

CableLabs®

ETV Application Interoperability Guidelines

CL-GL-ETV-AIG-V02-100618

RELEASED

Notice

This CableLabs document is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2010 Cable Television Laboratories, Inc.
All rights reserved.

Document Status Sheet

Document Control Number:	CL-GL-ETV-AIG-V02-100618			
Document Title:	ETV Application Interoperability Guidelines			
Revision History:	V01 – Released 02/19/10			
	V02 – Released 06/18/10			
Date:	June 18, 2010			
Status:	Work in Progress	Draft	Released	Closed
Distribution Restrictions:	Author Only	GL/Member	GL/Member/Vendor	Public

Trademarks

CableLabs®, DOCSIS®, EuroDOCSIS™, eDOCSIS™, M-CMTS™, PacketCable™, EuroPacketCable™, PCMM™, CableHome®, CableOffice™, OpenCable™, OCAP™, CableCARD™, M-Card™, DCAS™, tru2way™, and CablePC™ are trademarks of Cable Television Laboratories, Inc.

Contents

1	SCOPE.....	1
1.1	Introduction and Overview.....	1
1.2	Current State of Interoperability.....	1
2	REFERENCES	2
2.1	Informative References.....	2
2.2	Reference Acquisition	2
3	TERMS AND DEFINITIONS.....	3
4	ABBREVIATIONS AND ACRONYMS.....	5
5	ETV APPLICATION DEVELOPMENT DESIGN CONSIDERATIONS	6
5.1	Button Widgets.....	6
5.1.1	<i>Drawing the Background.....</i>	6
5.1.2	<i>Long Text Strings.....</i>	7
5.2	Rectangle Widgets.....	8
5.2.1	<i>Drawing the Background.....</i>	8
5.3	Selector Widgets.....	8
5.3.1	<i>Drawing Issues</i>	8
5.4	Drawing Text.....	9
5.4.1	<i>Special Characters.....</i>	9
5.4.2	<i>Font Metrics</i>	9
5.4.3	<i>Multi-Line Text Widgets</i>	9
5.4.4	<i>Anti-alias Properties.....</i>	10
5.5	Other Drawing Issues	10
5.5.1	<i>Z-Ordering of Widgets.....</i>	10
5.5.2	<i>Translucency to Graphics.....</i>	10
5.5.3	<i>Changing Widget Styles to Transparent.....</i>	10
5.6	Header Flags.....	11
5.6.1	<i>rhPrivateUseCritical</i>	11
5.7	Application Resolution Issues	11
5.7.1	<i>Visual Deformation of the User Interface.....</i>	11
5.7.2	<i>Aligning Graphics with Video</i>	12
5.8	Metadata Items.....	12
5.9	Redraw Timing	13
5.10	Platform-specific Sections not supported by UAs.....	13
5.11	Key Events.....	13
5.11.1	<i>Key Repeat Events</i>	13
5.11.2	<i>Key Down Events.....</i>	14
5.11.3	<i>Key Up Events</i>	14
5.12	HTTPS not supported by UAs.....	14
5.13	fwNoResponse flag in Form.....	14
5.14	String Functions.....	15
5.14.1	<i>String Convert.....</i>	15
5.14.2	<i>String Append</i>	15
5.14.3	<i>String Extract.....</i>	15
5.15	MPEG Service Locator.....	15
5.16	Persistence	16
5.17	pwPageTimeout.....	16

6 USER AGENT ANOMALIES.....17

- 6.1 Focus Related Issues.....17
 - 6.1.1 *Effect on Application Lifecycle*17
- 6.2 Focus Key Handling18
 - 6.2.1 *Info Key*.....18
 - 6.2.2 *Exit Key*.....18
- 6.3 MSO Key Handling18
 - 6.3.1 *Guide Key*18
 - 6.3.2 *Menu Key*.....19
 - 6.3.3 *Channel Up/Down Keys*19
 - 6.3.4 *Power Key*.....19

Figures

- Figure 1 - EBIF Button Widgets6
- Figure 2 - Button background drawing (background color not visible)7
- Figure 3 - Button background drawing (background color visible)7
- Figure 4 - Rectangle widget background drawing (background color not visible)8
- Figure 5 - Rectangle widget background drawing (background color visible)8

1 SCOPE

1.1 Introduction and Overview

This document presents a set of guidelines for the development and deployment of interoperable ETV *cable applications*. Although there are specifications that govern the development and formatting of interoperable ETV applications, [EBIF] and [EAM], the ETV User Agents (UAs) in use by MSOs today occasionally operate in an inconsistent manner. This document attempts to identify those areas where there is inconsistent behavior among UAs or where the specifications leave too much ambiguity for an application developer to count on predictable operational or behavioral results.

By adhering to these guidelines, interoperable applications will create a consistent user experience for interactive television viewers across a national footprint. While these guidelines are voluntary, all application developers are strongly encouraged to embrace them in order to provide all viewers with a familiar and intuitive user experience, regardless of the specific service they are using.

The audience for this document is EBIF application developers.

1.2 Current State of Interoperability

This document captures known interoperability issues surrounding the current state of the deployed ETV User Agents as of the published date of this document. The ETV User Agents currently deployed by the major MSOs target compliance with the I05 revisions of the EBIF and AM specifications, although each UA contains known deviations from those specifications.

Although work is progressing on an I06 version of the ETV standards and user agents, I05 will continue to be the compliance target of all of the deployed User Agents at least through 2010 and into early 2011.

This document will be updated as additional interoperability issues are identified. It will also be updated as new versions of user agents are deployed, with the hope that the number of interoperability issues will decrease over time.

2 REFERENCES

2.1 Informative References

This guidelines document uses the following informative references.

- [EBIF] Enhanced TV Binary Interchange Format 1.0, OC-SP-ETV-BIF1.0-I05-091125, November 25, 2009, Cable Television Laboratories, Inc.
- [EAM] Enhanced TV Application Messaging Protocol 1.0, OC-SP-ETV-AM1.0-I05-091125, November 25, 2009, Cable Television Laboratories, Inc.
- [UEG] Cable Application User Experience Guide, OC-GL-UEG-V02-100510, May 10, 2010, Cable Television Laboratories, Inc.

2.2 Reference Acquisition

Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
Phone +1-303-661-9100; Fax +1-303-661-9199; <http://www.cablelabs.com>

3 TERMS AND DEFINITIONS

This guidelines document uses the following terms:

Bound Application	A bound application is bound to, or associated with, the currently tuned channel. When the viewer tunes away from the current channel, the bound application goes out of scope and is typically terminated unless it is also associated with the newly tuned channel.
Content	Content is typically used to refer to audio, video, and graphic materials used by a service. Sometimes data and applications are also grouped into this term.
Enhanced TV (ETV)	A general term that refers to interactive services and applications that are provided in conjunction with video programming.
Enhancement	A software application that executes in conjunction with video programming.
Focus	The application with Focus is the one drawn on the top of the stack and will receive event notifications for all remote control keys in the Focus Key group.
Focus Key Group	The set of remote control key events that will be sent to cable applications when in focus.
Graphics Plane	The visual layer in the cable platform in which all application elements are rendered. It is displayed on top of the video plane.
Highlight	A contrasting visual style applied to a user interface element in an application to show that the interface element belongs to an application with focus. An action associated with the highlighted element is normally performed when the viewer selects the highlighted element.
ITV Safe area	The defined area on a television display that may contain visual elements without danger of their being distorted or not seen on almost all televisions sets. This is about 84% of the horizontal and 84% of the vertical distances covered by the video. All text and actionable on-screen buttons should not extend beyond this region.
Navigator	A navigator is an unbound application, provided by the network operator, that the viewer can activate at any time. The navigator is used to select services and launch other applications.
OOB channel	The OOB channel provides a bi-directional IP-based communication channel between the network and the digital set-top converter. The OOB Channel also carries SI data that describes the broadcast services available on the network.
Prompt	An application element that signals to a viewer that an application is present and may be launched.
Scaled Video	The video signal displayed in an area less than full screen.
Select	An action by the viewer that indicates a particular choice. For instance, a viewer may press a remote control key to choose an action represented by the currently highlighted button in an application.
Select Key	A key on the remote control that is used to select a choice or indicate an action to be performed by an application. NOTE that the Select Key is a logical definition and that the actual key on the remote control may be labeled as "Select," "OK," or "Enter," depending on the manufacturer and model of the device.

Service Information (SI)	Information that describes the broadcast services available on the network.
Signaling	Application life cycle messages either embedded in a program stream, for bound applications, or sent through the OOB channel, for unbound applications.
Time Out	An action taken by an application, such as termination, after a period of time in which no viewer input has been detected.
Title Safe area	The traditional area on a television display that may contain visual elements without danger of their being distorted or not seen. This is about 80% of the horizontal and 80% of the vertical distances covered by the video. This region is considered to be conservative and, therefore, ITV Safe can be used to provide additional screen real estate.
Unbound application	An unbound application is not associated with the currently selected channel and does not go out of scope when the viewer tunes away from any specific service.
User Interface	A presentation generated by an application that includes visual elements and responds to viewer actions.
Video Plane	The visual layer in the cable platform in which all broadcast video is played. It is displayed below the graphics plane.
Visual Element	A graphical object generated by an application. Examples include a text string, a shaded rectangle, and a bitmap.
Window	The area in the graphics plane in which an application renders all of its user interface elements.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

DVR	Digital Video Recorder
EBIF	Enhanced TV Binary Interchange Format
EPG	Electronic Programming Guide
ETV	Enhanced TV
GUI	Graphical User Interface
MSO	Multiple Service Operator, same as cable network operator
OCAP™	OpenCable Applications Platform
UI	User Interface
VOD	Video on Demand

5 ETV APPLICATION DEVELOPMENT DESIGN CONSIDERATIONS

There are several areas of the ETV specifications that are ambiguously written or are currently being interpreted in more than one manner. There are also aspects of some User Agents (UAs) that may be considered non-conformant to the [EBIF] or [EAM] specifications, but since those UAs are currently scheduled for deployment in the field, this document will caution against their use until such time as the defects are repaired or the specification is tightened up to remove those ambiguities.

5.1 Button Widgets

5.1.1 Drawing the Background

Issue Drawing elliptical cornered buttons with a non-transparent background produces visible artifacts on the display.

Recommendation When using elliptical cornered buttons, specify a background color that is transparent to the graphics or video planes (depending upon the background behind the button).

Discussion Section 10.1 of [EBIF] discusses the button widget and rules for its use and behavior in an EBIF application. Section 10.1.1 of [EBIF] specifically discusses the use of a background color used in the drawing of a button widget. Figure 1 is reproduced from [EBIF], where it is labeled as Figure 9.

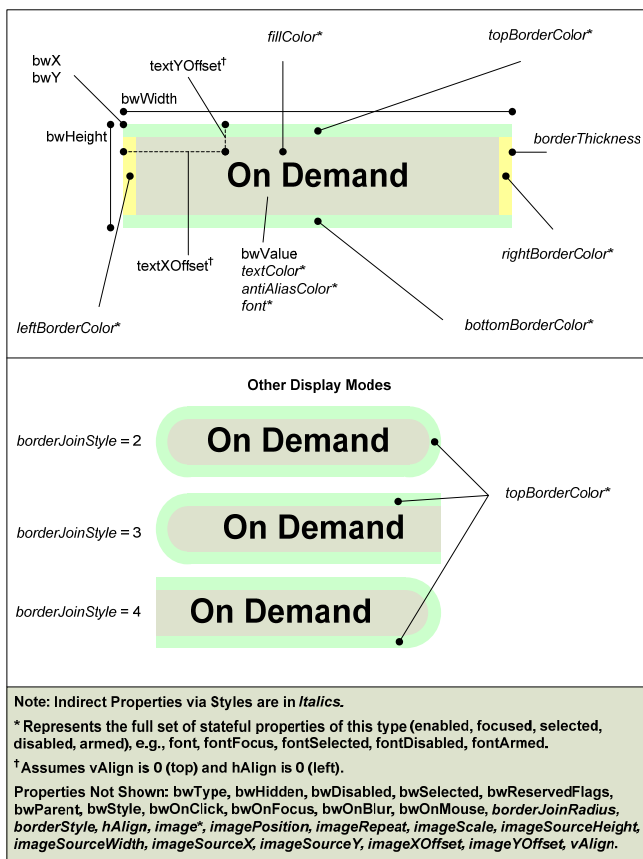


Figure 1 - EBIF Button Widgets

As shown in Figure 1, buttons may have rectangular or elliptical corners (or a combination of the two). What this figure does not show is the treatment of the "backgroundColor" field specified in the button widget definition. The definition of the backgroundColor properties, in [EBIF] section 10.1.1, states: "The background region is the region wholly contained within this widget's bounding rectangle."

Buttons are currently drawn in several of the commercial ETV UAs in an inconsistent manner.

If an application specifies an elliptical cornered button, and specifies "Black" as the background color, some ETV UAs will draw that button as in Figure 2 below, while others will draw it as in Figure 3. Note that Figure 3 is currently considered to be spec-compliant.



Figure 2 - Button background drawing (background color not visible)



Figure 3 - Button background drawing (background color visible)

5.1.2 Long Text Strings

Issue	Drawing buttons with text that exceeds the width of the button creates inconsistent results.
Recommendation	Make sure that the text fits the button width.
Discussion	In some cases, the text is truncated (as per [EBIF]). In other cases, the text is wrapped to a second line. In some cases, the text is simply not displayed. Note: [EBIF] I06 will likely modify the specification to say that only one line of text SHALL be displayed in a button (or any widget except a multi-line text widget) and that if the text is longer than the button, it SHALL be truncated without the use of ellipses. If the button text contains multiple words, truncation may appear at a word boundary. If the button text contains only one word, truncation will occur mid-word.

5.2 Rectangle Widgets

5.2.1 Drawing the Background

Issue Drawing elliptical cornered rectangles with a non-transparent background produces visible artifacts on the display.

Recommendation When using elliptical cornered rectangles, specify a background color that is transparent to the graphics or video planes (depending upon the background behind the rectangle).

Discussion Section 10.13 of [EBIF] discusses the rectangle widget and rules for its use and behavior in an EBIF application. Section 10.13.1 of [EBIF] specifically states that all indirect properties of a button widget also apply to a rectangle widget.

Rectangles are currently drawn in several of the commercial ETV UAs in an inconsistent manner.

If an application specifies an elliptical cornered rectangle, and specifies "Black" as the background color, some ETV UAs will draw that button as in the first example below, and some will draw it as in the second example below. Note that Figure 5 is currently considered to be spec-compliant.



Figure 4 - Rectangle widget background drawing (background color not visible)



Figure 5 - Rectangle widget background drawing (background color visible)

5.3 Selector Widgets

5.3.1 Drawing Issues

Issue Drawing selector widgets produces inconsistent results within each ETV UA.

Recommendation It is recommended to not use selector widgets at this time.

Discussion Section 10.15 of [EBIF] discusses the selector widget and rules for its use and behavior in an EBIF application. The specification is ambiguous, however, producing inconsistent results.

Some UAs simply do not support selector widgets at all.

5.4 Drawing Text

5.4.1 Special Characters

Issue Special characters, as listed in Annex G of [EBIF], (e.g., ellipses, arrow symbols) do not appear consistently across UAs.

Recommendation Avoid using special characters. Where needed, consider the use of images instead.

Discussion Special characters include the following glyphs:

Symbol
.
•
•••
▲
▼
◀
▶
●
★
♪
✓
◄
►
×
↶

5.4.2 Font Metrics

Issue Different UAs use different fonts, which may cause text spacing problems and unexpected text clipping.

Recommendation Allow room within your bounding rectangle and test on multiple platforms.

5.4.3 Multi-Line Text Widgets

Issue Scrolling (and display) of multi-line text widgets is handled differently on each UA.

Recommendation Avoid multi-line text widgets when a single-line text widget will suffice. Always set the wrap flag to true.

- Discussion** Some UAs don't pay attention to the "wrap" flag (but always default to true).
UAs use at least three different approaches to handling multi-line text widgets:
- The text truncates when reaching bottom of the text box.
 - The UA truncates the first line and no additional lines of text are displayed.
 - The UA displays the entire text, but truncates on last line. Ellipses ("...") are inconsistently displayed when there is more text to display outside of the visible text box.

5.4.4 Anti-alias Properties

- Issue** Anti-aliasing is not supported by several user agents.
- Recommendation** Do not depend on anti-aliasing properties to be supported across all user agents.
- Discussion** This does not appear to present undesirable results in any applications observed to date.

5.5 Other Drawing Issues

5.5.1 Z-Ordering of Widgets

- Issue** The z-order of widgets is not deterministic. This can cause unwelcome visual artifacts when re-drawing overlapping widgets. A common scenario for this is experienced when drawing hotspots around images.
- Recommendation** When using hotspot widgets that overlap images, specify the background color and fill color to be 100% transparent to the graphics plane. The border color(s) should be opaque. In other cases, avoid overlapping widgets.
- Discussion** When a Hotspot Widget is placed around an Image Widget, it is possible that the Hotspot is drawn after the Image, instead of before the Image. If the hotspot specifies a non-transparent fill color or background color, then the fill color may obscure the image that it overlaps. ETV I06 will likely address this issue to make the z-ordering of widgets more deterministic.

5.5.2 Translucency to Graphics

- Issue** Some UAs do not support translucency of widgets to other graphics. Regardless of the specified palette entries, all colors will be drawn either fully opaque or fully transparent.
- Recommendation** Do not specify palette entries that indicate translucency to graphics.
- Discussion** Although the EBIF specification allows applications to specify colors that are translucent (not 100% transparent) to other graphics they overlap, some UAs do not properly interpret those values.

5.5.3 Changing Widget Styles to Transparent

- Issue** Dynamically changing a widget's drawing colors to 100% transparent (in an event handler, for example) may not have any effect. The widget colors that were changed may remain opaque.

Recommendation Do not dynamically change a widget's drawing colors to 100% transparent.

Discussion [EBIF] section 6.4.3.3.1.5, Widget Presentation, states rules for drawing widgets. In this section, it states: "draw the widget's background color, if not 100% transparent". Further rules in section 10.1.3, Button Widget Presentation Behavior, for example, state similar rules for the fill color, border colors, text colors, etc.

While an application developer may assume that any change to the widget style would cause a refresh of the whole widget, it appears that the old widget is evidently not always cleared from the frame buffer before the changed colors are redrawn.

This behavior will likely be addressed in ETV I06.

5.6 Header Flags

5.6.1 rhPrivateUseCritical

Issue Some UAs will not run applications in which the rhPrivateUseCritical flag is set to *true* in the Resource Header.

Recommendation It is recommended not to set this flag to *true* for interoperable EBIF applications.

Discussion Some applications or toolchains set the rhPrivateUseCritical Flag in the resource header, indicating that private use extensions are being used. This flag prevents the application from running on UAs that do not support those private extensions. Modifying this setting enables the applications to run correctly. (Note the TVWorks XDK 3.4.1 sets this flag to true by default.)

5.7 Application Resolution Issues

5.7.1 Visual Deformation of the User Interface

Issue Applications may be authored for many different resolutions that do not necessarily match the native resolutions supported on a given platform. This may result in left-justifying, centering, clipping, or scaling the application visual assets, each of which may create some visual anomalies.

Recommendation Authoring for 640x480 provides the highest degree of assurance that the application will not be clipped and will fill most, if not all, of the screen.

Discussion

Applications are authored for one of the following resolutions:

- 320x240
- 640x480
- 704x480
- 720x480
- 960x540

Set-tops in use by MSOs today run in all of those modes except 320x240.

[EBIF] does not yet have specific directives for how to handle conversion from an application resolution to the set-top resolution, although this may be addressed in ETV I06.

For now, ETV UAs may even act differently on OCAP devices than they do on legacy devices with the same resolution, so there are no exact rules for what may happen if your application and set-top resolutions are not the same. Applications authored for 704x480 but displayed on a 640x480 device, for example, may be scaled down on an OCAP device, but simply clipped on a legacy device. Scaling will affect the width of every visual element, from images to buttons and text boxes. This may cause widget border widths to change as well as a variety of other artifacts.

When the aspect ratio also changes, as from 640x480 (4:3) to 960x540 (16:9), both scaling and centering techniques may be used. There is no requirement that an application be scaled in both the horizontal and vertical directions, although maintaining aspect ratio is ideal.

Note: ETV I06 may allow an application to influence the policy of scaling/centering used by the UA.

5.7.2 Aligning Graphics with Video**Issue**

The graphics plane is not always aligned pixel for pixel with the video plane and varies between different models of set-tops from any given manufacturer.

Recommendation

Use data resources for alignment data, adjusted according to the platform in use. Also, author applications for the native resolution of the device if at all possible, using resources targeted to the specific platform as necessary.

Discussion

For applications such as a Video Mosaic, where precise alignment of a graphic element and a region of the video is required, an application may have to adjust the horizontal placement of the graphic element by +/- 8 pixels (or more). Vertical alignment may also be off by several pixels. Note: these numbers appear to be constants for any given model of set-top.

5.8 Metadata Items**Issue**

Many of the metadata items in [EBIF] are not handled by the UA as might be expected.

Recommendation

Unless there is specific text that says the UA SHALL interpret this metadata item in a specific way, assume that there is no automatic behavior associated with that metadata item. If the application wishes to utilize the data in that field, it must interpret the data itself.

Discussion

The current UAs have no mandatory behaviors for the following metadata items, so applications should not rely on UAs to take any specific actions when they are specified in the resource metadata table:

- C.6 – Application Expiration Date and Time
- C.7 – Application Icons
- C.14 – Application Minimum Profile Required
- C.27 – Resource Active No Earlier Than
- C.28 – Resource Active No Later Than
- C.31 – Resource Expiration Date and Time

5.9 Redraw Timing

Issue

The REDRAW WIDGET action behaves inconsistently across User Agents, where refreshes may or may not take place immediately.

Recommendation

Do not depend upon an *immediate* refresh of the display after each REDRAW WIDGET action. If an application wishes to execute multiple REDRAW WIDGET actions, it is necessary to exit the initial event handler and use a secondary event handler, such as a timer, for each subsequent REDRAW WIDGET action.

Discussion

If a REDRAW WIDGET action is executed during an event handler, such as OnClick, the UA may not refresh the display until the event handler has completed and returned control back to the UA. This allows for more optimizations in compositing the display.

5.10 Platform-specific Sections not supported by UAs

Issue

Platform-specific sections are not currently supported by any UA.

Recommendation

Do not format EBIF Resource using platform-specific sections. Instead, use resources that target specific platforms.

Discussion

Although the current generation of application developer tools does not support the creation of applications that use this feature, it is also important to note that even if the tools are updated, the current generation of UAs does not support it in any case.

5.11 Key Events

5.11.1 Key Repeat Events

Issue

Key Repeat Events are not consistently delivered to EBIF applications.

Recommendation

Do not rely on consistent delivery of Key Repeat events across all UAs/platforms.

Discussion

When a user presses and holds a key on the remote control, an application may expect to get a Key Down event, followed by a series of Key Repeat events. This is not always the case. Different cable platforms signal these events in different manners. Each UA also interprets and delivers those events to applications in different manners.

In some cases, an application may receive a series of Key Down events. In others, an application may receive only a single Key Down event, with no indication of repeats. In others, applications may receive a combination of Key Down and Repeat events.

5.11.2 Key Down Events**Issue**

All user agents generate a key event when a key is pressed down, but not all user agents generate key events with the key down modifier set to true when a key is pressed down.

Recommendation

If it is critical to detect the key modifier being down, create your application so that it will execute if the modifier is all zeros, or if the key down modifier is set to true.

Discussion

Some user agents do not populate the key modifier, so an application should not require any specific modifier to be present. Most user agents generate one key event upon a key press action. That key event will either have a modifier of all zeros, or the down modifier set. At least one user agent generates events for key down, key up, and key repeat.

Therefore, an interoperable app using an OnKey event handler should limit the execution to occur only when the key down modifier is true or zero.

5.11.3 Key Up Events**Issue**

Most user agents do not support key up events.

Recommendation

Do not rely on a user agent to provide a key up event to the application.

Discussion

See the discussion for key down events for more details.

5.12 HTTPS not supported by UAs**Issue**

HTTPS is currently not supported by most user agents.

Recommendation

Use HTTP for form posting.

5.13 fwNoResponse flag in Form**Issue**

The fwNoResponse flag in forms is handled inconsistently across user agents.

Recommendation

Set the fwNoResponse flag to true in the application if no response is expected from the server as a result of a form post.

Discussion

Some user agents do not pay attention to the fwNoResponse flag, but others do. If a form is posted and the form widget expects a response with content, but only receives an empty HTTP 200 OK, then those user agents will generate an OnSubmitError event.

Note: The TVWorks XDK 3.4.1 sets this flag to false by default and does not provide the ability to set it to true. We have been told by Comcast that this will be resolved in the next release of the XDK.

5.14 String Functions

5.14.1 String Convert

Issue	String convert functions from strings to number values are inconsistent across user agents, especially when the number exceeds eight characters in length.
Recommendation	<p>Try to avoid performing string convert operations on large numbers, where the string representation of the number exceeds eight characters in length.</p> <p>If converting a hexadecimal string, prefix the string with “0x” (not “0X” since that is going to be removed in I06).</p> <p>Do not convert a decimal string that will exceed the bounds of a 32-bit signed integer.</p> <p>Do not include a “+” sign prefix on a decimal string to convert.</p> <p>Make sure the destination buffer is large enough to hold the converted result.</p>
Discussion	Some user agents will take the first eight characters and convert them; others will take up to thirty two characters and convert them. Some strip leading zeros; others do not. There are clarifications coming out in I06 in terms of leading zeros, but currently be aware that you may not get consistent results if converting large numbers (both radix 10 and radix 16).

5.14.2 String Append

Issue	String Append does not have consistent results if the op2 value is an integer to be encoded as a character. Also, some user agents do not signal the InsufficientSpaceError if the buffer is too small to append the string/character.
Recommendation	Don't expect the op2 integer value (e.g., stringappend(buf,int)) to be encoded as a character and appended to the end of the string buffer as per spec. Also, ensure the destination string buffer is large enough to hold the appended string/character.
Discussion	In some user agents we have seen the integer value (op2) converted to a decimal string representation and appended.

5.14.3 String Extract

Issue	String Extract has inconsistent results if using 0xFFFF to indicate “to end of string”. Also, some user agents fail to handle and report error conditions, such as insufficient space when performing this action.
Recommendation	<p>Avoid using the special op4 value of 0xFFFF to indicate “to end of string”.</p> <p>Ensure the starting character offset <= source string length.</p> <p>Ensure the destination string buffer is large enough to hold the extracted string.</p>

5.15 MPEG Service Locator

Issue	The MPEG Service Locator is currently not supported by most user agents.
Recommendation	This is removed in I06. The recommendation is to avoid using it in a national footprint distribution EBIF application.

5.16 Persistence

Issue Persistent data items are not supported across power cycles on most user agents.

Recommendation Avoid applications that rely on persistent data across power cycles.

Discussion Most user agents support persistence during a session. This would suffice for many applications that require storage of a user action for a short time. For example, if the user selected their favorite team during a sporting event, tuned away during a commercial break, returned to the channel and the application loaded again, the application would be able to retrieve that data about the favorite team.

5.17 pwPageTimeout

Issue OnPageTimeout events are not generated consistently across user agents.

Recommendation Do not rely on the pwPageTimeout parameter on a page widget to create a reliable timeout event. Use a timer widget instead.

6 USER AGENT ANOMALIES

This section describes differences in how various User Agents (UAs) respond to remote control key events. It is essential to understand these differences as you design your application.

6.1 Focus Related Issues

[EBIF] does not specifically address how a UA responds to the loss of focus, when executing either bound or unbound EBIF applications. The User Experience Guidelines [UEG] state that when an application is in focus, it should expect to receive several Key Events when specific keys on the remote control are pressed. Other keys are considered to be "reserved" by MSO applications such as the Navigator. When a Key Event is directed to the Navigator, the Navigator may choose to silently consume that Key Event, may choose to put an overlay on top of the EBIF application without taking focus, or may choose to put up an overlay that does take focus. Each of these scenarios affects the UA, and thus any active EBIF applications.

6.1.1 Effect on Application Lifecycle

There is a general, though not universal, consensus that when an EBIF application loses focus (i.e., is obscured by another application that is requesting focus) that the UA will *suspend* that EBIF application. When transitioning to a suspended state, the application SHOULD remove any visual elements from the screen, although it is likely that the UA will perform this action on behalf of the application in any case, by hiding the window in which the EBIF application is being rendered. In the case of bound applications, there is an assumption that the window should be hidden, regardless of whether the EBIF application is fully obscured, partially obscured, or not at all obscured by the application taking focus.

When the second application yields or gives up focus, there is, again, no set rule as to whether or not (or when) to re-display the suspended application. One convention dictates that if another AUTOSTART or PRESENT signal is received once the EBIF application is eligible to be seen, it will be transitioned back to a *running* state by the UA, again showing the window in which the EBIF application is being rendered.

An alternative view would dictate that as long as signaling was not lost while the application was obscured, it should immediately be made visible, as soon as focus is given up by the second application.

In either case, the EBIF application should expect to be notified of the state transition before being notified of focus being returned.

For unbound EBIF applications, there is the added complication that there is no signaling associated with the currently selected service. Signaling comes only from another application or configuration resource. In the case of unbound EBIF applications, it is recommended that they be displayed immediately, as soon as the obscuring application disappears. Because unbound applications are not specifically addressed in [EBIF] yet, however, this is not certain to occur.

Note: [EBIF] I06 is likely to include additional requirements and features for support and handling of unbound applications.

6.2 Focus Key Handling

6.2.1 Info Key

Issue The Info key is a focus key and should always be given to the EBIF application when it is in focus. Unfortunately, it is handled by different platforms in an inconsistent manner, primarily based on actions by the Navigator applications, which do not all follow the [UEG] guidelines.

Discussion Some Navigators intercept this key event before it can be directed to the EBIF application or UA. When this occurs, the Navigator may display the Info/Channel Banner on screen or may take no action at all, and just consume the key event, thus preventing the EBIF application from receiving it.

6.2.2 Exit Key

Issue The Exit key is a focus key and should always be given to the EBIF application when it is in focus. Unfortunately, it is handled by different platforms in an inconsistent manner, primarily based on actions by the UA.

Discussion The [UEG] states: "When the viewer presses the Exit key, it is an indication that they wish to stop interacting with the current user interface component. If an application has only a single visual layer within its window, it should hide it. If the application has multiple layers rendering, it should remove the one that currently contains the highlighted UI element. Once an application no longer has any visible user interface components, it should yield focus, hide itself, or terminate."

Currently, it appears that the Exit key is often handled by some UAs without ever notifying the EBIF application. In all cases, the UA completely clears the EBIF application from the screen. Some UAs subsequently pass the Exit key to the application, but other UAs terminate the application without passing the Exit key to the application.

6.3 MSO Key Handling

6.3.1 Guide Key

Issue The Guide key is always handled by the Navigator, which will display its UI based on receipt of that key event. At issue is whether the application should be suspended and/or cleared from the screen when that occurs.

Discussion Technically, this issue is directly addressed by the focus handling rules in 6.1.1. Evidently, however, some UAs do not currently transition the application to a suspended state when it is obscured by the Navigator, though the active page's OnBlur action script is executed.

6.3.2 Menu Key

Issue The Menu key is always handled by the Navigator, which will display its UI based on receipt of that key event. At issue is whether the application should be suspended and/or cleared from the screen when that occurs.

Discussion Technically, this issue is directly addressed by the focus handling rules in 6.1.1. Evidently, however, some UAs do not currently transition the application to a suspended state when it is obscured by the Navigator, though the active page's OnBlur action script is executed.

6.3.3 Channel Up/Down Keys

Issue The Channel Up/Down keys are always handled by the Navigator, which will do a service selection based on receipt of that event. At issue is whether the application is informed of the service selection and how its lifecycle is affected.

Discussion In all cases, the UA clears any bound EBIF applications from the screen when it detects a successful service selection. [EBIF] Section B.13, OnServiceSelection, states that typically, by the time an application is notified of a successful service selection, it has been transitioned to a suspended state. This is referring specifically to a bound application, since there is no particular reason to suspend or terminate an unbound application based on service selection.

Currently, however, some UAs will terminate and unload a bound application when a new service is selected. Some will suspend the application, but leave it in memory for quick re-activation, if appropriate.

If the UA suspends the application, it also triggers an OnServiceSelection event, which helps the application understand why it was suspended.

If the same application is signaled on the new service, the OnServiceSelection event is an indicator to the application that it has a new context, based on the new service to which it is bound. When the next AUTOSTART or PRESENT signal is received, the application should be re-activated without going through its loading and initialization steps.

6.3.4 Power Key

Issue The Power key is handled by different UAs in an inconsistent manner. Some UAs terminate the application immediately. Other UAs set OnStateChange to Suspend, but without agreement on whether the UA terminates the application or lets it time out.

Discussion When the Power key is pressed, [EBIF] does not require the application to be suspended or terminated. If, for example, the user "accidentally" presses the Power key and immediately presses it again, then ideally the application would still be active and the user would not lose the current application context. If the application was terminated based on the first key press, it would have to be re-initialized and wait for the next AUTOSTART signal before activating. If the application was merely suspended, an AUTOSTART or PRESENT signal would re-activate the application. If its state did not change at all, based on pressing the Power key, then it would still be visible and the user will lose nothing.