Tizen® 2.3.1 Compliance Specification for Wearable Profile

Version 1.0

Revision History

Revision	Date	Author	Reason for Changes
Tizen 2.2.1 Compliance Specification for Wearable Profile, v1.0	25 Mar 2014	Tizen TSG	Initial release
Tizen 2.3.1 Compliance Specification for Wearable Profile, v1.0	22 Sep 2015	Tizen TSG	Official release, adding Native API

Glossary

Term	Definition
ABI	Application Binary Interface, the runtime interface between a binary software program and the underlying operating system.
API	Application Programming Interface, the interface between software components, including methods, data structures, and processes.
Compliance	Certified for full conformance, which was verified by testing.
Conformance	How well the implementation follows a specification.
CSS	Cascading Style Sheets, a simple mechanism for adding style (for example fonts, colors, and spacing) to web documents.
DOM	Document Object Model, a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents.
DTV	Digital Television, a target of the TV Profile.
GPS	Global Positioning System.
HSPA	High Speed Packet Access, a mobile broadband technology.
IOMMU	Input/Output Memory Management Unit.
IPTV	Internet Protocol Television, a target of the TV Profile.
IVI	In-Vehicle-Infotainment, a target of the IVI Profile. System used for entertainment, such as music, video, and games, along with information, such as navigation and web. A platform target for Tizen.
jQuery	Portable client-side JavaScript library.
LTE	Long Term Evolution, a telephone and mobile broadband communication standard.
Mobile	Portable, connected devices, such as phones and tablets. A platform target for Tizen.
NFC	Near Field Communication, a form of contactless communication between devices containing an NFC tag, such as smartphones , tablets, smart signs, kiosks etc.
REST	Representational State Transfer, design model used by the World Wide Web based on a client/server architecture where the client requests information and the server processes the request and returns information.
SDB	Smart Development Bridge, a device management tool in the Tizen SDK.

Term	Definition
STB	Television set-top box, a target of the TV Profile.
Side loading	Installing applications or components other than from a certified application installer package.
Smack	Simplified Mandatory Access Control Kernel, an access control technology used by Tizen to protect data and prevent malicious programs from causing harm.
UI	User Interface, the widgets, theme, and layout of software components displayed on the device screen (if present) through which the user may interact with the device. Usually refers to the visual software elements but may also include hardware buttons or controls.
UX	User experience, the effect that the design of a system (both software and hardware) has on the user of the system.
Tizen Native API	Collection of Tizen native interfaces, standard C and C++ libraries, and a selected set of open source libraries such as EFL, OpenAL, libxml2, etc.
Tizen Web API	Collection of Tizen web programming interfaces for applications. Includes approved specifications generically known as HTML5, as well as additional interfaces such as Tizen Web Device API and Tizen Web UI FW.
Wearable	Miniature computing devices worn by the user on the body or clothing.
WPS	Wi-Fi based Positioning System.

Table of Contents

1. Overview

This specification defines the operating environment of the Tizen platform. It is intended to be used by both application developers and wearable device implementers to enable the development of portable application software.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" used in this document are to be interpreted as described in [ref. 5].

Tizen is a registered trademark of the Linux Foundation, which controls the usage of the brand and trademark. A requirement for permission to use this trademark in conjunction with products is compliance with the requirements of this specification.

1.1. Why Compliance?

Tizen Compliance is designed to ensure wearable device implementations and applications work together.

1.2. Target Audience

This specification is intended to be used by:

- **Application developers**: know how to create compatible applications that work across multiple devices and know how Tizen devices will behave.
- **Wearable device implementers**: know how to implement device hardware, security configurations, services, APIs, etc.
- **Operators**: know how to customize and enhance a device while remaining within compliance guidelines.

1.3. Tizen Compliance Model

To become Tizen compliant, a device MUST obtain Tizen Compliance certification from the Tizen certification authority for at least one Tizen Profile by satisfying the requirements of the Tizen Compliance Specification and passing all of the Tizen Compliance Tests for that profile.

A Tizen Profile describes the requirements for a category of Tizen devices that have a common application execution environment. Applications are created for a specific target profile and can run on devices compliant to that profile.

- **Device implementations**: if implemented to a profile, a device will provide applications with consistent behavior defined by that profile, as well as a consistent user experience.

- **Applications**: built to a profile, applications will run on devices that are compliant to that profile.

The Tizen Compliance Tests for a profile will measure conformance to the Tizen Compliance Specification for that profile.

Note: This specification describes only the compliance requirements for the Tizen Wearable Profile. Other supported profiles have their own related specifications.

1.4. Revision Policy

There will be a distinct release of the specification, as well as matching compliance tests, for each distinct release (version) of the Tizen platform. If deemed necessary, updates may be issued between releases. All compliance requirements for the Tizen Wearable Profile specification must be approved by the Tizen Technical Steering Group (TSG) and may change from time to time, only by approval of the Tizen Technical Steering Group.

1.5. Tizen Source Code Modification Policy

All Tizen implementations MUST provide the full behavior of the Tizen API and application execution environment as defined by the Tizen Profile for its device category. The best way to accomplish this is by using the source code for the Tizen reference implementation. If modifications or replacements to the source code must be made, the implementer is responsible for making sure that there is no impact on compliant applications. The Tizen Compliance Tests may be used to measure the correctness of the implementation, but in case of ambiguities, errors, or incompleteness of this specification or of the Tizen Compliance Tests, the final arbiter of compatibility is the behavior of the Tizen reference implementation.

1.6. References

The following external specifications and other documents are referenced by this specification.

[N]: Normative Reference
[I]: Informative Reference

1. [I] Dynamic Analyzer Reference: https://developer.tizen.org/development/tools/native-tools/dynamic-analyzer
2. [N] Globalize 0.1.0a2: https://github.com/jquery/globalize/tree/v0.1.0a2
3. [I] Eglibc 2.13: http://www.eglibc.org/home
4. [I] GNU C/C++ compiler 4.6.4: http://gcc.gnu.org
5. [N] IETF RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels": http://www.ietf.org/rfc/rfc2119.txt

6. [N] ISO 18092 “Near Field Communication – Interface and Protocol (NFCIP-1)”: http://standards.iso.org/ittf/PubliclyAvailableStandards/c038578_ISO_IEC_18092_2004(E).zip
7. [I] libxml2 2.7.8: http://www.xmlsoft.org/html/index.html
8. [N] jQuery 1.8.2: http://blog.jquery.com/2012/09/20/jquery-1-8-2-released/
9. [N] jQuery Mobile 1.2.0: https://github.com/jquery/jquery-mobile/tree/1.2.0
10. [I] Log View Reference: https://developer.tizen.org/development/tools/common-tools/log-view
11. [I] NFC Specifications: http://www.nfc-forum.org/specs/
12. [N] OpenAL 1.1 Specification and Reference: https://source.tizen.org/sites/default/files/page/openal_1.1_specification.pdf
13. [N] OpenGL ES 1.1.12: http://www.khronos.org/registry/gles/specs/1.1/es_full_spec_1.1.12.pdf
14. [N] OpenGL ES 2.0.25: http://www.khronos.org/registry/gles/specs/2.0/es_full_spec_2.0.25.pdf
15. [N] OpenMP Application Program Interface 3.0: http://www.openmp.org/mp-documents/spec30.pdf
16. [I] Smart Development Bridge: https://developer.tizen.org/development/tools/common-tools/smart-development-bridge
17. [N] Tizen Native API Modules List: https://developer.tizen.org/development/api-references/native-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.native.wearable.apireference/modules.html
18. [I] Tizen Native API Packaging: https://developer.tizen.org/development/getting-started/native-application/application-development-process
19. [N] ISO/IEC 14882 2011, section "20.7.1 Class template unique_ptr": http://en.cppreference.com/w/cpp/memory/unique_ptr
20. [I] Tizen Native Application Development Process: https://developer.tizen.org/development/getting-started/native-application/application-development-process
21. [I] Tizen Native Application Lifecycle: https://developer.tizen.org/development/getting-started/native-application/tizen-application-model#life
22. [N] Tizen Native Application Model: https://developer.tizen.org/development/getting-started/native-application/tizen-application-model
23. [I] Tizen UX Guide: https://developer.tizen.org/design
24. [N] Tizen Web Supplementary API References: https://developer.tizen.org/development/api-references/web-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.web.apireference/html/w3c_api/w3c_api_cover.html
25. [N] Tizen W3C/HTML5 API References: https://developer.tizen.org/development/api-references/web-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.web.apireference/html/w3c_api/w3c_api_cover.html
26. [N] Tizen Wearable Web Device API Reference: https://developer.tizen.org/development/api-references/web-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.web.apireference/html/device_api/device_api_cover.html

27. [N] Tizen Web Runtime Core Specification 2.3: https://source.tizen.org/sites/default/files/page/tizen-2.3-wrt-core-spec.pdf
28. [N] Tizen Web UI Framework Reference: https://developer.tizen.org/development/api-references/web-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.web.apireference/html/ui_fw_api/ui_fw_api_cover.html
29. [N] W3C Widget Access Request Policy (W3C Recommendation 7 February 2012 version): http://www.w3.org/TR/2012/REC-widgets-access-20120207/
30. [N] Optional Tizen W3C/HTML5 Features: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/application-filtering
31. [N] Optional Tizen Web Device Features: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/application-filtering
32. [N] Optional Tizen Native Features: https://developer.tizen.org/development/getting-started/native-application/understanding-tizen-programming/application-filtering
33. [I] Tizen Application Filtering: https://developer.tizen.org/development/getting-started/native-application/understanding-tizen-programming/application-filtering
34. [I] Application Controls for Tizen Web applications: https://developer.tizen.org/development/guides/web-application/tizen-features/application/application#controls
35. [I] Application Controls for Tizen Native applications: https://developer.tizen.org/development/guides/native-application/application-framework/application#app_controls
36. [I] Tizen Web Application Security and Privacy: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/web-runtime
37. [I] Tizen Web Application Package Manager: https://developer.tizen.org/development/getting-started/web-application/tizen-application-model
38. [I] Tizen Native Application Package Manager – Package ID and Application ID: https://developer.tizen.org/development/getting-started/native-application/tizen-application-model#packageID
    [I] Tizen Native Application Package Manager – Application Directory Policy: https://developer.tizen.org/development/getting-started/native-application/tizen-application-model#appdirectory
    [I] Tizen Native Application Package Manager – Package Commands: https://developer.tizen.org/development/getting-started/native-application/tizen-application-model#commands
39. [I] Packaging Hybrid Applications: https://developer.tizen.org/development/getting-started/web-application/tizen-application-model
40. [N] Tizen Push Messaging: https://developer.tizen.org/development/guides/native-application/messaging/push
41. [N] Tizen Native API Reference: https://developer.tizen.org/development/api-references/native-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.native.wearable.apireference/modules.html

42. [N] Tizen Native API: EFL API reference: https://developer.tizen.org/development/api-references/native-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.native.wearable.apireference/group__EFL__Group.html
43. [N] Tizen Provided EFL Elementary Widgets: https://developer.tizen.org/development/api-references/native-application?redirect=https%3A//developer.tizen.org/dev-guide/2.3.1/org.tizen.native.wearable.apireference/group__elm__widget__group.html
44. [N] Tizen Native API: EFL Elementary Widget Styles: https://developer.tizen.org/development/guides/native-application/ui-framework/ui-components/wearable-native/elementary/ui-component-styles
45. [N] Tizen Native API: Font Settings: https://developer.tizen.org/development/guides/native-application/ui-framework/font-setting
46. [N] Tizen Web API Privileges: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges
47. [N] Tizen Native API Privileges: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges
48. [N] Tizen Native API Non-API Bound Privileges: https://developer.tizen.org/development/getting-started/web-application/understanding-tizen-programming/security-and-api-privileges

2. Wearable Profile Software Compliance

This chapter describes the software requirements that implementers MUST meet to create a compliant Tizen Wearable device.

2.1. General Principles

Wearable device implementations MUST include support for both the Tizen Web API and the Tizen Native API.

- The wearable device implementation MUST report the availability of optional hardware and software features (see section 2.7) as platform attributes.
- If a wearable device implementation supports a particular optional hardware or software feature, it MUST implement the entire corresponding API..
- Whether a wearable device implementation supports or does not support a particular optional hardware or software feature, the compliance tests MUST be passed. If the feature is not supported, the corresponding API MUST return the unsupported return value, as described in sections 2.2.4 for Web API and 2.3.3 for Native API.

2.2. **Tizen Web API**

2.2.1. Namespace

Wearable device implementations MUST NOT modify the API namespace listed in the Tizen Web Device API Reference [ref. 26], including `tizen.*`.

2.2.2. Tizen Web API Categories

- **W3C/HTML5 APIs**: include the standard APIs defined by W3C, such as HTML5, CSS3, and Widget Specification. See [ref. 25].
- **Supplementary APIs**: non-W3C specifications, such as WebGL, Typed Array, FullScreen API, and viewport Meta Tag. See [ref. 24].
- **Web Device API**: defined by the Tizen project to facilitate the development of web applications by accessing various device features that are not fully covered by W3C APIs. The APIs enable interacting with device features, such as calendar, contact, Bluetooth, NFC, messaging, alarm, or system information. See [ref. 26].

2.2.3. Preliminary Web APIs

The Tizen Web API includes some preliminary Web API specifications which are in an early stage in the W3C development cycle. Preliminary revisions are referred to in the W3C maturity model as Editor's Draft (ED), Working Draft (WD), and Last Call Working Draft (LCWD). Application developers are cautioned that APIs in these specifications could be modified in a future version of Tizen to align with the developing progress of specifications. Preliminary APIs are indicated in the W3C/HTML5 APIs reference [ref. 25].

Wearable device implementations MUST support all Tizen Web APIs from the Tizen Web API specifications, including those indicated as preliminary.

2.2.4. Behavior of Unsupported Web APIs

Wearable device implementations MUST NOT omit any Web API listed in the Tizen Web API specification, except those specified as optional in section 2.7.1 and not supported on the device. Optional APIs are dependent on particular hardware or software availability.

If an optional API is not supported on the device, it MUST return "undefined" when a whole module is not supported. For example, an attempt to access `tizen.nfc` MUST return "undefined" if the NFC module is not supported on the device. In case APIs in a module depend on a certain optional feature which is not present, those APIs MUST report

NotSupportedError. For example, if MMS is not supported on the device, an attempt to call `tizen.messaging.getMessageServices("messaging.mms", successCallback)` MUST report `NotSupportedError`.

2.3. **Tizen Native API**

2.3.1. Namespace

The Tizen Native API does not have a distinct namespace. Developers MUST take care not to conflict with symbols used by the Native API [ref. 17].

2.3.2. Tizen Native API Categories

The Tizen Native API is defined by a C library [ref. 41].In addition, the following native open source library APIs MUST be available for use by native applications:

- C library, as implemented by eglibc 2.13 [ref. 3]
  - Wearable device implementations are not required to provide any specific commands for use by the system() and popen() interfaces and the exec() family of interfaces.
- C++ Standard Library: ISO/IEC 14882-2003 and ISO/IEC TR 19768:2007(C++ TR1) compliant C++ standard library implementation within g++ 4.6.4 [ref. 4]
  - std::unique_ptr from C++11 [ref. 19]
- OpenAL 1.13 [ref. 12]
- OpenMP 3.0 (part of GCC 4.6.4) [ref. 15]
- libxml2 2.7.8 [ref. 7]
- EFL 1.7
  - A subset of the full Enlightenment Foundation Libraries (EFL) API set MUST be available to Tizen native applications as described in [ref. 42].
- EXIF 0.6.21
- Json-glib 0.10.4
- Glib 2.32.3
- Curl 7.40
- Fontconfig 2.9.0
- Freetype 2.5.5
- Minizip 1.2.5
- Sqlite 3.7.13
- Cairo 1.12.14
- Openssl 1.0.1p
- libOAuth 0.9.4

For all Native API libraries listed above, wearable device implementations MUST use the version shown above, or fully API/ABI compatible versions.

2.3.3. Behavior of Unsupported Native APIs

Wearable device implementations MUST NOT omit any native API listed in the Tizen Native API specification [ref. 41], except those specified as optional in section 2.7.2 and not supported on the device. Optional APIs are dependent on particular hardware or software availability.

If an optional API is not supported on the device, it MUST return the corresponding "not supported" error when accessed. For example, BT_ERROR_NOT_SUPPORTED MUST be returned for Bluetooth and NFC_ERROR_NOT_SUPPORTED MUST be returned for NFC.

2.3.4. Native Application Model

The Tizen Native API MUST comply with the Native Application Model [ref. 22].
The Tizen native application model handles application life-cycle and system events in the native framework. The Tizen platform supports both UI applications (which have a graphical user interface) and service applications (which do not have a graphical user interface).

2.3.5. EFL

2.3.5.1. Elementary Widgets

The Tizen EFL Elementary widgets that are supported in the Native framework MUST all be supported [ref. 43].

2.3.5.2. Elementary Widget Styles

The Tizen EFL Elementary widget styles that are supported in the Native framework MUST all be supported [ref. 44].

2.3.5.3. Font Settings

The Tizen font settings that are supported in the Native framework MUST all be supported [ref. 45].

2.4. **Application Binary Interface**

The Application Binary Interface (ABI) describes the compatibility of executable object or binary programs. Use of the Tizen Native API will result in binary programs. Wearable device implementations MUST be compatible with the one of the following ABIs.

The ABI for ARM®Architecture CPUs is supported with these characteristics:

- ABI: aapcs-linux

- CPU architecture: armv7
- CPU instruction set: cortex-a5
- FPU option: vfpv3-d16
- Floating point ABI: softfp
- Endian-ness: little endian

The ABI for Intel® IA32 Architecture CPUs is supported with these characteristics:

- ABI: i386 psABI (gcc: -m32)
- CPU architecture: IA32
- CPU instruction set: SSSE3
- Floating point ABI: SSE math (gcc: -mfpmath=sse)
- Endian-ness: little endian

2.5. Application Control

The application control interface in the Tizen Web API and the Tizen Native API enables launching an application directly using an application ID or invoking specific application functionality remotely through inter-process communication (IPC).

A Tizen application may register itself as an application control provider. The available application control values can be queried and invoked by a Tizen application.

There are no mandatory platform provided application controls in this profile, however wearable device implementations MUST allow Tizen applications to register application controls for use by other Tizen applications.

Further details on Application Controls are provided in the developer documentation [ref. 34 and ref. 35].

2.6. Platform Attributes

Wearable device implementations MUST provide accurate platform attributes through the System Information API for the Tizen Web and Native APIs.

Platform attributes include the following:

- Device capabilities (see section 2.7)
- Information about data storage devices
- Display information
- Information about the device orientation
- Locale information
- Network information

2.7. **Optional APIs**

The Tizen API may depend on available hardware capabilities and, in some cases, on software capabilities. Optional software features may be capabilities not part of the publicly available stack, or may require hardware capability that is beyond the minimum wearable device requirement such as higher processing power or memory requirements (See section 3.1 for minimum hardware requirements).

Wearable device implementations MUST implement all APIs listed in the referenced API specifications, except those specified as optional in this section. Optional APIs are ones dependent on the availability of particular underlying hardware or software features.

2.7.1. Tizen Web API

The Tizen Web APIs specified as optional in [ref. 32] will not be implemented if a wearable device implementation does not include the corresponding hardware or software feature. The wearable device implementation MUST accurately report the availability of these underlying features through the Tizen Web API System Information API.

2.7.2. Tizen Native API

The Tizen Native APIs specified as optional in [ref. 32] will not be implemented if the wearable device implementation does not include or does not choose to report that it includes the corresponding hardware or software feature. The wearable device implementation MUST report the availability of these underlying features that it wishes to make available for use through the Tizen Native API System Information API and various other reporting APIs.

2.8. **Privilege**

Certain APIs have access to privacy-sensitive information (for example contacts, camera and geolocation) or have security or stability implications. If an application uses such APIs, then appropriate privileges MUST be declared in application's *`config.xml`* file and *`tizen-manifest.xml`* for Native applications.

Privilege is affected by the privilege levels described below. In addition to declaring the privilege, the application MUST have access to the required privilege level:

- Public: for all Tizen developers
- Partner: for trusted application developers
- Platform: for OEMs/operators

If an application declares a privilege that requires a level higher than public, and the application is not signed with a certificate granting it access to that level, then the implementation MUST block installation and execution of the application.

Privilege levels are deprecated as of version 2.3 of this specification.

2.8.1. Tizen Web API

If a web application does not declare a required privilege in the *config.xml* file, access to the corresponding API MUST throw `SecurityError` as specified in the Tizen Web Device API Reference [ref. 26]. Wearable device implementations MUST support this mechanism.

Wearable device implementations MUST NOT change the semantics of permissions as documented in the Tizen Web Application Security and Privacy [ref. 36] for applications using the Tizen Web API. A full list of privileges defined for the Tizen Web API is available in the developer's guide [ref. 46].

2.8.2. Tizen Native API

If a native application does not declare a required privilege in the *tizen-manifest.xml* file, wearable device implementations MUST deny access and return a permission denied message if the corresponding API is accessed. For example, `LOCATIONS_ERROR_ACCESSIBILITY_NOT_ALLOWED` is returned when permission is denied with the Location Manager API and `CONTACTS_ERROR_PERMISSION_DENIED` is returned when permission is denied with the Contacts API.

Wearable device implementations MUST enforce a mechanism that limits an application using the Tizen Native API to use privileged APIs only if the privileges it requires are declared. Wearable device implementations MUST NOT change the semantics of permissions and support, as documented in the manifest specification for applications using the Tizen Native API. The full list of privileges defined for the Tizen Native API is available in the developer's guide [ref. 47 and ref. 48].

2.9. **Application Packaging Compatibility**

Tizen defines several mandatory application packaging formats. Wearable device implementations MUST correctly process packages in these formats. They MUST NOT extend these packaging formats in a way that would prevent packages generated on the implementation from running on other conforming wearable device implementations.

Nothing in this section precludes wearable device implementations from supporting additional packaging formats outside the requirements of this specification.

2.9.1. Web App Package Support

Wearable device implementations MUST be able to install, remove, list, and update Web application packages in the .wgt format as described in the Tizen Web Runtime Core

Specification [ref. 27]. .wgt packages MUST NOT contain more than one UI application or the behavior is undefined.

2.9.2. Native App Package Support

Wearable device implementations MUST be able to install, remove, list and update Native application packages in .tpk format, as described in the Tizen Native Application Development Process [ref. 20]. .tpk packages MAY contain multiple applications.

2.9.3. Hybrid Web/Native Package Support

Wearable device implementations MUST be able to install, remove, list and update hybrid Web/Native application packages in .wgt format as described in Packaging Hybrid Applications [ref. 39]. Hybrid .wgt packages contain more than one application, but MUST NOT contain more than one UI application or the behavior is undefined.

2.10. **WebKit and Browser**

2.10.1. WebKit

The WebView and Web Runtime implementations on wearable device implementations SHOULD be based on the WebKit built from the Tizen reference implementation. This is strongly recommended for maintaining application compatibility across Tizen Wearable devices. Any customizations made by device implementations SHOULD NOT alter the original web exposed behavior from the reference implementation version.

If WebKit is used, the user agent string reported by the WebKit MUST follow this format:

Mozilla/5.0 (Linux; Tizen *PLATFORM_VER*; *MODEL*) AppleWebKit/*APPLE_WEBKIT_VER* (KHTML, like Gecko) *APP_NAME*/*APP_VER* Wearable Safari/*APPLE_WEBKIT_VER*

- The value of the *PLATFORM_VER* string MUST be "2.3.1".
- The value of the *MODEL* string SHOULD be the same as the name of the device. There is no specific format for this field.
- The value of the *APPLE_WEBKIT_VER* string MUST be the WebKit version.
- The value of the *APP_NAME* string SHOULD be the same as the name of the application.
- The value of the *APP_VER* string SHOULD be the same as the version of the application.
- Wearable device implementations MAY omit the word “Wearable” from the user agent string.

2.10.2. Browser

Wearable device implementations MAY include a browser.

If included, the browser MUST meet the W3C/HTML5 and Supplementary API specifications [ref. 24 and ref. 25]. The default browser on wearable device implementations SHOULD be based on the WebKit built from the Tizen reference implementation. Any customizations made by device implementations SHOULD NOT alter the original web exposed behavior from the reference implementation version.

2.11. **Web Runtime**

Wearable device implementations MUST support all mandatory requirements in the Tizen Web Runtime Core Specification. [ref. 27]

2.12. **User Interface**

2.12.1. Keys

Wearable device implementations MAY provide the following function keys. Keys may be implemented as hardware buttons, software-defined buttons, or specialized swipe motions, in any combination.

- **Home** - used to navigate to the Home screen in an application. The key will always send the application in use to the background and bring the Home screen to the front.
- **Back** - used to navigate to previous view in the application.
- **Power** - used to turn on/off the device or display.

Regardless of the actual mechanism of implementing keys, the wearable device implementation must deliver the Back key event as a `tizenhwkey` event object with a `keyName` attribute which has value "`back`".

2.13. **Security**

The following are security requirements for Tizen platforms.

- The device MUST follow the Linux standard security model, including:
  - Applications MUST run under a non-root user ID.
  - An application MUST be allowed to read and write files in its home directory.
- Smack-based access control and process isolation:
  - The device MUST have all Smack features from Linux kernel version 3.12 or later, and the Smack features MUST be enabled.

    - All applications MUST run with Smack labels different from the predefined Smack labels.
    - The device MUST use file systems which supports extended attributes (XATTR) and traditional discretionary access control (DAC) attributes such as owner, group, and permissions.
- Secure execution environment:
    - Native applications SHALL be launched by the application framework.
    - Web applications SHALL be launched by the web runtime.
    - There SHOULD NOT be any set-user-ID binaries in the device.
- Smack supported modules:
    - The device SHOULD contain coreutils or equivalent, d-bus, udev, and Xorg with Smack capability enabled by Tizen.
    - The device SHOULD contain the Tizen rpm security plugin.
- Privileged information:
    - The device MUST only allow an application to carry out an operation if it has the privilege and permission to do so. Privileges will be declared in the application's manifest file.

2.14. **Multimedia**

The following media formats/codecs MAY be supported by wearable device implementations.

This following list of codecs is a minimum requirement on a Tizen Wearable Device. Please note that the Tizen Technical Steering Group makes no representation that these codecs are unencumbered by patents. Implementation of these codecs MAY require patent licenses from the relevant patent holders.

Format	**Codec**
Audio codec (Decoder)	AAC LC
	AAC+
	Enhanced AAC+
	AMR-NB
	AMR-WB
	MP3
	Vorbis
Audio codec (Encoder)	AAC LC
	AMR-NB
	AMR-WB
Video codec (Decoder)	H.263
	H.264 HP
Video codec (Encoder)	H.263
	H.264 HP
Image codec (Decoder)	BMP
	GIF

	JPEG
	PNG
Image codec (Encoder)	JPEG
	PNG
	BMP

Type	File Type/Container Format
Audio	MPEG-4 (.mp4, .m4a)
	AAC(.aac)
	MP3 (.mp3)
	Ogg (.ogg)
Video	3GPP (.3gp)
	MPEG-4 (.mp4)
Image	BMP (.bmp)
	GIF (.gif)
	JPEG (.jpg)
	PNG (.png)

2.15. **Developer Tools**

Wearable device implementations MAY include services that enable communication with the Smart Development Bridge (SDB) in the Tizen SDK. If the implementation includes such support, the following development tasks MUST be available:

- MUST support all SDB functions [ref. 16] to interact with the Tizen SDK. The SDB daemon (sdbd) SHOULD support all commands documented in the SDB Commands section of the SDB reference. The implementation SHOULD allow sdbd to be activated by a device user.
- MUST support the Log View [ref. 10] function to retrieve the Tizen platform log (dlog).
- MUST include the Dynamic Analyzer [ref. 1] Analysis framework and make it available for applications to use.

While SDB support is OPTIONAL in production devices, device implementation MUST have a device driver available enabling connection to SDB in order to execute the Tizen Compliance Tests (TCT). This driver need not be available in production devices

2.16. **Software Update**

Wearable device implementations SHOULD provide a mechanism for updating system software. If provided, user data, application private data, and application shared data SHOULD be preserved.

2.17. **Tizen Compliance Tests**

The Tizen Compliance Tests (TCT) for the Tizen Wearable Profile verify conformance to the requirements of this specification. Platforms MUST pass the TCT to be considered Tizen compliant.

2.17.1. Satisfying TCT preconditions

Wearable device implementations MUST satisfy preconditions to pass TCT.

The full list of TCT preconditions that MUST be satisfied are available in the Tizen 2.3.1 TCT for Wearable Profile.

3. **Wearable Profile Hardware Compliance**

This chapter describes mandatory and optional hardware components.

3.1. **Mandatory Hardware Requirements**

These minimum hardware features MUST be provided by a compliant wearable device implementation, and any corresponding API must be fully implemented.

3.1.1. Memory Storage

A Tizen wearable device SHOULD have at least 128 MB of RAM to run Web applications smoothly.

Wearable device implementations MUST allow a host computer to access files in the shared media folders on the device. The precise method is unspecified. Two optional methods are USB mass storage (UMS) and Media Transfer Protocol (MTP).

3.1.2. Sound

Wearable device implementations MUST support at least one audio output.

3.1.3. Connectivity / Networking

Wearable device implementations MUST support at least one form of data networking capable either of accessing the Internet directly or accessing a host device which has data networking capable of accessing the Internet. Examples of acceptable data networking technologies include Wi-Fi, LTE, HSPA, Ethernet, Bluetooth, Bluetooth Low Energy, NFC, etc.

Implementations MAY omit any individual mechanism, as long as at least one method is supported.

3.2. Optional Hardware Requirements

If a wearable device implementation reports that it includes an optional hardware component that has a corresponding API, the implementation MUST fully implement that API, as described in this specification. Wearable device implementers MAY report a hardware component as absent if they choose not to support the full API. Partial API implementations are not permitted.

3.2.1. Camera

A wearable device implementation MAY omit camera devices.

If a wearable device implementation reports that it includes a camera hardware feature, it MUST support at least one of the preview pixel formats for camera previews:

RGB565	The RGB565 pixel format
ARGB8888	The ARGB8888 pixel format
R8G8B8A8	The R8G8B8A8 pixel format The order of color component is guaranteed by the byte unit.
YCbCr420_PLANAR	The 8-bit Y-plane followed by 8-bit 2x2 sub sampled U-plane and V-plane
JPEG	The encoded formats
NV12	The NV12 pixel formats
UYVY	The UYVY pixel format

3.2.2. Display

Wearable device implementations MAY omit a pixel-addressable screen display. If provided, a screen resolution of 320x320, 360x360 or 360x480 is recommended.

3.2.3. Input Devices (Keyboard)

Wearable device implementations MAY provide applications a means of receiving keyboard input from users.

- A soft keyboard or an input method setup MAY be able to augment keyboards not capable of a full QWERTY layout. For example, a 12 key number pad can allow a user to enter alphabetical letters through multiple presses of a numeric key.

Wearable device implementations MAY include a touchscreen capable of single touch. Multi-touch capability is recommended, if possible.

3.2.4. Graphics

A wearable device implementation SHOULD provide 3D Graphics hardware acceleration. While it MAY be omitted, doing so will provide a degraded user experience.

Wearable device implementations MUST accurately report the presence or absence of acceleration hardware.

3.2.5. GPS

A wearable device implementation MAY omit GPS hardware. If the implementation reports that it provides GPS, it MUST support the Location API.

3.2.6. Sensors

A wearable device implementation MAY omit any and all sensors listed in this specification. If an implementation provides any sensor from this specification, it SHOULD meet the specific requirements for that sensor type. See section A.1 for details.

3.2.7. Telephony

A wearable device implementation MAY omit telephony hardware features. If an implementation reports that it includes telephony hardware, it MUST support voice calls and the messaging API (SMS) using cellular technologies.

3.2.8. Bluetooth

A wearable device implementation MAY omit Bluetooth capability. If an implementation reports that it includes Bluetooth hardware features, it MUST support the Bluetooth API.

Wearable device implementations SHOULD implement the Object Exchange (OBEX) protocol.

3.2.8.1. Bluetooth Sub-features

A wearable device implementation MAY omit Bluetooth sub-features, such as Bluetooth Hands-Free (HFP) and Bluetooth Advanced Audio (A2DP), capability. If an implementation reports that it includes Bluetooth sub-feature hardware features, it MUST support the Bluetooth API as well as the respective Bluetooth sub-feature’s API.

3.2.8.2. Bluetooth Low Energy (BLE)

A wearable device implementation MAY omit Bluetooth Low Energy. If an implementation reports that it includes Bluetooth Low Energy hardware features, it MUST support the Bluetooth API as well as the respective Bluetooth Low Energy API.

3.2.9. Wi-Fi

A wearable device implementation MAY omit Wi-Fi capability. If an implementation reports that it includes Wi-Fi hardware features, it MUST support the Wi-Fi API.

3.2.10. Wi-Fi Direct

A wearable device implementation MAY omit Wi-Fi Direct capability. If an implementation reports that it includes Wi-Fi Direct hardware features, it MUST support the Wi-Fi API as well as the Wi-Fi Direct API.

3.2.11. NFC

A wearable device implementation MAY omit NFC capability. If an implementation reports that it includes NFC hardware, it MUST support the NFC API.

Wearable device implementations supporting NFC MUST read/write NFC Data Exchange Format (NDEF) messages in NFC standard formats, such as NFC Forum Tag Types 1, 2, 3, and 4.

Wearable device implementations supporting NFC MUST support sending and receiving data using the following standards [ref. 11]:

- NFCIP-1 (ISO 18092)
- LLCP 1.0
- SNEP 1.0

3.2.12. Audio Input Devices

A wearable device implementation MAY omit a microphone. Wearable device implementations MUST accurately report the presence or absence of a microphone.

3.2.13. Secure Element

A wearable device implementation MAY omit Secure Element capability. If an implementation reports that it includes the Secure Element feature, it MUST support the Secure Element API.

3.2.14. USB

Wearable device implementations SHOULD provide USB client functionality.

The implementation MAY omit physical USB hardware, including an external connector, in a production device. However, USB hardware MUST be available on a version of the device used for running the Tizen Compliance Tests, which NEED NOT be a production device as long as it is otherwise substantially the same hardware. If USB hardware is removed in a production device, the software behavior for 3rd party applications on the production and testing device MUST remain consistent. The TCT cannot be completed without USB support and SDB support, as described in section 2.15.

The implementation MUST support the following when TCT or Developer Tools are supported:

- USB 2.0 or later
- the Smart Development Bridge (SDB)

4. Wearable Profile Application Compliance

This chapter provides information for application developers to aid them in creating applications that will run on Tizen compliant devices.

4.1. API Use

Applications MUST use only the APIs defined in the Tizen Web API and the Tizen Native API specifications when making calls external to the application. Compliant web applications MAY also use any RESTful web APIs implemented using HTTP and the principles of REST (Representational State Transfer).

Web applications MAY also use RESTful APIs provided by other open services, as well as JavaScript libraries included in the resources of the application, subject to the condition that the web application's configuration specifies the REST API domain in the `<access>` tag, according to the W3C Widget Access Request Policy [ref. 29].

4.1.1. Limited use of C library functions

Native applications SHOULD avoid the use of the C library system() and popen() interfaces and the exec() family of interfaces. Wearable device implementations are not required to provide any specific commands to be invoked by those interfaces, so applications using those interfaces MUST use them only to execute application-provided commands. In

addition, launching another program using these interfaces will bypass the normal Tizen Native application lifecycle, which is strongly discouraged [ref. 21].

4.1.2. OpenGL ES

Applications MAY not link directly to the OpenGL ES libraries [ref. 13 and ref. 14]. They MAY access OpenGL ES functionality through EFL's Evas GL APIs [ref. 42]. Applications can determine the availability of OpenGL ES APIs using the System Information API as described in section 2.7.2.

4.2. Application Packaging

Applications MUST follow the packaging guidelines, as defined for the platform [ref. 37 and ref. 38].

Web .wgt packages MUST NOT contain more than one UI application or the behavior is undefined.

4.3. Application Lifecycle

Native applications MUST be implemented with the Tizen Native API application lifecycle [ref. 21].

4.4. Namespace

Applications SHOULD include a namespace, such as: `<company>.<application>`. Applications MUST NOT overwrite the Tizen API namespaces.

4.5. Application Features and Privileges

A Tizen application MUST declare the features and privileges that it uses in the configuration document included in the application package [ref. 30, ref. 31, ref. 32, ref. 46, ref. 47 and ref. 48]. Further details on how to implement this requirement are provided in the developer documentation [ref. 32 and ref. 23].

The application SHALL be granted privileges only for the listed APIs. In some circumstances, user consent MAY be required before a privilege is granted. User consent may be requested at install time or at access time.

The Tizen Web API configuration document (*config.xml*) uses syntax as shown in these examples:

```
<feature name="http://tizen.org/feature/network.nfc"/>
<tizen:privilege name="http://tizen.org/privilege/application.launch"/>
```

The Tizen Native API configuration document (*tizen-manifest.xml*) uses syntax as shown in these examples:

```
  <Requirements>
     <Feature Name="http://tizen.org/feature/camera">true</Feature>
  </Requirements>
  <Privileges>
     <Privilege>http://tizen.org/privilege/notification</Privilege>
</Privileges>
```

4.6. Profile Declaration

A Tizen application MUST declare the Tizen profile it is capable of running on. If this declaration is omitted, application stores MAY not correctly select the application for installation. For the Tizen Wearable profile, the following declarations style is used.

In the Tizen Web API configuration document (*config.xml*):

```
  <widget xmlns="http://www.w3.org/ns/widgets"
    xmlns:tizen="http://tizen.org/ns/widgets" ...>
      <tizen:profile name="wearable"/>
```

In the Tizen Native API configuration document (*tizen-manifest.xml*):

```
    <manifest xmlns="http://tizen.org/ns/packages" api-version="2.3.1"...>
      <profile name="wearable">
```

4.7. Hardware Keys

A Tizen application MAY add listeners for the hardware back key event. If enabled in the application configuration, the event SHALL be delivered as a `tizenhwkey` event object with a “keyName” attribute and a value of “back".

Appendix A. Additional Information

This chapter contains tables of information providing further details for API aspects referenced elsewhere in this specification.

A.1. Sensor Hardware Capabilities

The following table details strongly recommended capabilities of sensors which have corresponding programming interfaces in Tizen.

Sensor Type	Required Capabilities
Accelerometer	Axis: 3 (x, y, z)
	Data range: -2G ~ 2G
	Minimum data rate: 50Hz
	Minimum resolution 0.1m/s^2
	Unit: G, 9.8m/s^2 = 1G
Gyroscope	Axis: 3 (x, y, z)
	Data range: -8.73 rad/s ~ 8.73 rad/s
	Minimum data rate: 50Hz
	Minimum resolution 0.01 rad / s
	Unit: rad/s, radians per second
Heart Rate Monitor	Provide the heart rate value
	Data range: 0 ~ 240 bpm
	Minimum data rate: 10HZ
	Minimum resolution 1bpm
	Unit: bpm (beats per minute)
Magnetometer	Axis: 3 (x, y, z) with azimuth, pitch, roll
	Data range: -1200 μT ~ 1200 μT
	Minimum data rate: 50Hz
	Minimum resolution 1μT
	Unit: μT, micro tesla
Proximity	Provide the lux value, can turn on/off
	Data range: 0 ~ 5 cm
	Minimum data rate: 10Hz
	Minimum resolution 1 cm
	Unit: cm

A.2. Native API Libraries

The following libraries are part of the Native Application Programming Interface and Application Binary Interface. Tizen Wearable device implementations MUST provide these libraries with the shared object name (soname) indicated, and compiled Native API applications MUST only be bound to these sonames, not to any other soname for the same library, as only these sonames are guaranteed to be present on Tizen Wearable device implementations.

Lib	Version	Soname	Purpose
EFL	1.7	libecore.so.1 libecore_con.so.1 libecore_evas.so.1 libecore_fb.so.1 libecore_file.so.1	EFL is the fundamental set of libraries underlying the Native API.

		libecore_imf.so.1 libecore_imf_evas.so.1 libecore_input.so.1 libecore_input_evas.so.1 libecore_ipc.so.1 libecore_x.so.1 libedbus.so.1 libedje.so.1 libeet.so.1 libefreet.so.1 libefreet_mime.so.1 libefreet_trash.so.1 libeina.so.1 libeio.so.1 libelementary.so.1 libembryo.so.1 libethumb.so.1 libethumb_client.so.1 libevas.so.1	
EXIF	0.6.21	libexif.so.12	Exif is an image file format used by camera and scanner devices (extends existing formats such a jpeg and tiff). Many Tizen devices have a camera and emit this format, libexif allows decoding.
Json-glib	0.10.4	libjson-glib-1.0.so.0	Json-glib is a library for serializing and deserializing JavaScript Object Notation (JSON) using Glib and Gobject data types.
Eglibc	2.13	libBrokenLocale.so.1 libanl.so.1 libcidn.so.1 libcrypt.so.1 libc.so.6 libdl.so.2 libm.so.6 libnsl.so.1 libnss_compat.so.2 libnss_dns.so.2 libnss_files.so.2 libnss_hesiod.so.2 libnss_nisplus.so.2 libnss_nis.so.2 libpthread.so.0 libresolv.so.2 librt.so.1 libthread_db.so.1 libutil.so.1	Standard C library, needs to be available to programs written in ISO C language
Glib	2.32.3	libglib-2.0.so.0 libgio-2.0.so.0 libgmodule-2.0.so.0 libgthread-2.0.so.0	Application building blocks which add data types and other programming facilities for C-language programs

		libgobject-2.0.so.0	
Curl	7.40	libcurl.so.4	A client-side URL transfer library supporting http, https, ftp, file URIs and many more protocols. Allows applications to perform URL related activities without having to involve a web browser.
XML2	2.7.8	libxml2.so.2	Library for parsing xml documents
Fontconfig	2.9.0	libfontconfig.so.1	Font-handling library to let applications find a font or a closely matching font
Freetype	2.5.5	libfreetype.so.6	Text-rendering library
Minizip	1.2.5	libminizip.so.1	Lightweight library building on top of zlib for processing files in the zip format
Sqlite	3.7.13	libsqlite3.so.0	Implements a lightweight sql database within a library, widely used for embedded client-local storage.
Cairo	1.12.14	libcairo.so.2	Library for 2-D vector graphics drawing
openssl	1.0.1p	libssl.so.1.0.0 libcrypto.so.1.0.0	Library implementation of secure sockets layer (ssl) and transport layer security (tls) to enable secure internet communications
OpenAL	1.13	libopenal.so.1	Audio API designed for efficient rendering of 3-D positional audio.
OpenMP	3.0	libgomp.so.1	API for shared-memory multiprocessing programming (C and C++), useful for complex tasks on multicore processors
C++ Standard Library	4.6.4	libstdc++.so.6	Standard C library, needs to be available programs written in ISO C++ language
libOAuth	0.9.4	liboauth.so.0	Functions implementing the OAuth Core RFC 5849 protocol (that is, OAuth 1.0)