

# WPEFramework (Thunder)

- 1.Repos:
- 2. WPEFramework
- 3. HTTP Restful API
  - 3.1 Methods
  - 3.2 Web API Path
  - 3.3 Examples
- 4. WebSockets for events & notifications
- 5. Filesystem installation (config files & shared libs)
  - 5.1 Config files
  - 5.2 Executables
  - 5.3 Shared libs
- 6. Memory Usage
- 7. Disk space usage
- 8. WebKitBrowser plugin & rdkbrowser2
- Further Reading:

## 1.Repos:

Repo Name	URL	Comment
WPEFramework	<a href="https://github.com/WebPlatformForEmbedded/WPEFramework">https://github.com/WebPlatformForEmbedded/WPEFramework</a>	Main repo for Thunder/WPEFramework
WPEFrameworkPlugins	<a href="https://github.com/WebPlatformForEmbedded/WPEFrameworkPlugins">https://github.com/WebPlatformForEmbedded/WPEFrameworkPlugins</a>	Various WPEFramework plugins
Lightning SDK	<a href="https://github.com/WebPlatformForEmbedded/Lightning">https://github.com/WebPlatformForEmbedded/Lightning</a>	WPE UI Framework (JS & WebGL library for developing web apps)
WPEReferenceUX	<a href="https://github.com/WebPlatformForEmbedded/WPEReferenceUX">https://github.com/WebPlatformForEmbedded/WPEReferenceUX</a>	Sample UI/Demo app which demonstrates WPEFramework and Lightning use
WPEPluginLauncher	<a href="https://github.com/WebPlatformForEmbedded/WPEPluginLauncher">https://github.com/WebPlatformForEmbedded/WPEPluginLauncher</a>	Plugin to "Launch" linux applications and scripts

## 2. WPEFramework

WPEFramework provides a unified web-based interface with a consistent navigation model. In this model, plugins (custom or generic) are controlled and queried, through the WPEFramework application.

The main responsibilities of WPEFramework application are:

- Modular loading and unloading of plugins.
- Plugin process localization. In or out-of-process communicating with the framework over a lightweight RPC communication channel.
- Runtime enabling/disabling of tracing information within the plugins and the WPEFramework application.
- Light-weight implementation of the HTTP [RFC2616] specification.
- Light-weight implementation of the WebSocket [RFC6455] specification.

Each instance of a plugin in the WPE id identified by a name. This name is referred to as Callsign of the plugin. The callsign must be unique in the context of all configured plugins.

## 3. HTTP Restful API

All request and response bodies should use the JSON format for data.

### 3.1 Methods

Method	Function
GET	Retrieve information from WPEFramework or a plugin
POST	Update new information or new objects at WPEFramework or a plugin
PUT	Update new information or new objects at WPEFramework or a plugin

DELETE	Delete information at WPEFramework or a plugin
--------	--

### 3.2 Web API Path

All WPEFramework commands start with the "Service" prefix followed by the Plugin name:

```
(GET|POST|PUT|DELETE) /Service/<PluginName>[/OptionalPaths] HTTP/1.1
```

### 3.3 Examples

Retrieve the list of all active plugins:

```
curl -X GET http://192.168.1.122:9998/Service/Controller
```

Deactivate OCDM plugin:

```
curl -X PUT http://192.168.1.122:9998/Service/Controller/Deactivate/OCDM
```

Send a Scan command to the WIFI controller plugin:

```
curl -X PUT http://192.168.1.122:9998/Service/WifiControl/Scan
```

Update the WebkitBrowser config:

```
curl -X POST http://192.168.1.122:9998/Service/WebKitBrowser/Config --data "config.json"
```

## 4. WebSockets for events & notifications

The architecture of the WPEFramework offers event based feedback to clients. This functionality is realized using web socket connections to a plugin. Over these web socket connections, the plugin will notify the client on the other side of the web socket of events specific to the plugin.

The syntax for opening a web socket to the plugin is equal to the syntax of the RESTfull API:

```
Request: ws://<IP Address and port of WPEFramework>/Service/<Callsign>
protocol: notification
```

In Javascript, the socket would be opened like:

```
socket = new WebSocket("ws://192.168.1.122:9998/Service/Controller", "notification");
```

using wscat tool: (shows notification for OCDM plugin being deactivated)

```
09:23 $ wscat -c ws://192.168.1.122:9998/Service/Controller -s notification --no-color
connected (press CTRL+C to quit)
< {"callsign":"OCDM","state":"deactivated","reason":"Requested"}
< {"subsystems":{"Decryption":false}}
>
```

Using the *Controller* callsign, one can listen to all the notifications. If a specific plugin callsign is used when opening the websocket connection, only events from that plugin will be received.

## 5. Filesystem installation (config files & shared libs)

### 5.1 Config files

Config files are stored under /etc/WPEFramework:

```
root@pacexi5:/etc/WPEFramework# find .
.
./config.json
./plugins
./plugins/OCDM.json
./plugins/DeviceInfo.json
./plugins/Tracing.json
./plugins/Monitor.json
```

config.json is the main configuration file for the WPEframework and port number and network interface for the main HTTP API are set here:

```

"version": "1.0.0000000",
"port": 9998,
"binding": "0.0.0.0",
"ipv6": false,
"idletime": 180,
"persistentpath": "/opt",
"datapath": "/usr/share/WPEFramework",
"systempath": "/usr/lib/wpeframework/plugins",
"proxystubpath": "/usr/lib/wpeframework/proxystubs",
"redirect": "/Service/Controller/UI",

```

Each plugin has a json config file under /etc/WPEFramework/plugins. For example, OCDM plugin config file OCDM.json:

```

root@pacexi5:/etc/WPEFramework# cat plugins/OCDM.json
{
  "locator": "libWPEFrameworkOCDM.so",
  "classname": "OCDM",
  "precondition": [
    "Provisioning"
  ],
  "autostart": true,
  "configuration": {
    "outofprocess": true,
    "mapping": [
      {
        "key": "com.youtube.playready",
        "system": "PlayReady"
      },
      {
        "key": "com.microsoft.playready",
        "system": "PlayReady"
      },
      {
        "key": "com.widevine.alpha",
        "system": "WideVine"
      }
    ]
  }
}

```

Generic plugin properties like preconditions, autostart and outofprocess are set here as well as plugin specific ones. In this case, supported KeySystems (DRMs) are listed in the config file.

## 5.2 Executables

Two executables are installed in /usr/bin:

```

/usr/bin/WPEFramework-1.0.0000000
/usr/bin/WPEProcess -> WPEProcess-1.0.0000000 (each out-of-process plugin will run in a separate WPEProcess)
/usr/bin/WPEProcess-1.0.0000000
/usr/bin/WPEFramework -> WPEFramework-1.0.0000000 (main WPEFramework process, started with systemd, Controller and in-process plugins run in this process)

```

## 5.3 Shared libs

wpeframework core libraries are in /usr/lib:

```

root@pacexi5:/usr/lib# ls -l libWPEFramework*
libWPEFrameworkCore.so -> libWPEFrameworkCore.so.1
libWPEFrameworkCore.so.1 -> libWPEFrameworkCore.so.1.0.0000000
libWPEFrameworkCore.so.1.0.0000000
libWPEFrameworkCryptalgo.so -> libWPEFrameworkCryptalgo.so.1
libWPEFrameworkCryptalgo.so.1 -> libWPEFrameworkCryptalgo.so.1.0.0000000
libWPEFrameworkCryptalgo.so.1.0.0000000
libWPEFrameworkPlugins.so -> libWPEFrameworkPlugins.so.1
libWPEFrameworkPlugins.so.1 -> libWPEFrameworkPlugins.so.1.0.0000000
libWPEFrameworkPlugins.so.1.0.0000000
libWPEFrameworkProtocols.so -> libWPEFrameworkProtocols.so.1
libWPEFrameworkProtocols.so.1 -> libWPEFrameworkProtocols.so.1.0.0000000
libWPEFrameworkProtocols.so.1.0.0000000
libWPEFrameworkTracing.so -> libWPEFrameworkTracing.so.1
libWPEFrameworkTracing.so.1 -> libWPEFrameworkTracing.so.1.0.0000000
libWPEFrameworkTracing.so.1.0.0000000

```

OCDM library is also part of WPEFramework:

```

root@pacexi5:/usr/lib# ls -l libocdm.so*
libocdm.so -> libocdm.so.1
libocdm.so.1 -> libocdm.so.1.0.0000000
libocdm.so.1.0.0000000

```

Plugin shared libs are located in /usr/lib/wpeframework:

```

root@pacexi5:/usr/lib/wpeframework# find .
./plugins/libWPEFrameworkMonitor.so
./plugins/libWPEFrameworkOCDM.so
./plugins/libWPEFrameworkWebKitBrowser.so
./plugins/libWPEFrameworkTraceControl.so
./plugins/libWPEFrameworkDeviceInfo.so
./proxystubs
./proxystubs/libWPEFrameworkInterfaces.so.1.0.0000000
./proxystubs/libWPEFrameworkInterfaces.so
./proxystubs/libWPEFrameworkProxyStubs.so.1
./proxystubs/libWPEFrameworkProxyStubs.so.1.0.0000000
./proxystubs/libWPEFrameworkInterfaces.so.1
./proxystubs/libWPEFrameworkProxyStubs.so

```

## 6. Memory Usage

WPEFramework runs as a systemd service. The main process is named WPEFramework and runs the main controller plugin as well as the other in-process plugins (such as monitor, tracing etc).

The out-of-process plugins such as OCDM plugin run in separate "WPEProcess" processes. For example, in a configuration with Trace, Monitor and DeviceInfo in-process plugins and OCDM out-of-process plugin, the following processes are present:

```

root@pacexi5:/etc/WPEFramework# ps ax | grep -i wpe

11317 ? Ssl 0:00 /usr/bin/WPEFramework -b
11340 ? Sl 0:00 WPEProcess -a /usr/bin/ -c OCDMImplementation -d /usr/share/WPEFramework/OCDM/ -i 85 -l
libWPEFrameworkOCDM.so -m /usr/lib/wpeframework/proxystubs/ -p /opt/OCDM/ -r /tmp/communicator -s /usr/lib
/wpeframework/plugins/

```

Top command output:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25712	root	20	0	105156	3040	2432	S	0.3	0.5	0:00.08	WPEFramework
25737	root	20	0	322284	6120	5064	S	0.0	1.1	0:00.19	WPEProcess

The main WPEFramework process "WPEFramework" uses ~**3MB** RES memory with monitor, trace and deviceinfo plugins enabled. WPEProcess (OCDM) plugin is using **6MB** (in idle state, it uses more when there are active DRM sessions).

## 7. Disk space usage

Plugins: (Monitor, OCDM, TraceControl, DeviceInfo, WebkitBrowser) use about **409KB** of file space:

```

root@pacexi5:/usr/lib/wpeframework# du -ah
75.0K ./plugins/libWPEFrameworkMonitor.so
103.0K ./plugins/libWPEFrameworkOCDM.so
104.0K ./plugins/libWPEFrameworkWebKitBrowser.so
67.0K ./plugins/libWPEFrameworkTraceControl.so
59.0K ./plugins/libWPEFrameworkDeviceInfo.so
409.0K ./plugins

```

Core shared libraries use about **600KB** of file space:

```

root@pacexi5:/usr/lib# du -ah libWPEFramework*.so.1.0.0000000
146.0K libWPEFrameworkCore.so.1.0.0000000
34.0K libWPEFrameworkCryptalgo.so.1.0.0000000
152.0K libWPEFrameworkPlugins.so.1.0.0000000
162.0K libWPEFrameworkProtocols.so.1.0.0000000
39.0K libWPEFrameworkTracing.so.1.0.0000000

67.0K libocdm.so.1.0.0000000

```

ProxyStubs use an additional of **207KB**:

```

root@pacexi5:/usr/lib/wpeframework/proxystubs# du -ah
143.0K ./libWPEFrameworkInterfaces.so.1.0.0000000
63.0K ./libWPEFrameworkProxyStubs.so.1.0.0000000

```

Executables use **367KB**:

```

root@pacexi5:/usr/bin# du -ah WPE*
276.0K WPEFramework-1.0.0000000
91.0K WPEProcess-1.0.0000000

```

## 8. WebKitBrowser plugin & rdkbrowser2

WebKitBrowserPlugin allows WPEFramework clients to launch WPE instances, set the URL, suspend&resume browser processes. It works similar to rdkbrowser2 but instead of the rtRemote interface, HTTP API is used.

## Further Reading:

- [Common structure of RDK Services](#)
- [How to Make a Thunder Nano Service](#)
- [Splitting out-of-container rdkservices/thunderplugin .so for memory reduction and expected container security split](#)