flexyPage

innovative display solutions

API description

flexyPage Display API

for centralized configuration, monitoring and control

Version 1.0

Table of Contents

Objective

This document is aimed exclusively at developers and quality managers who want to develop their own extensions for flexypage displays and control or test them automatically.
The flexyPage displays offer a wide range of functions that can be configured via a user interface. The functionality of the displays can be extended or customized via the installation of plug-ins, e.g. for the translation of proprietary protocols, the connection to your own cloud, or the transmission of monitoring data in your own format.
An HTTP API is available for automated control of the plug-ins. This API provides functions for the automated testing and continuous integration of plug-ins.

Features

The interface provides the following functions for the flexyPage displays:

- command line ready
- HTTP API interface
- authentication
- automated installation process
    - upload of a plugin file
    - return plugin installation ID
    - use ID to regularly check installation progress
- log checking of running plugin

General

Authentication

All endpoints except for the login endpoint are protected by an authentication mechanism. This means that each request needs to contain an API key in the header `api-key`. If the request is missing it an Unauthorized Request response is returned.

401

Content:

```
{
      “error”: {
            “code”: 401002,
            “message”: “'api-key' header is missing”
      }
}
{
      “error”: {
            “code”: 401003,
            “message”: “'api-key' is empty”
      }
}
{
      “error”: {
            “code”: 401004,
            “message”: “API key is invalid”
      }
}
{
      “error”: {
            “code”: 401005,
            “message”: “API key is expired”
      }
}
```

Allowed Methods

In the following, all endpoints and their methods are specified. If an endpoint is accessed with a method it does not support, the endpoint returns a Method Not Allowed response.

405

Content:

```
{
    “error”: {
        “code”: 405001,
        “message”: “Invalid method”
    }
}
```

Authentication

The installation of a plugin is only possible when authorized. This means at least a form of authentication is required.
The mechanism is inspired by the OAuth password grant. Based on the following documentation for the request:

https://www.oauth.com/oauth2-servers/access-tokens/password-grant/

Based on the following documentation for the response:

https://www.oauth.com/oauth2-servers/access-tokens/access-token-response/

Endpoint

`/api/v1/oauth/token POST`
Query Parameters:

```
grant_type: password
username: <username>
password: <password>
```

The credentials are the ones used to log in to the device via the flexyPage editor.

Responses

200

Content:

```
{
        "token_type": "Bearer",
        "access_token": "<access token>",
        "expires_in": <expiry period in seconds>
}
```

The returned API key is valid for the duration amount of seconds specified in the `expires_in` field and needs to be used for each request in the `api-key` header.

400

Content:

```
{
      “error”: {
            “code”: 400001,
            “message”: “‘grant_type’ missing in query
                        parameters”
      }
}
{
      “error”: {
            “code”: 400002,
            “message”: “grant_type not supported”
      }
}
{
      “error”: {
            “code”: 400003,
            “message”: “‘username’ missing in query
                        parameters”
      }
}
{
      “error”: {
            “code”: 400004,
            “message”: “‘password’ missing in query
                        parameters”
      }
}
```

401

Content:

```
{
      “error”: {
            code”: 401001,
            “message: “Invalid credentials”
      }
}
```

Plugin installation

The installation of a flexyPage plugin is triggered by uploading a valid plugin file to the device. Since the installation procedure can take some time, an installation ID is provided on successful upload. This ID is used in conjunction with a redirect response and a retry after period that allows an HTTP client to periodically check until the installation is finished.

After a successful installation the viewer application is restarted and the plugin log is deleted.

Endpoint

```
/api/v1/plugin?keep-data=true POST
/api/v1/plugin?keep-data POST
/api/v1/plugin POST
```

If keep-data is given the data directory of the previously installed plugin with the same name is kept during installation.

Format:	`multipart/form-data`
Content:	`file=<Plugin file>`

Reponses

301

If the request is successful and plugin installation is initiated a redirect response is sent that allows to check the update status.
Headers:

```
Retry-After: 5
Location: /api/v1/plugin?installation-id=<id>
```

400

Content:

```
{
      “error”: {
            “code”: 400011,
            “message”: “Plugin file is missing”
      }
}
{
      “error”: {
            “code”: 400012,
            “message”: “Plugin file needs to be uploaded
                        as multipart/form-data in the file
                        key”
      }
}
```

422

Content:

```
{
      “error”: {
            “code”: 422011,
            “message”: “Not a valid plugin file”,
            “details”: [
                  {
                        “receivedFile”: “<sha256 hash
                                          of received file>”
                  }
            ]
      }
}
{
      “error”: {
            “code”: 422012
            “message”: “Plugin not encrypted for the
                        installed decryption key”,
            “details”: [
                  {
                        “receivedFile”: “<sha256 hash
                                          of received file>”
                  }
            ]
      }
}
```

Installation Status

Due to the potentially long running nature of the installation process the installation procedure is started after upload. The returned installation ID can be used to check the installation status.

Endpoint

```
/api/v1/plugin?installation-id=<ID> GET
```

Responses

200

Content:

```
{ “status”: “Installation successfully finished” }
{
      “error”: {
            “code”: 200001,
            “message”: “Installation termination reason”
      }
}
```

When installing a new plugin any of the previous installation IDs are deleted. Subsequent requests for this ID result in a 404.

301

This response happens as long as the installation process is still ongoing.
Attention: the curl (Version: 7.81.0 (tested) up to current 8.4.0) command line tool does not follow the Retry-After header after the second redirect, which leads to instantaneous redirects.
Headers:

```
Retry-After: 5
Location: /api/v1/plugin?installation-id=<id>
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Installation ID unknown”
      }
}
```

Uninstall Plugin

To uninstall a plugin a `DELETE` request can be sent.

Endpoint

```
/api/v1/plugin/<plugin name>?keep-data=true DELETE
/api/v1/plugin/<plugin name>?keep-data DELETE
/api/v1/plugin/<plugin name> DELETE
```

If keep-data is given the data directory of the previously installed plugin is kept during installation.

Responses

204

Content: none

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name
                        ‘<plugin name>’ is not installed”
      }
}
```

List of Installed Plugins

To get an overview of the installed plugins an endpoint to list the installed plugins is provided.

Endpoint

`/api/v1/plugin GET`

Responses

200

Content:

```
{
      “plugins”: [
            {
                  "name": "<plugin name>",
                  "version": "20230803031803-2bc39fc",
                  "configured": true,
                  "enabled": true,
                  "conflicts": false
            }
      ]
}
```

Check Specific Plugin

To confirm if a plugin installation resulted in an updated plugin the installed version is checkable.

Endpoint

`/api/v1/plugin/<plugin name> GET`

Responses

200

Content:

```
{
      "name": "<plugin name>",
      "version": "20230803031803-2bc39fc",
      "configured": true,
      "enabled": true,
      "conflicts": false
}
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name
                        ‘<plugin name>’ is not installed”
      }
}
```

Configure Enabled State of Plugin

To allow the plugin to be enabled or disabled. The viewer application is restarted after setting this option.

Endpoint

```
/api/v1/plugin/<plugin name>/config PATCH
```

Request

Content:

```
{
      “enabled”: <true|false>
}
```

Responses

200

Content:

```
{
      "name": "<plugin name>",
      "version": "20230803031803-2bc39fc",
      "configured": true,
      "enabled": true,
      "conflicts": false
}
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name <plugin name>
                        is not installed”
      }
}
```

422

Content:

```
{
      “error”: {
            “code”: 422012,
            “message”: “Only boolean values (true|false)
                        are supported for ‘enabled’”
      }
}
```

Configure Parameters for Adapter Application

To override the configured parameter options of the installed plugin. The viewer application is restarted after setting this option.

Endpoint

`/api/v1/plugin/<plugin name>/config PATCH`

Request

Content:

```
{
      “adapter-parameters”: “<string of parameters that
                              are understood by the adapter
                              application>”
}
```

Any request to the config endpoint can be combined in one by merging the content in one JSON object.

Responses

200

Content:

```
{
      "name": "<plugin name>",
      "version": "20230803031803-2bc39fc",
      "configured": true,
      "enabled": true,
      "conflicts": false,
      “adapter-parameters”: “<string of parameters that
                              are understood by the adapter
                              application>”
}
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name <plugin name>
                        is not installed”
      }
}
```

422

Content:

```
{
      “error”: {
            “code”: 422013,
            “message”: “‘adapter-parameters’ needs to be
                        a string”
      }
}
```

Reset Parameters for Adapter Application to Default

To reset to the configured parameter options of the installed plugin. The viewer application is restarted after using this endpoint.

Endpoint

```
/api/v1/plugin/<plugin name>/config PATCH
```

Request

Content:

```
{
      “adapter-parameters”: null
}
```

Responses

200

Content:

```
{
      "name": "<plugin name>",
      "version": "20230803031803-2bc39fc",
      "configured": true,
      "enabled": true,
      "conflicts": false
}
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name <plugin name>
                        is not installed”
      }
}
```

Check Plugin Log

The following endpoint allows to check the log of a specific plugin.

Endpoint

```
/api/v1/plugin/<plugin name>/logs?tail=<number of lines>    GET
/api/v1/plugin/<plugin name>/logs?tail                      GET
/api/v1/plugin/<plugin name>/logs                           GET
```

If tail is given without an argument 100 is used as default value.

Responses

200

Content:

```
{
      “logEntries”: [
            “line 1”,
            “line 2”,
            …
            “line n”
      ]
}
```

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name <plugin name>
                         is not installed”
      }
}
```

Clear Plugin Log

The following endpoint allows to delete the accumulated logs.

Endpoint

```
/api/v1/plugin/<plugin name>/logs DELETE
```

Responses

204

Content: none

404

Content:

```
{
      “error”: {
            “code”: 404011,
            “message”: “Plugin with the name <plugin name>
                         is not installed”
      }
}
```

Trigger Restart of Viewer Application

The following endpoint allows restarting the viewer application.

Endpoint

`/api/v1/viewer/restart POST`

Responses

204

Content: `none`

Create Screenshot of flexyPage Viewer

The following endpoint allows retrieving a screenshot of the current state of the viewer application.

Endpoint

`/api/v1/viewer/screenshot POST`

Responses

200

Content: `PNG Image of viewer`

503

Headers:

```
Retry-After: 10
```

Your contact persons

Even extensive documentation cannot answer all questions. Do you have questions or suggestions concerning our flexyPage system? We look forward to your requests. You can contact us at:

ELFIN Technology GmbH
Im Zollhafen 22
50678 Cologne
Germany
Phone: +49 (221) 6778932-0
FAX: +49 (221) 6778932-2
service@elfin.de
www.elfin.de

flexyPage Sales
flexypage.de/en/sales
Tel.: +49 (221) 6430816-2
FAX: +49 (221) 6778932-2
sales@flexyPage.de

flexyPage Support
flexypage.de/en/support
Phone: +49 (221) 6430816-3
support@flexyPage.de