# SVFunctions Guide

## Document Information

| | |
|---|---|
| Software Version: | 4.0.3.3 |
| Creation Date: | 17 September, 2020 |
| Last Edit Date: | 17 March, 2022 |
| Version: | 1.1 |

## Table of Contents

# 1. Scope

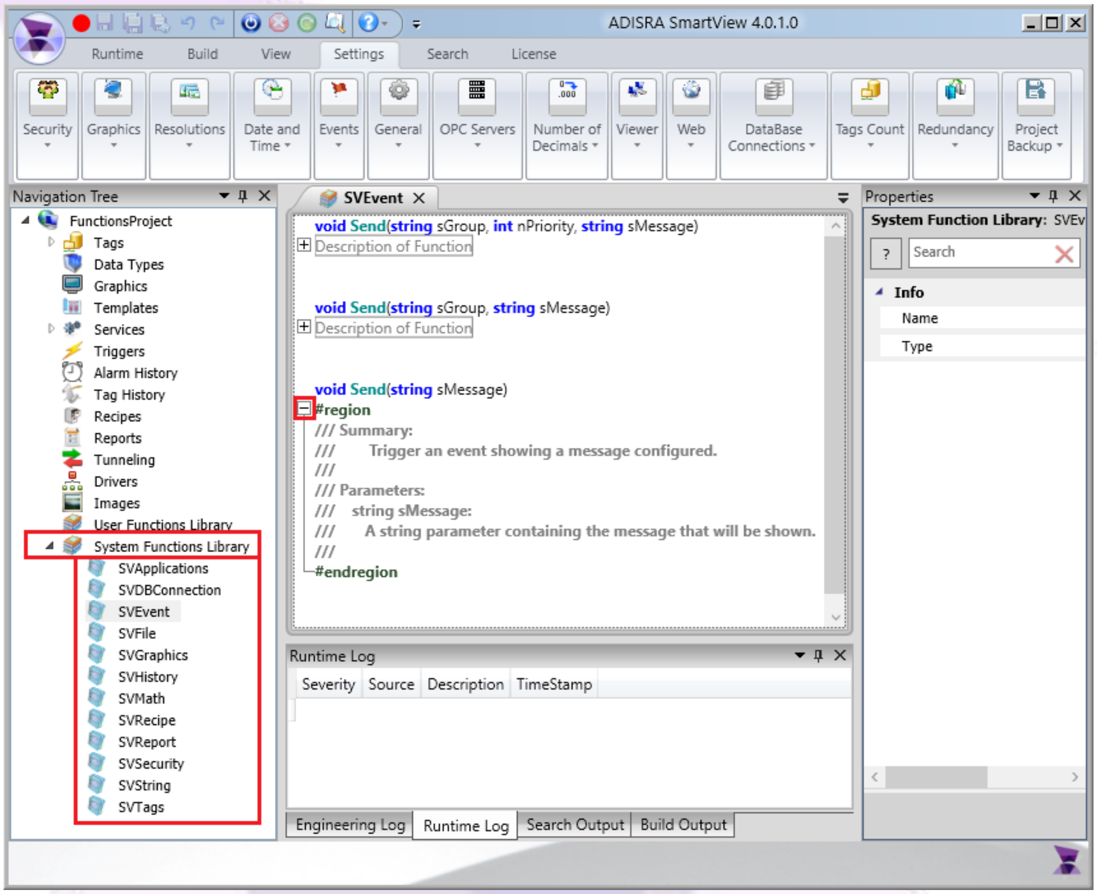
This document describes the ADISRA SmartView System Function Libraries, explain how they work and the steps to follow to use them.

# 2. Summary

System Function Libraries (SVFunctions) are functions that the ADISRA SmartView makes available to the user, and they are grouped into different libraries.  All these functions can be used in any script (i.e., service document, trigger, screens, etc.).

# 3. Using SVFunctions

To access the SVFunctions available, expand the node “System Function Libraries” on the navigation tree and open one of the documents.



```
void Send(string sGroup, int nPriority, string sMessage)
Description of Function

void Send(string sGroup, string sMessage)
Description of Function

void Send(string sMessage)
#region
/// Summary:
///      Trigger an event showing a message configured.
///
/// Parameters:
///    string sMessage:
///      A string parameter containing the message that will be shown.
///
#endregion
```

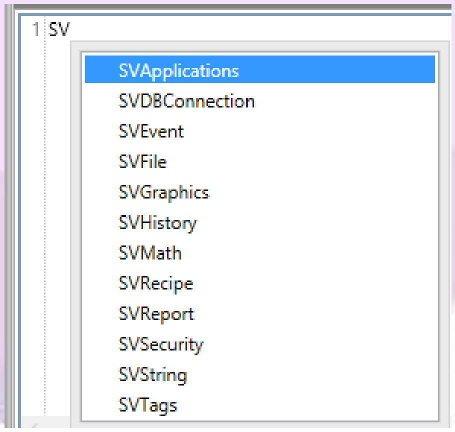
Each of the documents under “System Functions Library” contains the functions signature. The user will not be able to see the function’s implementation, but they will learn how to use them. The function signature shows the parameters expected for each of the functions and the function’s return type. If further function information is needed, please expand the regions (+) and there will be a summary for each of the functions and a summary for each of the parameters. Additionally, the Help file can be checked for further information and examples.

The next chapter will provide some SVFunctions usage examples:

# 4. Intellisense

The Intellisense is an ADISRA SmartView feature that autocompletes the script codes. It can be used for adding tags into different fields/scripts (when @ sign is written into a script, it shows all the application tags available) and it can also be used to add the System Function Libraries.

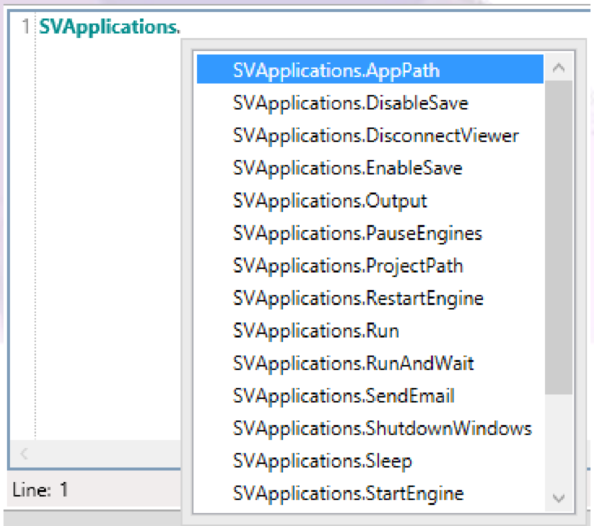
All the System Function Libraries starts with an SV prefix, so as soon as the user types "SV" in the script box, all the libraries will be listed and can be selected.



```
1 SV
```

- SVApplications
- SVDBConnection
- SVEvent
- SVFile
- SVGraphics
- SVHistory
- SVMath
- SVRecipe
- SVReport
- SVSecurity
- SVString
- SVTags

After selecting the library (i.e. SVApplication), the user will have access to the functions after typing a period (.).



```
1 SVApplications.
```

- SVApplications.AppPath
- SVApplications.DisableSave
- SVApplications.DisconnectViewer
- SVApplications.EnableSave
- SVApplications.Output
- SVApplications.PauseEngines
- SVApplications.ProjectPath
- SVApplications.RestartEngine
- SVApplications.Run
- SVApplications.RunAndWait
- SVApplications.SendEmail
- SVApplications.ShutdownWindows
- SVApplications.Sleep
- SVApplications.StartEngine

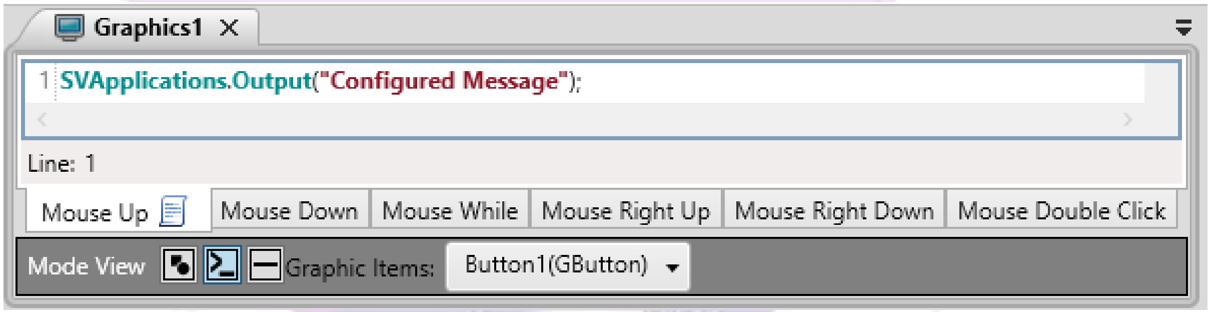
# 5. SVFunctions - Examples

## 5.1. Output

How to use a function to log a message in the “Engineering Log” will be described in this example.

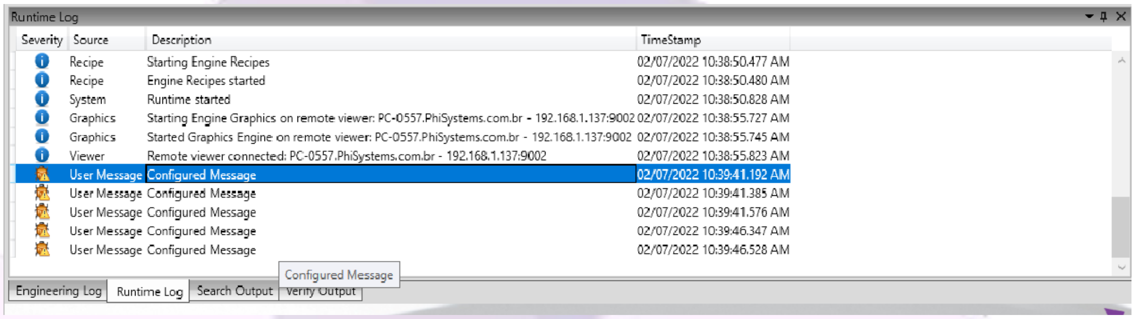
**SVApplications.Output()**

Configure the “mouse up” script in a button object as shown in the image below:



```
SVApplications.Output("Configured Message");
```

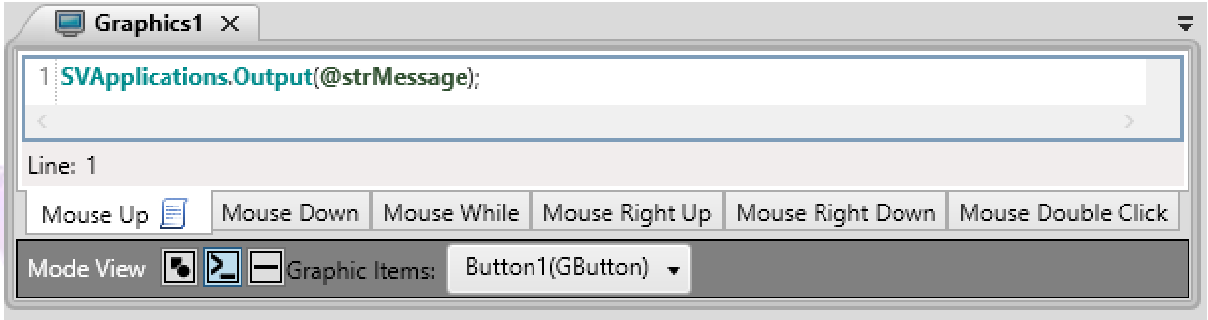
Every time the button is pressed during runtime it will log the message in the “Runtime Log” as shown in the image below:



| Severity | Source | Description | TimeStamp |
|---|---|---|---|
| ⓘ | Recipe | Starting Engine Recipes | 02/07/2022 10:38:50.477 AM |
| ⓘ | Recipe | Engine Recipes started | 02/07/2022 10:38:50.480 AM |
| ⓘ | System | Runtime started | 02/07/2022 10:38:50.828 AM |
| ⓘ | Graphics | Starting Engine Graphics on remote viewer: PC-0557.PhiSystems.com.br - 192.168.1.137:9002 | 02/07/2022 10:38:55.727 AM |
| ⓘ | Graphics | Started Graphics Engine on remote viewer: PC-0557.PhiSystems.com.br - 192.168.1.137:9002 | 02/07/2022 10:38:55.745 AM |
| ⓘ | Viewer | Remote viewer connected: PC-0557.PhiSystems.com.br - 192.168.1.137:9002 | 02/07/2022 10:38:55.823 AM |
| ⚠ | User Message | Configured Message | 02/07/2022 10:39:41.192 AM |
| ⚠ | User Message | Configured Message | 02/07/2022 10:39:41.385 AM |
| ⚠ | User Message | Configured Message | 02/07/2022 10:39:41.576 AM |
| ⚠ | User Message | Configured Message | 02/07/2022 10:39:46.347 AM |
| ⚠ | User Message | Configured Message | 02/07/2022 10:39:46.528 AM |

(If the message does not appear, right click the “Runtime Log” area and change the Log Level to debug)

It is also possible to pass a tag as parameter to the SVApplication.Output() function and the result will be a new output log showing the value of the tag.



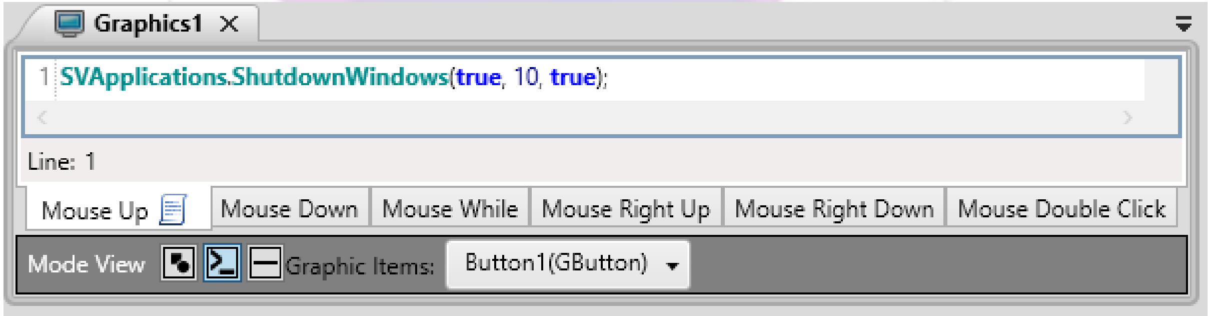
```
SVApplications.Output(@strMessage);
```

## 5.2. Shutdown Windows

How to use a function to shut down or restart Windows will be explained in this example.

**SVApplications.ShutdownWindows()**

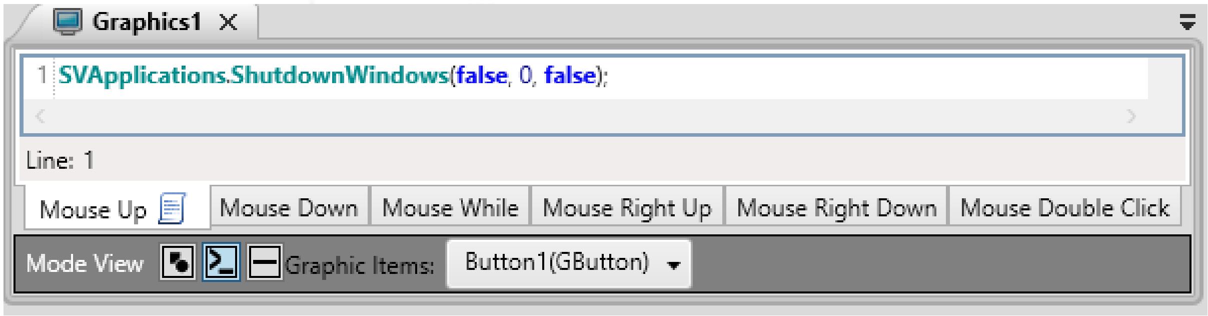
Configure the “mouse up” script in a button object as shown in the image below:



```
SVApplications.ShutdownWindows(true, 10, true);
```

In this case, the first parameter “true” indicates that the machine will shut down, the second parameter “10” indicates that the shutdown will occur 10 seconds after the button is pressed, and the last parameter “true” indicates that the shutdown will be forced.

See another example below:



```
SVApplications.ShutdownWindows(false, 0, false);
```

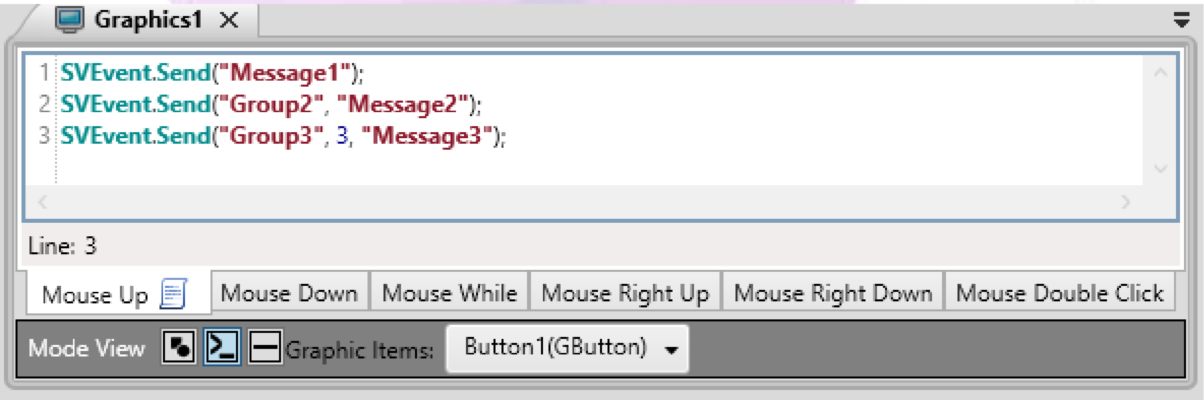
In this case, the first parameter "false" indicates that the machine will restart, the second parameter 0 indicates that the restart will occur instantly after the button is pressed, and the last parameter "false" indicates that the restart will not be forced.

## 5.3. Send

How to use a function to trigger an event will be described in this example.

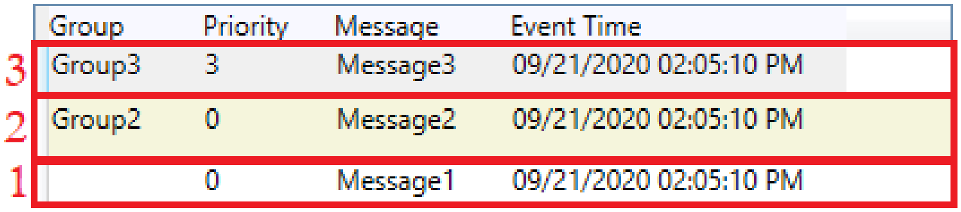
**SVEvent.Send()**

Configure the "mouse up" script in a button object as shown in the image below:



```
SVEvent.Send("Message1");
SVEvent.Send("Group2", "Message2");
SVEvent.Send("Group3", 3, "Message3");
```

The example below shows all the ways to trigger an event. It is important to understand that the first line of the script triggers the event, as seen in line 1 below with the message "Message1", the default priority, and no group. The second line of the script triggers event 2, with the message "Message2", the default priority, and "Group2". The third line of the script triggers event 3, with the message "Message3", priority "3" and "Group3".



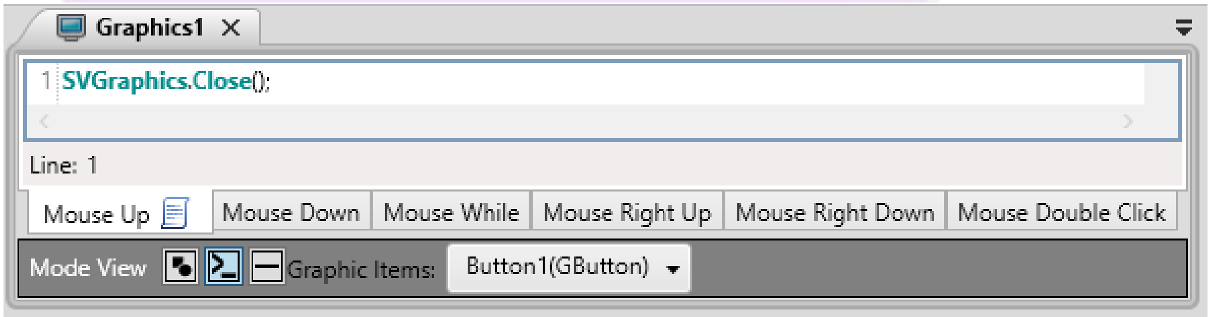
| | Group | Priority | Message | Event Time |
|---|---|---|---|---|
| 3 | Group3 | 3 | Message3 | 09/21/2020 02:05:10 PM |
| 2 | Group2 | 0 | Message2 | 09/21/2020 02:05:10 PM |
| 1 | | 0 | Message1 | 09/21/2020 02:05:10 PM |

## 5.4. Close

How to use a function to close a graphic will be explained in this example.

**SVGraphics.Close()**

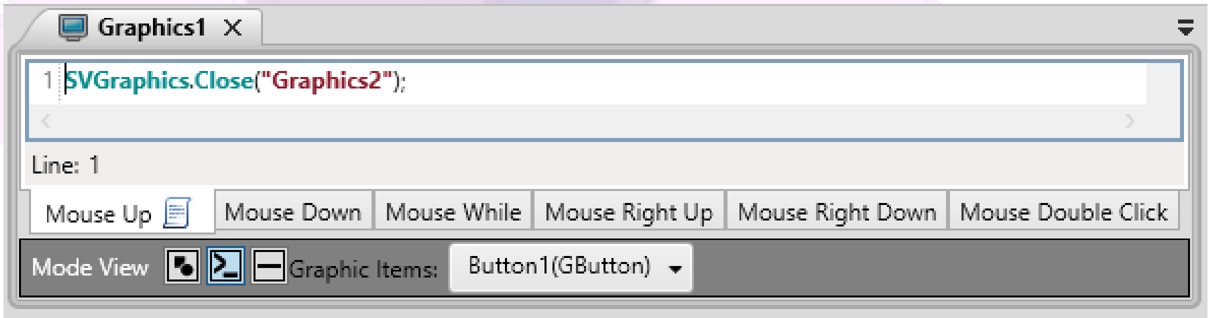
Configure the “mouse up” script in a button object as shown in the image below:



```
SVGraphics.Close();
```

In this example, the button will close current graphic document. If the document is a parent document with children documents opened, all graphics will be closed.

There is an alternative to close graphics. A specific graphic can also be closed if the script passes the graphic’s name as parameter to the SVGraphics.Close(). The image below shows a script closing “Graphics2” screen. If the Graphics2 document is a parent document with children documents opened, all graphics will be closed.



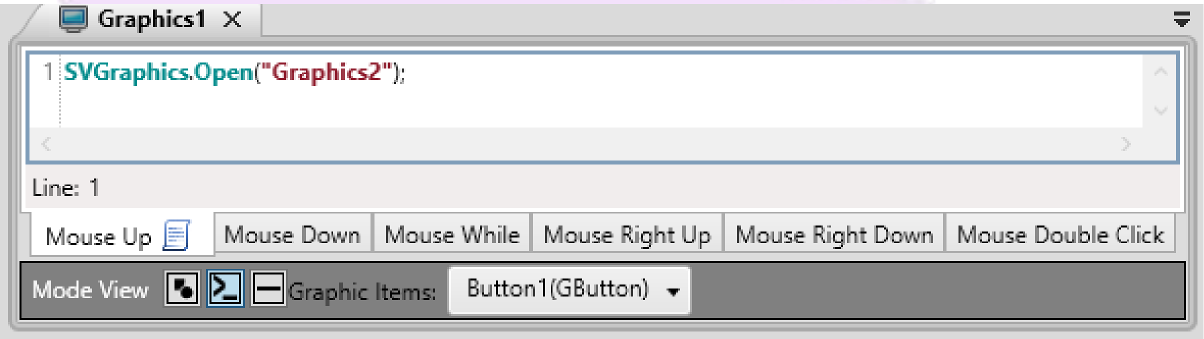
```
SVGraphics.Close("Graphics2");
```

## 5.5. Open

How to use a function to open a graphic will be explained in this example.

**SVGraphics.Open()**

Configure the “mouse up” script in a button object as shown in the image below:



```
SVGraphics.Open("Graphics2");
```

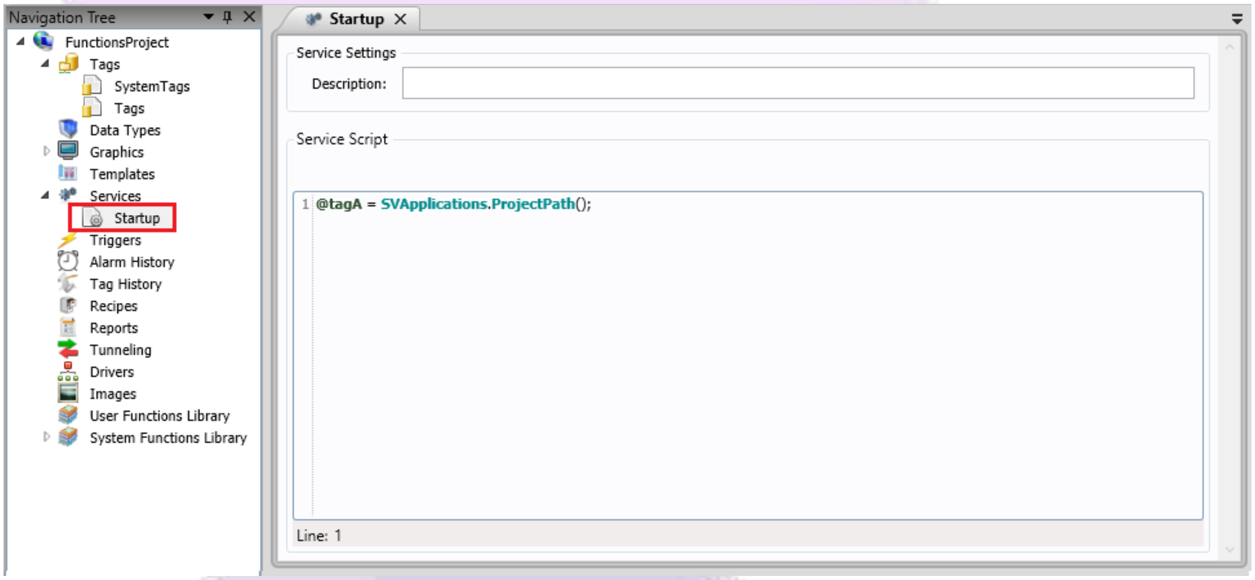
In this example, the button will open the graphic document “Graphics2”.

## 5.6. ProjectPath

How to use a function that returns a string value containing the path of the project will be explained in this example.

**SVApplications.ProjectPath()**

Configure the script for the "Startup" Service document as show in the image below:



```
@tagA = SVApplications.ProjectPath();
```

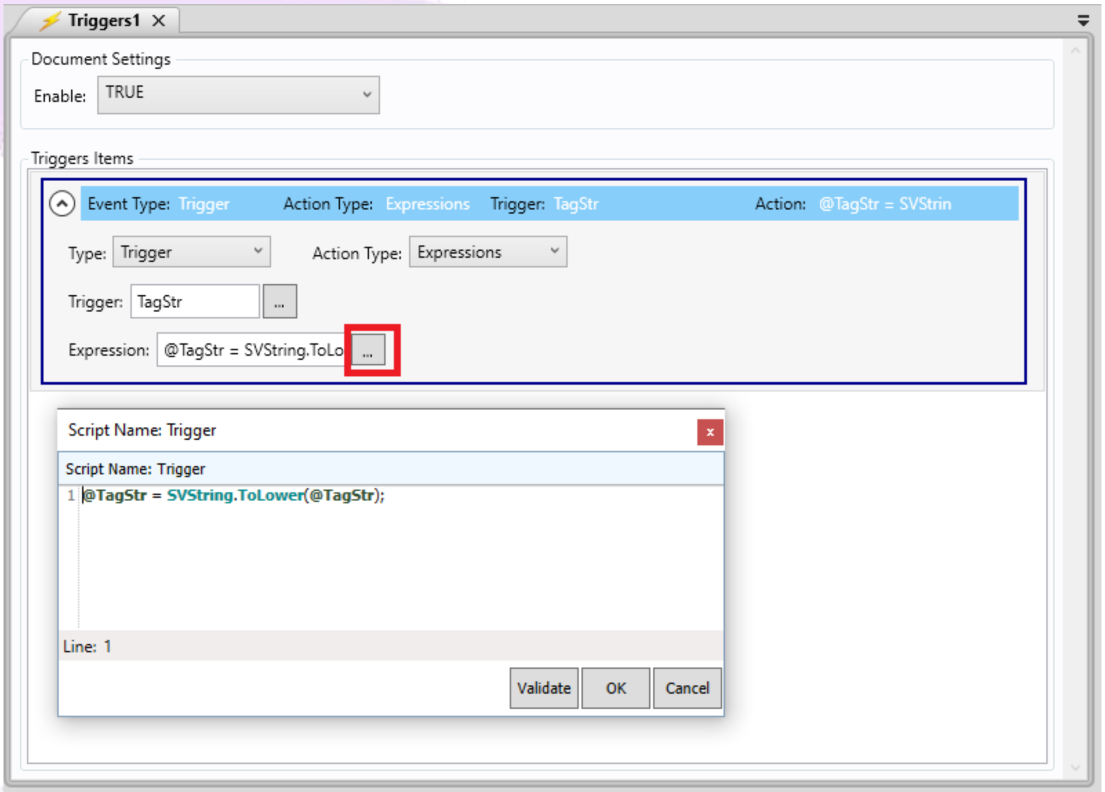
Every time the application starts, the script in the "Startup" service will execute once. In the example above, the tagA will be initialized with the SVApplication.ProjectPath().

## 5.7. ToLower

This function modifies the string parameter to lower case.

**SVString.ToLower()**

Configure a trigger document as show in the image below:



Enable: TRUE

Type: Trigger

Action Type: Expressions

Trigger: TagStr

Expression: @TagStr = SVString.ToLo

```
@TagStr = SVString.ToLower(@TagStr);
```

Every time the tag "TagStr" changes its value it will execute the expression configured. In this example, it will transform the value of "TagStr" to lower case.

ADISRA®, InsightView™, and KnowledgeView™ are the registered trademarks of ADISRA, LLC.

© 2022 ADISRA, LLC. All Rights Reserved.