Requirement For Standardizing Widgets

W3C Working Group Note 27 September 2011

This version:
http://www.w3.org/TR/2011/NOTE-widgets-reqs-20110927/
Latest published version:
http://www.w3.org/TR/widgets-reqs/
Previous versions:
http://www.w3.org/TR/2010/WD-widgets-reqs-20101026/
Latest editor's draft:
http://dev.w3.org/2006/waf/widgets-reqs/
Editor:
Marcos Cáceres

Abstract

This document lists the design goals and requirements that specifications would need to address in order to standardize various aspects of widgets. A *Widget* is an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device. Typical examples of widgets include clocks, CPU gauges, sticky notes, battery-life indicators, games, and widgets that make use of Web services, like weather forecasters, news readers, e-mail checkers, photo albums, and currency converters.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document reflects three years of gathering and refining requirements for the Widget family of specifications. The requirements were gathered by extensive consultation with

W3C members and the public, via the Working Group's mailing lists (WAF archive, WebApps archive). The Working Group's goal is to make sure that vendor's requirements for Widgets are complete and have been effectively captured. The Widget family of specifications will set out to address as many requirements as possible (particularly the ones marked with the keywords MUST and SHOULD).

This document is produced by the Web Applications (WebApps) Working Group (WG). This WG is part of the Rich Web Clients Activity and this activity is within the W3C's Interaction Domain. The public is encouraged to send comments to the WebApps Working Group's public mailing list public-webapps@w3.org (archive). See W3C mailing list and archive usage guidelines.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

1 Introduction

A ***widget*** is an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device. A widget may run as a stand-alone application (meaning it can run outside of a Web browser), and it is envisioned that the kind of widgets being standardized by this effort will one day be embedded into Web documents.

In this document, the runtime environment in which a widget is run is referred to as a ***widget user agent***. Note that running widgets may be the specific purpose of a widget user agent, or it may be a mode of operation of a more generic user agent (e.g. a Web browser). A widget running on a widget user agent is referred to as an ***instantiated widget***. Prior to instantiation, a widget exists as a ***widget package***.

Prior to this standardization effort, there was no standardized way to author, package, digitally sign, or internationalize a widget package for distribution and deployment on the Web. In the widget space, although many widget user agents are now on the market, widgets built for one widget user agent (e.g., Windows 8 Metro applications) are still not able to run on any other widget user agent (Apple's Dashboard). This document, along with the Widget family of specifications, attempt to address the interoperability issues.

This document lists the design goals and requirements that specifications need to address in order to standardize how widgets are authored/scripted, digitally signed, secured, packaged and deployed in a way that is device independent, follows W3C principles, and is as interoperable as possible with existing market-leading user agents and existing Web browsers.

Note: To be clear, this specification describes the requirements for installable/desktop or mobile widgets (akin to Apple's Dashboard, Opera Widgets, Windows 8's Metro Applications). This document does not address the requirements of "web widgets", such as iGoogle Gadgets or Windows Live Gadgets, which are being specified by the Open Ajax Alliance's IDE Working Group.

2 Conformance

This section is normative.

The key words MUST, MUST NOT, REQUIRED, NOT REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in the normative parts of this document are to be interpreted as described in [RFC2119].

This specification only applies to one class of product: W3C Technical Reports. The Working Group will attempt to standardize widgets by addressing the requirements listed in this document through the ***Widget family of specifications***, which include:

- [Widget Packaging and Configuration]
- [Widget Updates]
- [Widget Interface]
- [Digital Signatures for Widgets]
- [Widget URIs]
- [View Modes]

Note: Only normative statements marked with the keywords* MUST *and* SHOULD *are required to be standardized by the Widget family of specifications. The Working Group will publish an informative Working Group Note at the end of the standardization process listing any requirements that were not formally addressed by the Widget family of specifications.

Although a number of specifications will be created to address the requirements enumerated in this document, in some instances, it will be the amalgamation of multiple parts of individual specifications that address a single requirement. Nevertheless, this document speaks only of a conforming specification (defined below). The Working

Group's choice to have multiple specifications address the following requirements, as opposed to having a monolithic specification, was made for the following reasons:

- Easier for multiple editors to maintain and edit, as each can check in and out the relevant specifications they are editing.
- Easier for the Working Group to discuss during teleconferences and face-to-face meetings.
- Easier for the public and experts to review.
- Easier for implementers to implement, as each specification will be as modular as possible.
- Easier to print, for the purpose of review.

A ***conforming specification*** is one that addresses one or more requirements listed in this document. A conforming specification SHOULD attempt to address requirements marked with the keywords “SHOULD” or “MAY” unless there is a technically valid reason not to do so. The validity of technical reasons for not addressing any requirements will be determined by the Working Group members, and through communication with vendors and the public on the Working Group's public mailing list public-webapps@w3.org (archive).

3 Design Goals

This section is informative.

This section lists the ***design goals*** that the Working Group recommends a conforming specification adopt when attempting to standardize the various standardizable aspects of widgets. The requirements are directly motivated by the following design goals. The design goals are listed in alphabetical order.

Accessibility:

A conforming specification needs to support relevant accessibility guidelines, such as [WCAG 2.0].

Compatibility with other standards:

A conforming specification needs to maintain compatibility with, or directly make use of, other standards where possible.

Current development practice or industry best-practice:

A conforming specification needs to consider the development practices currently used by widget developers and promote relevant industry best-practice, such as [MWBP].

Device independence:

A conforming specification needs to support relevant device independence guidelines.

Ease of authoring:

A conforming specification needs to specify solutions that are easy to use and avoid unnecessary complexity, meaning that a widget package should be easy for authors to create without requiring special or expensive proprietary software, and easy for end-users to acquire and install/run.

Internationalization and localization:

A conforming specification needs to implement relevant internationalization and localization guidelines, such as [i18n-XML] and [BCP 47], as well as consider current practical internationalization solutions used by the widget development community.

Interoperability:

A conforming specification needs to attempt to be as interoperable as possible with existing market-leading widget user agents.

Longevity:

A conforming specification needs to be specified in such a way that it ensures that a widget package can still be processed well into the future (e.g. 100 years from the date the specification reaches “Recommendation Status as defined in the ” [W3C Process] document).

Security:

A conforming specification needs to address the security concerns of end-users, authors, distributors and device manufacturers by recommending appropriate security policies and programming behavior. A conforming specification must also consider the distribution and deployment security requirements as they relate to authors and vendors.

Web and offline distribution:

A conforming specification needs to deal with cases where an end-user acquires a widget package over [HTTP] or via some other non HTTP-based (offline) means, such as a local file system, Blue tooth or a Multimedia Message Service. In addition, a conforming specification needs to provide a means by which widgets can be updated when a new or different version of a widget becomes available. It must be possible to perform updates from online and offline sources.

Wider community benefit

Any new technologies or processes defined by a conforming specification needs to designed in such a way that they are beneficial the wider Web community at large. In other words, conforming specifications should try not to be insular to their problem domain, but should also consider the needs of the wider Web community.

4 Requirements

This section is normative.

This section enumerates the ***requirements*** that a conforming specification would need to address to standardize widgets. These requirements are directly motivated by the design goals and are based on an iterative process of feedback from the public, discussions with various vendors, and a survey of market-leading widget user agents.

4.1 Packaging

This section enumerates the requirements that a conforming specification needs to address to standardize the packaging format of a widget. The objective of this section is to ensure that a conforming specification recommends a general packaging format that is, amongst other things:

- Already a *de facto* standard on market-leading widget user agents on both desktops and mobile devices.
- Able to be used in multilingual contexts.
- Suitable for mobile devices.
- Able to support digital signatures.

R1. Packaging Format

A conforming specification MUST recommend a packaging format that is royalty free, open, and widely implemented across market-leading widget user agents and compatible with mobile devices. In addition, a conforming specification MUST specify exactly which aspects of the packaging format are to be supported by conforming widget user agents.

Motivation:
Compatibility with other standards, Web and offline distribution, device independence, ease of use, current development practice or industry best-practice, internationalization and localization, interoperability, and longevity.

Rationale:
To specify an interoperable and pervasive packaging format that is relatively easy for vendors to implement, and easy for authors to use/access on any platform.

R2. Media Type

A conforming specification MUST recommend that a conforming widget package be sent over [HTTP] with a formally registered [media type] that is specific to the Widget family of specifications. The Working Group MUST formally register the media type with the Internet Assigned Numbers Authority (IANA). A conforming specification MUST specify how a widget user agent will process a widget package that was served with an unsupported media type or when the media type is unspecified.

Motivation:
Compatibility with other standards, Web and offline distribution, and ease of use.

Rationale:
To provide a formal means for an author to denote that a widget package conforms to a W3C endorsed specification when a widget package is served over [HTTP]. In addition, the media type could potentially be used in conjunction with an auto-discovery mechanism to facilitate deployment of a widget package.

R3. File Extension

A conforming specification MUST specify a file extension that authors MAY assign to widget packages in contexts that rely on file extensions to indicate the content type, such is the case on many popular file systems.

Motivation:
: Device independence, ease of use, Web and offline distribution, interoperability, and longevity.

Rationale:
: When a [media type] is not present, as is often the case when a widget is instantiated locally from an end-user's storage device, the operating system will sometimes use the file extension to associate the widget package with the appropriate widget user agent. However, when the widget is distributed over [HTTP] and a media type is present, a file extension will usually not be required. However, in some cases, a Web server may rely on a file extension to correctly set a widget package's media type in the [HTTP] headers. In addition, a user agent that is aware of the content type of widgets may add the appropriate file extension automatically or include it as the default if the user is prompted for the file name during a saving process.

R4. Internal Abstract Structure

A conforming specification MUST recommend a packaging format that supports structuring resources into collections such as files and directories (understood in this document in a broader sense than in some popular file systems, namely as forms of generic logical containers). In addition, the packaging format SHOULD allow authors to add and remove resources of a widget package without needing to recreate the widget package.

Motivation:
: Ease of use, interoperability, and current development practice or industry best-practice.

Rationale:
: To provide authors with a format that is easy to use in a development process.

R5. Reserved Resource Names

A conforming specification MUST indicate if any resources (files or directories or similar logical containers) are mandatory or reserved and what specific function they serve in the widget package. A conforming specification SHOULD specify graceful error handling/recovery procedures if those resources are used erroneously or missing.

Motivation:
: Ease of use, compatibility with other standards, and current development practice or industry best-practice.

Rationale:
: To make it more efficient for widget user agents to locate reserved resources at runtime. For example, the packaging format may require authors to place all resources inside a '`resources`' directory located at the root of the widget package.

R6. Addressing Scheme

A conforming specification MUST recommend a hierarchical addressing scheme that can be used to address the individual resources within a widget package from within a configuration document. The hierarchical addressing scheme MUST be capable of expressing both absolute and relative relationships between a resource and the widget package. In addition, the hierarchical addressing scheme MUST be interoperable with resources that might also need to address other resources within the widget package (e.g., HTML documents, CSS documents, JavaScript documents, etc.). The hierarchical addressing scheme SHOULD be one that Web authors would feel comfortable using or to which they are already accustomed.

Motivation:
: Ease of use, compatibility with other standards, current development practice or industry best-practice, and security.

Rationale:
: To make it easy for authors to address resources from the configuration document or other relevant resources within the widget package. For example, addressing a custom icon within a widget package from the configuration document (e.g. `<icon src="icons/cat.ico'/>`). Or, for example, addressing an image within a widget package from within a HTML start file (e.g. `<img src="/backgrounds/sky.png'>`).

R7. Multilingual File Names

A conforming specification MUST recommend a packaging format that allows for non-ASCII characters in file and directory names, allowing authors to create widgets suitable for various cultures and languages, as well as multilingual contexts. The packaging format MUST either provide some means to declare the character encoding or specify what the character encoding is. The [UTF-8] character encoding SHOULD be either the default (if multiple encodings are allowed) or sole encoding used.

Motivation:
: Internationalization and localization, current development practice or industry best-practice, ease of use, interoperability, and longevity.

Rationale:
: To allow authors to create files and folders using characters beyond the ASCII character repertoire. Since packaged widgets are widely distributed, variation in character encoding between different platforms or configurations may render a widget with non-ASCII resources inoperable or otherwise degrade the user experience unless a character encoding is used.

R8. Localization Guidelines

A conforming specification MUST provide guidelines that explain to authors how collections of resources need to be structured for the purpose of internationalization.

Motivation:
: Internationalization and localization, current development practice or industry best-practice, ease of use, and interoperability.

Rationale:

To both guide and encourage authors to localize content. For example, the specification could mandate that authors place localized content into a strictly named directory structure that denotes localized content (e.g. '`resources/en/`' for all English content, and '`resources/en-AU/`' for further localized Australian-English content, and so on).

R9. Automatic Localization

A conforming specification SHOULD specify a processing model that automatically localizes content when authors follow the localization guidelines.

Motivation:

Internationalization and localization, current development practice or industry best-practice, ease of use, and interoperability.

Rationale:

To define an internationalization model, complete with graceful error handling, to reduce the amount of engineering work an author needs to do in order to localize a widget.

R10. Device Independent Delivery

A conforming specification MUST recommend a packaging format that is suitable for delivery on many devices, particularly Web-enabled mobile devices.

Motivation:

Device independence, Web and offline distribution, and interoperability.

Rationale:

To recommend a packaging format that is interoperable with desktops and for mobile devices, where the widget space is currently growing.

R11. Data Compression

A conforming specification MUST recommend a packaging format that supports both decompressed data and OPTIONAL data compression. A conforming specification SHOULD also recommend at least one royalty-free default compression/decompression algorithm that is compatible with market-leading widget user agents and implementations of the packaging format on mobile devices.

Motivation:

Web and offline distribution, device independence, and current development practice or industry best-practice.

Rationale:

To make a widget package smaller for delivery over [HTTP], where the cost of data access is sometimes expensive for end-users. Compressing might also help with transfer speed when a widget package is sent over a communication channel with limited bandwidth, such as Bluetooth or infrared. Compressed widgets may also have a lesser impact on a device's battery during download.

R12. Derive the Media Type of Resources

In the case that the packaging format does not support labeling resources with a media type, a conforming specification MUST either specify or recommend a means of deriving the media type of resources for the purposes of rendering. A conforming specification MAY define a means to override how a widget user agent derives the media type of a resource (e.g., treat resources with the file extension `.php` as `text/html`), but MUST NOT force a widget user agent to process resources of one media type as that of another type (e.g. treating a jpeg image at `text/html`).

Motivation:
Web and offline distribution, device independence, and current development practice or industry best-practice.

Rationale:
To allow appropriate rendering of resources by the widget user agent. For instance, for the sake of interoperability, all widget user agents should treat resources with a `.html` file extension as `text/htm`l (and not as `application/xhtml+xml`).

4.2 Configuration Document

This section enumerates the requirements that a conforming specification needs to address in order to standardize the configuration document. The objective of this section is to ensure that a conforming specification specifies a configuration document format that defines:

- Metadata elements that can capture metadata about a widget, including its title, some form of identification, and versioning information.
- Metadata elements that can capture authorship information.
- A bootstrapping mechanism that would enable a widget user agents to automatically instantiate a widget.
- Relevant configuration parameters.

R13. Format and Schema

A conforming specification MUST specify the configuration document language using a common data interchange format, as well as the rules for processing the configuration document language and any micro-syntaxes represented as character data. A conforming specification MUST specify graceful error handling procedures for when metadata values are in error, or the values are impossible to satisfy or realize by the user agent. The metadata MUST be extractable, processable and reusable in other contexts (for instance, to create an online gallery of widgets). In addition, a conforming specification SHOULD make it clear to authors which elements are optional and which elements are mandatory. A conforming specification SHOULD specify a formal schema for the language, as well as define any configuration defaults. The schema SHOULD NOT be a normative part of the conforming specification, but MUST be suitable for use by conformance checkers. A conforming specification MUST recommend that configuration documents be encoded in [UTF-8]. A conforming specification MAY specify the configuration document language using alternative standardized data interchange formats (e.g. JSON) and schema.

Motivation:
Compatibility with other standards, current development practice or industry best-practice, ease of use, internationalization and localization, longevity, interoperability,

and accessibility.

Rationale:

To have a language in a format that is relatively easy for authors to read and write, and provides effective internationalization support. An example of such a language is [XML]. XML is generally accepted and understood by widget authors and parsed by all market-leading widget user agents, and XML parsers generally have reasonable support for [Unicode], which allows for effective internationalization and localization.

R14. Widget Metadata

A conforming specification MUST specify the structure and semantics of elements that represent metadata about a widget. More specifically, a conforming specification MUST specify the structure and semantics of elements that represent data about the widget, including the name, version number, a unique identifier, and a description of what a widget does.

Motivation:

Current development practice or industry best-practice, interoperability, and accessibility.

Rationale:

To provide authors with a practical set of metadata elements that describe various aspects of the widget that may be used in various contexts.

R15. Authorship Metadata

A conforming specification MUST specify the structure and semantics data about the authorship of a widget, including an author's name, e-mail, and organization.

Motivation:

Current development practice or industry best-practice, and interoperability.

Rationale:

To provide authors with a practical set of metadata elements that describe a widget and its authorship that may be utilized within an application context (such as a menu) or of importance to end-users.

R16. Copyright Notice and License Metadata

A conforming specification MUST specify the structure and semantics of fields that explicitly reference, or otherwise include, a software license agreement or notice. In addition, a conforming specification MUST provide a means to declare who holds the copyright for the widget, as well as a model for how this data must be processed by a widget user agent.

Motivation:

Current development practice or industry best-practice, and interoperability.

Rationale:

To provide authors with a means to legally declare how a widget and its various internal resources can be used by end-users. For example, an author may include a GNU-style license that allows others to reuse any source code.

R17. Visual Rendering Dimensions

For widgets that make use of a rendering context, a conforming specification SHOULD specify an OPTIONAL means for an author to declare the initial visual dimensions for an instantiated widget in a way that is device independent (e.g. via [CSS] pixels). In the absence of user style sheets, a conforming specification MUST specify that styling by the author takes precedence over dimensional values declared in the configuration document or any dimensional values implicitly computed by the widget user agent. However, in the presence of user style sheets, user style sheets take precedence but MUST be applied in conformance to the cascade model and rules of behavior specified in [CSS].

Motivation:
: Ease of use, device independence, and current development practice or industry best-practice.

Rationale:
: To set up the rendering context for an instantiated widget in a way that is compatible on a range of devices.

R18. Declarative Bootstrap

A conforming specification MUST specify a declarative bootstrap mechanism that addresses the start file that is to be initially instantiated at runtime (the instantiated widget). The bootstrap mechanism MUST NOT be able to address or instantiate local files outside the scope of the widget package. However, the bootstrapping mechanism MAY be able to address a resource on the Internet, but only of a media type allowed by the automated bootstrap requirement (below) or resources that are of media types supported by a widget user agent. A conforming specification MAY also allow authors to declaratively bootstrap proprietary resources (e.g. a Flash movie) within the widget package, so long as they are able to be processed or instantiated by the widget user agent. If a bootstrap has not been declared by an author, then automated bootstrapping MUST occur as described in the automated bootstrap requirement.

Motivation:
: Ease of use, current development practice or industry best-practice, and security.

Rationale:
: For example, bootstrapping could occur by dereferencing, via a relative reference, the initial resource to be retrieved from within a widget package by a widget user agent (e.g. '`/ui/clock.svg`'). Alternatively, the resource might be a HTML document that when combined with the visual rendering dimensions requirement displays at the appropriate size.

R19. Automated Bootstrap

A conforming specification SHOULD specify an automated model for finding the start file of the widget in the absence of a declarative bootstrap. The automated bootstrap model MUST NOT be able to address resources outside the scope of the widget package and MUST NOT address resources on the Web over [HTTP] or any other protocol. The widget user agent SHOULD be allowed to select its preferred format for the start file, and then it SHOULD locate that resource first before attempting to locate other resources.

Motivation:

Ease of use, device independence, current development practice or industry best-practice, and internationalization and localization.

Rationale:

For example, the conforming specification could specify a model that searches for a default file name (`index.htm`, `index.html`, `index.svg`, etc.) firstly within localized directory names, as required by automatic localization, and then within the directories of the widget package. If that search fails, then the widget user agent could try to find files with extensions ".html, .svg, etc." starting from the root directory.

R20. Iconic Representations

A conforming specification MUST specify a means to declare iconic representations of the widget for use as alternate or fallback content, standby indicator or in a non-running context. The conforming specification SHOULD NOT limit iconic representations to static images and it SHOULD provide support for alternative text representations of an icon where possible. A conforming specification SHOULD also recommend a default icon media type and file name.

Motivation:

Ease of use, device independence, current development practice or industry best-practice, internationalization and localization, interoperability, and accessibility.

Rationale:

To provide authors with a visual means of representing widgets to end-users prior to instantiation. The icon may also serve as visual means for end-users to associate an icon with a widget. For example, a small graphic of a calendar showing today's date may represent a calendar widget.

R21. Configuration Parameters

A conforming specification MUST specify a means to for authors to declare values of custom and predefined configuration parameters, all of which would be applied as a widget is instantiated. A conforming specification MUST specify the default values for specification-defined parameters in case a parameter is missing or the value supplied by the author is invalid. A conforming specification SHOULD allow a widget user agent to override author defined parameters in circumstances where it might be beneficial for users or has the potential to improve performance or stability of the widget user agent.

Motivation:

Ease of use, and current development practice or industry best-practice.

Rationale:

To allow authors to declaratively control how a widget is configured during instantiation. And, in the absence of any declarations, allow the widget user agent to automatically configure a widget using default values. For example, the author might declare that the value for the parameter `width` = `50` indicating the `50` as the value for visual width. Or, in the absence of an author-declared width, the widget user agent will automatically set the `width` to some standardized value (e.g. `300`).

R22. Author-defined Start-up Values (Preferences)

A conforming specification MUST specify a means to for authors to declare persistently stored name-values pairs that an author MAY use to configure a widget instance when the widget is first run. A conforming specification MUST allow authors to define which values are read only, meaning that those values MUST be protected from modification at runtime. A conforming specification MUST specify that these values MUST be made available to the author at runtime via scripting.

Motivation:
: Ease of use, and current development practice or industry best-practice.

Rationale:
: To allow authors to configure a widget through declarative means using name value pairs (i.e., "preferences" for a widget). This would allow authors to, for instance, dynamically generate a widget to have predefined values that could be custom for an end-user. For example, a user might want a weather widget that defaults to their home city. When the widget is generated, the value of the home city is included in the configuration document so when the widget starts, the end-user is not required to specify which city they want to receive the weather forecast for.

R23. Feature Access Declarations

A conforming specification MUST specify or recommend a means to allow authors to declare that an instantiated widget will require access to formally standardized features that allow access to device-specific capabilities or proprietary features (e.g. a proprietary API to access the camera on a device). A conforming specification MAY be specified in such a way that fallback relationships can be declared so that if one feature is unavailable, another can be declared as a possible substitute. In addition, a conforming specification MUST provide authors with a means of stating which features are optional and which features are mandatory for a widget to run.

Motivation:
: Device independence, ease of use, security, and interoperability.

Rationale:
: To allow authors to securely request access to device specific services and features, and to allow widgets to use proprietary features but with a degree of graceful degradation if a feature is unavailable to a particular widget user agent.

R24. Configuration Document Independence

A conforming specification MUST specify the configuration document format in such a way that it can be used independently of the widget package that contains it. A conforming specification MAY provide guidelines for how the configuration document can be used separately from a widget package.

Motivation:
: Ease of use, Web and offline distribution, and device independence.

Rationale:
: To allow the configuration document to be accessed by other applications (either on the server or on the client side) for unrelated purposes. For example, in the context of a widget gallery Web site, a server may automatically extract the configuration document from a widget package and serve it upon request. The extracted configuration document may then be used, for instance, for checking information

about the widget without needing to download the complete widget package. This may be particularly useful for users of widgets on mobile devices, where the cost of downloading data can sometimes be expensive.

R25. Preferred Display Mode

A conforming specification MUST provide a means for author to indicate at least one preferred display mode for a widget. In the absence of a preferred mode, a conforming specification SHOULD provide a consistent default display mode across all user agents. A conforming specification SHOULD make it possible for an author to indicate to the widget user agent which display modes the widget has been designed to run in. The Widget User Agent MAY ignore the indications of display mode supported, but SHOULD NOT ignore the preferred display mode.

Motivation:
Ease of use, Web and offline distribution, and device independence.

Rationale:
To provide authors a means to indicate a preference over how their widget is initially rendered, though this would not be not guaranteed by the widget user agent. A means of declaring the preferred display mode also provides authors some reassurance, as some widgets may be better suited to being displayed in one display mode over the others. As already stated, widget user agents may choose to ignore the author's display mode preference, for example, if they do not support the indicated display mode.

4.3 Application Programming Interfaces

This section enumerates the requirements that a conforming specification needs to address to standardize an ***API for widgets***. The objective of this section is to ensure that a conforming specification specifies an API that allows authors to, amongst other things:

- Manipulate the preferences and properties of an instantiated widget.
- Capture widget-specific events.
- Safely access services, resources, and other applications on the user's device.

R26. Instantiated Widget API

A conforming specification MUST specify a set of interfaces that expose properties, methods, and events of an instantiated widget. These interfaces MUST be encapsulated as a self-contained object, or some similar data structure, in a non-object-oriented programming environment.

Motivation:
Ease of use, compatibility with other standards, and current development practice or industry best-practice.

Rationale:
To allow authors to make their widgets interactive. See for example Apple's `widget` Object described in the [Dashboard Reference] and Microsoft's `System.Gadget` Object described in the [Sidebar Reference].

R27. IDL Definitions

A conforming specification MUST specify the APIs in this section in a standardized interface definition language (e.g. [Web IDL]).

Motivation:
: Compatibility with other standards, current development practice or industry best-practice, and internationalization and localization.

Rationale:
: To facilitate implementation in [ECMAScript], which is already widely supported in widget user agents. Facilitating the implementation of the APIs in ECMAScript will allow authors to use existing ECMAScript code libraries such as jQuery, Dojo, and Prototype.

R28. Configuring Runtime Properties

A conforming specification SHOULD specify a set of interfaces that expose relevant properties and methods of the widget user agent.

Motivation:
: Ease of use, compatibility with other standards, and current development practice or industry best-practice.

Rationale:
: To allow authors to access to any relevant state information or helper methods provided by the widget user agent. Such properties could include localization information, operating environment details, availability of network access, etc. See for example Microsoft's [Sidebar Reference].

R29. Manipulation of Author-defined Start-up Values

A conforming specification MUST specify or recommend a set of interfaces for dynamically getting and setting persistently stored values for a widget instance set using the means provided by the author-defined start-up values. A widget user agent MUST persistently retain a widget's author-defined start-up values in the case where a widget is re-instantiated or the widget user agent is restarted. A conforming specification MUST recommend an API that allows authors to iterate and modify the author-defined start-up values that are not set to read-only, clear the values, add new values, get all the names of the values, and get the number of values currently stored. A conforming specification MUST specify that, if an author attempts to modify a value that is set as read-only, the API throw an appropriate access violation exception.

Motivation:
: Ease of use, compatibility with other standards, Web and offline distribution, security, and current development practice or industry best-practice.

Rationale:
: To allow widgets to be closed and re-instantiated without the end-user having to re-input the author-defined start-up values for an instantiated widget. For example, when using a weather widget, the end-user will want to store the preferred location for weather information, and not be asked to input that information again every time the widget is re-instantiated. The same would apply if the user has instantiated two instances of the same widget and would like to see the weather forecast for two

different cities (e.g., Paris and Sydney). When the widgets are re-instantiated, the corresponding weather information would be downloaded to match each widget's city value.

R30. Widget State Change Events

A conforming specification MUST define a set of states in the lifecycle of the instantiated widget as well as how and when an instantiated widget enters each state. Changes in states MUST have associated events which can be consumed by event handlers, such as scripts. Additionally the API MUST expose the current state. A conforming specification MUST NOT require the widget user agent to send events to the widget immediately, and SHOULD allow the widget user agent to dispatch the events at its convenience.

Motivation:
Current development practice or industry best-practice, and ease of use.

Rationale:
To allow authors to capture state-change events generated by the instantiated widget.

R31. Network State Change Events

A conforming specification MUST specify a means that allows authors to check if a widget instance is connected to the network. A conforming specification MUST define the scope of the term "network" and MUST specify a means by which connection and disconnection events can be captured by an author through script. A conforming specification MUST NOT guarantee event delivery, as there may be cases where a device is running low on resources (e.g., power) and can not afford to deliver them.

Motivation:
Current development practice or industry best-practice, and ease of use.

Rationale:
To allow authors to programmatically capture when the widget user agent has acquired or lost a network connection, particularly for cases when the device intermittently loses and regains the network connection.

R32. Modal Priority

A conforming specification SHOULD specify how an instantiated widget (or any of its presentation contexts) should classify itself to the widget user agent as critical, floating, output-only, etc.. Some of these mode changes may require the end-user's attention, in which a case conforming specification SHOULD recommend that widget user agent find a suitable way to draw the end-user's attention.

Motivation:
Current development practice or industry best-practice, ease of use, and accessibility.

Rationale:
An example of this kind of behavior can be seen on Mac OS X, where a program's icon bounces in the dock when a program has a critical window to display.

R33. Device Specific APIs and Services

A conforming specification SHOULD specify a mechanism, either through an API or through the configuration document, which allows instantiated widgets to bind to third-party APIs that allow access to device-specific resources and services. A conforming specification is NOT REQUIRED to specify any APIs to device specific resources or services, but SHOULD provide some means of binding to those APIs if they are available and the user agrees. A conforming specification SHOULD specify that bindings MUST NOT occur without consulting the user or a policy which exists to represent the end user (or the owner of the device).

Motivation:
: Current development practice or industry best-practice, ease of use.

Rationale:
: To endow widgets with functionality beyond what is currently available to HTML documents, allowing widgets to be used as means to bridge special device capabilities and operating environment services with data on the Web. Examples of device-specific services and resources that could be made available through script include cameras, SMS, GPS and address books.

R34. Configuration Document Data

A conforming specification SHOULD specify a means that allows authors to access data they declared in the configuration document for the widget package.

Motivation:
: Current development practice or industry best-practice.

Rationale:
: To allow authors at runtime to easily access metadata declared in the configuration document.

R35. Scheme Handler

A conforming specification SHOULD specify a means that allows authors to open [IRI]s in an appropriate scheme handler (e.g., using the default Web Browser to open [HTTP] URIs).

Motivation:
: Current development practice or industry best-practice.

Rationale:
: To allow authors to open a URL in the default system Web browser. For example, in a news aggregator widget, to allow the end user to navigate to the source of a particular news item. Alternatively, if the widget deems that a specific content may be better experienced outside the context of the widget user agent, the user can be offered the option of opening the content in the default system Web browser.

R36. Resolve Addressing Scheme

A conforming specification MUST define a mechanism to set the base URI for any DOM instances that occur within the Widget, and it MUST define a mechanism that enables the construction of URI references between different resources within a widget package.

Motivation:
: Current development practice or industry best-practice, interoperability, and security.

Rationale:
: To allow resources to be resolved and normalized within DOM attributes. For example, addressing a resource via an [IRI] reference (e.g. `<img src="images/bg.png'/>` where the `src` attribute resolves to something similar to "`widget://engine/myWidget.wgt/images/bg.png`" or "`http://localhost/myWidget.wgt/images/bg.png`").

R37. Display mode API and Events

A conforming specification MUST specify an API to allow authors to programmatically switch between display modes. A conforming user agent MUST be allowed to ignore requests by the author to switch to an unsupported display mode, but MUST throw an exception or error if it will not perform the mode change. A conforming specification MUST also provide a guaranteed means for authors to detect a change in display mode. A conforming specification MUST provide a means for an author to check the current display mode of a widget.

Motivation:
: Current development practice or industry best-practice, interoperability, and security.

Rationale:
: To give authors a degree of control over the user experience of their widgets.

4.4 User experience

This section enumerates the requirements that a conforming specification needs to address in order to standardize a user experience widgets. The objective of this section is to ensure that a conforming specification make recommendations that would make widgets accessible and able to be rendered on a range of devices.

R38. User Interface Accessibility

A conforming specification MUST specify that the language used to declare the user interface of a widget be a language that is accessible at the various levels specified by the [WCAG 2.0] (perceivable, operable, understandable, and robust): specifically, the language MUST provide keyboard access to interactive graphical elements, and provide means to access the widget's functionality through a non-graphical UI. For the user interface language, the role and state of all interface elements MUST be available to screen readers and other assistive technologies, to allow relevant sections of text and functionality to be accessed.

Motivation:
: Compatibility with other standards, current development practice or industry best-practice, ease of use, and accessibility.

Rationale:

To recommend a language, or a set of languages, that will allow authors to realize their designs, while at the same time remaining accessible to screen readers and other assistive technologies.

R39. Display Modes

A conforming specification MUST specify a set of display modes for widgets that stipulate how widgets SHOULD be rendered at runtime when in a specific mode. A conforming specification SHOULD also define particular allowed, or disallowed, user-interaction behaviors for each display mode; such as the ability for a widget to be dragged or re-sized. For each display mode, the way in which the widget is displayed MUST be specified so that the rendering of the Widget is as consistent as possible across widget user agents. The display modes SHOULD also be specified to interoperate with device independence best practices and/or specifications. Proprietary display modes MAY be supported by the Widget User Agent.

Motivation:
Compatibility with other standards, current development practice or industry best-practice, ease of use, and accessibility.

Rationale:
To provide authors with a variety of commonly widget display modes and to help ensure that their widgets are renders as consistently as possible across different Widget User Agents. In addition, allowing proprietary display modes provides a means to support innovative user experiences.

4.5 User Agents

This section enumerates the requirements that a conforming specification needs to address in order to standardize certain aspects of widget user agents. The objective of this section is to ensure that a conforming specification recommends features that will make widget user agents interoperate more effectively with each other and with services on the Web.

R40. Remote and Local Updates

A conforming specification MUST specify a model to allow widget user agents to check if a new version of a widget package has become available online or from local storage. A conforming specification MUST recommend that an updated widget is downloaded only with the user's consent and that users be able to cancel or defer updates. An automatic update MUST preserve the identity of a widget, meaning that that preferences previously set by the user are retained after the update process. A conforming specification SHOULD recommend that, when possible, updates be conducted over a secure communication channel. In addition, a conforming specification SHOULD specify a means for updates to be authenticated. A conforming specification SHOULD also define a mechanism to protect against downgrade attacks using ancient versions of widgets. A conforming specification SHOULD specify that signature verification policies be applied to updates in a manner that is consistent with those applied upon original installation of the widget.

Motivation:

Security, current development practice or industry best-practice, and interoperability.

Rationale:

To allow authors to provide updates for a widget package online. For example, the author could declare in the configuration document an [IRI] for where the widget user agent can check for updates. If an update to a widget package becomes available, then the widget user agent could ask the end-user if they want to download the widget.

R41. Multiple Widget Instances

A conforming specification MUST recommend that a widget user agent allow multiple instances of a widget package to be instantiated. A conforming specification MAY recommend that implementations which have sufficient resources (CPU, memory, etc.) run widgets concurrently as separate processes.

Motivation:

Current development practice or industry best-practice and interoperability.

Rationale:

To allow multiples instances of the same widget to be run at the same time, but possibly be configured differently. For example, instantiating two clock widgets where one displays the time for Amsterdam and the other displays the time for Boston.

R42. Display Mode Switching

A conforming specification MUST allow a widget user agent to dynamically change display mode of a widget. Switching from one mode to another, however, MUST not cause the re-instantiation of the Widget. Furthermore, it MUST be possible for a Widget to seamlessly move between modes, maintaining runtime state and any processes that are in progress.

Motivation:

Current development practice or industry best-practice and interoperability.

Rationale:

To allow a widget user agent to have a degree of control over how widgets are displayed for the purpose of mediating the user experience. For example, the widget user agent my attempt to switch all widgets into floating mode and then display them in a 3D carousel.

4.6 Security and Digital Signatures

This section enumerates the requirements that a conforming specification needs to address in order to standardize industry standard signing and an adequate security model for widgets. The objective of this section is to ensure that a conforming specification specifies a security model that:

- Defines a robust and flexible digital signature scheme and processing model.
- Makes security a fundamental part of the standardization process and permeates all aspects of a conforming specification.
- Limits the potential for widgets to perform harmful operations on end-user's

machine or device.

R43. Runtime Security Exceptions

A conforming specification MUST specify runtime exceptions for when a widget's script attempts to perform an action it's not authorized to perform.

Motivation:
: Current development practice or industry best-practice and security.

Rationale:
: To provide the API with an error recovery mechanism for when a script attempts to perform a disallowed security-sensitive action. For example, a security exception might be thrown if a widget attempts to access the network but has not been granted permission by the widget user agent to do so.

R44. Digital Signatures

A conforming specification MUST specify a means to verify the authenticity and data integrity of all resources in a widget package, with the exception of any resources explicitly excluded by the specification (e.g. the digital signature file itself). A conforming specification MUST provide these capabilities by specifying or recommending a processing model for generating and verifying a digital signature associated with a widget package. The digital signature scheme MUST be compatible with existing Public Key Infrastructures (PKI), particularly [X.509v3].

Motivation:
: Security and current development practice or industry best-practice.

Rationale:
: To provide a means to verify the authenticity, check the data integrity and provide persistent proof of origin of the widget package. Some vendors may choose to use digital certificates as a means of quality assurance, whereby only widgets that meet a particular level of quality and security receive a digital signature.

R45. Multiple Signatures and Certificate Chains

A conforming specification SHOULD recommend that it should be possible for a widget package to contain multiple independent digital signatures (i.e. it be possible to include multiple signatures and associated certificate chains). A conforming specification MUST specify the expected behavior when multiple signatures and certificate chains are provided. A conforming specification MUST specify that if none of the signatures and certificate chains can be verified, e.g. because of missing root certificates or where any certificates in the chain have expired or are not yet valid, then the widget package SHOULD be treated as unsigned (meaning that widget is treated as if it had no digital signature).

Motivation:
: Web and offline distribution, device independence, and current development practice or industry best-practice.

Rationale:
: To enable the inclusion of certificate chains that the receiving device can use to build a certificate chain from the end entity certificate, which is then used to verify

the signature against the appropriate locally stored root certificate.

R46. Signature Document Format

A conforming specification MUST recommend a digital signature format that can be extracted and conveyed independently of the widget package. A conforming specification SHOULD provide guidelines for how any digital signature can be used separately from a widget package. An example of such use is to perform certificate chain validation and other checks related to the signature key information, without necessarily validating the referenced widget content at that time. Risks associated with separating time of verification and validation steps MAY need consideration.

Motivation:
: Web and offline distribution, device independence, and current development practice or industry best-practice.

Rationale:
: To allow signature files to be extracted and used by other applications, either on the server-side or on the client-side, for different purposes. For example, a server may automatically extract the signature information from a widget package and serve it upon request. The independent signature information may then be used, for instance, to provide the user with information about the signer and associated trust level of the widget package without needing to download the entire widget package. Additionally, if combined with security declaration information, the signature information may allow a security decision to be made about whether or not it will be possible for the widget user agent to instantiate the widget; hence enabling the end-user or the widget user agent to decide if widget package should be downloaded. This may be particularly useful for users of widgets on mobile devices, where the cost of downloading data can sometimes be expensive.

R47. Support for Multiple Message Digest Algorithms

A conforming specification MUST specify at least one mandatory to support message digest algorithm. A conforming specification SHOULD recommend that other message digest algorithms may be supported.

Motivation:
: Security and longevity.

Rationale:
: To provide a transitional means for developers to move towards using the more secure hashing algorithms.

R48. Support for Multiple Signature Algorithms

A conforming specification MUST recommend that where a widget package is digitally signed, it MUST be possible to select from multiple message signature algorithms. A conforming specification MUST mandate that at least one signature algorithm be supported by a widget user agent.

Motivation:
: Security, longevity, and current development practice or industry best-practice.

Rationale:

To lessen the seriousness against the risk that weaknesses will be found with a selected algorithm.

R49. Key Lengths

A conforming specification MUST recommend that widget user agents support processing signatures with key lengths of 2048 bits or greater. A conforming specification MUST recommend to authors that widget packages be signed with key lengths of 2048 bits or greater.

Motivation:
Security and longevity.
Rationale:
To be in-line with current security recommendations and provide longevity of the system security.

R50. Key Usage Extension

A conforming specification MUST specify the expected use of valid key usage extensions and when present (in end entity certificates) MUST specify that implementations verify that the extension has the `digitalSignature` bit set (as defined in [X.509v3]). A conforming specification MUST specify that implementations recognize the extended key usage extension and when present (in end entity certificates) verify that the extension contains the `id-kp-codeSigning` object identifier.

Motivation:
Security.
Rationale:
To maintain compliance to [X.509v3] (experience suggests that if the use of the extended key usage extension is not explicitly required, then X.509v3 is not followed when it comes to key extension usage). Compliance ensures that only certificates intended to be used (issued for) code signing can be used to sign widget packages.

R51. Inclusion of Revocation Information

A conforming specification SHOULD specify a means of packaging up-to-date revocation information with a digital signature and associated certificate chain (e.g. a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) response stating that certificate has not been revoked). In addition, a conforming specification SHOULD specify the behavior in the case that the revocation information is not included or not complete. A conforming specification SHOULD specify that if the revocation information is present the widget processing environment MUST attempt to verify the revocation information. A conforming specification SHOULD specify the behavior if revocation information is out of date or otherwise invalid.

Motivation:
Security
Rationale:
To enable an instantiated widget to obtain revocation information without having to

query an online CRL or OSCP server from each device. This is significantly more efficient and eases the load on CRL or OCSP servers. Note, however, that the revocation information may not be as up to date as an online query. However, if this information is updated at the server in a timely manner before widget installations, then an online query would not be necessary at the client.

R52. Default Security Policy

A conforming specification MUST specify a default security policy that limits the privileges afforded to a widget at runtime. The default security policy MUST be specified in such a way that it forces a widget package to explicitly request permission to use particular device capabilities (see also the Security Declarations requirement).

Motivation:
Current development practice or industry best-practice and security.

Rationale:
To make the default behavior of a widget as benign as possible. For example, the default security policy might be that a widget cannot access the network.

R53. Widget Black/White Listing

A conforming specification MAY specify a mechanism that allows a remote trusted authority to update black and/or white lists and the security policy related to widget packages installed on the widget user agent.

Motivation:
Current development practice or industry best-practice and security.

Rationale:
To provide the mechanisms that would enable the creation of trusted public authorities for widgets. These authorities could serve to authorize or revoke widget packages that other members of the community have found to exhibit undesirable aspects or malicious behavior, which could potentially damage an end-user's device or breach their privacy or security.

R54. Security Declarations

A conforming specification MUST specify or recommend a means for declaring that an instantiated widget will require access to resources or services that have to the potential to introduce a security risk for an end user. A conforming specification SHOULD also make it possible to externalize and associate security declarations with a particular widget package (e.g., by allowing security declarations to be securely acquired from an external trusted authority over [HTTP]). This MUST include a means of declaring the APIs that a widget expects to access. When possible, a conforming specification MUST specify a means to verify the authenticity and integrity of security declarations included in the widget package (e.g. by using Digital Signatures).

Motivation:
Security, and current development practice or industry best-practice.

Rationale:
To declare the security intentions of the widget, allowing the widget user agent to, for example, confirm with the user before installing the widget, or adjust its policies

before instantiating it. Example of security sensitive services that could require access-control include accessing end-user's storage device, or performing a cross-domain request.

References

[*Ajax*]
Ajax: A New Approach to Web Applications. J. J. Garrett. Adaptive Path.

[*BCP 47*]
[Tags for Identifying Languages], A. Phillips and M. Davis. September 2009.

[*CSS*]
Cascading Style Sheets, level 2, revision 1, B. Bos, T. Çelik, I. Hickson, and H. Wium Li.e. W3C.

[*Dashboard Reference*]
Dashboard Reference, Apple Computer, Inc.

[*Digital Signatures for Widgets*]
XML Digital Signatures for Widgets. M. Cáceres, P. Byers, S. Knightley, F. Hirsch, M. Priestley (Work in Progress). W3C.

[*ECMAScript*]
ECMAScript Language Specification, Third Edition. ECMA.

[*Google Gadgets*]
Google Desktop Sidebar Scripting API, Google Inc.

[*HTML*]
[HTML - Living Standard]. I. Hickson. WHATWG.

[*HTTP*]
Hypertext Transfer Protocol -- HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee. IETF.

[*i18n-XML*]
Practices for XML Internationalization. Y. Savourel, J. Kosek, R. Ishida. Working Group Note. W3C.

[*IRI*]
Internationalized resource Identifiers (IRIs), M. Duerst, M. Suignard. IETF.

[*Media Type*]
Multipurpose Internet Mail Extensions (MIME) Part Two: media types, N. Freed and N. Borenstein. IETF.

[*MWBP*]
Mobile Web Best Practices 1.0. J. Rabin, C. McCathieNevile. W3C.

[*WCAG 2.0*]
Web Content Accessibility Guidelines 2.0, B. Caldwell,M. Cooper, L. Guarino Reid, G. Vanderheiden. W3C.

[*Widget Packaging and Configuration*]
Widget Packaging and XML Configuration. M. Cáceres. W3C.

[*Widget Updates*]
Widget Updates. M. Cáceres, R Tibbett, R. Berjon (Work in Progress). W3C.

[*RFC2119*]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner. IETF.

[*Sidebar Reference*]
Windows Sidebar Reference, Microsoft Corporation.

[*Unicode*]
The Unicode Standard, The Unicode Consortium.

[*UTF-8*]

UTF-8, a transformation format of ISO 10646, F. Yergeau. IEFT.

[*Web IDL*]
Web IDL. C. McCormack (Work in Progress). W3C.

[*Widget Interface*]
Widget Interface. M. Cáceres (Work in Progress). W3C.

[*Widget URIs*]
Widget URIs. M. Cáceres (Work in Progress). W3C.

[*W3C Process*]
World Wide Web Consortium Process Document. I. Jacobs, W3C.

[*View Modes*]
The 'view-mode' Media Feature. R. Berjon and M. Cáceres (Work in Progress). W3C.

[*XML*]
Extensible Markup Language (XML) 1.0 Specification (Second Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler. W3C.

[*X.509v3*]
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, D. Cooper, S. Santesson, S. Boeyen, R. Housley, and W. Polk. IETF.

Acknowledgments

The editor would like to thank to the following people who have reviewed or otherwise contributed to this document (ordered by first name):

Alexander Dreiling, Anne van Kesteren, Arthur Barstow, Arun Ranganathan, Bárbara Barbosa Neves Benoit Suzanne, Bert Bos, Bradford Lassey, Bryan Sullivan, Cameron McCormack, Charles McCathieNevile, Cliff Schmidt, Claudio Venezia, Coach Wei, Corin Edwards, Cynthia Shelly, Cyril Concolato, Dan Brickley, David Pollington, David Rogers, Dean Jackson, Doug Schepers, Ed Voas, Gene Vayngrib,Guido Grassel, Jay Sweeney, Jim Ley, Jon Ferraiolo, Jose Manuel Cantera Fonseca, Josh Soref, Kevin Lawver, Krzysztof Maczyński, Lachlan Hunt, Mark Baker, Marc Silbey, Mikko Pohja, Michael Smith, Nick Allott, Olli immonen, Paddy Byers, Philipp Heltewig, Richard Tibbett, Sally Cain, Sean Mullan, Stephen Paul Weber, Steven Faulkner, Thomas Landspurg, Yang Wong, Zachary Fitz-Walter.