



Software Development Kit

Application Development and UI Guidelines

Status: Final
Version: 5.0
Date: 15 January 2016
Author: Smart TV Alliance inc.
Category: Official

© Smart TV Alliance inc. 2012-2016
All rights are reserved. Reproduction or transmission in whole or in part, in any form or by any means, electronic, mechanical or otherwise, is prohibited without the prior written consent of the copyright owner

1. CHANGE HISTORY	5
2. INTRODUCTION.....	6
2.1. OVERVIEW	6
2.2. CONVENTIONS AND STYLES.....	6
2.3. USAGE OF CODE SAMPLES	6
2.4. DEFINITIONS.....	7
2.5. REFERENCES	8
2.6. TRADEMARKS AND COPYRIGHTS	9
3. GUIDELINES.....	11
3.1. INTRODUCTION	11
3.2. GENERAL	11
3.2.1. <i>HTML5</i>	11
3.2.2. <i>Diversity handling</i>	11
3.2.3. <i>Identifying the client</i>	11
3.2.4. <i>Determining optional capabilities (specification 3.0+ only)</i>	12
3.2.5. <i>General</i>	15
3.2.6. <i>Timers</i>	17
3.2.7. <i>XMLHttpRequest</i>	17
Introduction.....	17
JSON.....	17
CORS.....	17
3.2.8. <i>Eval</i>	18
3.2.9. <i>Popups</i>	18
3.3. KEYS AND POINTER.....	18
3.3.1. <i>Overall</i>	18
3.3.2. <i>Back navigation</i>	19
General	19
Exit method (specification 3.0+ only).....	19
3.3.3. <i>In-page keyboard navigation</i>	20
3.3.4. <i>In-page pointer navigation</i>	21
3.3.5. <i>Text entry</i>	21
3.3.6. <i>Data storage and autocomplete</i>	22
3.4. AUDIO AND VIDEO	22
3.4.1. <i>HTML5 video object</i>	22
3.4.2. <i>Streaming</i>	22
HTTP Live Streaming (HLS).....	23
Smooth Streaming.....	23
MPEG-DASH.....	23
3.4.3. <i>HTTP server</i>	23
3.4.4. <i>Play control keys</i>	23
3.4.5. <i>Subtitles</i>	24
General	24
Optional: in-band/out-of-band subtitle support (specification 3.0+ only).....	24
3.4.6. <i>Multi (track) audio</i>	26
3.5. IMAGES	27
3.6. CASCADING STYLE SHEETS	28
3.6.1. <i>Introduction</i>	28
3.6.2. <i>CSS3 Transforms</i>	28
3.6.3. <i>CSS3 Transitions</i>	29
3.6.4. <i>CSS3 Animations</i>	30
3.6.5. <i>CSS3 Image Values and Replaced Content</i>	30
3.6.6. <i>CSS3 Multi-column layout</i>	31
3.7. COOKIES.....	31
3.8. FONTS	31
3.9. MULTISCREEN USING DIAL (SPECIFICATION 2.5+ ONLY)	32
3.10. NETWORK SERVICE DISCOVERY (OPTIONAL IN SPECIFICATION 4.0+ ONLY).....	32
3.11. PAYMENT SOLUTIONS.....	33

3.12.	OVERALL APP RECOMMENDATIONS	33
3.13.	DEVELOPMENT TOOLS AND FRAMEWORKS	33
4.	UI DESIGN REQUIREMENTS	36
4.1.	INTRODUCTION	36
4.2.	EXPLANATORY TEXT FOR MANDATORY UI ELEMENTS	37
4.2.1.	<i>Screen Layout</i>	37
4.2.2.	<i>Font</i>	37
4.2.3.	<i>Visual Treatment</i>	37
4.2.4.	<i>Navigation Scheme</i>	38
4.2.5.	<i>On-screen Button</i>	38
4.2.6.	<i>Back Behaviour</i>	39
A.	APP OPTIMIZATION REFERENCES	41
B.	INDEX OF IMPORTANT GUIDELINES	43

1. Change history

Version	Date	Changes
2.0	2013-02-26	Final for V2.0
2.5	2013-08-26	Final for V2.5
3.0	2014-01-07	Final for V3.0
3.0.1	2014-11-20	Final for V3.01 (Section 3.11 is updated)
4.0	2015-06-09	Final for V4.0
5.0	2016-01-09	Final for V5.0

2. Introduction

2.1. Overview

This document provides important guidelines for developing an App for the TV Device. It has a number of goals:

- Provide you with a number of easy to implement solutions for common problems and caveats in developing a Smart TV App, and thus allowing for a shorter development time.
- Help to achieve a certain level of quality for all Smart TV Apps.
- Provide an explanation for certain choices that were made in the TV Device, so you may better understand the reasons behind these choices.
- Give additional explanation for some requirements that your UI design needs to conform to (e.g. font size, safe area, on-screen controls).

Also refer to the Smart TV Alliance SDK manual for addendums with regard to the SDK software, and to the included samples, code snippets and templates. Details on the supported API's are documented in the API Reference document. Additionally, you can discuss application development for the Smart TV Alliance on the forums provided on <https://developers.smarttv-alliance.org>.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

This guidelines document refers to Specification version 5.0, but most information also applies to Specification 4.0 to 2.0 compliant devices. Where information is specifically only applicable for 5.0 compatible devices, this is indicated in the paragraph.

Finally, while a lot of care has been taken to ensure the correctness of the information in this document, errors cannot be completely prevented. The latest version of this document, with possible corrections, is always available online. If you have questions and/or remarks regarding these guidelines, please post them through the designated support channels.

2.2. Conventions and styles

In this document, a number of code samples are provided. A code sample is displayed like this:

```
<!doctype html>
<html>
<head>
<title>Basic Example </title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body style="width:1280px;height:720px;margin:0px;overflow:hidden;">
Hello, world.<br/>
<br/>
This is a basic HTML 5 page.
</body>
</html>
```

Important hints for developing the app are formatted as follows:

! This is an important hint for developing your app.

Where needed, references are made to the standards on which the TV Device is based. These references are formatted as [n], where [n] is the referred document in 2.5.

2.3. Usage of code samples

All code samples can be used freely in your own code. However, the following terms apply on this code - explaining that you get no warranty on any of the code:

Acceptance of Terms of Use: By accessing and using this software you agree to be bound by the following Terms of Use and all terms and conditions contained and/or referenced herein or any additional terms and conditions set forth on this software and all such terms shall be deemed accepted by you. If you do NOT agree to all these Terms of Use, you should NOT use this software. If you do not agree to any additional specific terms which apply to particular Content (as defined below) then you should NOT use the part of the software which contains such Content.

These Terms of Use may be amended by the author at any time. Such amended Terms of Use shall be effective upon posting of this software. Other Software enclosed herein may have their own terms of use which apply to such Software. Also, specific terms and conditions may apply to specific content, products, materials, services or information contained in or available through this software (the Content). Such specific terms may be in addition to these Terms of Use or, where inconsistent with these Terms of Use, only to the extent the content or intent of such specific terms is inconsistent with these Terms of Use, such specific terms will supersede these Terms of Use.

The author reserves the right to make changes or updates with respect to or in the Content of the software or the format thereof at any time without notice. The author reserves the right to terminate or restrict access to the software for any reason whatsoever at its sole discretion.

Although care has been taken to ensure the accuracy of the information on this software, the author assumes no responsibility therefore. ALL CONTENT IS PROVIDED AS IS AND AS AVAILABLE. THE AUTHOR HEREBY EXPRESSLY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, NON-INFRINGEMENT, OR AS TO THE OPERATION OF THIS SOFTWARE OR THE CONTENT. THE AUTHOR DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS AS TO THE SECURITY OF THIS SOFTWARE. YOU ACKNOWLEDGE ANY INFORMATION SENT MAY BE INTERCEPTED. THE AUTHOR DOES NOT WARRANT THAT THE SOFTWARE OR THE SERVERS WHICH MAKE THIS SOFTWARE AVAILABLE OR ELECTRONIC COMMUNICATIONS SENT BY THE AUTHOR ARE FREE FROM VIRUSES OR ANY OTHER HARMFUL ELEMENTS.

IN NO EVENT SHALL THE AUTHOR OR ANY OF ITS AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS, CONTRACT, REVENUE, DATA, INFORMATION OR BUSINESS INTERRUPTION) RESULTING FROM, ARISING OUT OF OR IN CONNECTION WITH THE USE OF, OR INABILITY TO USE THIS SOFTWARE OR THE CONTENT, EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. ANY ACTION BROUGHT AGAINST THE AUTHOR PERTAINING TO OR IN CONNECTION WITH THIS SOFTWARE MUST BE COMMENCED AND NOTIFIED TO THE AUTHOR IN WRITING WITHIN ONE (1) YEAR AFTER THE DATE THE CAUSE FOR ACTION AROSE.

This software may provide other Software that is not under the control of the author. The author shall not be responsible in any way for the content of such other Software. The author provides such software only as a convenience to the user of this software, and the inclusion of any such Software does not imply endorsement by the author of the content of such Software.

The software may contain references to specific products and services that may not be (readily) available in a particular country. Any such reference does not imply or warrant that any such products or services shall be available at any time in any particular country. Please contact your local business contact for further information.

WARRANTIES, IF ANY, WITH RESPECT TO SUCH SOFTWARE SHALL ONLY APPLY AS EXPRESSLY SET FORTH IN THE APPLICABLE LICENSE AGREEMENT. THE AUTHOR HEREBY EXPRESSLY DISCLAIMS ALL FURTHER REPRESENTATIONS AND WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR NON-INFRINGEMENT WITH RESPECT TO THE SOFTWARE.

2.4. Definitions

DOM	Document Object Model
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group (compression format)
PNG	Portable Network Graphics
SDK	Software Development Kit
XHTML	Extensible HyperText Markup Language
JSON	JavaScript Object Notation

(TV) Device The Smart TV device capable of running applications and receiving services in accordance with the Smart TV Alliance specification ([4], [4a], [4b], [4c], [4d]). For example, this may be a television, or a device that connects to a television such as a Blu-ray player or set top box.

2.5. References

[1] HTML5 Candidate Recommendation 17 December 2012
<http://www.w3.org/TR/2012/CR-html5-20121217/>

[2] HTML5 Logo - (CC BY 3.0 - <http://creativecommons.org/licenses/by/3.0/>) ;
<http://www.w3.org/html/logo/>

[3] ECMAscript Language Specification (5.1 Edition), June 2011,
<http://www.ecma-international.org/ecma-262/5.1/>

[4] Smart TV Alliance Technical Specification, Version 5.0

[4a] Smart TV Alliance Technical Specification, Version 2.5

[4b] Smart TV Alliance Technical Specification, Version 2.0

[4c] Smart TV Alliance Technical Specification, Version 3.0

[4d] Smart TV Alliance Technical Specification, Version 4.0

[5] REC-DOM-Level-2-20001113 Document Object Model (DOM) Level 2 Core Specification, Version 1.0,
<http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113>

[6] REC-DOM-Level-2-20001113 Document Object Model (DOM) Level 2 Style Specification, Version 1.0,
<http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113>

[7] REC-DOM-Level-2-20001113 Document Object Model (DOM) Level 2 Events Specification, Version 1.0,
<http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113>

[8] REC-DOM-Level-2-20030109 Document Object Model (DOM) Level 2 HTML Specification, Version 1.0,
<http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109>

[9] W3C - CSS 2.1 - April 2008
<http://www.w3.org/TR/2008/REC-CSS2-20080411/>

[10] HTTP State Management Mechanism
<http://tools.ietf.org/html/rfc6265>

[11] Persistent Client State: HTTP Cookies
http://wp.netscape.com/newsref/std/cookie_spec.html

[12] Apache mod_mime documentation ;
http://httpd.apache.org/docs/2.2/mod/mod_mime.html

[13] Software Development Kit - Application Development Diversity Handling Guidelines, latest version

[14] DIAL Discovery And Launch protocol specification Version 1.6.4
<http://www.dial-multiscreen.org/dial-protocol-specification>

[15] HTML5 Server-sent events Candidate Recommendation 11 December 2012
<http://www.w3.org/TR/2012/CR-eventsource-20121211/>

[16] HTML5 Web storage Proposed Recommendation 9 April 2013

<http://www.w3.org/TR/2013/PR-webstorage-20130409/>

[18] HTML5 Web Workers Candidate Recommendation 1 May 2012

<http://www.w3.org/TR/2012/CR-workers-20120501/>

[19] DOM Range

<http://dvcs.w3.org/hg/editing/raw-file/tip/editing.html#selections>

[20] HTML5 Developer's Cookbook; Chuck Hudson, Tom Leadbetter; Addison-Wesley 2012; ISBN 978-0-321-76938-1

[21] HTML5 Cookbook; Christopher Schmitt, Kyle Simpson; O'Reilly Media 2011; Ebook ISBN: 978-1-4493-9893-4

[22] Programming HTML5 Applications; Zachary Kessin; O'Reilly Media 2011; Ebook ISBN: 978-1-4493-9975-7

[23] Sergey's HTML5 & CSS3 : Quick Reference. HTML5, CSS3 and APIs 2nd ed.; Sergey Mavrody ; Belisso 2012; ISBN 978-0983386728

[24] EmbedJS framework -

<https://github.com/uxebu/embedjs>

[25] HTML5: The Definitive Guide, 7th Revised Edition; Chuck Musciano, Bill Kennedy, Estelle Weyl ; Safari Books Online, February 2015; ISBN 978-1-4493-0259-7

[26] Microsoft Smooth Streaming -

<http://www.iis.net/download/smoothstreaming>

[27] Timed Text Markup Language (TTML) 1.0 (Second Edition)

<http://www.w3.org/TR/ttaf1-dfxp/http://www.w3.org/TR/ttaf1-dfxp/>

[28] Webpagetest

<http://www.webpagetest.org/>

[29] Smart TV Alliance Application Requirements

<https://developers.smarttv-alliance.org/app-development>

[30] ETSI TS 102 796 v1.2.1 Hybrid Broadcast Broadband TV

http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.02.01_60/ts_102796v010201p.pdf

[31] Bento4 DASH tool

<http://www.bento4.com/developers/dash/>

2.6. Trademarks and copyrights

All trademarks and copyrights are the property of their respective owners.

3. Guidelines

3.1. Introduction

This chapter gives technical recommendations for developing a Smart TV Application (“*STA App*”). It is recommended that you keep a copy of the referred standards available as a reference.

3.2. General

3.2.1. HTML5



A Smart TV App is based on the HTML5 profile as specified in the Smart TV Alliance Specification, Version 5.0 [4], Version 4.0 ([4d]), Version 3.0 ([4c]), 2.5 ([4a]) or Version 2.0 ([4b]).

([2])

⚠ **Refer to the Specification document for technical documentation of the supported HTML5 profile.**

Generic information/background on creating HTML5 applications can be found in a number of recent publications ([20][21][22][23][25]). Also refer to [29].

3.2.2. Diversity handling

The Smart TV Alliance Specification describes the common set of API's which are supported on all TV Devices. However, devices are made by different manufacturers and as such could still have slight differences. These differences ('diversity') can be captured in three categories:

- means of identifying the device to the Smart TV App (only for Specification 2.0 ; for devices compliant to Specification 2.5 and later, a new device identification method is available (also refer to 3.2.3)
- additional diversity handling, including optional features of the STA specification, that may be implemented only in specific platforms or device-ranges. For devices compliant to Specification 3.0 and later, a new capability method is available that allows you to check the presence of certain optional capabilities on the device your App is running on. Refer also to 3.2.4.
- specific user interface requirements due to device design

Because of this, these guidelines also include a means for your App to easily handle the device differences ('diversity handling'). This guideline-document will mostly be generic in nature, and thus not describe specific diversity handling for all STA-platforms. It will instead refer to the separate diversity handling document where applicable ([13]).

3.2.3. Identifying the client

For Smart TV Alliance Specification 2.5 and later devices, a new Smart TV Alliance generic platform identification mechanism has been added using an extension to the user agent string. This extension to the user agent string is "**SmartTvA/5.0.0**" for Specification 5.0, "**SmartTvA/4.0.0**" for Specification 4.0, etc. An example of a (part of the) user agent string in a 5.0 compliant devices would be:
" Opera/9.80 (Linux mips (.....) Presto/2.6.33 Version/10.70 SmartTvA/5.0.0"

Additional information on the specific platforms is available in the diversity handling document [13].

This document also lists the means by which you can identify a device compliant to Specification 2.0 using a Javascript code sample ("smarttv_getPlatform()").

The Smart TV Alliance Specification is available in multiple versions. In general the lower version of the Specification has less features but a wider install-base. Please target this version (minimum version 2.0) of the specification for your application to assure the widest reach. To use advanced features of newer versions of the specification like multiscreen (supported in specification 2.5 and later), you can use the above identification method to enable these features for specific platforms complying with that Smart TV Alliance Specification version. For determining certain optional features, also refer to 3.2.4.

⚠ Please target the lowest Smart TV Alliance Specification Version (minimum version 2.0) and use the available (version/capability) methods to determine newer and/or optional features.

3.2.4. Determining optional capabilities (specification 3.0+ only)

Certain device capabilities are set to optional in the Smart TV Alliance specification. For some of these capabilities it is possible to determine the presence of these options using a special method. This method is called `hasCapability()` ([4] 3.3.6.2.1) and is part of the `SmartTvA_API` object that is available on platforms compliant to Specification 3.0 and later. Where indicated below, some capabilities only exist in later specifications than 3.0.

boolean SmartTvA_API.hasCapability(query[,param1,....,paramN])			
Description	Query an optional or conditionally mandatory function on the TV Device.		
Arguments	Name	Type	Description
	Query	String	This string is the optional or conditionally mandatory function name to query support for. <i>Note: this string is case sensitive.</i>
	Params	String	This string contains additional query information.as a variable number of param arguments (param1, ..., paramN). The number of arguments depends on the query string (refer to hasCapability method arguments table). <i>Note: this string is case sensitive.</i>
Return value	Boolean	It is set to true if the given function is supported, otherwise false.	

hasCapability() method arguments table

Query	Param1	Param2	Description	Remark
3DSupport			Return value is set to true if receiver can display 3D video in side-by-side and top-bottom formats, otherwise false.	See [4] section 3.4.4.1
Key	numerickeys		Return value is set to true if global VK_ key constants corresponding to all defined numeric keys (VK_0, VK_1, ..., VK_9) are supported, otherwise false.	See [4] section 3.3.4
	colorkeys		Return value is set to true if global VK_ key corresponding to all color keys (VK_RED, VK_GREEN, VK_YELLOW, and VK_BLUE) are supported, otherwise false.	See [4] section 3.3.4
	prev_next		Return value is set to true if VK_PREV and VK_NEXT are supported, otherwise false.	See [4] section 3.3.4
Multiscreen	AllJoyn		Return value is set to true if	See [4] section 3.7.2

			AllJoyn is supported, otherwise false.	
DRM	PlayReady	DASH	Return value is set to true if PlayReady in combination with MPEG-DASH are supported according to the specification, otherwise false.	See [4] section 3.5
	Widevine	AdaptiveStreaming	Return value is set to true if Widevine in combination with Widevine Adaptive Streaming and Widevine API are supported according to the specification, otherwise false. Note: No space is included in "AdaptiveStreaming".	See [4] section 3.5
Multiaudio			Return value is set to true if Multi Audio is supported, otherwise false. On 4.0+ platforms, always true.	See [4] section 3.4.7
TTML	inband		Return value is set to true if TTML in band (part of the stream) subtitles are supported according to the specification, otherwise false. On 4.0+ platforms, always true.	See [4] section 3.4.6.1.3
	outofband		Return value is set to true if TTML out of band (separately loaded) subtitles are supported according to the specification, otherwise false.	See [4] section 3.4.6.1.4
	EBU-TT-D		Return value is set to true if EBU-TT-D subtitles are supported, otherwise false.	See [4] section 3.4.6.1.2
	CFF-TT		Return value is set to true if CFF-TT subtitles and supported, otherwise false.	See [4] section 3.4.6.1.3
UHD			UHD is set to true if supported according to the specification ([4],[4d]), otherwise false.	See [4] section 3.4.8
UHD	MainTier	5.1	Return value is set to true if Video codecs level up to Main Tier 5.1 is supported, otherwise false. (4.0+ only)	See [4] section 3.4.8
UHD	HLS		Return value is set to true if HLS/MPEG2-TS supported, otherwise false. (4.0+ only)	See [4] section 3.4.8
NSD			Network Service Discovery is supported according to this specification. This includes support for the mandatory	See [4] section 3.3.2

			discovery protocols. (4.0+ only)	
NSD	Zeroconf		Network Service Discovery and zeroconf are supported according to this specification. If this flag is set to true, the simple query for 'NSD' shall also return true. (4.0+ only)	See [4] section 3.3.2.2
NSD	DIAL		Network Service Discovery and DIAL are supported according to this specification. If this flag is set to true, the simple query for 'NSD' shall also return true. (4.0+ only)	See [4] section 3.3.2.2
EMEMSE			Return value is set to true if EME/MSE are supported according to this specification, otherwise false. (4.0+ only)	See [4] section 3.6.2, 3.6.3
EMEMSE	Widevine		Return value is set to true if Widevine EME is supported according to this specification, otherwise false. (4.0+ only)	See [4] section 3.6.2
HLS	DISCONTINUITY		Return value is set to true if DISCONTINUITY tag is supported according to this specification, otherwise false. (5.0+ only)	See [4] section 3.4.2.2 <i>Note: HLS is always supported</i>
HDR	HDR10		Return value is set to true if HDR10 is supported according to this specification, otherwise false. (5.0+ only)	See [4] section 3.4.8
HDR	DV		Return value is set to true if Dolby Vision is supported according to this specification, otherwise false. (5.0+ only)	See [4] section 3.4.8

Example of using the hasCapability() method in your App:

```
// informative, you have to determine this version number based on the UA string, e.g.
"SmartTvA/5.0.0" or later ; refer to 3.2.3
var userAgent = navigator.userAgent;
var smarttv_platform = [];
// check if this is a Smart TV Alliance 2.5+ compliant platform
substr_pos=userAgent.search(/SmartTvA/i);
if (substr_pos > -1)
{
    // this is a Smart TV Alliance 2.5+ compliant platform with a new UA identification
    smarttv_platform["type"]=userAgent;
    smarttv_platform["manufacturer"]="Smart TV Alliance";
    smarttv_platform["version"]=userAgent.substr(substr_pos+9,6);
}
```

```

    smarttv_platform["major"]=parseInt(smarttv_platform["version"].split('.')[0]);
    smarttv_platform["minor"]=parseInt(smarttv_platform["version"].split('.')[1]);
    smarttv_platform["revision"]=parseInt(smarttv_platform["version"].split('.')[2]);
}
// test if the SmartTvA_API object exists and STA version is 5 (major)
if (typeof SmartTvA_API != "undefined" && smarttv_platform["major"] == 5)
{
    // determine capabilities for version 5.0.0 compliant device

    // test 3d support
    var 3dsupport = SmartTvA_API.hasCapability('3DSupport');

    // test numerickeys support
    var numerickeys = SmartTvA_API.hasCapability('Key', 'numerickeys');
    // test colorkeys support
    var colorkeys = SmartTvA_API.hasCapability('Key', 'colorkeys');

    // test AllJoyn multiscreen support according to specification 5.0.0
    var alljoyn = SmartTvA_API.hasCapability('Multiscreen', 'AllJoyn');

    // test Playready + MPEG-DASH combination DRM support according to specification 5.0.0
    var playready_dash = SmartTvA_API.hasCapability('DRM', 'PlayReady', 'DASH');

    // test Widevine + Adaptive Streaming combination DRM support according to
specification 5.0.0
    var widevine_adaptive = SmartTvA_API.hasCapability('DRM', 'Widevine',
'AdaptiveStreaming');

    // test Multi track audio support
    var multiaudio = SmartTvA_API.hasCapability('Multiaudio');

    // test TTML inband support according to specification 5.0.0
    var ttml_inband = SmartTvA_API.hasCapability('TTML', 'inband');

    // test TTML out-of-band support according to specification 5.0.0
    var ttml_outofband = SmartTvA_API.hasCapability('TTML', 'outofband');

    // test Ultra High Definition support according to specification 5.0.0
    var ultrahd = SmartTvA_API.hasCapability('UHD');

    // for version 4.0+ specific devices only
    var ultrahd_maintier_51 = SmartTvA_API.hasCapability('UHD', 'MainTier', '5.1');
    var ultrahd_hls = SmartTvA_API.hasCapability('UHD', 'HLS');

    var nsd = SmartTvA_API.hasCapability('NSD');
    var nsd_zeroconf = SmartTvA_API.hasCapability('NSD', 'Zeroconf');
    var nsd_dial = SmartTvA_API.hasCapability('NSD', 'DIAL');

    var ememse = SmartTvA_API.hasCapability('EMEMSE');
    var ememse_widevine = SmartTvA_API.hasCapability('EMEMSE', 'Widevine');

    var hlsdiscontinuity = SmartTvA_API.hasCapability('HLS', 'DISCONTINUITY');

    var hdr_hdr10 = SmartTvA_API.hasCapability('HDR', 'HDR10');
    var hdr_dv = SmartTvA_API.hasCapability('HDR', 'DV');
}

```

3.2.5. General

In terms of writing code, it is important to optimize the code as much as possible for the TV Device which potentially has less computing performance compared to that of a typical PC platform. In general, optimization for speed can be accomplished by avoiding complex, compound statements. Nested loops should be 'rolled out' as much as possible. Website compression is not always that helpful, as some of the mainstream compression technologies are aimed at a fast PC platform. . However, filtering out unnecessary

tags and CSS code - as often generated by HTML authoring frameworks - as well as using shorter function and variable names can be useful. It is better to optimize code to run on your server than on the client, where this is possible and when bandwidth requirements allow.

While debugging your code, it is also good practice to enable "strict mode" for the Javascript engine. This can help prevent the use of (old/ECMAScript 3) statements in your code that may not work on some newer browsers that deprecate certain statements. Note that setting strict mode does potentially generate more exceptions. To use this mode, include the following statement on top of your Javascript code-block or function: *"use strict"*;

```
<script type="text/javascript">
"use strict";
..
// alternatively per function
function ()
{
  "use strict";
  ..
}
```

As a precautionary note, execution time between the Platform and SDK differs significantly. If e.g. a Javascript function is called outside the <body>-onload, a DIV-element that you call in your Javascript code could still be loading and thus not be present in the DOM-tree when executed on a slower device. This would cause Javascript errors, with as a symptom that your code works on a fast PC, but not on the device. To prevent this, one solution is to call your code from within a <body>-onload function. However, extra caution is needed for this method, since the page will not be shown (completely) until all onload functions have been executed. Also, the behaviour of the onload function handler is browser specific with regards to the execution time. Try not to execute all Javascript code all at once, but use e.g. a timeout-timer to delay each step a few seconds.

The following is a code sample of how you should **not** call your Javascript functions:

```
<script type="text/javascript">
  function myNonWorkingFunction()
  {
    document.getElementById('myNonPresentDiv').innerHTML='My non working text';
  }
  myNonWorkingFunction();
</script>
</head>
<body>
  ...
  <div id="myNonPresentDiv">My initial text</div>
</body>
```

A more optimized code using the body-onload is provided as follows:

```
<script type="text/javascript">
  function myWorkingFunction()
  {
    document.getElementById('myPresentDiv').innerHTML='My working text';
  }
</script>
</head>
<body onload="myWorkingFunction();" >
  ...
  <div id="myPresentDiv">My initial text</div>
</body>
```

⚠ Try to execute all required Javascript functions starting in the body onload.

In the past, a large number of books and sites have been dedicated to optimizing web site code. Some of these sources can be useful for your own development purposes. A list of websites and books is given in appendix A. You should be aware that most of these sources of information have not been written with Smart TV in mind, and may contain code that is not valid for a TV Device.

3.2.6. Timers

Javascript can be used for rapid dynamic page changes, such as animations. For this purpose, timers can be used with very small time granularity value. In an embedded environment, such as the TV Device, any value less than 100ms is typically not practical. Note that in version 2.0 and later of the specification, CSS3 animations are also supported.

Using many timers at once on the same page has the same implications: it does not work smoothly on an embedded system.

! Use timers sparsely, increase granularity of frequently recurring timers.

3.2.7. XMLHttpRequest

Introduction

The following is an excerpt for using the XMLHttpRequest (XHR) object:

```
xhr_req = new XMLHttpRequest();
...
xhr_req.onreadystatechange = xhr_response;
xhr_req.open("GET", "http://myserver/mydata.xml", true);
xhr_req.send(null);
...
```

JSON

To increase performance, one can use JavaScript Object Notation (JSON) instead.

The browsers used for the TV Device have support for native JSON parsing using `JSON.parse()` as part of Javascript. This native implementation is much faster than using a library, and therefore recommended.

Example, which will set the innerhtml of an element to the text "content 1" using native JSON parsing:

```
<script type="text/javascript">
  "use strict";
  function exampleJSON()
  {
    var myjson_object = JSON.parse('{"Tag_name": "object", "Tag_properties": {"data":
" MyData"}, "Tag_content": ["content 0", "content 1"]}');
    document.getElementById("smarttvalliance").innerHTML=myjson_object.Tag_content[1];
  }
  ...
</script>
```

CORS

To allow XHR -access to resources that are not on the same domain as your app, CORS can be used. It is a two-step approach that adds a few statements to your XHR request, and needs a few changes to your HTTP-server. A simple example using PHP on the server-side is shown below.

Javascript part, hosted on e.g. "http://my_sta_app_server/cors.html":

```
<script type="text/javascript">
  ...
  function XHRRequest()
  {
    var xhr_req = new XMLHttpRequest();
    xhr_req.withCredentials = true; // allow exchange of e.g. cookies
    xhr_req.open("GET", "http://testserver/cors.php", true);
    ...
  }
</script>
```

Server part (PHP), hosted on e.g. "http://testserver/cors.php" :

```
<?php
```

```
header("Access-Control-Allow-Origin: my_sta_app_server.com");
// alternatively "*" iso "my_sta_app_server" allows access from all origins
header("Access-Control-Allow-Credentials: true"); // allow exchange of e.g. cookies
?>
```

3.2.8. Eval

Your app should not use eval, except for use in JSON, although you should rely on the native JSON parse methods included in the browser for this. Eval in general is bad coding practice. Note that if you use the recommended approach to debugging your app in Javascript strict mode, eval will not be allowed in most cases.

3.2.9. Popups

As the Smart TV Alliance browsers only support one window, popups are not supported. Do not use e.g. window.open methods in your code. For overlays on your page (for e.g. debugging purposes) you could use simple <div>-elements. Of course for debugging your application it is better to use the advanced tooling offered by the SDK emulator and IDE.

Do not use popups when exiting your application.

! When exiting your application, no pop ups must be generated.

3.3. Keys and pointer

3.3.1. Overall

Key constants are used in an app to refer to remote control keys. An example is the key constant VK_ENTER, which refers to the 'select' or 'OK'-button on a remote control. These constants are translated to key codes that apply to the device your app is running on.

As these key constants remain the same for future and current common platforms of different manufacturers, your app should use these instead of key codes (e.g. 13).

! Use VK_ constants instead of keycodes.

Key constants are defined globally. For certain platforms, they are supplied with this SDK and you need to define them in your application. Please refer to the diversity handling documentation ([13]).

Example of using key constants:

```
<!DOCTYPE html>
<html>
<head>
<title>Basic key constants</title>
<style type="text/css">
  a:link { color: white }
  a:visited { color: white }
  a:focus { color: yellow }
  a:hover { color: yellow }
</style>
<script type="text/javascript">

  // define key constants here as per keyconstants.js

function keydownhandler(e)
{
  switch (e.keyCode)
  {
    case VK_UP:
      goto_key("key_0");
      break;
    case VK_DOWN:
      goto_key("key_1");
      break;
    default:
      break;
```

```

}
}

function goto_key(key_id)
{
  document.getElementById(key_id).style.background="red";
}

document.addEventListener("keydown", keydownhandler, true);
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body style="margin:0px;overflow:hidden;">
  <div id="key_0"
  style="float:left;margin:10px;height:50px;width:15px;background:blue;color:white;"><a
  href="#"><h2>0</h2></a></div>
  <div id="key_1"
  style="float:left;margin:10px;height:50px;width:15px;background:blue;color:white;"><a
  href="#"><h2>1</h2></a></div>
</body>
</html>

```

Note (**specification version 3.0 and later only**): for certain optional keys present on the device, like colorkeys (VK_BLUE, VK_RED etc.) and numerickeys (VK_0, VK_1, etc.), the VK_-constants are (only) available if the respective option is enabled in the hasCapability()-method. Refer also to 3.2.4.

3.3.2. Back navigation

General

For proper user experience, the back behaviour of your application has to conform to the following:

1. **proper exit**: your app shall return control back to the source application that called your Smart TV app when your Smart TV app is exited AND only when this is supported by the device. For platforms compliant to specification version 3.0 and later, refer to the exit() method below. Also refer to the diversity handling guidelines.
2. **time of exit**: your app shall exit when the user navigates back 'beyond' the first (entry) page of your Smart TV app, only when this is supported by the device.
3. **no pop-ups**: your app may never generate pop-ups when navigating back (e.g. to ask for conformation of exit of the app)
4. **support VK_BACK**: your app shall support the physical back key (VK_BACK) on the remote for navigating back in your application ; note that for AJAX applications the VK_BACK key can only be overridden in a keypress eventhandler while strictly adhering the exit conditions (when supported on the device).

Exit method (specification 3.0+ only)

For platforms compliant to Specification version 3.0 and later, a method is available to exit your App. This method is called exit() and is part of the SmartTvA_API object. When called, you signal that your App is ready to exit and control is given back to the device on which your App is run. Refer also to ([4c] 3.3.5.2.2).

Exit()	
Description	Notify the TV Device that an application is ready to exit and returns control back to the device. Note: The end user's expectation is to navigate back to the source application that called the application. The device is expected to meet this user experience, but there could be exceptions and the behavior following this call is device dependent.
Arguments	None
Return value	None

Example of exiting your App:

```

// informative, you have to determine this version number based on the UA string, e.g.
"SmartTvA/5.0.0" or later ; refer to 3.2.3
var userAgent = navigator.userAgent;
var smarttv_platform = [];
// check if this is a Smart TV Alliance 2.5+ compliant platform
substr_pos=userAgent.search(/SmartTvA/i);
if (substr_pos > -1)
{
    // this is a Smart TV Alliance 2.5+ compliant platform with a new UA identification
    smarttv_platform["type"]=userAgent;
    smarttv_platform["manufacturer"]="Smart TV Alliance";
    smarttv_platform["version"]=userAgent.substr(substr_pos+9,6);
    smarttv_platform["major"]=parseInt(smarttv_platform["version"].split('.')[0]);
    smarttv_platform["minor"]=parseInt(smarttv_platform["version"].split('.')[1]);
    smarttv_platform["revision"]=parseInt(smarttv_platform["version"].split('.')[2]);
}

// test if the SmartTvA_API object exists
if (typeof SmartTvA_API != "undefined" && smarttv_platform["major"] >= 3)
{
    // my app is ready to exit
    SmartTvA_API.exit();
    // control is handed back to the device from this point
}

```

⚠ Use the available exit-methods for returning control back to the device from your App.

3.3.3. In-page keyboard navigation

Navigation in a page can be accomplished using three different methods:

1. CSS3 based spatial navigation
2. Javascript based spatial navigation
3. Proprietary spatial navigation offered by the browser (on some platforms only)

The recommended and best way to navigate in a page is the CSS3 based spatial navigation. An example is given here:

```

<!DOCTYPE>
<html>
<head>
<title>Basic Spatial navigation using CSS3</title>
<style type="text/css">
    a:focus { background: yellow }
    a:hover { background: red }
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body style="margin:0px;overflow:hidden;">
<a id="a1" href="#" style="position:absolute;top:0 ;left:0 ;nav-index:1;nav-
down:#b1;nav-right:#a2;nav-left:#a1;nav-up:#a1">A1</a>
<a id="a2" href="#" style="position:absolute;top:0 ;left:50%;nav-index:2;nav-
down:#b2;nav-right:#a3;nav-left:#a1;nav-up:#a2">A2</a>
<a id="a3" href="#" style="position:absolute;top:0 ;left:90%;nav-index:3;nav-
down:#b3;nav-right:#a3;nav-left:#a2;nav-up:#a3">A3</a>

<a id="b1" href="#" style="position:absolute;top:50%;left:0 ;nav-index:4;nav-
down:#c1;nav-right:#b2;nav-left:#b3;nav-up:#a1">B1</a>
<a id="b2" href="#" style="position:absolute;top:50%;left:50%;nav-index:5;nav-
down:#c2;nav-right:#b3;nav-left:#b1;nav-up:#a2">B2</a>
<a id="b3" href="#" style="position:absolute;top:50%;left:90%;nav-index:6;nav-
down:#c3;nav-right:#b1;nav-left:#b2;nav-up:#a3">B3</a>

```

```
<a id="c1" href="#" style="position:absolute;top:90%;left:0 ;nav-index:7;nav-
down:#a1;nav-right:#c2;nav-left:#c3;nav-up:#b1">C1</a>
<a id="c2" href="#" style="position:absolute;top:90%;left:50%;nav-index:8;nav-
down:#a2;nav-right:#c3;nav-left:#c1;nav-up:#b2">C2</a>
<a id="c3" href="#" style="position:absolute;top:90%;left:90%;nav-index:9;nav-
down:#a3;nav-right:#c1;nav-left:#c2;nav-up:#b3">C3</a>
</body>
</html>
```

In essence, CSS version 3 allows the developer to set the navigation direction for each navigable element on a page. This method is the most versatile way to control how items on a page are navigated, and is preferred for both performance and usability reasons.

It is also possible to use the proprietary spatial navigation of the browser. This method is not guaranteed to work for each device, as the browser is not guaranteed to be the same. Therefore the behaviour of navigation can be different or non-existent on other platforms, and its usability is limited.

Because proprietary spatial navigation is also based upon internal calculations over which the developer has no control, it could be that the navigation works illogical or some elements cannot be reached.

In some clients, CSS3 spatial navigation can be combined with built-in spatial navigation.

Finally, there is the possibility to use Javascript to control spatial navigation. This method is not recommended, as it could hamper performance on all but the simplest pages (pages with more than a few navigable links). The size of the page alone will increase because of the Javascript code included for navigation even when there are is a limited number of links on the app.

3.3.4. In-page pointer navigation

Make sure any navigation your app implements works also with a pointer (e.g. mouse). Please refer to the design guidelines in this document for information on how to keep the design visually pointer navigable.

Your page must be navigable by both keys as a pointer device.

Some attention points (further described in the UI Requirements below):

- Navigation in your page should be done using anchor-elements of a size that also allow clicking. For this, the elements should be of a reasonable size. Navigating your page using DIV-elements with custom navigation schemes is not recommended because of this.
- When you can navigate to an element using the Up-Down-Left-Right keys, you should be able to click on that element using the pointer button.
- When you implement an action using e.g. the OK-button (VK_ENTER), you should be able to access that action using the pointer (by e.g. a click-action). In some instances, typically play control, this means that you need to support on-screen play control keys (described below).
- Double highlights in the page can be prevented using the 'onmouseout' event handler in a navigable (anchor) element.

```
<a onmouseout='javascript:RemoveHighlight(this);'
onfocus='javascript:AddHighlight(this);' onblur='javascript:RemoveHighlight(this);'
...></a>
```

3.3.5. Text entry

For text entry, the TV Device can provide a number of different built-in methods:

- on-screen built-in keyboard: the user sees a keyboard that they can control using the up-down-left-right keys to input text. In some platforms, this keyboard only pops up after the user pressed OK on a text-field. The keyboard is implemented as an overlay.
- on-screen keyboard provided in the app: for some platforms that do not support the built-in on-screen keyboard or other means of text entry, a separate keyboard shall be included within the app. A detailed code is provided and to avoid work, developers can copy and paste this information. The keyboard is implemented as an overlay.
- the so called 'multitap' or 'SMS-tap' method as available on certain platforms. With this method, a user can use the 0-9 keys to select letters and symbols for a text entry field. The user will get on-screen feedback when they change a character. This multitap-method is built-in: you do not need to make your own multitap method.

To determine whether a TV Device supports the built-in keyboard or the keyboard provided in the app, refer to the diversity handling documentation. Your app *can* default to the self-provided on-screen keyboard regardless of diversity, but on some platforms that also provide the built-in keyboard the behaviour to the user can be confusing.

For Smart TV Apps, you can restrict your TV Device to accept the text field input as only numeric. This is done by defining the input type field as "number". An example:

```
<input type="text"/>
<input type="number"/>
```

Only the second field in this example provides numeric input.

Different input methods across various apps can be confusing to the user. If in doubt, depend on the provided input methods.

! Use the provided text entry methods as much as possible.

3.3.6. Data storage and autocomplete

If you wish to allow the user to enter a large amount of data, you should be aware that this could make your app very hard to use if done repeatedly. For this, it is recommended you implement a form of autocomplete in your app: a means by which the user can skip re-entering large amounts of text-data.

Any storage of personal data on/in your app needs to be in line with legal and privacy requirements.

! Any storage of (personal) data on/in your application needs to be in line with any legal and privacy requirements.

A use case in which this could come in handy is the input of address information. For all autocomplete methods in which you store sensitive data, you should always require the user to login with a username and password combination *before* allowing the stored data to be retrieved and used in the app.

! Autocomplete in your page for sensitive data? Always authenticate the user.

3.4. Audio and video

3.4.1. HTML5 video object

Support for the HTML5 video object is included in the device. An example of using a video object:

```
<body style="margin:0px;overflow:hidden;"
onload="document.getElementById('videoobj').play();">
<video id="videoobj" style="width:300px;height:300px" src="/myvideo/myvideo.mp4">
</video>
```

3.4.2. Streaming

The capabilities of the streaming client differ per device and are documented in the device specification ([4c], [4]).

The TV Device basically supports two streaming methods:

- regular HTTP (1.1)
- adaptive streaming

The regular HTTP streaming method is using a similar server as the server that hosts your content. Refer also to 3.4.3. For adaptive streaming, there are a few different options with their implementation details described below. Most of these formats are also supported in the SDK software, so you can test your stream:

HTTP Live Streaming (HLS)

Set the src-tag of the video element to the M3U playlist file of the HLS stream. Treat as a regular video element.

Smooth Streaming

Set the src-tag of the video element to the manifest file of the Smooth Streaming stream ([26]). Treat as a regular video element.

MPEG-DASH

Set the src-tag of the video element to the MPD file of the MPEG-DASH stream. Treat it as a regular video element.

Note that the MPEG-DASH file has to be compliant to the ISO/BMFF Live Profile as described in Annex E of [30]. You can use tools such as Bento4 ([31]) to generate a compliant stream.

3.4.3. HTTP server

For any video or audio file, the HTTP server that provides the content should identify the file with the proper mimetype. If you use Apache as HTTP server, this can be accomplished using the directive "AddType" from mod_mime [11]. Or you can edit the file "mime.types" and add the supported mime-types to this file.

Example of the additions in Apache:

```
...
AddType video/mp4 .mp4 .mpeg4
AddType video/x-ms-wmv .wmv .asf
AddType audio/x-ms-wma .wma
AddType audio/mpeg .mp3
...
```

mod_mime.conf

For MP3 content, note that your HTTP stream must be compatible with the specification. As an example, any metadata besides MP3 data in the HTTP stream will not function.

3.4.4. Play control keys

When playing video or audio, your app needs to support remote control keys to control the playout of the media content. As not all platforms support (all) remote control keys in every situation, play control keys also need to be supported on-screen. This is also described in the UI guidelines below. You are free to implement your own on-screen play keys, but an example is included for reference purposes, refer to the separate code snippets. The easiest method to support both the remote control as on-screen play control keys is to let them use the same functionality. The on-screen keyboard shall be hidden when the user is playing the content full screen, and appear for full screen when the user presses the OK key on the remote control, after which a selection can be made. This key handling does not apply to windowed mode playback (as the onscreen play controls can be always shown there). It is recommended that the on-screen keyboard has a timeout that hides it after a few (e.g. 3) seconds.

For certain platforms, your app needs to implement specific on-screen keys in the play control. Refer to the separate included example code and the diversity handling guidelines for more information.

⚠ Always implement both remote control as on-screen play controls

```
<script type="text/javascript">
function play_video()
{
  document.getElementById("myvideo").play();
}

function keydownhandler (e)
{
  if (e.keyCode == VK_ENTER)
  {
    // show playcontrol keyboard on OK for full screen video
    // link the play-action to play_video();
  }
}
```

```

}
if (e.keyCode == VK_PLAY)
{
    play_video();
}
}

document.addEventListener("keydown", keydownhandler, true);
</script>
...

```

3.4.5. Subtitles

General

If you want to use subtitles in your app, you can use various subtitling solutions, depending on the device on which you run your application. The following external subtitling solutions work with all the current TV Devices:

1. Popcorn - <http://popcornjs.org> (with support for TTML [27] based subtitles)

In the separate code snippets, an example is included how to call e.g. popcorn JS in your application. It is recommended that you make use of TTML based subtitles in your Smart TV App. Make sure that you style your subtitles according to the minimum font size: the default of these libraries may be much smaller than required on a TV Device. For some libraries, this could mean you may have to apply a small patch to the library beforehand to change the font-size.

⚠ When you use one of the app based subtitling solutions, make sure the font size is correct.

Optional: in-band/out-of-band subtitle support (specification 3.0+ only)

Certain platforms conformant to Specification 3.0 and later support the option of displaying native TTML based subtitles either "in band" (as part of the stream) or "out of band" (delivered separately). These options are available if the capability method (3.2.4) returns true for the given option. In-band subtitles are optionally supported as part of the stream based on Smooth Streaming [26]. Out-of-band subtitles are delivered separately over HTTP.

Selecting of subtitles in the App occurs via the TextTrack/TextTrackList-interface.

The TTML profile that is supported by a device implementing either one of these methods is as follows:

- `<p begin="tt:begin" end="tt:end">Timed subtitle text</p>` - text of the given subtitle
- `tt:begin` - start time of the text (hh:mm:ss:ff)
- `tt:end` - end time of the text (hh:mm:ss:ff)

Example of a conformant TTML subtitle:

```

<?xml version="1.0" encoding="utf-8"?>
<tt xml:lang="en"
    xmlns="http://www.w3.org/ns/ttml"
    xmlns:tts="http://www.w3.org/ns/ttml#styling"
    xmlns:ttm="http://www.w3.org/ns/ttml#metadata">
  <head>
  </head>
  <body>
    <div>
      <p begin="00:00:01:00" end="00:00:04:00">First text.</p>
      <p begin="00:00:05:00" end="00:00:09:00">Second text.</p>
    </div>
  </body>
</tt>

```

For a TTML subtitle that is delivered in-band, selection occurs as follows:


```

..
<script type="text/javascript">
function select_subtitle()
{
// informative, you have to determine this version number based on the UA string, e.g.
"SmartTvA/4.0.0", "SmartTvA/5.0.0" or later ; refer to 3.2.3
var userAgent = navigator.userAgent;
var smarttv_platform = [];
// check if this is a Smart TV Alliance 2.5+ compliant platform
substr_pos=userAgent.search(/SmartTvA/i);
if (substr_pos > -1)
{
// this is a Smart TV Alliance 2.5+ compliant platform with a new UA identification
smarttv_platform["type"]=userAgent;
smarttv_platform["manufacturer"]="Smart TV Alliance";
smarttv_platform["version"]=userAgent.substr(substr_pos+9,6);
smarttv_platform["major"]=parseInt(smarttv_platform["version"].split('.')[0]);
smarttv_platform["minor"]=parseInt(smarttv_platform["version"].split('.')[1]);
smarttv_platform["revision"]=parseInt(smarttv_platform["version"].split('.')[2]);
}

// test if the SmartTvA_API object exists
if (typeof SmartTvA_API != "undefined" && smarttv_platform["major"] >= 3)
{
// test TTML support according to specification 3.0.0+
var ttml_inband = SmartTvA_API.hasCapability('TTML', 'inband');

if (ttml_inband)
{
// in-band TTML subtitles are supported in this device according to specification
3.0.0+
var myvideo=document.getElementById('videoobj');
// get the total number of text tracks in the stream
var texttrack_length = myvideo.textTracks.length;
if (texttrack_length > 0)
{
// select the first in-band TTML text track
var mytexttrack = myvideo.textTracks[0];
if (mytexttrack.mode != "showing")
{
// select this subtitle and start displaying it (mode:showing)
mytexttrack.mode="showing";
}
}
}
}
}
}
</script>
..
<video id="videoobj" style="width:1280px;height:720px" src="http://myserver/
/mystream.ism/Manifest"></video>
..

```

Here, one or more subtitles are available inside the Smooth Streaming manifest file. Selection of the TTML subtitle that is to be displayed occurs in Javascript. From specification 4.0 onwards, TTML in-band selection is always present in a compliant device.

For a TTML subtitle that is delivered out-of-band, selection and loading occurs as follows:

```

..
<script type="text/javascript">
function select_subtitle()
{
var userAgent = navigator.userAgent;
var smarttv_platform = [];
// check if this is a Smart TV Alliance 2.5+ compliant platform

```

```

substr_pos=userAgent.search(/SmartTvA/i);
if (substr_pos > -1)
{
    // this is a Smart TV Alliance 2.5+ compliant platform with a new UA identification
    smarttv_platform["type"]=userAgent;
    smarttv_platform["manufacturer"]="Smart TV Alliance";
    smarttv_platform["version"]=userAgent.substr(substr_pos+9,6);
    smarttv_platform["major"]=parseInt(smarttv_platform["version"].split('.')[0]);
    smarttv_platform["minor"]=parseInt(smarttv_platform["version"].split('.')[1]);
    smarttv_platform["revision"]=parseInt(smarttv_platform["version"].split('.')[2]);
}

// test if the SmartTvA_API object exists
if (typeof SmartTvA_API != "undefined" && smarttv_platform["major"] >= 3)
{
    // test TTML out-of-band support according to specification 3.0.0
    var ttml_outofband = SmartTvA_API.hasCapability('TTML', 'outofband');

    if (ttml_outofband)
    {
        // out-of-band TTML subtitles are supported in this device according to
specification 3.0.0+
        var myvideo=document.getElementById('videoobj');
        // get the total number of text tracks in the stream
        var texttrack_length = myvideo.textTracks.length;
        if (texttrack_length > 0)
        {
            // select the first out-of-band TTML text track
            var mytexttrack = myvideo.textTracks[0];
            if (mytexttrack.mode != "showing")
            {
                // select this subtitle and start displaying it (mode:showing)
                mytexttrack.mode="showing";
            }
        }
    }
}
}
</script>
..
<video id="videoobj" style="width:1280px;height:720px" src="http://myserver/myvid.mp4">
    <track src="http://myserver/mysubtitle1.ttml" language="en"></track>
    <track src="http://myserver/mysubtitle2.ttml" language="nl"></track>
</video>
..

```

Here, two TTML track elements are available over HTTP and signalled as part of the video element. The selection occurs in a similar way as for the in-band TTML selection.

3.4.6. Multi (track) audio

Just as with subtitles, you can play streaming media files with multiple audio tracks as defined in ([4] 3.4.7.4). For e.g. specification 5.0 devices, this would mean that multi audio can be supported in Smooth Streaming (MSS) and MPEG-DASH compliant files. The method by which to include multiple audio tracks in the file is depending on the protocol and not described here, please refer to ([4] 3.4.7.4) for further information.

When you want to select a specific audiotrack in your video file, regardless of file format, you can use the following example code:

```

..
<script type="text/javascript">
function select_audiotrack()
{
    var userAgent = navigator.userAgent;
    var smarttv_platform = [];
    // check if this is a Smart TV Alliance 2.5+ compliant platform

```

```

substr_pos=userAgent.search(/SmartTvA/i);
if (substr_pos > -1)
{
    // this is a Smart TV Alliance 2.5+ compliant platform with a new UA identification
    smarttv_platform["type"]=userAgent;
    smarttv_platform["manufacturer"]="Smart TV Alliance";
    smarttv_platform["version"]=userAgent.substr(substr_pos+9,6);
    smarttv_platform["major"]=parseInt(smarttv_platform["version"].split('.')[0]);
    smarttv_platform["minor"]=parseInt(smarttv_platform["version"].split('.')[1]);
    smarttv_platform["revision"]=parseInt(smarttv_platform["version"].split('.')[2]);
}

// test if the SmartTvA_API object exists
if (typeof SmartTvA_API != "undefined" && smarttv_platform["major"] >= 3)
{
    // test Multi track audio support
    var multiaudio = SmartTvA_API.hasCapability('Multiaudio');

    if (multiaudio)
    {
        var myvideo=document.getElementById('videoobj');
        // get the total number of text tracks in the stream
        var audiotrack_length = myvideo.audioTracks.length;
        // disable all audiotracks
        for (var i=0; i < audiotrack_length; i++)
        {
            myvideo.audioTracks[i].enabled=false;
        }
        if (audiotrack_length > 0)
        {
            // select the first audio track
            var myaudiotrack = myvideo.audioTracks[0];
            if (!myaudiotrack.enabled)
            {
                // enable the audiotrack
                myaudiotrack.enabled=true;
            }
        }
    }
}
}
</script>
..
<video id="videoobj" style="width:1280px;height:720px" src="http://myserver/
/mystream.mpd"></video>
..

```

Note that TV devices will align the track list in the media file with the index of the tracks inside the video object. This way, in case you need to select a track based on any non STA-conformant parameters (such as e.g. audiotrack , you can do so by using the index-attribute.

3.5. Images

To include images in your pages, there are a few things which you should keep in mind to keep the performance of the app well.

Some recommendations regarding images: large and many images per page can result in slow loading pages, and large rendering times. They also limit the amount of data that can be cached for your app, leading to increased bandwidth usage.

In general the size of the images should be limited to the size of the screen. The device is not created for printing the images shown, therefore you can generally assume for e.g. a 1280x720 pixel screen a maximum image size of 1 megapixel. An exception to this rule can be the use of CSS sprites: images that

contain many sub-images and are used in combination with CSS to crop and select the correct image-asset - but even here using very large images can lead to a diminished performance.

If an image is meant purely as a background image - throughout your whole app - keep its size small to prevent slowing down loading times of your page, as (cache) memory is limited on the device.

⚠ (Cache) memory is limited on a TV Device, do not rely on it e.g. in use of large images.

For the number of images on a page, the maximum is defined by the memory available in the device browser. However, it should be clear that loading a large number of images will do the same as loading one very large image: it will slow down the app.

⚠ Limit image sizes to less than 1 megapixel. Limit the number of images displayed on a single page.

In general, it is better to use the *background-image* CSS property rather than an *img*-element because it is better performing. However, client-side scaling of this type of images is not easily possible.

For *img*-elements, if the size of the image you create is known, set the width and height of the image to prevent additional layout calculations while the page is loading - this increases the rendering speed of your page.

3.6. Cascading Style Sheets

3.6.1. Introduction

CSS is a worthwhile asset in making your app look attractive, while at the same time allowing your development to go smoothly. Refer to the web site optimization information in appendix A for some helpful guidelines in creating your CSS.

To speed up the layout of your page, CSS elements should be optimized for speed, not size. This means that the use of deeply nested styles is not recommended: the browser needs to recalculate the absolute position and size of the element based on the parent styles each time a layout of the page is required. Instead, specify the position and size of an element absolutely, so the browser does not have to do this calculation: this will increase the size of the CSS file, but using short-tag names (which can be accomplished using basic CSS compression) the transfer time of the file can be limited. While doing this, still try keeping your CSS files small by excluding unused styles.

Also, optimize your page to let the browser use the most optimal selector matching strategy. This means that for large pages, your app should use the most unambiguous CSS selectors such as "#id" or "a:link" instead of just ":link", as the browser will take much more time evaluating the last statement than the two first.

Below are some primers to CSS use cases that are supported in the latest version of the specification. For more background details, please refer to one of the numerous books written on CSS(3).

3.6.2. CSS3 Transforms

Transforms allow you to modify certain page elements in two-dimensions. You can modify the scale, position, rotation and angle of an element using various transformation functions. Advanced modifications are also possible using the matrix-transform function. It is possible to put multiple transformation functions in one statement.

Because CSS3 Transforms is officially a work in progress, different browsers could need specific browser-extensions in front of the CSS-style attribute that defines the transformation. Below example uses a best-practice method to guarantee that the transformation attribute will work on all current and future browsers. For more information, refer to the diversity handling documentation ([13]).

Example, this would move the image down and right by 100 pixels, rotate it 45 degrees and scale the image along the X-axis by a factor of 0.9:

```

<head>
...
<style type="text/css">
#smarttvalliance
{
  -o-transform: translate(100px,100px) rotate(45deg) scaleX(0.9);
  -webkit-transform: translate(100px,100px) rotate(45deg) scaleX(0.9);
  -moz-transform: translate(100px,100px) rotate(45deg) scaleX(0.9);
  -ms-transform: translate(100px,100px) rotate(45deg) scaleX(0.9);
  transform: translate(100px,100px) rotate(45deg) scaleX(0.9);

  background-image: url('smarttvalliance.png');
  background-repeat: no-repeat;
  width: 100px;
  height: 100px;
}
</style>
</head>
<body style="width:1280px;height:720px;margin:0px;">
...
  <div id="smarttvalliance"></div>
...
</body>

```

3.6.3. CSS3 Transitions

When modifying CSS properties, transitions allow you to animate these modifications over a period of time. Commonly transitions are used in combination with CSS3 transforms.

An example is shown below that modifies the transform example and adds a 5 second transition effect (note that the transformation is applied using Javascript *after* setting the transition effect):

```

<head>
<script type="text/javascript">
  function transform()
  {
    var sta_elm=document.getElementById("smarttvalliance");
    sta_elm.style.webkitTransform="translate(100px,100px) rotate(45deg) scaleX(0.9)";
    sta_elm.style.OTransform="translate(100px,100px) rotate(45deg) scaleX(0.9)";
    sta_elm.style.MozTransform="translate(100px,100px) rotate(45deg) scaleX(0.9)";
    sta_elm.style.msTransform="translate(100px,100px) rotate(45deg) scaleX(0.9)";
  }
</script>
<style type="text/css">
#smarttvalliance
{
  -o-transition: all 5s ease-in-out;
  -webkit-transition: all 5s ease-in-out;
  -moz-transition: all 5s ease-in-out;
  -ms-transition: all 5s ease-in-out;
  transition: all 5s ease-in-out;

  background-image: url('test.png');
  background-repeat: no-repeat;
  width: 100px;
  height: 100px;
}
</style>
</head>
<body style="width:1280px;height:720px;margin:0px;" onload="transform();">
<div id="smarttvalliance"></div>
</body>

```

3.6.4. CSS3 Animations

CSS3 animations offer a way to change a certain CSS property value over time, in a controlled manner. It gives more control over the timing than for example Javascript or CSS3 Transitions, and is useful when you want to link a certain animation to a user-interaction such as in simple games.

Example, this performs a simple position translation in 3 defined steps (keyframes), over a duration of 2 seconds, for an infinite number of times:

```
<head>
<style type="text/css">
...
@keyframes sta_key
{
  0% {top: 0px; left: 0px;}
  40% {top: 20px; left: 50px;}
  100% {top: 0px; left: 0px;}
}
@-webkit-keyframes sta_key
{
  0% {top: 0px; left: 0px;}
  40% {top: 20px; left: 50px;}
  100% {top: 0px; left: 0px;}
}
@-o-keyframes sta_key
{
  0% {top: 0px; left: 0px;}
  40% {top: 20px; left: 50px;}
  100% {top: 0px; left: 0px;}
}
@-moz-keyframes sta_key
{
  0% {top: 0px; left: 0px;}
  40% {top: 20px; left: 50px;}
  100% {top: 0px; left: 0px;}
}
#smarttvalliance
{
  -moz-animation: sta_key 2s infinite;
  -webkit-animation: sta_key 2s infinite;
  -o-animation: sta_key 2s infinite;
  animation: sta_key 2s infinite;

  background-image:url('test.png');
  background-repeat: no-repeat;
  width: 100px;
  height: 100px;
  position: absolute;
}
</style>
</head>
<body style="width:1280px;height:720px;margin:0px;">
...
<div id="smarttvalliance"></div>
...
</body>
```

3.6.5. CSS3 Image Values and Replaced Content

This provides known-items like url(), but also gradient-fill options as shown in the example below:

```
<head>
...
<style type="text/css">
#smarttvalliance
```

```

{
  background-image:-webkit-linear-gradient (rgb (213,76,109) ,
rgb (0,179,240) ,rgb (172,208,69) ) ;
  background-image:-o-linear-gradient (rgb (213,76,109) , rgb (0,179,240) ,rgb (172,208,69) ) ;
  background-image:-moz-linear-gradient (rgb (213,76,109) ,
rgb (0,179,240) ,rgb (172,208,69) ) ;
  background-image:-ms-linear-gradient (rgb (213,76,109) , rgb (0,179,240) ,rgb (172,208,69) ) ;
  background-image:linear-gradient (rgb (213,76,109) , rgb (0,179,240) ,rgb (172,208,69) ) ;
  width:100px;height:100px;left:100px;top:100px;
  background-repeat: no-repeat;
  position:absolute;
}
</style>
</head>
<body style="width:1280px;height:720px;margin:0px;">
<div id="smarttvalliance"></div>
</body>

```

3.6.6. CSS3 Multi-column layout

For providing additional layout options you can use this standard that allows you to specify columns for formatting content:

```

<head>
<style type="text/css">
#smarttvalliance
{
  width:1280px;height:70px;
  -webkit-column-count: 3;
  -o-column-count: 3;
  -ms-column-count: 3;
  -moz-column-count: 3;
  column-count: 3;
  font-size:18pt;
}
</style>
</head>
<body style="width:1280px;height:720px;margin:0px;">
<div id="smarttvalliance">This is a text distributed amongst multiple columns.<br/>This
is a text distributed amongst multiple columns.</div>
</body>

```

3.7. Cookies

Common mechanisms as described in ([9],[10]) can be used to generate a cookie. You can use cookies as a means by which to implement autocomplete on your app, which is recommended if you are requiring input of a large amount of text. As noted before, if cookies are used to store sensitive data, be sure to authenticate the user by a username and password combination before you retrieve the data and allow it to be used. Limit the amount of different cookies per device to the absolute necessary amount needed, as space on the device for cookie storage is limited. Each cookie with a unique name counts as one different cookie. Of course you are free to reuse the same cookie (name) as often as needed. Be aware that laws in certain countries require additional measures such as user consent before storing certain user data.

⚠ Limit the amount of different cookies (cookie names) per device.

3.8. Fonts

The Tiresias (or equivalent) font is supported by default for sans-serif text. To make sure you are using the proper fonts, follow the naming in the device documentation. To automatically switch to the correct font in case the naming changes or is slightly different for that device, always include a generic font-family name like in this example:

```

...
<p style="font-family:'Tiresias', sans-serif;font-size:20pt;">font test</p>
...

```

Here, the font "Tiresias" is set as font-style for the paragraph, with as backup the generic font-family name "sans-serif".

Also refer to minimum size requirements in the UI requirements below.

For specification 4.0 and later, you can additionally use downloadable fonts. You must supply fonts in both WOFF and TrueType format, as both downloadable font-formats are used in Smart TV's compliant to the specification 4.0+ in the field, and you cannot determine the exact supported font format. You can specify these two downloadable fonts additionally as follows in a single CSS statement:

```
...
<style type="text/css">
@font-face
{
  font-family: 'My Font Name';
  src: local('My Font Name'), url(http://myfont.com/myfont.woff) format('woff'),
url(http://myfont.com/myfont.ttf) format('truetype');
}
</style>
...
<p style="font-family:'My Font Name', sans-serif;font-size:20pt;">downloadable font
test</p>
...
```

3.9. Multiscreen using DIAL (specification 2.5+ only)

The Smart TV Alliance Specification 2.5/3.0/4.0/5.0 (par. 3.6) supports DIAL (Discovery And Launch) Multiscreen ([14]). It allows a companion device (e.g. tablet, phone) to discover and launch their application on a Smart TV Alliance host device. The SDK 2.5+ manual describes the process of testing the DIAL application on the SDK, and also refers to the included DIAL sample client applications for both IOS and Android.

To make it possible to launch your multiscreen capable application on a compliant device, it is required that you register your application separately. For more information refer to the developers website as indicated in 2.1.

3.10. Network Service Discovery (optional in specification 4.0+ only)

In certain devices from version 4.0 onwards, Network Service Discovery (NSD) is optionally supported as indicated through the capability object. When implemented, NSD allows a service to, when allowed by both the device as the user, recognize other devices available in the network. One of the uses of NSD is within the Smart Home.

An example of a Network Service Discovery page which simply lists the available services in the network is shown below:

```
...
<script type="text/javascript">
function showAll( mylist ) {
  for(var i = 0, l = mylist.length; i < l; i++)
  {
    // do something with mylist[i].name
    document.getElementById("myservice").innerHTML+="service: "+mylist[i].name+"<br/>";
  }
}

function showError( e )
{
  // throw error
  document.getElementById("myservice").innerHTML+="error: "+e.name+"<br/>";
}

function init()
{
```



```

if (typeof SmartTvA_API != "undefined")
{
  if (SmartTvA_API.hasCapability.apply(this, ["NSD"]))
  {
    if (navigator.getNetworkServices == undefined)
      return;
    var promiseResponse = navigator.getNetworkServices(["upnp:urn:schemas-upnp-
org:service:Dummy:1", "upnp:urn:dial-multiscreen-org:service:dial:1", "upnp:urn:schemas-
upnp-org:service:ContentDirectory:1",]).then(showAll, showError);
  }
}
}
</script>
<body onload="init();">
List of services:<br/>
<div id="myservice"></div>
</body>
...

```

Applications implementing NSD should take note of the following and of the security precautions in [4]:

- Make sure you are authorized by the user of the application prior to collecting data within the network. As devices will often ask for permission outside of the application scope, not asking permission or notifying the user of the imminent request, could lead to a denial by the user.
- Only enable this service when strictly required to perform NSD-related functions the user is aware of.
- If you receive invalid or unknown services in your application from e.g. a non-CORS enabled service that is not required for your application functionality, notify the user of the potential security risk and remove the service from the list.

3.11. Payment solutions

To integrate payment solutions in your page, you can continue to use your current payment methods, such as credit card or debit card, if these are integrated in the interface of your own app or are otherwise compatible with the navigation and input methods of the device.

If the payment process is continued outside of your own app the payment provider must make the app Smart TV Alliance compliant.

3.12. Overall app recommendations

The size and complexity of an app page has a large influence on speed. If your app exists of only 1 page, with different blocks of (hidden) content, performance can suffer. Try dividing your app in different (AJAX loaded) pages and/or limit the number of hidden blocks of content and elements your page has.

The use of redirect pages is not recommended: it confuses the user with regard to the app's back-behaviour, and makes your app take longer to load.

Additionally, after creating your site consider using the SDK emulator and possibly tools like Webpagetest ([28]) to determine basic performance characteristics and optimization tactics.

⚠ Keep pagesize small, limit the number of elements in a page.

When you design secure applications, e.g. such that require a payment interface, and load (part of) your application over HTTPS, please refrain from SSL in your server and instead rely on TLS 1.1. Reason is that multiple vulnerabilities were found in SSL versions 3.0 and older, and that using these protocols will make your app less secure.

⚠ Use only TLS secure protocols in your server, refrain from using SSL.

3.13. Development tools and frameworks

Development of an app for the 'generic web' has progressed significantly the last couple of years. Where a developer traditionally had to construct their own code from scratch, now it is possible to generate the

majority of the code using off the shelf tools: frameworks and libraries. Examples of common frameworks include:

- JQuery
- Dojo Toolkit
- Angular JS

These tools are designed to be used for web apps meant for a PC-based environment. As such, they can create very complex and large pages, requiring a fast(er) device to execute the app on. It is clear that code as complex as that which can be generated with a framework is generally not optimized for a TV Device.

If you still want to use a framework, make sure you check the generated code. Prevent using handy off the shelf code snippets to do such things as spatial navigation or JSON parsing, as sometimes provided in a framework: these may not be performing well on the device as they are based on Javascript code instead of more optimal solutions that use CSS3 or native Javascript functions as available on the TV Device. Additionally, there are frameworks that are potentially better fitting for a TV Device.

Examples of optimized frameworks for more constricted platforms:

- EmbedJS [24] - compatible with Dojo Toolkit

Manual editing and properly reviewing any generated code against the device on which it is to run is the safe choice. Especially check these items in your framework:

1. Does your framework compress code? Make sure it does this not by implementing non-optimal compression methods as discussed in this document.
2. Does your framework load code on a needed basis? Make sure that the code that is critical to your application is available upon first load, or the user will notice timeouts while using your code. Also note that due to network specifics of certain TV Devices, opening multiple HTTP(S) connections at the same time or consecutively could take much more time than on a PC platform - therefore limit the number of connections your page opens (simultaneously).
3. Is the Javascript code generated by your framework very large (>100KB)? Even though most of this code is not used by your application, the browser still needs to process, execute and cache part of this code. This will take up time both in loading and executing your page, sometimes it takes more than a few seconds : strip out unused code, load critical code first.
4. Does your framework provide methods that you could also implement easily yourself using simpler/smaller Javascript? Examine the code you use from the framework carefully; sometimes a method/construct from the framework you use only *once* in your code adds a large number of methods and properties from the framework to your code. This leads to slower execution and loading times: instead optimizing this single use case would provide a big performance gain for your app.
5. Can you limit the CPU time of your framework by executing certain methods 'statically'? In optimization there often is a tradeoff between CPU time and size of your code: by implementing some of the code statically (or server-side) instead of generating it runtime each time your app is executed, you could increase the performance significantly while only slightly increasing the size of your total code.
6. Is the code used by your framework HTML5 compliant, for e.g. Angular you have to make sure you use the data-ng-* elements instead of the non-compliant ng-* elements.

In the end, sometimes you may find that having the framework included in the first place and optimizing it makes your code more complex than if you had to implement the code from scratch, even though that method may seem to take more time at first.

⚠ Use frameworks sparsely; if you really need these: simplify code, limit network connections.

4. UI Design requirements

4.1. Introduction



Developing an app for the TV Device has implications on the way it is navigated and appears to users. The user interface of your app will have to be optimized to be usable on multiple platforms.

This chapter is not setup to guide you in developing the functionality and identity of your app. You are free to design your app as required. However, this chapter does give you the constraints to which your design shall conform.

The Smart TV Alliance developer website lists the current and complete requirements that apply to your application. You can review these at <https://developers.smarttv-alliance.org/app-development> ([29]).

UI Design for a TV viewing environment

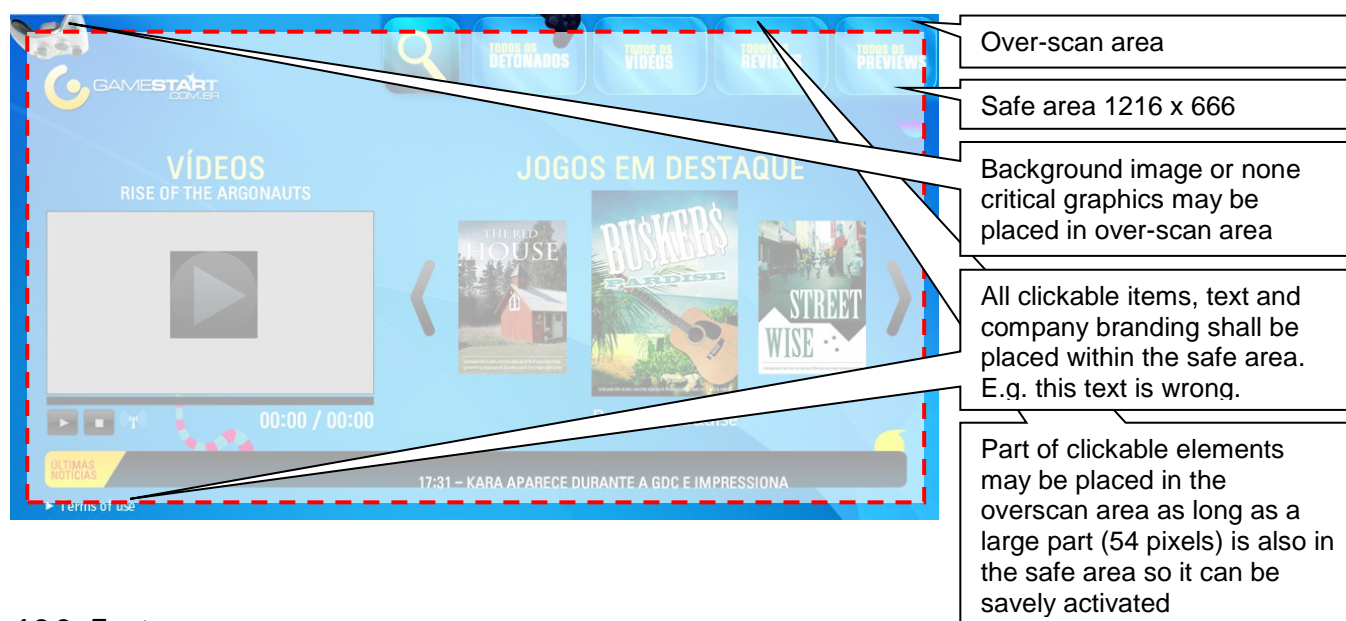
The TV viewing environment is defined below. As a guiding principle, a user experience that allows the user to control the TV and Media equipment with minimum effort is provided. Therefore, the user experience (UX) needs to be designed such that the Smart TV App is optimized for the TV viewing situation, ie:

		
Distance from Screen	10 Foot (3.5 Meter)	2 Foot (70cm)
Environment	Fun Environment Family/Social Environment	Work Environment Single-user Environment
User Behavior	Lean-back The user will typically be sitting 10 feet (3.5meter) away from the screen Passive Browsing Low concentration and involvement (user's role) Comfortable Posture Comfortable posture involving lying back or face downward	Lean-forward The user sits upright, control from a close range Active Interaction & Searching High concentration and involvement (user's role) Constrained Posture Constrained posture in front of the desk
User's Goal	Entertaining Primarily for entertainment Often there is no specific target/goal Single contents, exploring structure	Working Goal-oriented, based on various needs There is a specific target/goal Infinite type and number, searching structure

4.2. Explanatory text for Mandatory UI Elements

4.2.1. Screen Layout

- ✓ 1. The app (screen) size should support 1280 x 720 pixels.
- ✓ 2. Over-scan Area: All text, clickable items and company brand should be placed within the safe area of the screen. Safe area is 1216 x 666 pixels for the 1280 x 720 pixel resolution.



4.2.2. Font

3. Unused.

- ✓ 4. Minimum font size: The font size of all text in your app should always be at least 18 points. For comfortable reading we advise to use a 20 point font size or higher.

4.2.3. Visual Treatment

- ✓ 5. If there is a mouse over action or a focus action, the app should show a different highlight.: Idle/Selected



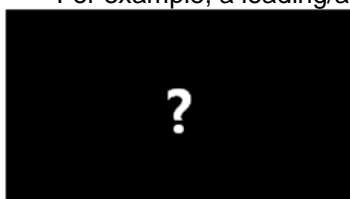
Both pointer focus and key focus should be allowed and implemented. As an example scenario, a user could be using a "standard remote" and switch to using a pointer remote. One thing to note when switching from the pointer remote, is that any first input of the standard remote should 1) visually hide the pointer and 2) move the pointer to position (0,0) and 3) prevent focus change for that single instance.

Refer to 3.3.4 for a technical explanation on how to fulfill part of this requirement.

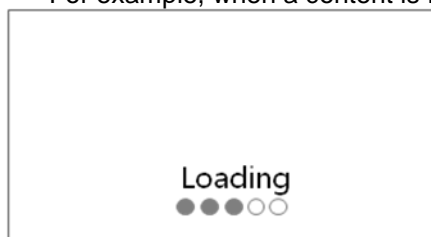
- ✓ 6. The App should have three different highlights for idle, focus and active and should show the highlight for 'active' if the element is active. (e.g. 'tab-like' items or 'category- indicating').



- ✓ 7. The app should never leave the screen black. For example, a loading/alert messages should be shown in between page switches.



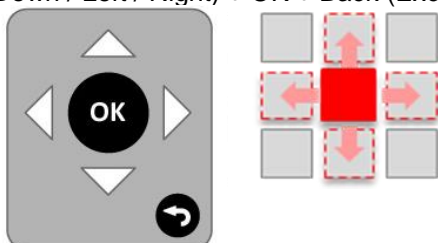
- ✓ 8. The App should show a loading UI when it is loading (start-up, video/audio playback start). For example, when a content is loading, a visual cue is needed:



- ✓ 9. Clickable elements should be at least 54 x 54 pixels in size. Recommended is a margin of at least 15 pixels between elements.

4.2.4. Navigation Scheme

- ✓ 10. The app should be fully navigable with the remote control using the following keys: 4 ways (Up / Down / Left / Right) + OK + Back (Exception: color key functionality indication buttons). Refer also to 3.3.2.



- ✓ 11. The app should be fully navigable with a pointer device (cursor) + OK + Back.

4.2.5. On-screen Button

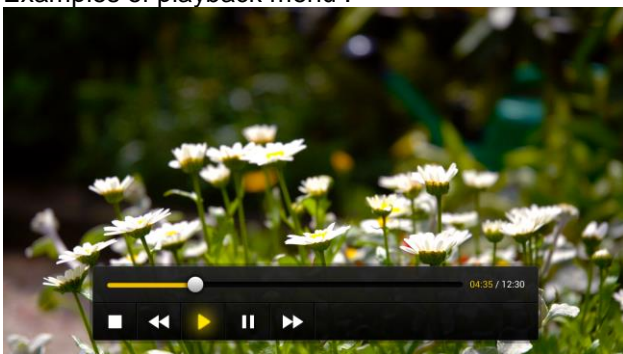
- ✓ 12. For (horizontal or vertical) scrolling lists, app should provide on-screen scrolling buttons to browse through the list. Scrolling lists should be usable with remote control keys and a pointer device.



✓ 13. If the app implements the playback option UI, playback options should work seamlessly as a user would expect. The playback options should be available through on- screen playback buttons and through the remote control's playback buttons.

The playback controls on-screen menu could be made visible when the OK key is pressed for full screen content, and recommended to be hidden after a specific timeout (e.g. 3 seconds). This key handling does not apply to windowed mode playback. An app can use their own playback menu or use a default menu. Refer to 3.4.4 for more information.

Examples of playback menu :

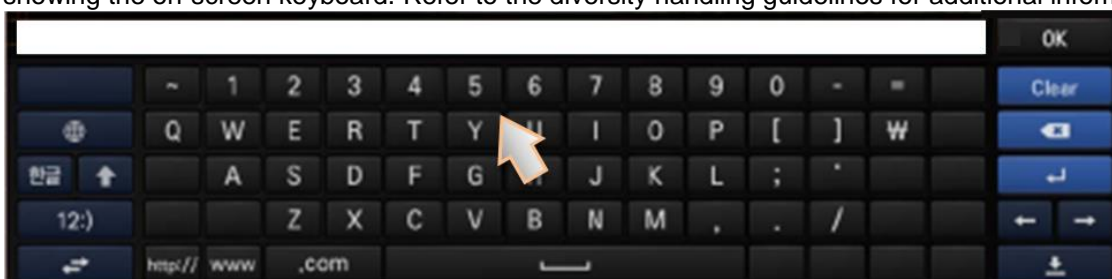


Example playback menu

✓ 14. LG's Q.Menu on-screen button should be implemented when providing the full screen video playback. And the app should hide this on-screen button for other platforms.

Please refer to the diversity handling guidelines for additional information.

✓ 15. If the App uses text entry it should support an on-screen keyboard. A library for an on-screen keyboard needs to be included within the code (see below for details). For certain manufacturers the keyboard is supported from a system device. Thus, diversity handling needs to be implemented to avoid showing the on-screen keyboard. Refer to the diversity handling guidelines for additional information.



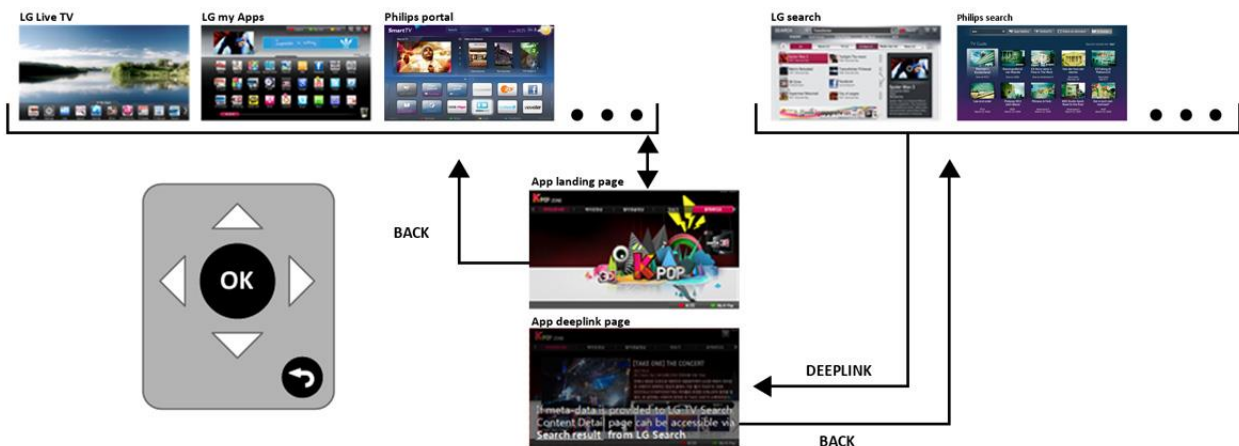
Refer to 3.3.5 for a technical explanation on how you can implement this keyboard with diversity handling.

4.2.6. Back Behaviour

✓ 16. When the back key is pressed at the top level of the application or on-screen exit button (optional) is pressed, the app should follow "Back navigation" section in the "Application Development and UI Guidelines". The app may give a confirmation message when exiting the app. The TV device is to navigate

back to the source application that called the application, but there could be exceptions. Refer to 3.3.2 and the diversity handling guidelines for details.

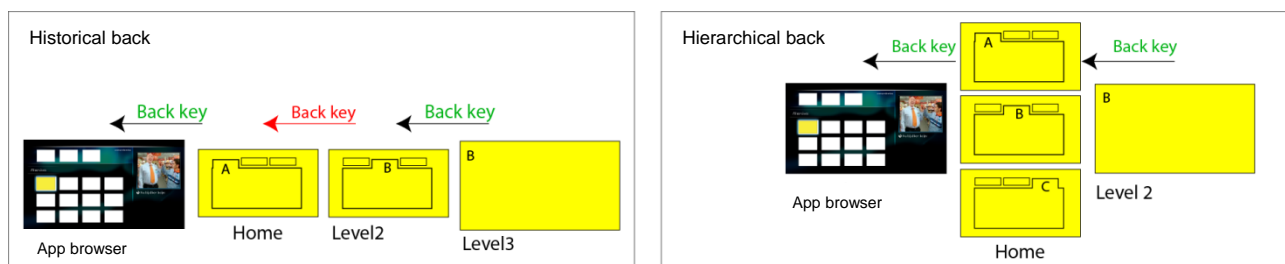
✓ 17. When the back key is pressed at the top level of the application or on-screen exit button (optional) is pressed, the app should follow "Back navigation" section in the "Application Development and UI Guidelines". The app may give a confirmation message when exiting the app. The TV device is to navigate back to the source application that called the application, but there could be exceptions. Refer to 3.3.2 and the diversity handling guidelines for details.



✓ 18. When pressing the remote control back key repeatedly, it should eventually exit the app. The app may give a confirmation message when exiting the app.

Refer to 3.3.2 for a technical explanation on how you can accomplish this behaviour requirement.

Ideally, hitting the back key many times should guarantee that the app eventually exits to its device content page: the source application that the app was executed from by the user. This shall be done for platforms that support this exit behaviour, refer to 3.3.2 and the diversity handling guidelines for more information. It is recommended to build a hierarchical back navigated app instead of a historical back navigated app to keep the route to the site's entry page as short as possible. Historical back behaviour is the idea of traversing back the path of user navigation in sequence whereas hierarchical back implies moving up a defined UI hierarchy within the app - top level as the entry page and bottom level as deeper pages. Refer to the figures below.



A. App optimization references

This is a (non-exhaustive) list of books and sites detailing optimization strategies for web apps. This list is informative, and not every strategy or tool given here will be applicable to the TV Device. Some sites can offer code which is incompatible with Smart TV, and some of these sites will offer code shrinking/compression examples which are not by definition recommended for the TV Device. Remember that most of these sites keep a fast PC platform (CPU) in mind, whereas the TV Device has a relatively slow CPU with medium bandwidth.

Use at your own discretion:

[a] Speed Up Your Site: Web Site Optimization, Andrew B. King, 2003, ISBN 0735713243.

[b] Javascript optimization, Jeff Greenberg, http://home.earthlink.net/~kendrasg/info/js_opt/

[c] Parallel web site optimization, <http://www.websiteoptimization.com/speed/tweak/parallel/>

[d] Javascript Optimization test, Andrew Hedges,
http://andrew.hedges.name/experiments/javascript_optimization/

[e] Yahoo performance rules, <http://developer.yahoo.com/performance/rules.html>

[f] Reflow guidelines, <http://code.google.com/speed/articles/reflow.html>
<http://dev.opera.com/articles/view/efficient-javascript/?page=3#reflow>

[g] CSS compression, <http://code.google.com/speed/page-speed/docs/payload.html#MinifyCSS>

[h] JS compression, <http://www.crockford.com/javascript/jsmin.html>

[i] CSS sprites, <https://launchpad.net/css-sprite-generator/>

B. Index of important guidelines

• THIS IS AN IMPORTANT HINT FOR DEVELOPING YOUR APP.	6
• REFER TO THE SPECIFICATION DOCUMENT FOR TECHNICAL DOCUMENTATION OF THE SUPPORTED HTML5 PROFILE.	11
• PLEASE TARGET THE LOWEST SMART TV ALLIANCE SPECIFICATION VERSION (MINIMUM VERSION 2.0) AND USE THE AVAILABLE (VERSION/CAPABILITY) METHODS TO DETERMINE NEWER AND/OR OPTIONAL FEATURES.	12
• TRY TO EXECUTE ALL REQUIRED JAVASCRIPT FUNCTIONS STARTING IN THE BODY ONLOAD.	16
• USE TIMERS SPARSELY, INCREASE GRANULARITY OF FREQUENTLY RECURRING TIMERS.	17
• WHEN EXITING YOUR APPLICATION, NO POP UPS MUST BE GENERATED.	18
• USE VK_ CONSTANTS INSTEAD OF KEYCODES.	18
• USE THE AVAILABLE EXIT-METHODS FOR RETURNING CONTROL BACK TO THE DEVICE FROM YOUR APP.	20
• YOUR PAGE MUST BE NAVIGABLE BY BOTH KEYS AS A POINTER DEVICE.	21
• USE THE PROVIDED TEXT ENTRY METHODS AS MUCH AS POSSIBLE.	22
• ANY STORAGE OF (PERSONAL) DATA ON/IN YOUR APPLICATION NEEDS TO BE IN LINE WITH ANY LEGAL AND PRIVACY REQUIREMENTS.	22
• AUTOCOMPLETE IN YOUR PAGE FOR SENSITIVE DATA? ALWAYS AUTHENTICATE THE USER.	22
• ALWAYS IMPLEMENT BOTH REMOTE CONTROL AS ON-SCREEN PLAY CONTROLS	23
• WHEN YOU USE ONE OF THE APP BASED SUBTITLING SOLUTIONS, MAKE SURE THE FONT SIZE IS CORRECT.	24
• (CACHE) MEMORY IS LIMITED ON A TV DEVICE, DO NOT RELY ON IT E.G. IN USE OF LARGE IMAGES.	28
• LIMIT IMAGE SIZES TO LESS THAN 1 MEGAPIXEL. LIMIT THE NUMBER OF IMAGES DISPLAYED ON A SINGLE PAGE.	28
• LIMIT THE AMOUNT OF DIFFERENT COOKIES (COOKIE NAMES) PER DEVICE.	31
• KEEP PAGESIZE SMALL, LIMIT THE NUMBER OF ELEMENTS IN A PAGE.	33
• USE FRAMEWORKS SPARSELY; IF YOU REALLY NEED THESE: SIMPLIFY CODE, LIMIT NETWORK CONNECTIONS.	34