

0 - Introduction	2
1 - Preparations	7
2 - Setup GSE Actions	9
3.1 - Usage - Create Ticket	16
3.2 - Usage - Create Note	23
3.3 - Usage - Check Status	29
3.4 - Usage - Trouble Shooting	38
A.1 - Example - Create Ticket	41
A.2 - Example - Create Note	44
A.3 - Example - Check Status	47
A.4 - Example - Check Status and Create Note	51
A.5 - Example - Check and Announce Status	55
B.1 - Access modify the code behind	60

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

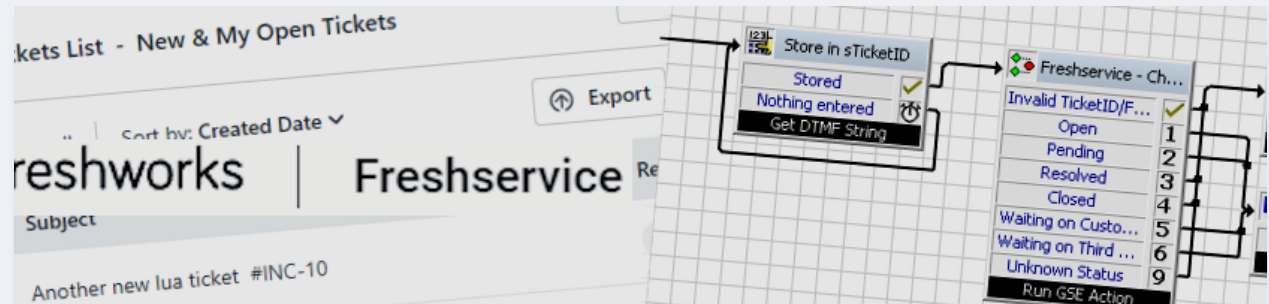
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 0 - Introduction

Followers

0

VBScript

Lua

This extension provides **Freshservice** ticket functionality for the SwyxWare call routing:

- **Create** a new Freshservice ticket
 - **Add Note** to an existing Freshservice ticket
 - **Check** the current **status** of a Freshservice ticket
-
- Comes for **VBscript based** and **Lua based** call routing
 - Requires **SwyxWare 12.40** (or newer) for **VBScript based** call routing
 - Requires **SwyxWare 13.10** (or newer) for **Lua based** call routing

To handle Freshservice tickets it makes use of the Freshservice REST API.

APPENDIX B

B.1 - Access/modify the code behind



Please refer to the [Forums](#) to discuss the Freshservice Integration or for support requests.



Please find the download for this project [here](#).



For the complete documentation explaining the setup, usage and all included examples just read the following chapters from the menu on the left.

As with all other Swyx Forum Open Source Projects, Support is **EXCLUSIVELY** provided in the Project Froum (see link above).

Lua based call routing has been introduced to SwyxWare from **13.10** on. As of the release of the Freshservice Integration the Lua based call routing is still in **BETA state** and should not be used in productive environments. However, this project already comes also for **Lua based** call routing as an example of its ease of use.

License

Freshservice Integration
v1.0.0

This is a Swyx Forum Open Source Project.

<https://www.swyxforum.com/projects/>

<https://www.swyxforum.com/freshservice-integration/introduction/0-introduction-r1/>

The MIT License (MIT)
Copyright (c) 2024 by Swyx Forum
Copyright (c) 2024 by Tom Wellige
All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice must be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

For the VBScript based integration this project also includes the following Open Source project:

JSON object class 3.5.4 - May, 29th - 2016

Licence:
The MIT License (MIT)
Copyright (c) 2016 RCDMK - rcdmk[at]hotmail[dot]com

<https://github.com/rcdmk/aspJSON>

<https://github.com/rcdmk/aspJSON/blob/master/README.md>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Modifications needed for SwyxWare Call Routing

Tom Wellige, 03.05.2017

For the Lua based integration this project also includes the following Open Source project:

Simple JSON encoding and decoding in pure Lua.

Copyright 2010-2016 Jeffrey Friedl

<http://regex.info/blog/>

Latest version: <http://regex.info/blog/lua/json>

This code is released under a Creative Commons CC-BY "Attribution" License:

http://creativecommons.org/licenses/by/3.0/deed.en_US

It can be used for any purpose so long as:

- 1) the copyright notice above is maintained
- 2) the web-page links above are maintained
- 3) the 'AUTHOR_NOTE' string below is maintained



By Tom Wellige

September 22, 2024

 Share

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige

Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

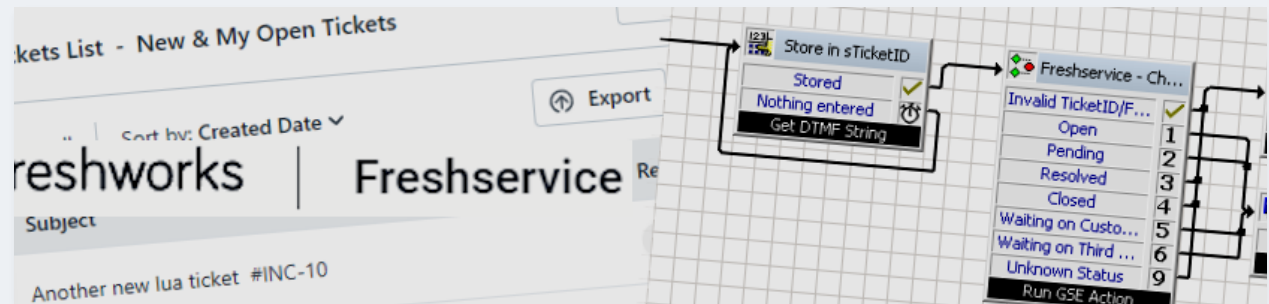
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 1 - Preparations

Followers

0

VBScript

Lua

[Download](#) the latest version of this project.

To get access to the **Freshservice Support** portal via its **REST API** some means of authentication are required. Freshservice offers **API Key** authentication.

You need to create such an **API Key** following these steps:

1. Login to your **Freshservice Support** portal
2. Click on your **Profile Picture** on the top right corner of your portal
3. Go to **Profile Settings** page

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind

4. Your **API Key** will be available below the change password section to your right



By Tom Wellige
September 22, 2024

 Share

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

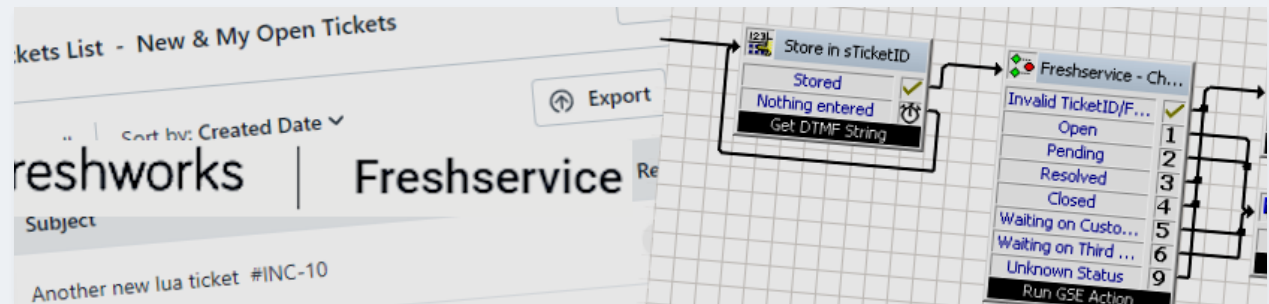
2 - Setup GSE Actions

USAGE

- 3.1 - Create Ticket
- 3.2 - Create Note
- 3.3 - Check Status
- 3.4 - Trouble Shooting

APPENDIX A

- A.1 - Example: Create Ticket
- A.2 - Example: Create Note
- A.3 - Example: Check Status
- A.4 - Example: Check Status and Create Note



Freshservice Integration - 2 - Setup GSE Actions

Followers 0

VBScript

Lua

The **Freshservice Integration** extension is designed as a set of GSE actions. To install the GSE actions you need to use the **SwyxWare Administration**.

Step 1

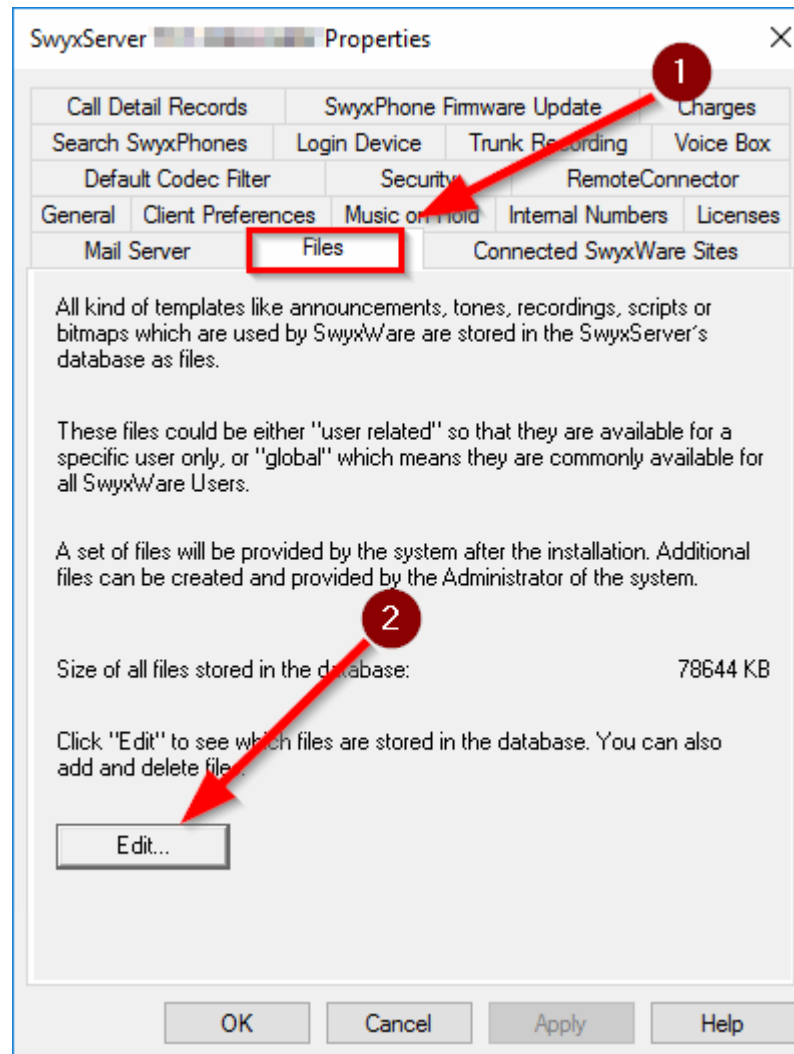
Open the SwyxWare Administration and open the properties of your Swyx Server.

Step 2

Switch to the **Files** page and click on **Edit....**

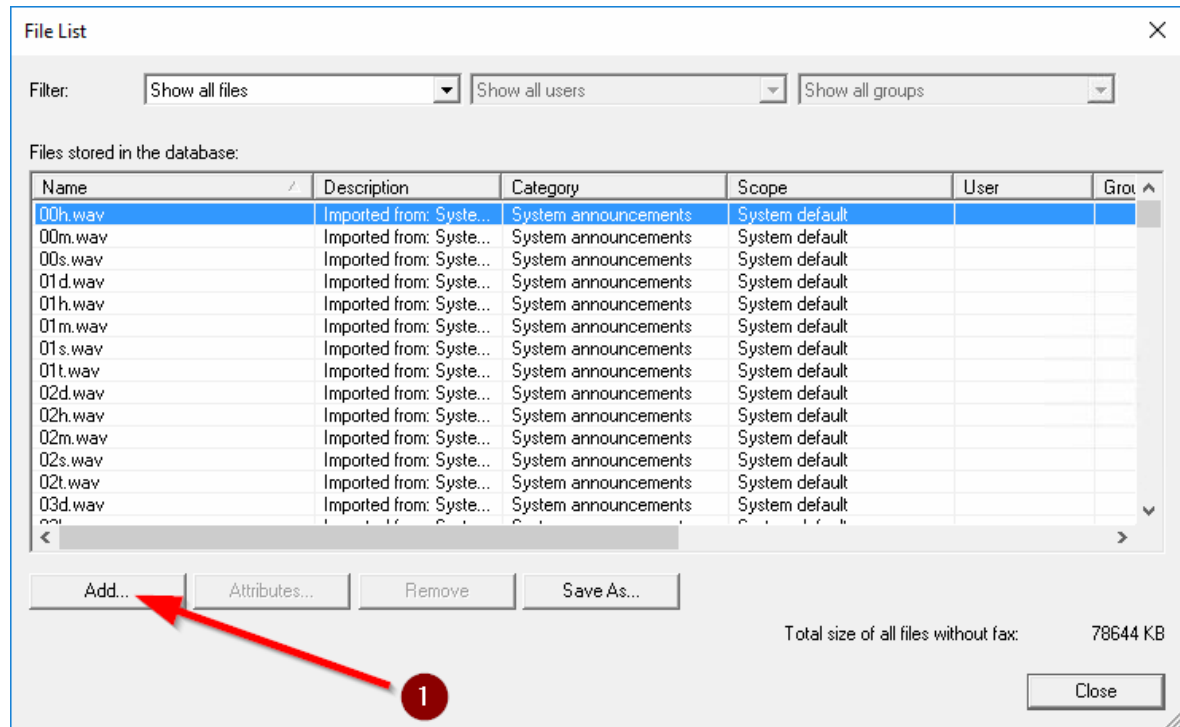
APPENDIX B

B.1 - Access/modify the code behind



Step 3

Click on **Add...**



Step 4.1 - VBScript usage

1. Select all files from the **VBScript based\ase** folder from the download package.
2. Select **Global** as **Scope** if you want all users to have access to this integration. You can also select **User** to load the files into the **User Scope** a your script user.
3. Select **Call Routing VBS scripts** as **Category**.
4. It is recommended to enter **Freshservice Integration** into the **Description** field, but not necessary.

The screenshot shows a 'Add File to Database' dialog box with the following fields and annotations:

- File to add:** A text field containing 'C:\Projects\FreshserviceIntegration\VBScript ba' with a red circle '1' over the file path.
- Name:** A text field containing 'actionFreshserviceCheckStatus.ase;'.
- Scope:** A dropdown menu with 'Global' selected, with a red circle '2' over the dropdown.
- Category:** A dropdown menu with 'Call Routing VBS scripts' selected, with a red circle '3' over the dropdown.
- User:** A dropdown menu with 'Botsquad' selected.
- Group:** A dropdown menu with 'Everyone' selected.
- File Properties:** A section with three checkboxes: 'Private', 'Hidden', and 'System', all of which are unchecked.
- Description:** A text field containing 'Freshservice Integration' with a red circle '4' over the text.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Step 4.2 - Lua usage

1. Select all files from the **Lua based\ase** folder from the download package.
2. Select **Global** as **Scope** if you want all users to access to this integration. You can also select **User** to load the files into the **User Scope** a your script user.
3. Select **Call Routing Lua scripts** as **Category**.
4. It is recommended to enter **Freshservice Integration** into the **Description** field, but not necessary.

The screenshot shows a Windows-style dialog box titled "Add File to Database" with a close button (X) in the top right corner. The dialog contains several input fields and dropdown menus, with four red circular callouts numbered 1 through 4 pointing to specific elements:

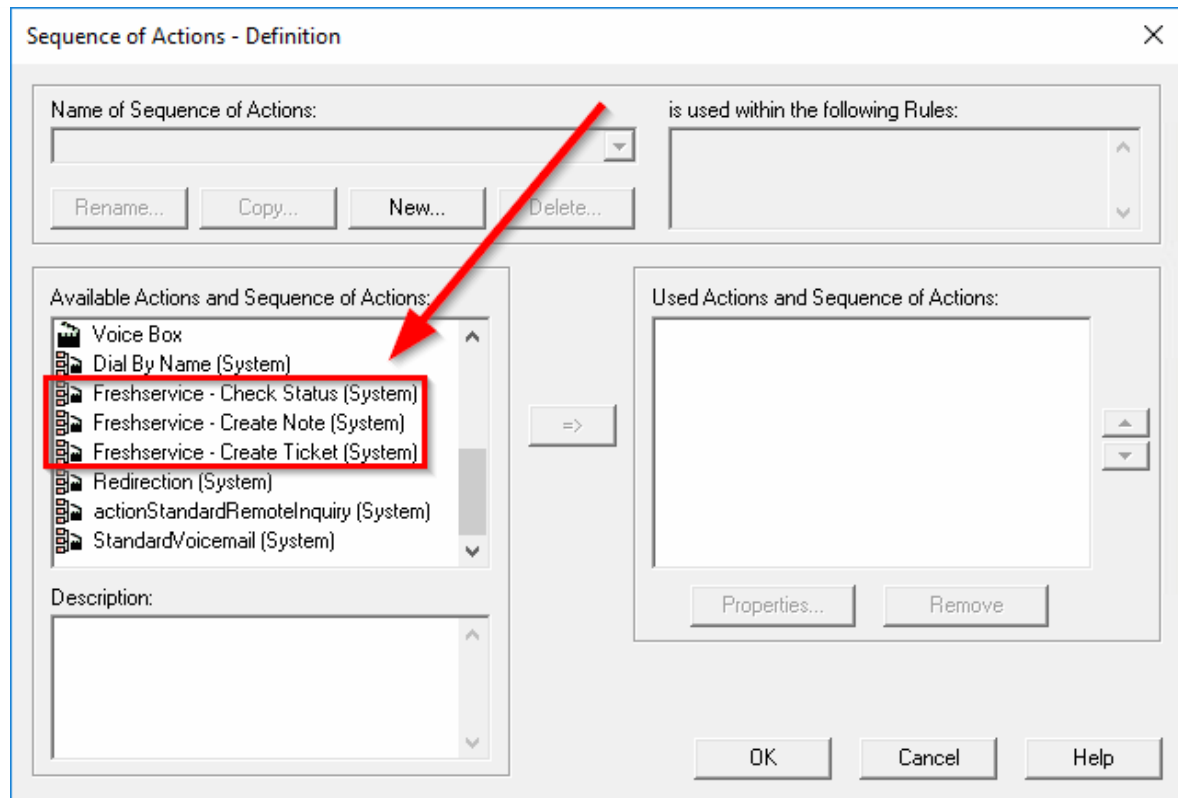
- Callout 1:** Points to the "File to add:" text box, which contains the path "C:\Projects\FreshserviceIntegration\Lua based\..." and a browse button ("...").
- Callout 2:** Points to the "Scope:" dropdown menu, which is set to "Global".
- Callout 3:** Points to the "Category:" dropdown menu, which is set to "Call Routing Lua scripts".
- Callout 4:** Points to the "Description:" text box, which contains the text "Freshservice Integration".

Other fields in the dialog include:

- Name:** "actionFreshserviceCheckStatus.ase;"
- User:** "Botsquad"
- Group:** "Everyone"
- File Properties:** A section with three unchecked checkboxes: "Private", "Hidden", and "System".
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Step 5

To check if the GSE actions are available within call routing scripts open the **Call Routing Manager** of any user and click the button **Sequence of Actions**. Scroll the list on the left side down until you reach the **Freshservice...** actions. The **(System)** behind the GSE action name shows that this is a global action being available for every SwyxWare user.



The Freshservice Integration actions are now installed and can be fully used within GSE call routing scripts.



By Tom Wellige
September 22, 2024

 Share

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige

Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

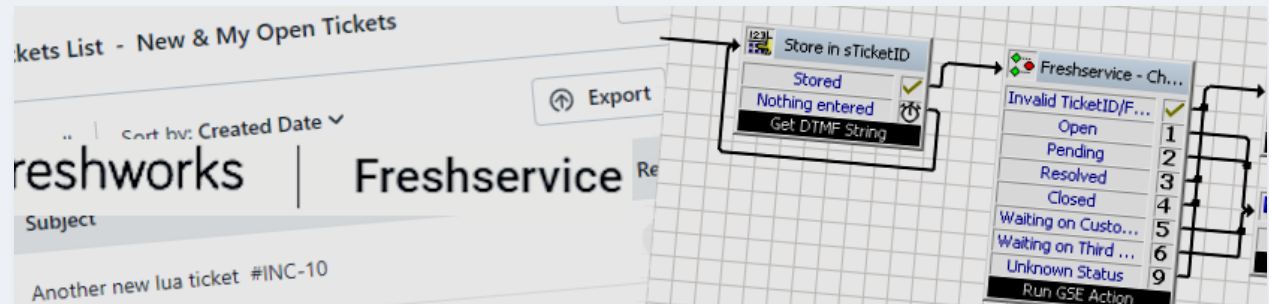
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 3.1 - Create Ticket

Followers

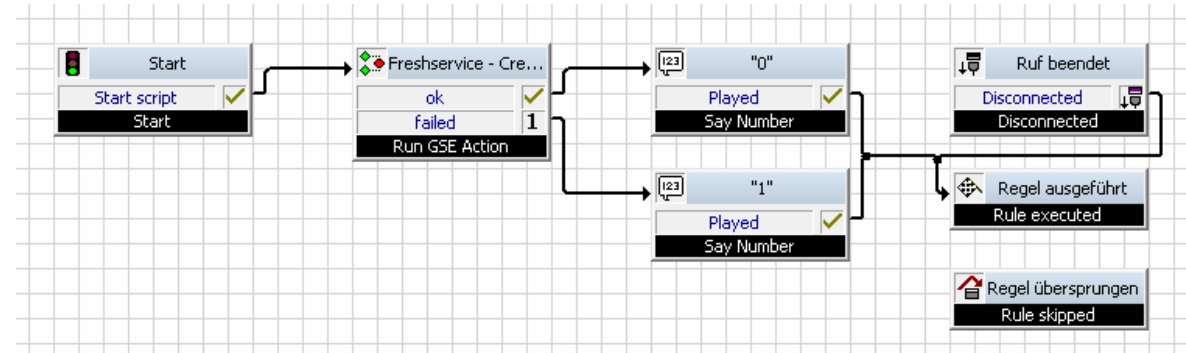
0

VBScript

Lua

This GSE action creates a new ticket. A common usage of this is to register a callback request.

An example call routing script can be found in [A.1 - Example: Create Ticket](#).



Run GSE Action Properties

General Parameters Links

Select GSE action: Freshservice - Create Ticket

Set action parameters:

Name	Value	Default Value
Domain	= ""	= ""
APIKey	= ""	= ""
ContactEmail	= ""	= ""
Subject	= ""	= ""
Description	= ""	= ""
Source	3	3

Edit Parameter...

Description:

Freshservice Integration v1.0.0
Creates a new Ticket..

Return Values:

0 - ok
1 - failed

OK Cancel Help

By double clicking a parameter in the list, you can edit it.

Domain Required

This is the name of your subdomain in the Freshservice URL of your support portal, for example: https://**domain**.freshservice.com

This is a string value.

APIKey Required

This is the API Key you obtained from your Freshservice Support portal. This is a string value. Please refer to [1 - Preparations](#) for more details.

ContactEmail Required

This is the email address of the ticket requester. This is a string value.
Freshservice will connect a contact linked to this email address to the new created ticket.

Subject Required

This is the subject or title of the new ticket. This is a string value.

Description Required

This is the description of the new issue. This is a string value.

Source

This is the source the new ticket was requested from. This is a number value.
Freshservice knows the following ticket sources:

Email	1
Portal	2
Phone	3 (Default)
Chat	7
Feedback Widget	9
Outbound Email	10

Status

This is the status for the new ticket. This is a number value.
Freshservice knows the following ticket states:

Open	2 (Default)
Pending	3
Resolved	4

Closed 5

Priority

This is the priority for the new ticket. This is a number value.

Freshservice knows the following ticket priorities:

Low 1

Medium 2 (Default)

High 3

Urgent 4

Configure action exits

Run GSE Action Properties

General Parameters Links

Visible	Fixed name	Link name	Linked to
☒	Default	ok	"0"
☒	Return code 1	failed	"1"
☐	Return code 2		↓ [no link]
☐	Return code 3		↓ [no link]
☐	Return code 4		↓ [no link]
☐	Return code 5		↓ [no link]
☐	Return code 6		↓ [no link]
☐	Return code 7		↓ [no link]
☐	Return code 8		↓ [no link]
☐	Return code 9		↓ [no link]
☐	Disconnected		Ruf beendet

OK Cancel Help

Exit 0 (Default)

This exit will be reached when everything worked fine and the ticket was created. It is recommended to name this exit "**ok**" or "**created**".

Exit 1

This exit will be reached when there was any kind of problem and no ticket has been created. It is recommended to name this exit "**failed**".

If you reach this exit you can refer to [3.4 - Trouble Shooting](#) to figure what went wrong.

Additional return value (as global variable)

g_sLatestFreshserviceTicketID (string)

This global variable holds ID of the newly created ticket after the **ok** (0) exit has been reached.



By Tom Wellige
September 22, 2024

 Share

Next Page 
[3.2 - Create Note](#)

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

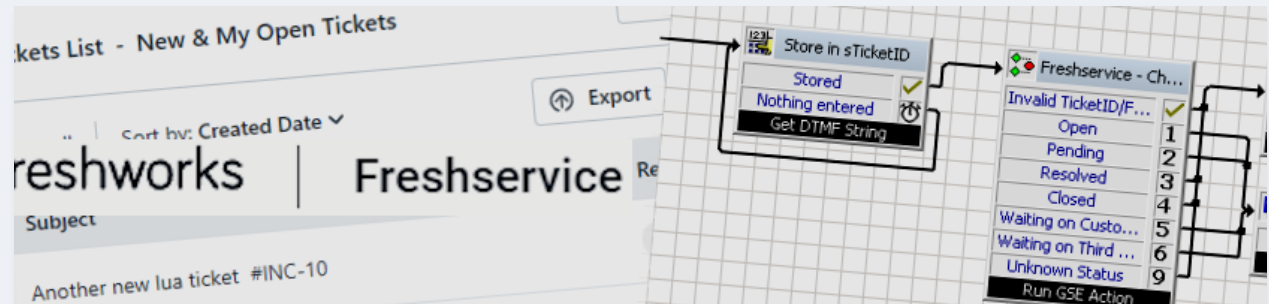
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 3.2 - Create Note

Followers

0

VBScript

Lua

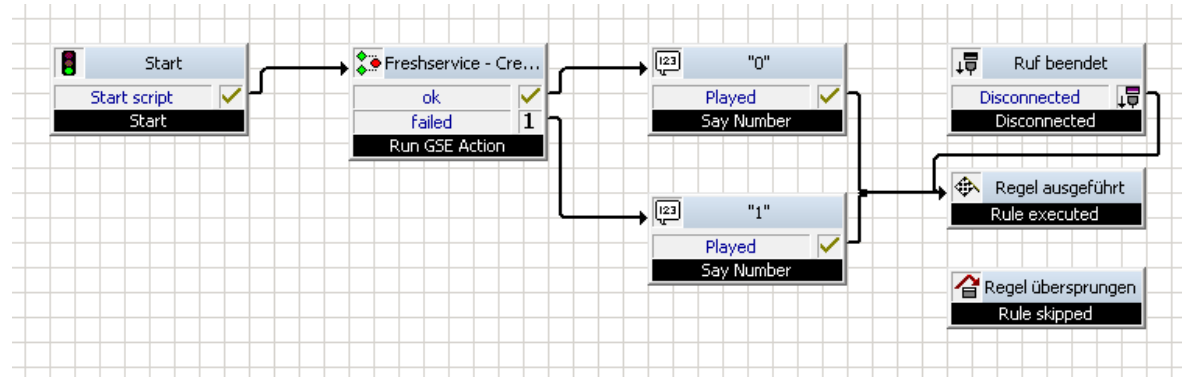
This GSE action creates a new note/comment to an existing ticket.

Example call routing scripts can be found here [A.2 - Example: Create Note](#) and here [A.4 - Example: Check Status and Create Note](#).

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind



Configure action parameters

Run GSE Action Properties

General Parameters Links

Select GSE action: Freshservice - Create Note

Set action parameters:

Name	Value	Default Value
Domain	= ""	= ""
APIKey	= ""	= ""
TicketID	= ""	= ""
Body	= ""	= ""
Private	1	1

Edit Parameter...

Description:

Freshservice Integration v1.0.0
Creates a note into an existing Ticket.

Return Values:

0 - ok
1 - failed

OK Cancel Help

By double clicking a parameter in the list, you can edit it.

Domain Required

This is the name of your subdomain in the Freshservice URL of your support portal, for example: <https://domain.freshservice.com>

This is a string value.

APIKey Required

This is the API Key you obtained from your Freshservice Support portal. This is a string value. Please refer to [1 - Preparations](#) for more details.

TicketID Required

This is the id of the ticket you want to add a note/comment to.
This is a string value.

Body Required

This is the text of the note/comment.
This is a string value.

Private

You can define, if the new note/comment will only be visible internally (1) or is also visible to your customer (0).
Valid values or 0 and 1. Default: 1
This is a number value.

Configure action exits

Run GSE Action Properties

General Parameters Links

Visible	Fixed name	Link name	Linked to
☒	Default	ok	"0"
☒	Return code 1	failed	"1"
☐	Return code 2		↓ [no link]
☐	Return code 3		↓ [no link]
☐	Return code 4		↓ [no link]
☐	Return code 5		↓ [no link]
☐	Return code 6		↓ [no link]
☐	Return code 7		↓ [no link]
☐	Return code 8		↓ [no link]
☐	Return code 9		↓ [no link]
☐	Disconnected		Ruf beendet

OK Cancel Help

Exit 0 (Default)

This exit will be reached when everything worked fine and the note/comment was created. It is recommended to name this exit "**ok**" or "**created**".

Exit 1

This exit will be reached when there was any kind of problem and no note/comment has been created. It is recommended to name this exit "**failed**".

If you reach this exit you can refer to [3.4 - Trouble Shooting](#) to figure what went wrong.

Additional return value (as global variable)

g_sLatestFreshserviceTicketID (string)

This global variable holds the ID of the ticket the note/comment was added to. after the **ok** (0) exit has been reached.



By Tom Wellige
September 22, 2024

 Share

< Previous Page
3.1 - Create Ticket

Next Page >
3.3 - Check Status

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

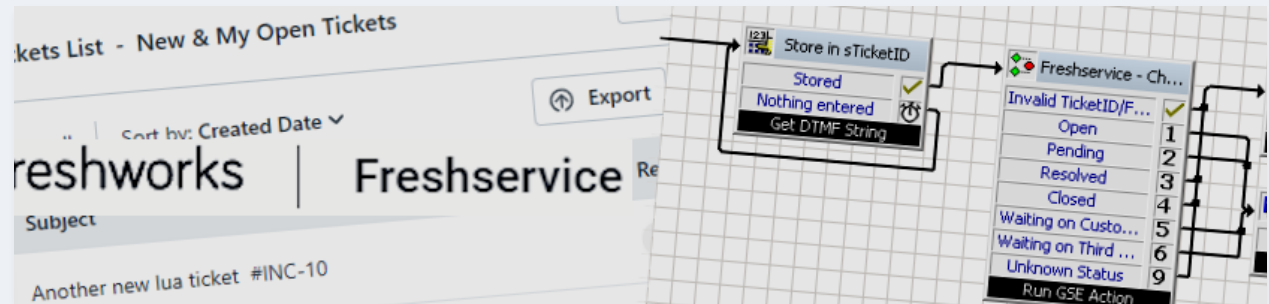
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 3.3 - Check Status

Followers

0

VBScript

Lua

This GSE action checks the status of an existing ticket.

Example call routing scripts can be found here [A.3 - Example: Check Status](#), here [A.4 - Example Check Status and Create Note](#), and here [A.5 - Example: Check and Announce Status](#).

Configure action parameters

Run GSE Action Properties

General Parameters Links

Select GSE action: Freshservice - Check Status

Set action parameters:

Name	Value	Default Value
Domain	= ""	= ""
APIKey	= ""	= ""
TicketID	= ""	= ""

Edit Parameter...

Description:

Freshservice Integration v1.0.0
Checks status of an existing Ticket.

Return Values:

0 - ok
1 - failed

OK Cancel Help

By double clicking a parameter in the list, you can edit it.

Domain Required

This is the name of your subdomain in the Freshservice URL of your support portal, for example: <https://domain.freshservice.com>

This is a string value.

APIKey Required

This is the API Key you obtained from your Freshservice Support portal. This is a string value. Please refer to [1 - Preparations](#) for more details.

TicketID Required

This is the id of the ticket you want to check the status of.
This is a string value.

Configure action exits

Run GSE Action Properties

General Parameters Links

Visible	Fixed name	Link name	Linked to
☒	Default	Invalid TicketID/Failed	"0"
☒	Return code 1	Open	"1"
☒	Return code 2	Pending	"2"
☒	Return code 3	Resolved	"3"
☒	Return code 4	Closed	"4"
☒	Return code 5	Waiting on Customer	"5"
☒	Return code 6	Waiting on Third Party	"6"
☐	Return code 7		[no link]
☐	Return code 8		[no link]
☒	Return code 9	Unknown Status	"9"
☐	Disconnected		Ruf beendet

OK Cancel Help

Exit 0 (Default)

This exit will be reached if an invalid issue id was given or any kind of problem occurred. It is recommended to name this exit "**Invalid ID/Failed**".

If you reach this exit you can refer to [3.4 - Trouble Shooting](#) to figure what went wrong.

Exit 9

This exit will be reached if an unknown status was returned by the Freshservice REST API. It is recommended to name this exit "**Unknown Status**".

If you reach this exit you can refer to [3.4 - Trouble Shooting](#) to figure the returned status and modify your status mapping.

Dynamic Status Mapping

This GSE action provides a flexible mapping of Fresdesk ticket status values to exits of the block. All block exits **1 to 8** are of your free choosing.

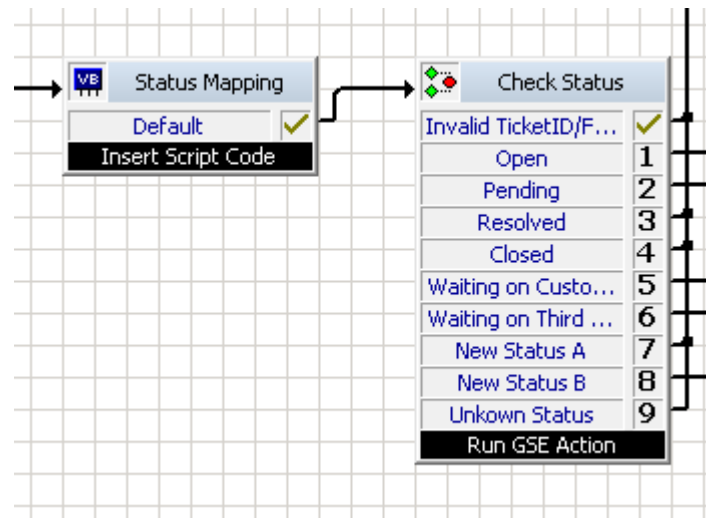
By default this GSE action maps the status values to exits as follows. If nothing changes on the Freshservice side, you don't need to do any modifications. If however FreshService makes changes to their status values or you want another mapping of status values to the exit, you can freely do so. This is the default mapping:

Exit States

- | | |
|---|------------------------|
| 1 | Open |
| 2 | Pending |
| 3 | Resolved |
| 4 | Closed |
| 5 | Waiting on Customer |
| 6 | Waiting on Third Party |

As already mentioned, you can modify this mapping to your precise needs. Just keep in mind that only the exits **1 to 8** are available to you.

All you need to do is to modify a global variable before you use the **Check Status** GSE action. This can best be done in a **Insert Script Code** block.



The content of the **Insert Script Code** block is then the definition of the mapping list, which is either a **VBScript Array** or a **Lua Table**. In both cases you are free to map any number of states to any exit (1..8).

Lets assume you want to map new states **New State A (Freshservice status value 8)** to exit **7** and **New State B (Freshservice status 9)** to exit **8**. This is the code you need to place into the **Insert Script Code** block:

For **VBScript** based call routing:

```
UseExit = 0

g_aFreshserviceStatusMapping = Array( _
    "1;2;Open", _
    "2;3;Pending", _
```

```
"3;4;Resolved", _  
"4;5;Closed", _  
"5;6;Waiting on Customer", _  
"6;7;Waiting on Third Party", _  
"7;8;New Status A", _  
"8;9;New Status B")
```

For **Lua** based call routing:

```
UseExit = 0  
  
g_aFreshserviceStatusMapping = {  
    "1;2;Open",  
    "2;3;Pending",  
    "3;4;Resolved",  
    "4;5;Closed",  
    "5;6;Waiting on Customer",  
    "6;7;Waiting on Third Party",  
    "7;8;New Status A",  
    "8;9;New Status B"  
}
```

As you can see, you are absolutely free in terms of the mapping of the different possible **states** of your service issue to an **exit** of the **Run GSE Action** block.

All you need to do is to do your mapping in the **Insert Script Code** block and afterwards open and name the **exits** of the **Run GSE Action block** accordingly.

An example of some own status mapping can be found in [A.5 - Example: Check and Announce Status](#).

Additional return values (as global variables)

g_sLatestFreshserviceTicketID (string)

This global variable holds the ID of the latest checked ticket after the **ok** (0) exit has been reached.

The [A.5 - Example: Check and Announce Status](#) makes use of this variable to announce the checked ticket id via [AzureTTS \(text-to-speech\)](#).

g_sLatestFreshserviceTicketStatus (string)

This global variable holds the status of the latest checked issue after the **ok** (0) exit has been reached.

This is a number value, as returned by the Freshservice REST API.

g_sLatestFreshserviceTicketStatusName (string)

This global variable holds the name of the status of the latest checked issue after the **ok** (0) exit has been reached.

The name is taken from the above explained **status mapping**, as the Freshservice REST API returns a number value only.

The [A.5 - Example: Check and Announce Status](#) makes use of this variable to announce the status via [AzureTTS \(text-to-speech\)](#).

g_sLatestFreshserviceTicketModified (string)

This global variable holds the latest modification date of the latest checked ticket after the **ok** (0) exit has been reached.

The date is as the Freshservice REST API provides it in UTC/GMT, e.g. "2024-09-20T13:02:03Z".

g_sLatestFreshserviceTicketModifiedReadable (string)

This global variable holds the latest modification date of the latest checked ticket after the **ok** (0) exit has been reached.

The date is in a better readable format and converted to local time, e.g. "20.09.2024 15:02:037".

The [A.5 - Example: Check and Announce Status](#) makes use of this variable to announce the latest modification date via [AzureTTS \(text-to-speech\)](#).



By Tom Wellige
September 22, 2024

 Share

< Previous Page
[3.2 - Create Note](#)

Next Page >
[3.4 - Trouble Shooting](#)

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

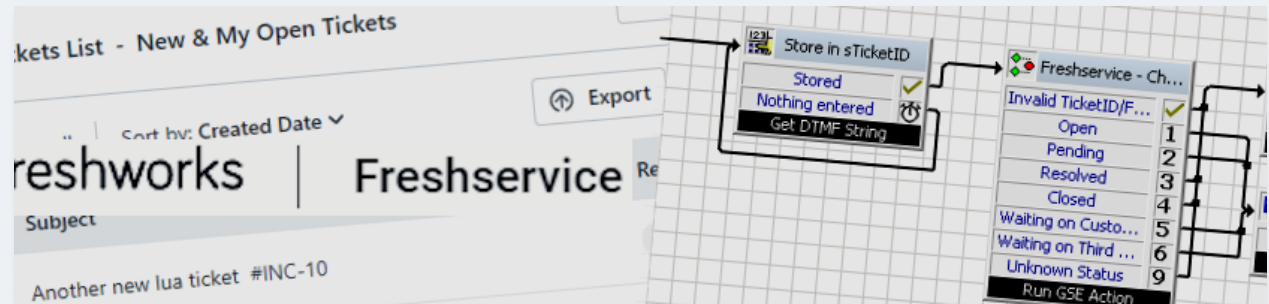
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - 3.4 - Trouble Shooting

Followers 0

VBScript

Lua

The Freshservice Integration extension writes meaningful trace information into the server trace file, according to the trace recommendation from this blog post:

- [Don't be shy, be chatty!](#)

So, to figure the exact reason for the problem you need to take a look into that file. The following link explains where to find it and how to filter its content down to the call in question and call routing output only:

- [How to filter SwyxWare traces for call routing output of single call](#)

APPENDIX B

B.1 - Access/modify the code behind

Once you have opened and filtered the trace file you need to search for the first trace lines written by the GSE action. Just look for these lines:

For **VBScript** based call routing:

```
-----> Freshservice.Class_Initialize
```

For **Lua** based call routing:

```
-----> Freshservice:Create
```

The following lines will contain information of all things happened, like setting the parameters and calling the Freshdesk REST API.

If at some point an error occurs, you will see an error message in the trace.

If the REST API call fails, for example because you entered an invalid API Key, you will see the **response body** of the Freshservice REST API in the trace which contains usually proper information on why the request failed.

You will also find the original **response code** of the web request. If everything went fine, the response code is **201**, otherwise its any other number (most likely ≥ 400), e.g.

```
nResponseCode = 401
```

By taking the detailed error information from the server trace file you should be able to fix your problem.

In case you can't get your problem fixed or have specific questions, please open your own topic in the [project forum](#).



By Tom Wellige

September 22, 2024

 Share



Previous Page

3.3 - Check Status

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige

Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

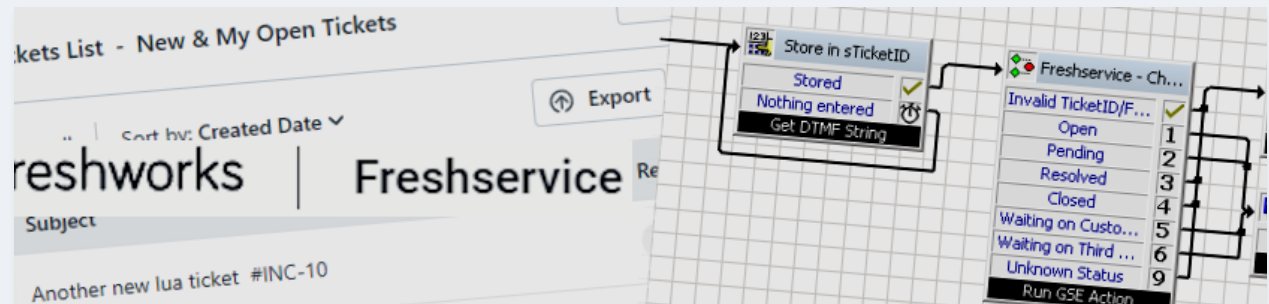
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - A.1 - Example: Create Ticket

Followers

0

VBScript

Lua

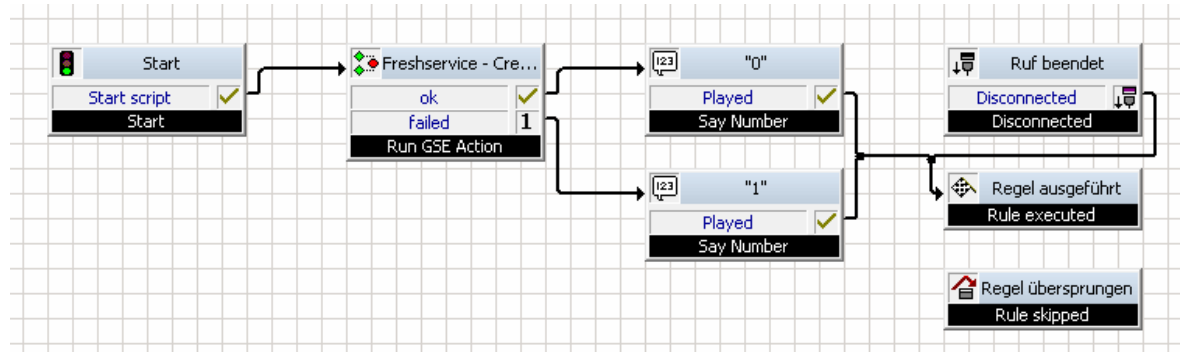
This example demonstrates the usage of the **FreshserviceIntegration - Create Issue** GSE action.

It creates a new ticket and announces afterwards "0" on success or "1" on failure.

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind



To install it:

1. Open the **Call Routing Manager** of your desired SwyxWare user.
2. Click the "New Rule..." button.
3. Select "**Graphical Script Editor**" and click **Ok**.
4. Within the GSE open the **File | Import...** menu and click **No**.
5. From the download package select the following file:

For **VBScript** usage:

```
VBScript based\rse\Create Ticket.rse
```

For **Lua** usage:

```
Lua based\rse\Create Ticket.rse
```

6. You need to make some modifications in the **FreshserviceIntegration - Create Ticket** (Run GSE Action) block . They are explained in detail in chapter [3.1 - Create Ticket](#).



By Tom Wellige
September 22, 2024

 Share

Next Page 
[A.2 - Example: Create Note](#)

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

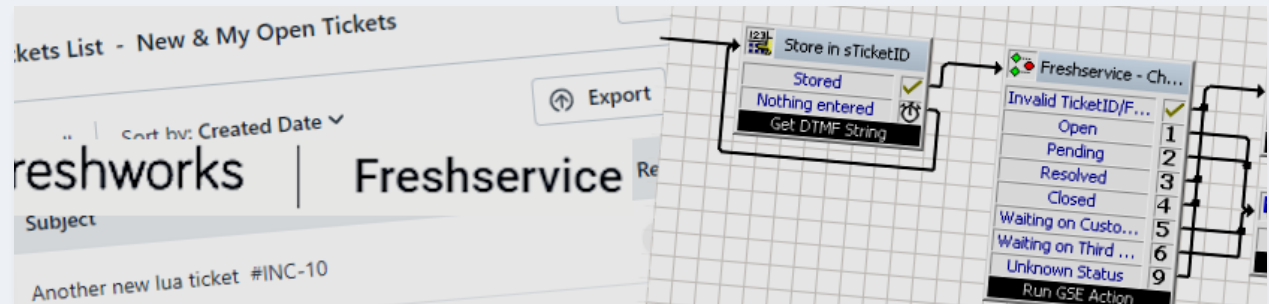
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - A.2 - Example: Create Note

Followers

0

VBScript

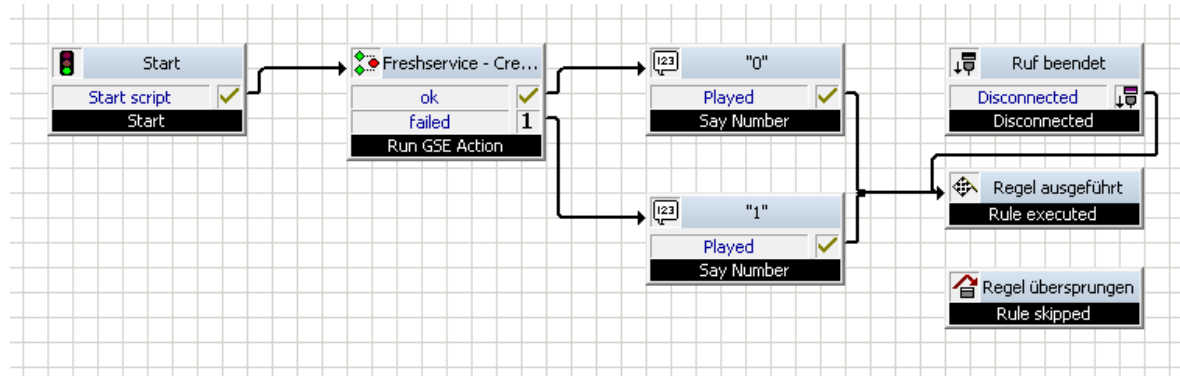
Lua

This example demonstrates the usage of the **FreshserviceIntegration - Create Note** GSE action.

It adds a new note/comment to an existing ticket and announces afterwards "0" on success or "1" on failure.

APPENDIX B

B.1 - Access/modify the code behind



To install it:

1. Open the **Call Routing Manager** of your desired SwyxWare user.
2. Click the "New Rule..." button.
3. Select "**Graphical Script Editor**" and click **Ok**.
4. Within the GSE open the **File | Import...** menu and click **No**.
5. From the download package select the following file:

For **VBScript** usage:

```
VBScript based\rse\Create Note.rse
```

For **Lua** usage:

```
Lua based\rse\Create Note.rse
```

6. You need to make some modifications in the **FreshserviceIntegration - Create Note** (Run GSE Action) block . They are explained in detail in chapter [3.2 - Add Comment](#).



By Tom Wellige
September 22, 2024

 Share

< Previous Page
A.1 - Example: Create Ticket

Next Page >
A.3 - Example: Check Status

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

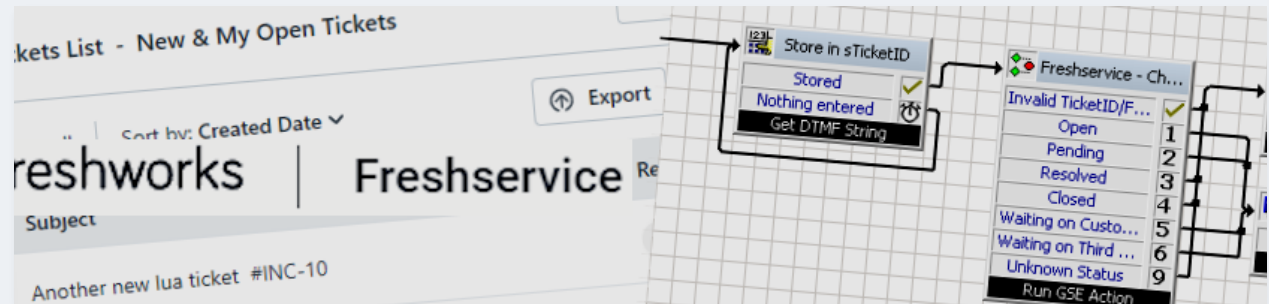
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - A.3 - Example: Check Status

Followers

0

VBScript

Lua

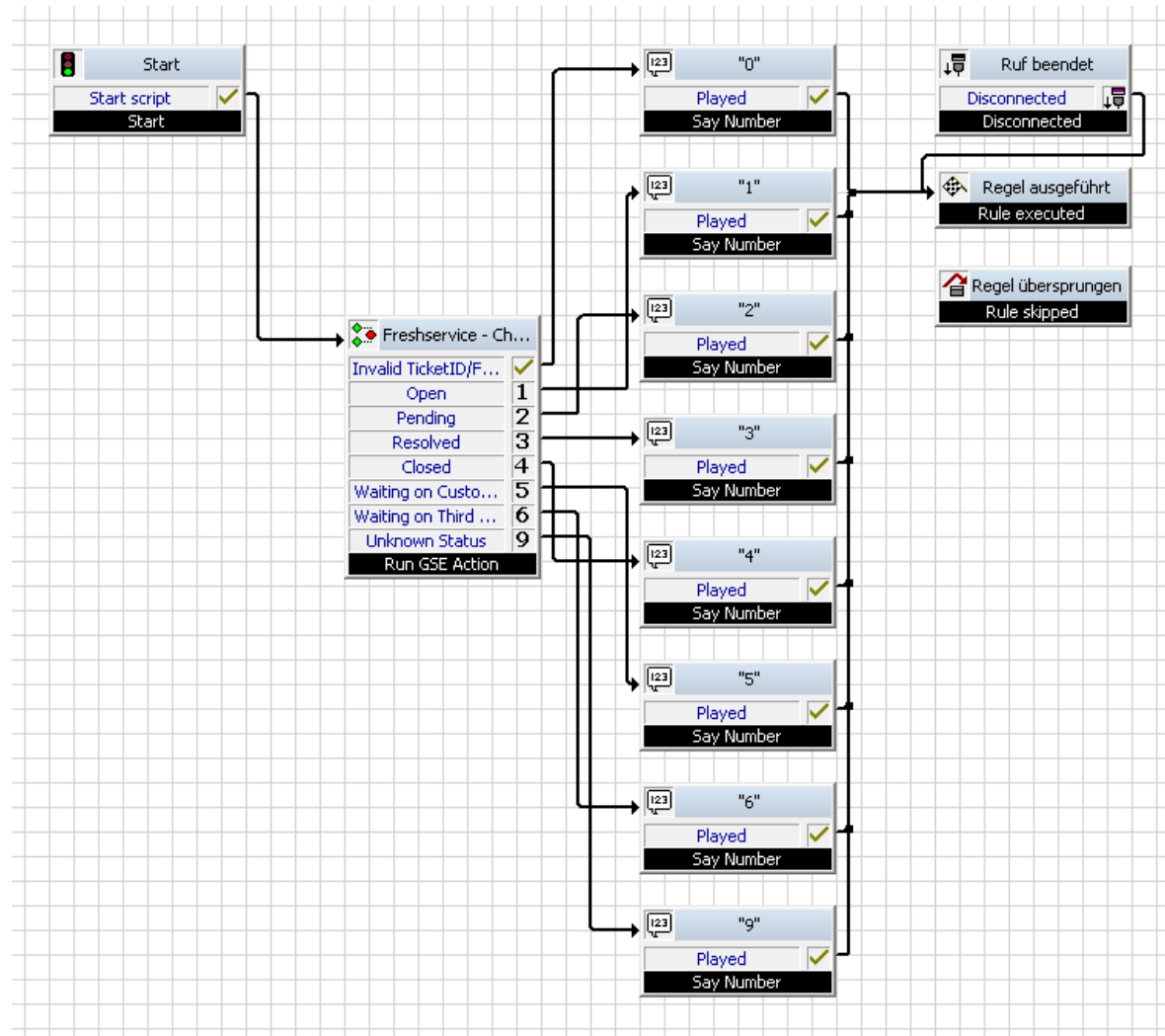
This example demonstrates the usage of the **FreshserviceIntegration - Check Status** GSE action.

It checks the current status of an existing ticket and announces the result.

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind



To install it:

1. Open the **Call Routing Manager** of your desired SwyxWare user.
2. Click the "New Rule..." button.

3. Select "**Graphical Script Editor**" and click **Ok**.
4. Within the GSE open the **File | Import...** menu and click **No**.
5. From the download package select the following file:

For **VBScript** usage:

```
VBScript based\rse\Check Status.rse
```

For **Lua** usage:

```
Lua based\rse\Check Status.rse
```

6. You need to make some modifications in the **FreshserviceIntegration - Check Status** (Run GSE Action) block . They are explained in detail in chapter [3.3 - Check Status](#).



By Tom Wellige
September 22, 2024

 Share

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige

Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

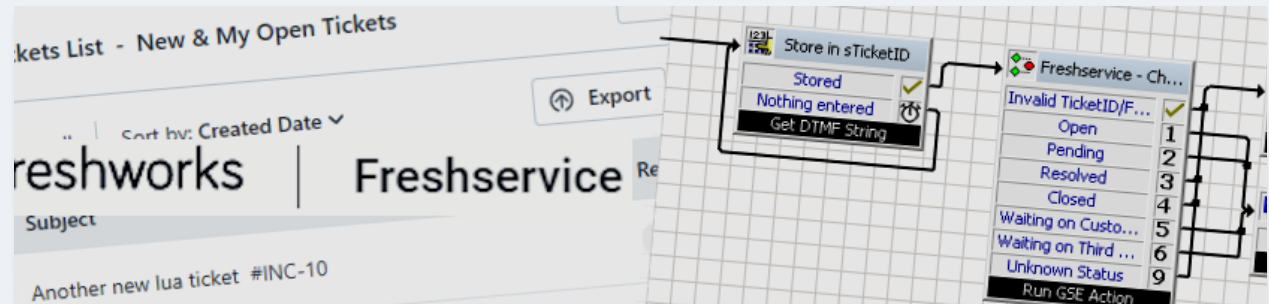
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - A.4 - Example: Check Status and Create Note

Followers 0

VBScript

Lua

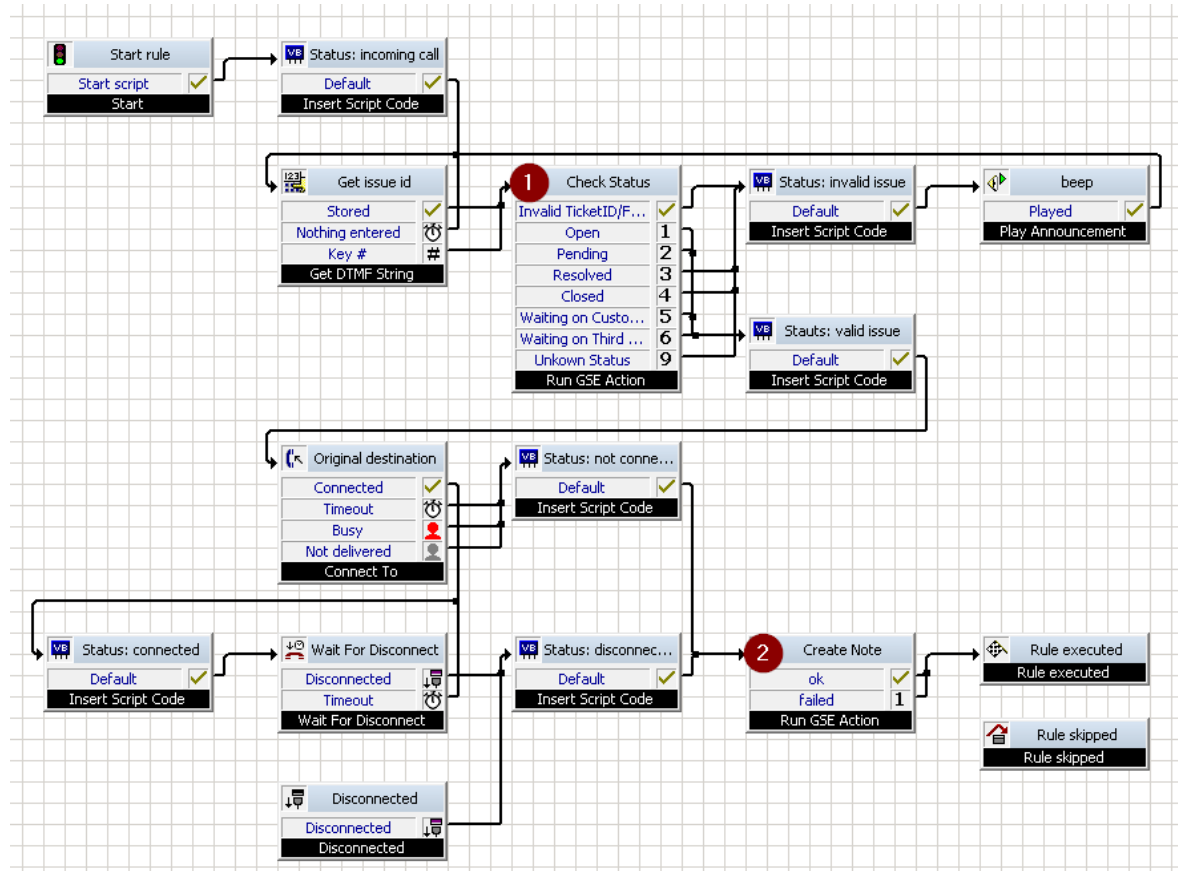
This example demonstrates the usage of the **FreshserviceIntegration - Check Status** and **FreshserviceIntegration - Create Note** GSE actions.

It is assumed to be used on a support help desk, where only callers with a valid ticket id are getting connected. Additionally all call details will be added as private note to the ticket at the end of the call.

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind



To install it:

1. Open the **Call Routing Manager** of your desired SwyxWare user.
2. Click the "New Rule..." button.
3. Select "**Graphical Script Editor**" and click **Ok**.
4. Within the GSE open the **File | Import...** menu and click **No**.

5. From the download package select the following file:

For **VBScript** usage:

```
VBScript based\rse\Check Status and Create Note.rse
```

For **Lua** usage:

```
Lua based\rse\Check Status and Create Note.rse
```

6. You need to make some modifications to make it work:

(1) - Make all needed configurations according to [3.3 - Check Status](#)

(2) - Make all needed configurations according to [3.2 - Create Note](#)

After a call has been completely handled by the call routing script and also afterwards by an agent (and finally disconnected), you will find the following call details added as a private note within the ticket:

```
04.09.2024 21:17:34 : Incoming call from 'Test User', 101
04.09.2024 21:17:55 : Ticket validated.
04.09.2024 21:18:04 : Call connected to Agent 1
04.09.2024 21:19:41 : Call connected to Agent 2
04.09.2024 21:22:17 : Call disconnected.
```



By Tom Wellige
September 22, 2024

 Share

< Previous Page
A.3 - Example: Check Status

Next Page >
A.5 - Example: Check and Announce Status

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

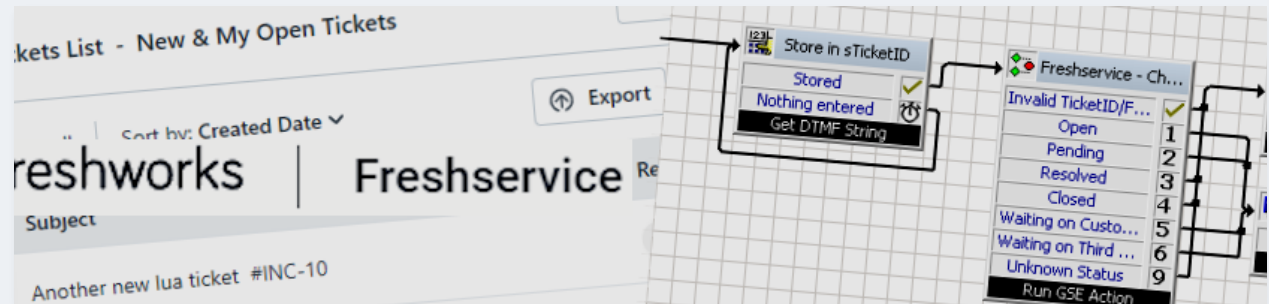
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - A.5 - Example: Check and Announce Status

Followers 0

VBScript

Lua

This example demonstrates the usage of the **FreshserviceIntegration - Create Issue** and **AzureTTS (text-to-speech)** GSE actions.

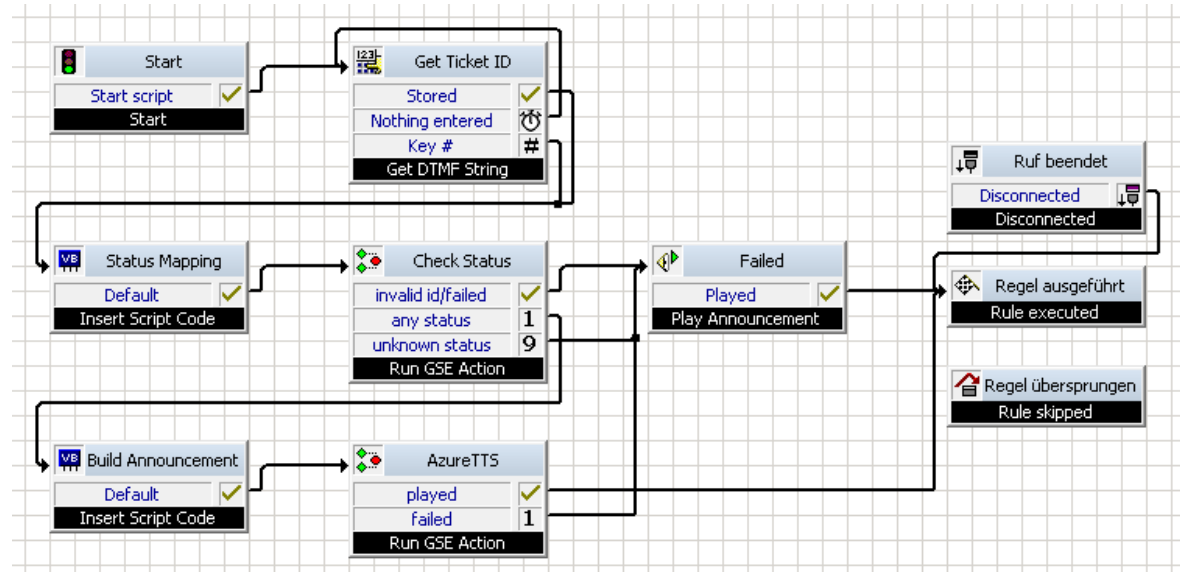
You need to have the **AzureTTS (text-to-speech)** Open ECR Extension installed beside the Jira Service Integration.

It asks for an issue id (close id with #), checks the status and announces its details.

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind



To install it:

1. Open the **Call Routing Manager** of your desired SwyxWare user.
2. Click the "New Rule..." button.
3. Select "**Graphical Script Editor**" and click **Ok**.
4. Within the GSE open the **File | Import...** menu and click **No**.
5. From the download package select the following file:

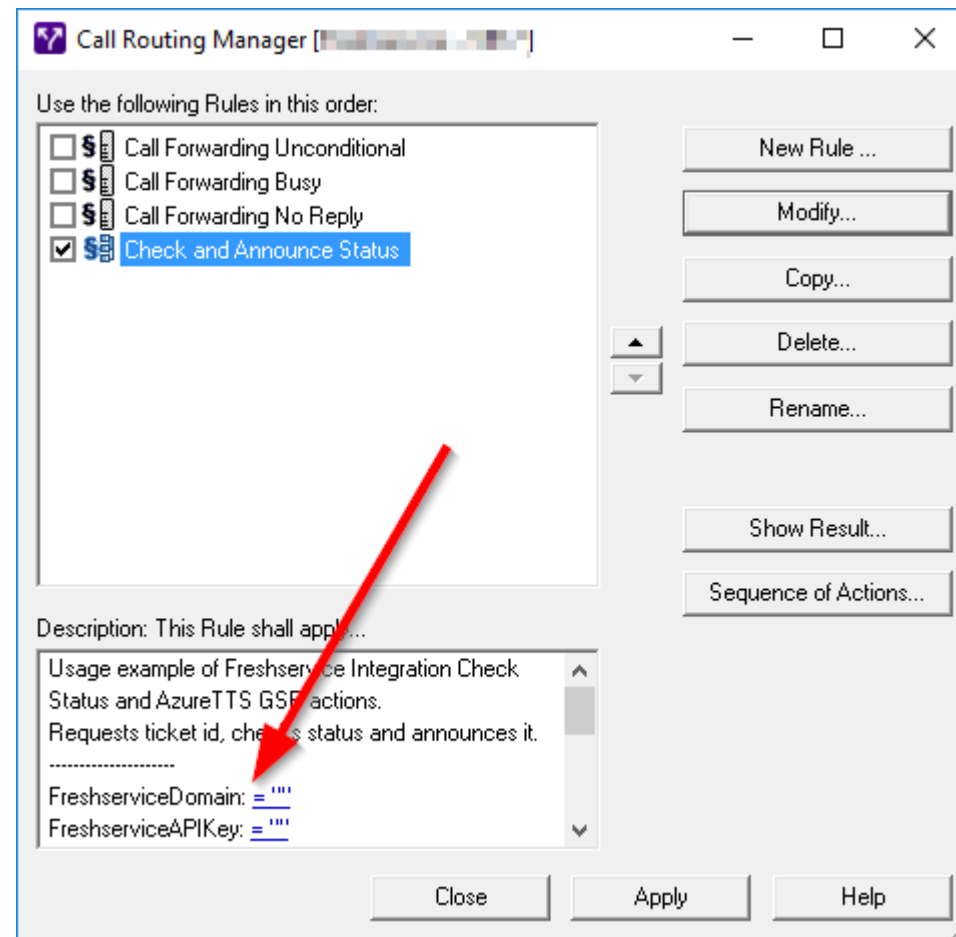
For **VBScript** usage:

```
VBScript based\rse\Check and Announce Status.rse
```

For **Lua** usage:

Lua based\rse\Check and Announce Status.rse

6. All needed configuration is done via **GSE Rule Parameters** directly in the **Call Routing Manager**.



For all **Freshservice...** parameters please refer to [Freshservice 3.1 - Create Ticket](#) for

instructions.

For all **Azure...** parameter please refer to [AzureTTS 3.1 - Usage](#) for instructions.

EnterTicketID

Name of the announcement wav file to be played when you have to enter the Freshservice ticket id.

It should announce something like "Please enter your Freshservice Ticket ID (the trailing number only) and finish with the # (hash) key."

Default: *beep.wav*

StatusAnnouncement

The text of the status announcement. You can make use of placeholders, to insert current values:

#ID# - the ticket id

#STATUS# - the current status of the ticket

#UPDATE# - the date/time of the latest modification of the ticket

Default: *The current status of ticket #ID# is #STATUS# and it was modified latest at #UPDATE#*

FailedAnnouncement

Name of the announcement wav file to be played if either the Freshservice or the Azure request fails.

Please refer to [Freshservice 3.4 - Trouble Shooting](#) or [AzureTTS 3.2 - Trouble Shooting](#) for information on how to figure what went wrong in detail.

Default: *beep.wav*

Hint:

It makes a lot of sense to use the **AzureTTS (text-to-speech)** extension once to create the announcement files for **EnterTicketID** and **FailedAnnouncement**. This ensures that the entire call routing uses the same voice.

To do so, just use the [A.1 - Example: Announce Text](#) example and use an **Insert Script**

Code block right behind the **Played** exit to copy the generated temporary wav file to a permanent location.

This only works with the **VBScript** version of the extension.

```
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CopyFile g_sAzureTTS_LatestWavFile, "C:\MyFiles\"
Set fso = Nothing
UseExit = 0
```



By Tom Wellige
September 22, 2024

 Share



Previous Page

A.4 - Example: Check Status and Create Note

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community

Menu

INTRODUCTION

0 - Introduction

PREPARATIONS

1 - Preparations

SETUP

2 - Setup GSE Actions

USAGE

3.1 - Create Ticket

3.2 - Create Note

3.3 - Check Status

3.4 - Trouble Shooting

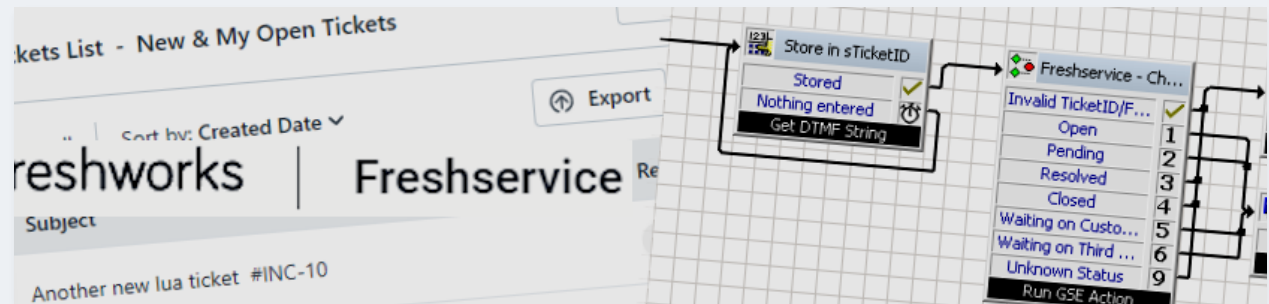
APPENDIX A

A.1 - Example: Create Ticket

A.2 - Example: Create Note

A.3 - Example: Check Status

A.4 - Example: Check Status and Create Note



Freshservice Integration - B.1 - Access/modify the code behind

Followers

0

How to make your own modifications or simply just take a look onto the implementation to get an idea of how everything works?

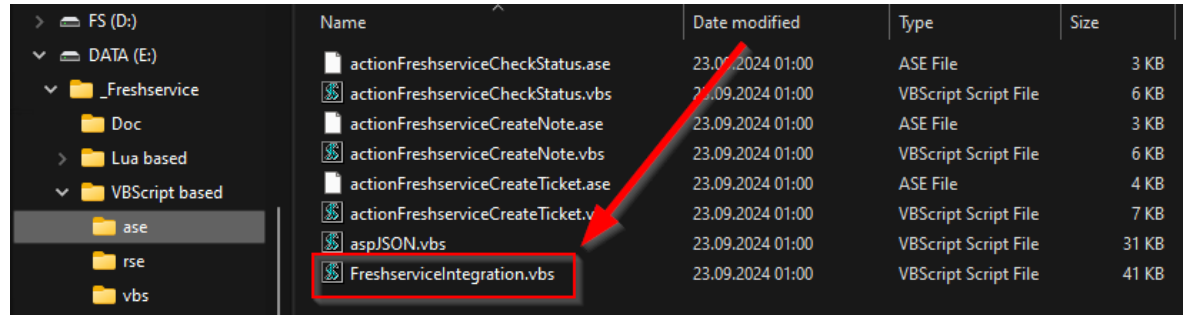
The following explanations will refer to the **VBScript** version, the **Lua** version however is more or less identical in its structure.

The entire code of this integration is located in one single file, **FreshserviceIntegration.vbs**.

A.5 - Example: Check and Announce Status

APPENDIX B

B.1 - Access/modify the code behind

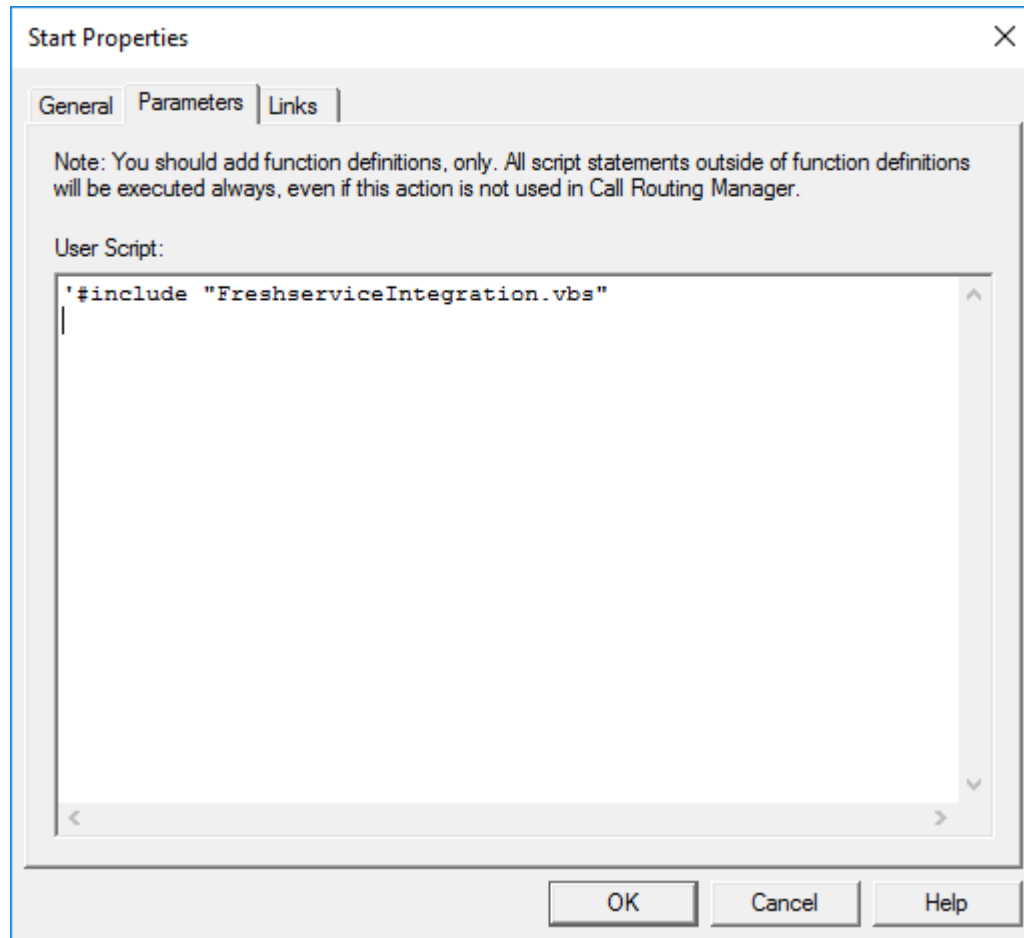


	Name	Date modified	Type	Size
FS (D:)				
DATA (E:)				
_Freshservice				
Doc				
Lua based				
VBScript based				
ase				
rse				
vbs				
	actionFreshserviceCheckStatus.ase	23.09.2024 01:00	ASE File	3 KB
	actionFreshserviceCheckStatus.vbs	23.09.2024 01:00	VBScript Script File	6 KB
	actionFreshserviceCreateNote.ase	23.09.2024 01:00	ASE File	3 KB
	actionFreshserviceCreateNote.vbs	23.09.2024 01:00	VBScript Script File	6 KB
	actionFreshserviceCreateTicket.ase	23.09.2024 01:00	ASE File	4 KB
	actionFreshserviceCreateTicket.v	23.09.2024 01:00	VBScript Script File	7 KB
	aspJSON.vbs	23.09.2024 01:00	VBScript Script File	31 KB
	FreshserviceIntegration.vbs	23.09.2024 01:00	VBScript Script File	41 KB

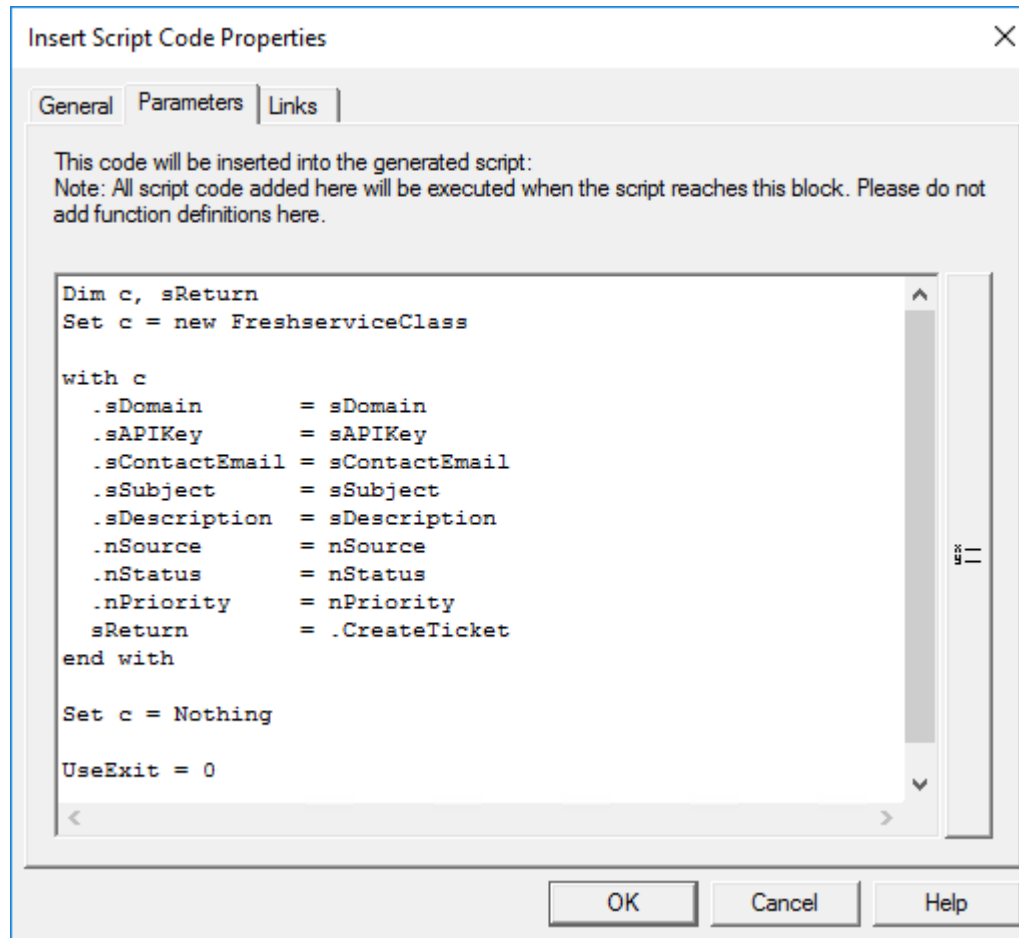
All functionality is encapsulated in a class (**FreshsericeClass**), which defines properties for all parameters (with some validation) as also private and public methods.

To make use of the **FreshserviceClass** class the **FreshserviceIntegration.vbs** file must be included first. This is done in the **"Start"** block of the GSE Action or your own GSE rule.

```
'#include "FreshserviceIntegration.vbs"
```



Afterwards the class needs to get instantiated, its properties set and the needed public method/function called. The return value of the function is either "0" (ok) or "1" (failed), so it can be used directly for the exits of a "**Insert Script Code**" or "**Run GSE Action**" block.



And this is exactly what the GSE actions of this project are doing.

In order to make modifications it is not simply possible to modify the FreshserviceIntegration.vbs file and re-upload it again into the SwyxWare database. This is because the file is digitally signed, which is necessary to get it included with the **#include** statement.

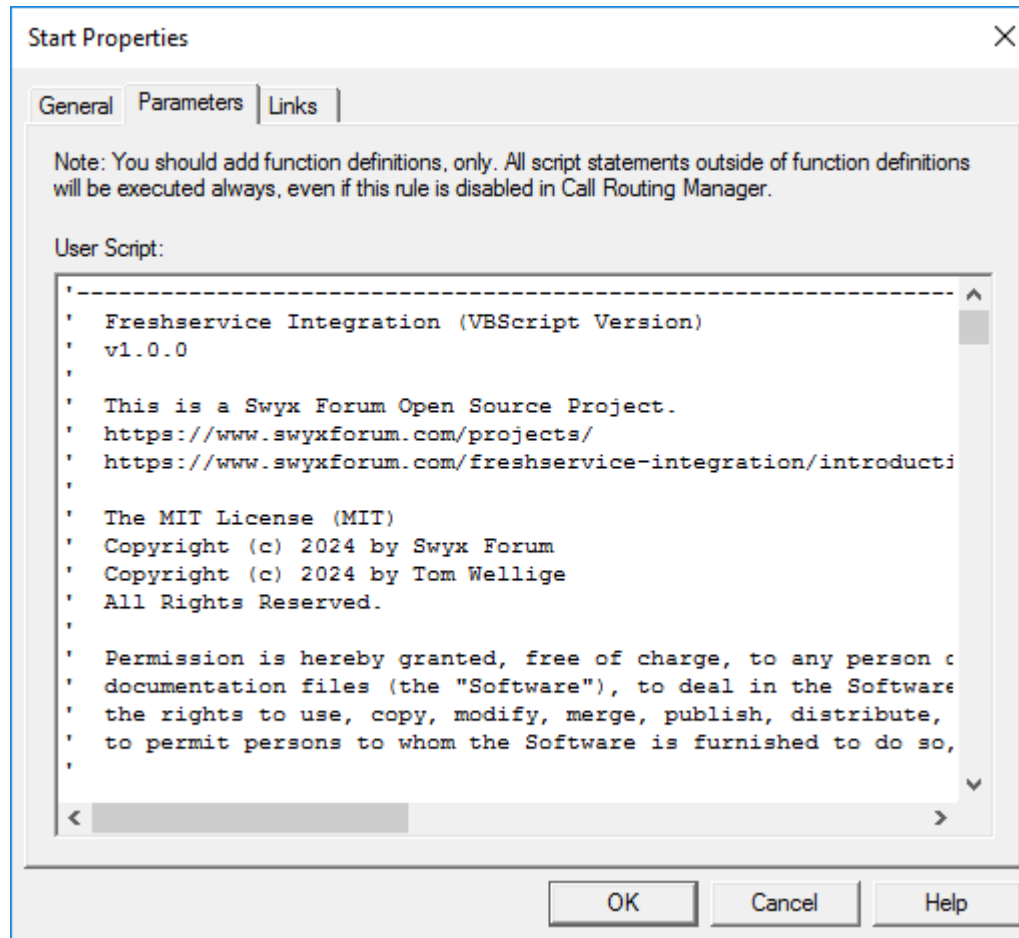
The easiest way to do your own modifications is to forget about the FreshserviceIntegration.vbs file all together and use just its content within a start block of a

GSE rule (instead of the #include statement), but without the digital signature.

You can strip the signature yourself from the code (first line and all lines from the bottom), or simply use the content of the start.vbs file from the "vbs" folder or the download package.

```
VBScript based\vbs\start.vbs
```

So just do all your modifications within the **start.vbs** file and afterwards just **copy & paste** the entire content of the file into the "**Start**" block



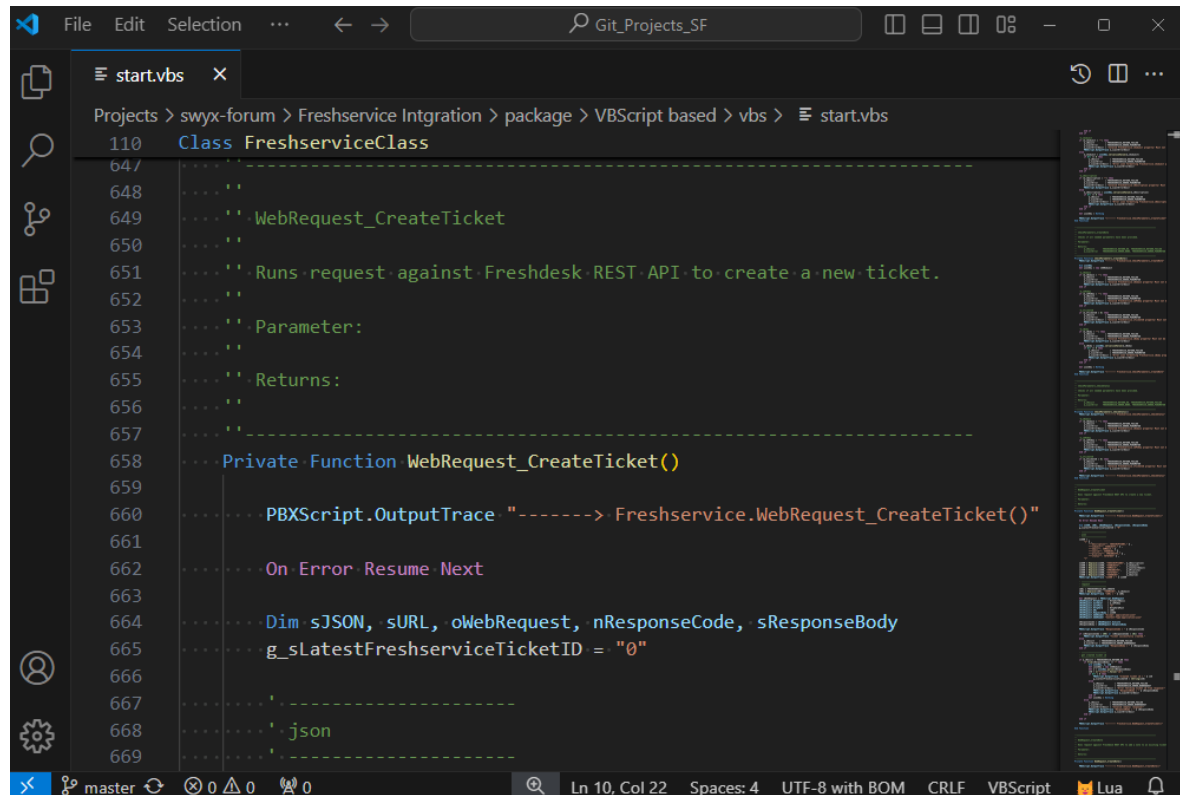
and then use the Insert "**Insert Script Code**" block to instantiate the FreshserviceClass and call its functionality

```
Dim c, sReturn
Set c = new FreshserviceClass

with c
    .sDomain      = "mycompany"
```

```
.sAPIKey      = "1234567890123"  
.sContactEmail = "test@test.com"  
.sSubject     = "My first Ticket"  
.sDescription  = "Please call me asap!"  
.nSource      = 3  
.nStatus      = 2  
.nPriority     = 2  
sReturn       = .CreateTicket  
end with  
  
Set c = Nothing  
  
UseExit = sReturn
```

It is highly recommended to not do any code modifications directly in the **Start** block, but instead use a proper editor like VS Code or Notepad++.

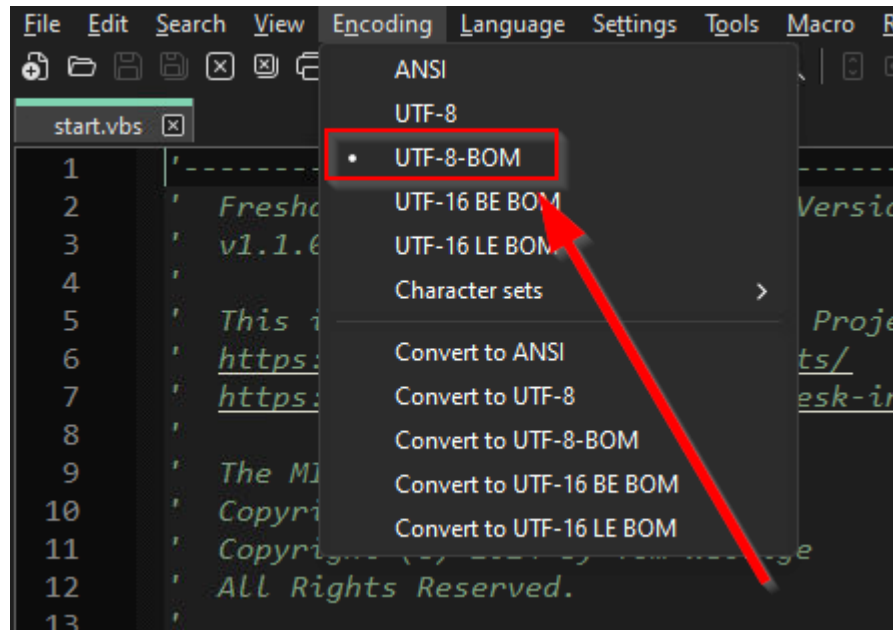


```
110 Class FreshserviceClass
647
648
649 Private WebRequest_CreateTicket
650
651 Private Runs request against Freshdesk REST API to create a new ticket.
652
653 Private Parameter:
654
655 Private Returns:
656
657
658 Private Function WebRequest_CreateTicket()
659
660 Private PBXScript.OutputTrace "-----> Freshservice.WebRequest_CreateTicket()"
661
662 Private On Error Resume Next
663
664 Private Dim sJSON, sURL, oWebRequest, nResponseCode, sResponseBody
665 Private g_sLatestFreshserviceTicketID = "0"
666
667
668 Private json
669
```

You are now free to to any modifications within the code and use it afterwards in your own GSE rules.

If you want to update the FreshserviceIntegration.vbs in the end, you need to update its signature as well. This can be done with the "**SignScript**" tool which can be found in the [Enreach Partner Net](#) (you need a partner login).

In order to get get your own .vbs file signed, make sure that the text encoding is **UTF-8-BOM**. This can easily be checked/configured with Notepad++.



Please feel free to ask any questions regarding the code, implementation or distribution (signing, including, etc.) in the [Open ECR Extensions forum](#).

Most of the above mentioned is also true for the following extensions. So there is lots more to explore 😊

- [Azure Translate](#) (REST API usage, OAuth2 authentication)
- [Azure TTS \(text-to-speech\)](#) (REST API usage, OAuth2 authentication)
- [Freshdesk Integration](#) (REST API usage, API Key/Token authentication)
- [Invision Power Services \(IPS\) Integration](#) (REST API usage, API Key/Token authentication)
- [Jira Service Integration](#) (REST API usage, API Key/Token authentication)
- [Longest Waiting](#) (Database usage)
- [Open Queue](#) (Database usage)
- [Zendesk Integration](#) (REST API usage, API Key/Token authentication)



By Tom Wellige
September 22, 2024

 Share

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige
Powered by Invision Community