

# Hello-IT integration with FileWave (macOS)

## Description

Hello-IT is a 3rd party tool for macOS designed as a Menu Item to launch scripts as a Self Service tool. The below is an example of how FileWave can deliver and integrate with the tool. Hello-IT has multiple options for the Menu Item list, including:

- Reporting on server status
- Links to Websites
- Reporting information in the Menu Item
- Additional Scripting for Self Service (Including Integration with Slack)

Instructions on Hello-IT may be found via the following link:

<https://github.com/ygini/Hello-IT>



### Third Party Software

Slack is a third party application. The details provided are for example only and are unsupported by FileWave.

## Requirements

- Hello-IT installer
- Hello-IT configuration file



### Self-Signed Certificates

Although many features of Hello-IT will work fine, the option to report on server status relies upon the ability to pull data from a server. If the server is not trusted, then this will fail. As such, servers with self-signed certificates will always report a failure when using public.test.http (see below)

Some examples below implement Slack, based upon our KB:

[Slack integration with FileWave](#)

## Directions

Configuration of Hello-IT can be via a Custom Settings payload profile, which may be delivered through FileWave to devices, the basic example of which can be uploaded into a Custom Settings payload and is located within the GitHub repository: <https://github.com/ygini/Hello-IT/blob/master/example/Basic%20Example/com.github.ygini.Hello-IT.plist>

The Menu Items are linked to functions, below are some examples of Public functions.

### public.open.resource

This function provides URLs that will launch in the default browser. Editing the provided Fileset, can allow for important websites, e.g. Intranet pages. In this example, FileWave Foundry and Website are offered:

### public.open.resource

```
Dict {
  settings = Dict {
    title = FileWave Website
    URL = https://www.filewave.com
  }
  functionIdentifier = public.open.resource
}
Dict {
  settings = Dict {
    title = FileWave Foundry
    URL = https://foundry.filewave.com
  }
  functionIdentifier = public.open.resource
}
```

### public.test.http

This function tests for access to a Web server by running a download command and checking the output of the download. If successful all will be well, but if failed, the Menu Item text will become red and a red dot will be highlighted next to the text in the Drop Down.

An md5 checksum is required for the resource, and may be obtained using a curl command. For example, if using the URL: <https://custom.filewave.com>, the following would be run

## Web Resource md5

```
curl https://custom.filewave.com | md5
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
100	1406	100	1406	0	0	11051	0
						--:--:--	--:--:--
						--:--:--	--:--:--
							11070

```
63e398fd52c3dc883d13401531339f51
```

The payload would then be amended thus:

## public.test.http

```
Dict {
  settings = Dict {
    repeat = 60
    ignoreSystemState = true
    mode = md5
    originalString = 63e398fd52c3dc883d13401531339f51
    title = FileWave Server
    URL = https://custom.filewave.com
  }
  functionIdentifier = public.test.http
}
```

Configuration could be set to point to your FileWave server for example, so users can see that a connection is established.

## public.script.item

This function provides the ability to run scripts. In this example, consider the [KB article that uses Slack](#) as an IT reporting tool and further utilising this for the following requests:

- Printer ink replacement
- Call the user back

The format of this section of the file could look like:

## public.script.item

```
Dict {
  settings = Dict {
    content = Array {
      Dict {
        settings = Dict {
          script = slack_printer_ink.sh
          title = Dict {
            fr = Demander l'encre d'imprimante
            en = Request Printer Ink
          }
        }
      }
      functionIdentifier = public.script.item
    }
  }
  Dict {
    settings = Dict {
      script = slack_request_callback.sh
      title = Dict {
        fr = Demander un rappel
        en = Request IT Callback
      }
    }
    functionIdentifier = public.script.item
  }
}
```

```

    title = Dict {
      fr = Libre-service
      en = Self-Service
    }
  }
  functionIdentifier = public.submenu
}

```

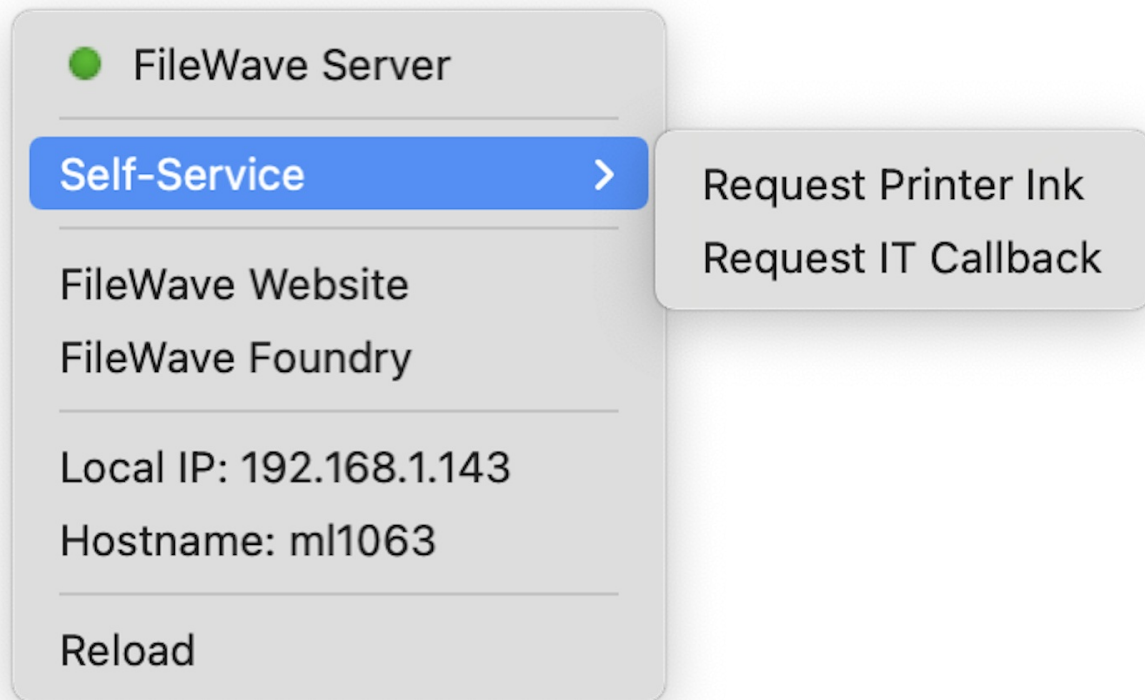
First thing to note is that these have been wrapped in their own 'Self-Service' sub Menu Item 'public.submenu'. In the example two scripts have been referenced:

- slack\_printer\_ink.sh
- slack\_request\_callback.sh

Scripts should be located in the following directory:

```
/Library/Application Support/com.github.ygini.hello-it/CustomScripts/
```

The user may then use the drop down menu to trigger these items, where IT would then receive a Slack message.



## Example Filesets

Based upon the above, the following example Filesets include not only the configuration file, but the example scripts that implement the Slack integration. Consider placing these along with the Hello-IT installer in a Fileset Group and associate the group with devices:

## ▼ Hello-IT

### Hello-IT Slack Printer Ink

### Hello-IT Slack Request Callback

### **PKG - Hello-IT-1.6.0-Release-Distribution**

### **Profile - Hello-IT**

Please see the KB on [FileWave and Slack](#) with regards to creating the required Slack Webhook. Configuration of the scripts to set the Webhook are the same here, editing the Filesets, selecting the pre-installation script and then editing the 'Environment Variables' such that the 'slack\_webhook' is set to the generated Webhook from Slack; replacing 'PLACE WEBHOOK HERE'.

Similarly, if using Legacy Webhooks, the slack\_channel variable needs to be edited to match the name of the created channel; for App Webhooks it will be ignored. In the example, this channel is called 'fw\_messages'

## Profile - Hello-IT

[Profile - Hello-IT.fileset.zip](#)

## Hello-IT Slack Request Callback


[Hello-IT Slack Request Callback.fileset.zip](#)


This script attempts to read the telephone number of the user based upon a directory service entry. If device is not bound or user is not a directory user, no number will be supplied.

## Hello-IT Slack Printer Ink

[Hello-IT Slack Printer Ink.fileset.zip](#)

Example messages in Slack:


 **Helpdesk** APP 11:39 AM


 **Request Callback: 1015VMDEP**

Message from annie

**Please call back: 0118 999 881 999 119 725 3**

macOS: 10.15.1, client: 13.2.3 | Today at 11:39 AM

 **Helpdesk** APP 11:34 AM

 **Printer Ink Request: 1015VMDEP**

**!!Printer Ink is Low!!**

**Request: Please Instal New Cartridge**

macOS: 10.15.1, client: 13.2.3 | Today at 11:34 AM

## Conclusion

This is just an example of how FileWave can be used to deliver an additional tool to devices, empowering users to easily request resources and where communication may be hindered, allow the user to easily request IT assistance; particularly useful for users who may be remote.