# SOC JS Playback Engine – SDK Services - High Level Specifications

V.1.03

Author: marius alexandru

Contents

[SOC JS Playback Engine – SDK Services - High Level Specifications 1](#_Toc56682516)

[I. SDK Services – General Scope 2](#_Toc56682517)

[II. SDK Services – Class Hierarchy 3](#_Toc56682518)

[II.A. SDK Services – Class Hierarchy - Diagram 3](#_Toc56682519)

[II.B. SDK Services – Class Hierarchy – Description 3](#_Toc56682520)

[III. SDK Services – Interfaces 5](#_Toc56682521)

[III.A. SDK Services – Interfaces - Diagram 5](#_Toc56682522)

[III.B. SDK Services – Interfaces – Description 5](#_Toc56682523)

# SDK Services – General Scope

The scope of SDK Services is to ensure a multiplatform compatibility of the SOC Multiplatform Player

The **“SDK” Services** - is a hierarchy of SDK services having as root class A\_SDK\_JsTV class (with A - meaning “*A*bstract”). This hierarchy will encapsulate the specificity of each TV Signage OS and also (via additional subclasses) the specificity of each new Firmware inside each TV OS.

**Why are needed?**

The implementation of the SDK Services is needed in order to encapsulate all the specificity of different WebTV OS Systems.

All the other services, that used the SDK functionalities as clients, will work only with the SDK abstract service “A\_SDK\_JsTV” in order.

This “A\_SDK\_JsTV” (A is the notation for abstract) will offer a set of interfaces that will not depend on the specificity of each system

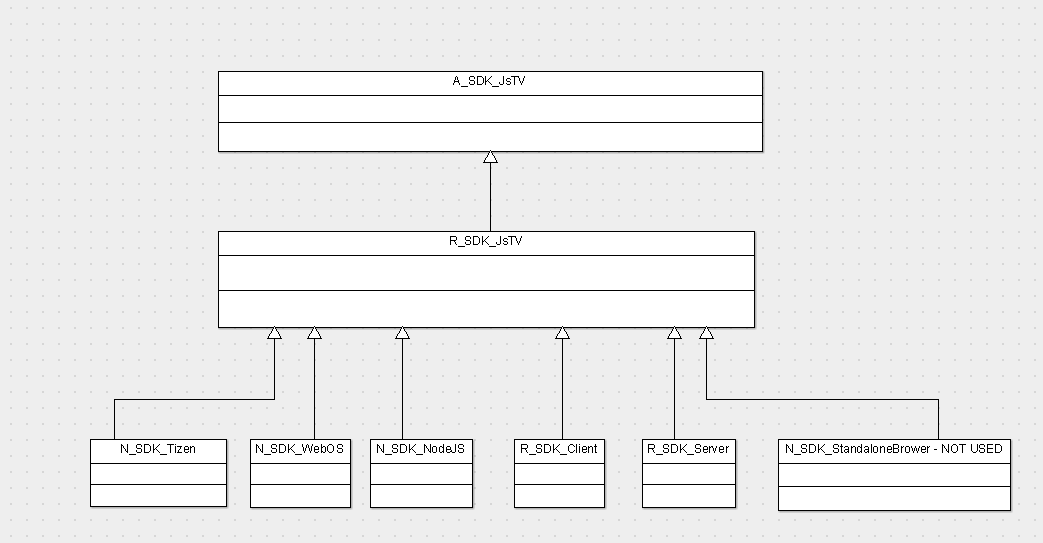
There are 2 types of Java Script/TV “system specificity” :

* A major one : TV OS Types / Node.js / etc.. (ex. Tizen OS, WebOS, Node.js etc...)
* A minor one : The “”ongoing” versions inside a specific OS (= firmware/SDK versions)

This situation is modeled via a hierarchy a SDK classes that will have on top the abstract service A\_SDK\_JsTV

# SDK Services – Class Hierarchy

## II.A. SDK Services – Class Hierarchy - Diagram



## II.B. SDK Services – Class Hierarchy – Description

1. A\_SDK\_JsTV – The abstract root class of the hierarchy. It defines the SDKs abstract Interfaces

(see next chaper).

All the SDK services are derived from this root class.

The abstract SDK layer is the layer that ensures the multi-platform compatibility

1. R\_SDK\_JsTV – Keeps the SDK common functionalities. This is not a full SDK implementation but

it keeps the reusable SDK functionalities that are based on native SDK functionalities.

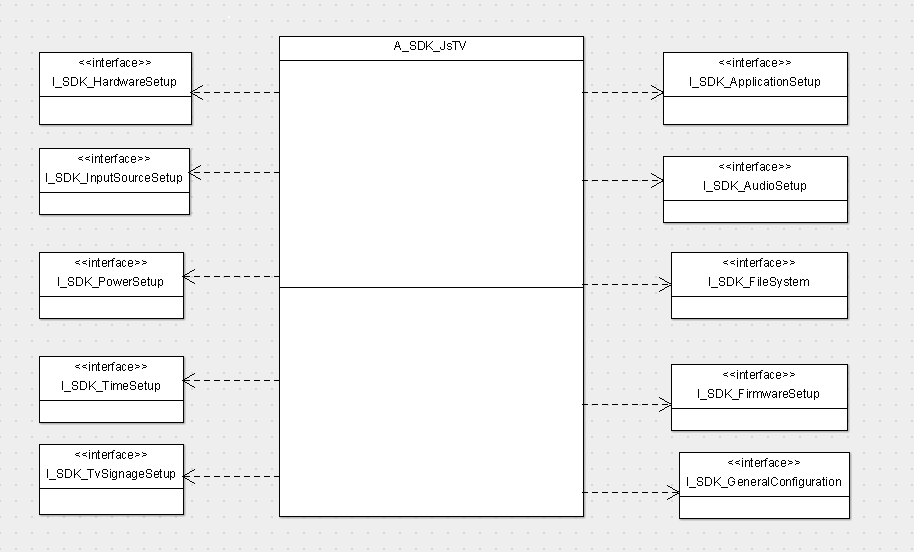
1. N\_SDK\_Tizen – The SDK encapsulation of Tizen SDK. Current TIZEN SDK that is targeted is SSSP6
2. N\_SDK\_WebOS – The SDK encapsulation of WebOS SDK, currently targeting WebOS 4.06
3. N\_SDK\_NodeJs – The SDK encapsulation of NodeJs. This ensures that the SOC multiplatform player can work on standard operating systems (Windows, Linux, Android, IOS)
4. R\_SDK\_Client – the SDK Client part in a client-server architecture. For instance is used only in conjunction with NodeJs SDK Service
5. R\_SDK\_Server – the SDK Server part in a client-server architecture. For instance is used only in conjunction with NodeJs SDK Service

NOTE: On the standard operating systems (Windows, Linux, Android, …) the application has a client-server architecture. With Nodejs on the server part, and a WebBrowser (ussualyy Chrome) running the client part.

1. R\_SDK\_StandaloneBrowser – NOT USED – The SDK encapsulation designed to acees the system functionalities directly from a Web Browser was removed – due to the fact that Google will not support in the future their system api from Chrome (outside Chrome OS)

# SDK Services – Interfaces

## III.A. SDK Services – Interfaces - Diagram



## III.B. SDK Services – Interfaces – Description

1. I\_SDK\_ApplicationSetup – This is the SDK interface that supplies application management functionalities (like application installation etc.)
2. I\_SDK\_AudioSetup – This is the SDK interface that supplies system audio setup (like volume up/down etc.)
3. I\_SDK\_FileSystem – This is the SDK interface that supplies file system functionalities (like filecopy, readFile, writeFile, createFolder, etc.)
4. I\_SDK\_FirmwareSetup - This is the SDK interface that supplies firmware management functionalities (like firmware installation and updates etc.)
5. I\_SDK\_GeneralConfiguration - This is the SDK interface that supplies general system fucntionalities (setVirtualKeyboardLanguage, on screen menu language, set/get general system properties)
6. I\_SDK\_HardwareSetup – This is the SDK functionalities that supplies the Hardware functionalities (like getWifiList, getNetworkProperties, getWifiProperties, getPlatformInfo, etc.)
7. I\_SDK\_InputSourceSetup - This is the SDK interface that supplies video input source setup (like HDMI1, HDMI2, etc…)
8. I\_SDK\_PowerSetup - This is the SDK interface that supplies power functionalities setup (like executePowerCommand : “restart”, “shutdown”, setSavePowerMode, etc)
9. I\_SDK\_TimeSystem – This is the SDK interface that supplies System Time setup and retrieval.
10. I\_SDK\_TvSignageSetup - This is the SDK interface that supplies TV Signage general functionalities (like capturePrintScreen, getSignageInfo, setPortraitMode etc..)

Note: After the main core of the SDK functionalities was implemented in the first phase, the rest of the SDK functionalities will be implemented whenever an Application Service will need them.