.InsertItems(serieNumber,

		list1,list2);) The object type can be either string or numeric.
ClearItems(int serieNumber)	void	Removes an entire series from the Line Charts Object
ChangeLineType(int serieNumber, int type)	void	Changes the line type of a series on the Line Charts Object
ChangeSeriesThickness(int serieNumber, double thickness)	void	Changes the thickness of a series on the Line Charts

Pie (Common Properties,Chart Ancestor)

1. Properties

No properties available apart from the ones inherited

2. Functions

No functions available apart from the ones inherited

Grouping

Grouping (Common Properties)

1. Properties

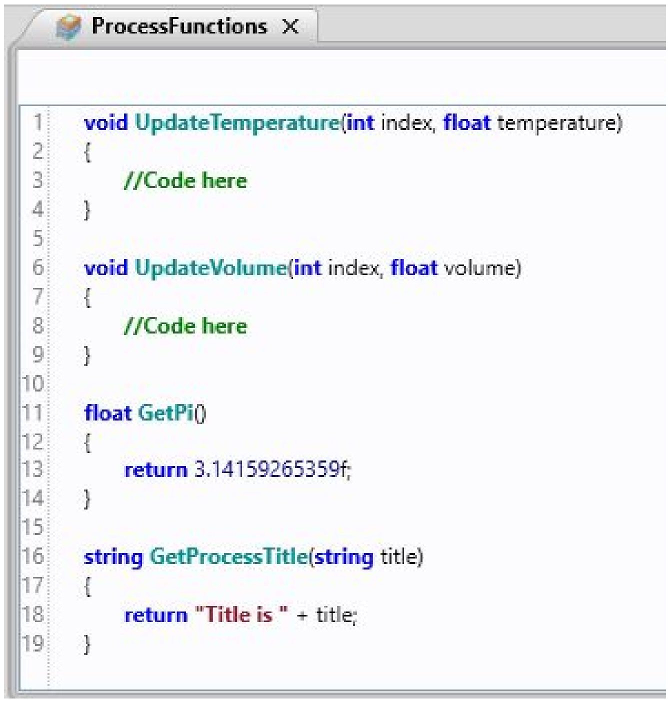
No properties available apart from the ones inherited

2. Functions

No functions available apart from the ones inherited

Scripts and User Functions Library

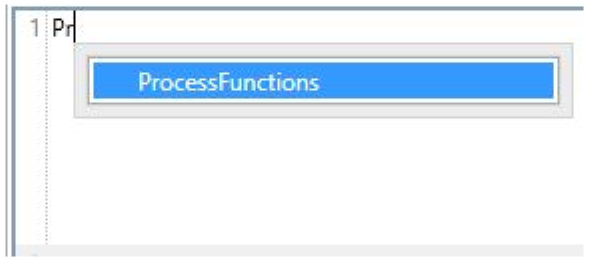
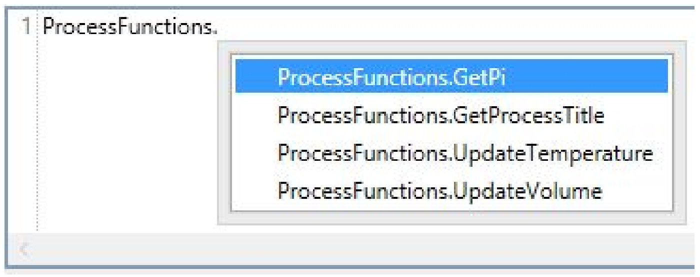
You can also create your own user functions in the SmartView, you just need to add functions to a library (document) and those functions will be available to be used in different places of your application. In the example below the library was named as ProcessFunctions and four functions were added to it.



ProcessFunctions

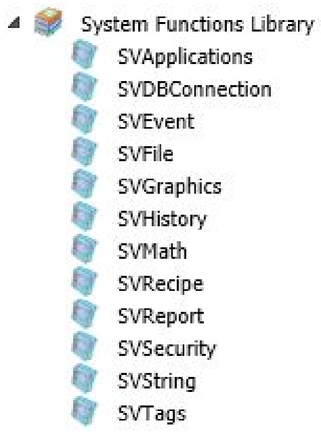
```
void UpdateTemperature(int index, float temperature)
{
      //Code here
}

void UpdateVolume(int index, float volume)
{
      //Code here
}

float GetPi()
{
      return 3.14159265359f;
}

string GetProcessTitle(string title)
{
      return "Title is " + title;
}
```

The autocomplete functionality will help you to add the ProcessFunction library and choose between the four functions as you can see below

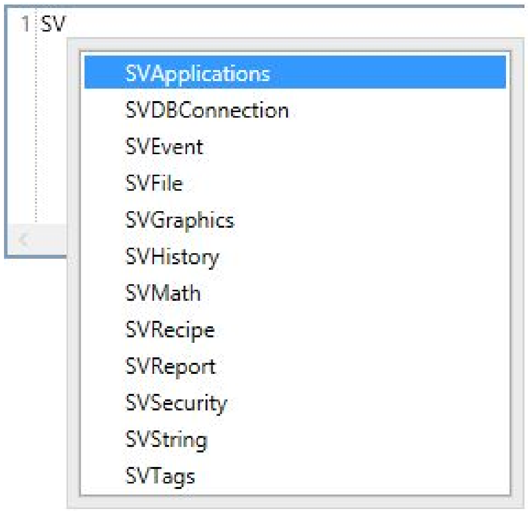
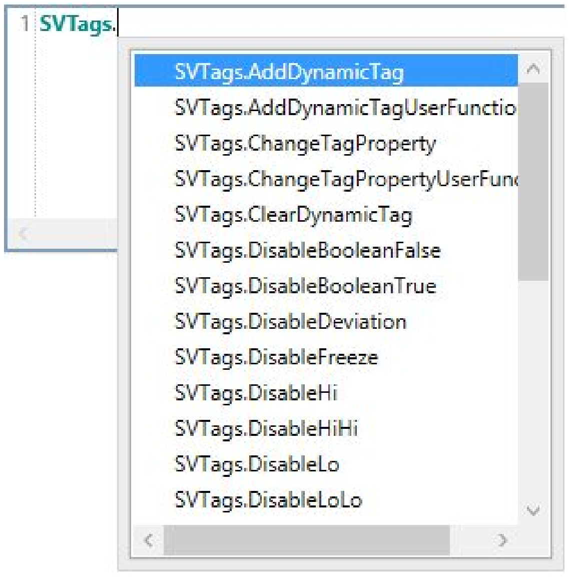
Add Library	Select Function
 1 Pr ProcessFunctions	 1 ProcessFunctions. ProcessFunctions.GetPi ProcessFunctions.GetProcessTitle ProcessFunctions.UpdateTemperature ProcessFunctions.UpdateVolume

Scripts and System Functions Library

The System Functions Library is also available to be used by the scripts. They are very useful libraries that will provide functions to make your life easier. You can locate them in the navigation tree and take a look at the available functions and their signature with comments.


System Functions Library
SVApplications
SVDBConnection
SVEvent
SVFile
SVGraphics
SVHistory
SVMath
SVRecipe
SVReport
SVSecurity
SVString
SVTags

As they have the same prefix, as soon as you type SV, they will all be listed. After the library is chosen, just add a dot and the functions will be listed.

Add Library	Select Function
 SV SVApplications SVDBConnection SVEvent SVFile SVGraphics SVHistory SVMath SVRecipe SVReport SVSecurity SVString SVTags	 SVTags. SVTags.AddDynamicTag SVTags.AddDynamicTagUserFunctio SVTags.ChangeTagProperty SVTags.ChangeTagPropertyUserFunc SVTags.ClearDynamicTag SVTags.DisableBooleanFalse SVTags.DisableBooleanTrue SVTags.DisableDeviation SVTags.DisableFreeze SVTags.DisableHi SVTags.DisableHiHi SVTags.DisableLo SVTags.DisableLoLo

Document Release

Version	Comments	Date	Author
1.0	Initial Version	02-April-2020	Fabio Carvalho
1.1	Added IsWebRunning constant to the new Sub-Section	06-April-2020	Fabio Carvalho
1.2	Change function name ChangeSerieThickness to ChangeSeriesThickness	09-April-2020	Fabio Carvalho