Tizen® 6.0 Compliance Specification for IoT Profile

Version 0.97 beta

Revision History

Revision	Date	Author	Reason for Changes
2.3 version 1.0	11 Dec 2014	Tizen TSG	Official release
2.4 version 1.0	18 Jan 2016	Tizen TSG	Official release
3.0 version 1.0	31 Jan 2017	Tizen TSG	Official release
4.0 version 0.9	30 Dec 2017	Tizen TSG	Release candidate
5.0 version 0.9	31 Dec 2018	Tizen TSG	Initial edits
5.5 version 0.9	31 Dec 2019	Tizen TSG	Initial edits
6.0 version 0.97	17 Nov 2020	Tizen TSG	IoT platform specifications

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.
IOMMU	Input/Output Memory Management Unit.
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.
Mobile	Portable, connected devices, such as phones and tablets. A platform target for Tizen.
SDB	Smart Development Bridge, a device management tool in the Tizen SDK.
Side loading	Installing applications or components other than from a certified application installer package.
TV	Connected smart televisions and set-top boxes. A platform target for Tizen.
Profile Tizen Profile	The variant of the Tizen system dedicated to specific type of device, i.e. TV, Mobile, Wearable..
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.
Wearable	Miniature computing devices worn by the user on the body or clothing. A platform target for Tizen.
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 IoT 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. 1].

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 IoT 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 IoT devices, and know how Tizen devices will behave.
- IoT device implementers: know how to implement device hardware, security configurations, services, APIs, etc.

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 with 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 IoT 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 IoT 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, which is available at https://source.tizen.org. 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. [N] IETF RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels": http://www.ietf.org/rfc/rfc2119.txt
2. [I] Log View Reference: https://developer.tizen.org/development/tizen-studio/native-tools/debugging-your-app/log-view
3. [I] Smart Development Bridge: https://developer.tizen.org/development/tizen-studio/native-tools/running-and-testing-your-app/sdb
4. [I] Tizen Application Filtering: https://developer.tizen.org/development/getting-started/native-application/understanding-tizen-programming/application-filtering
5. [I] Application Controls for Tizen applications: https://developer.tizen.org/development/guides/native-application/application-management/application-controls
6. [I] Tizen Application Security and Privacy: https://developer.tizen.org/development/training/native-application/understanding-tizen-programming/security-and-api-privileges
7. [I] Tizen Application Package Model: https://developer.tizen.org/development/training/native-application/tizen-application-model
8. [N] Tizen API Privileges: https://www.tizen.org/privilege
9. [I] Tizen OS Recovery/Upgrade Guide: https://wiki.tizen.org/Tizen_OS_Recovery_/_Upgrade_Guide
10. [N] TizenFX API Level 6 Reference: https://samsung.github.io/TizenFX/API6/
11. [N] .Net Standard 2.0 API: https://docs.microsoft.com/en-us/dotnet/api/?view=netstandard-2.0
12. [N] Tizen Native API Reference for Headed IoT Profile: https://docs.tizen.org/iot/api/latest/tizen-iot-headed/
13. [N] Tizen Native API Reference for Headless IoT Profile: https://docs.tizen.org/iot/api/latest/tizen-iot-headless/

14. [N] Tizen IoT Extension SDK documentation: https://docs.tizen.org/iot/api/latest/things-sdk/index.html

2. IoT Profile Software Compliance

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

2.1. General Principles

IoT device implementations MUST include support for the Tizen Native API and MAY include support for the Tizen .NET API.

- The IoT device implementation MUST report the availability of optional hardware and software features (see section 2.6) as platform attributes.
- If an IoT device implementation reports that it supports a particular optional hardware or software feature, it MUST implement the entire corresponding API.
- Whether an IoT 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 report the lack of support by returning appropriate error codes for Native API as described 2.2.3 and by throwing an exception as described in the section 2.3.2 for .NET API.

2.2. Tizen Native API

2.2.1. Modules

IoT device implementations MUST provide all of the API modules listed in the Tizen Native API Reference for a Headed or Headless profile.

2.2.2. Tizen Native API Categories

- **Tizen Common Headed API**: is based on Native API for Tizen Common Profile. This API is targeted for IoT devices featuring display. [ref. 12]
- **Tizen Common Headless API**: is a cut down version of Headed IoT API. This API is targeted for IoT devices without a display. [ref. 13]
- **Things SDK API**: allows to integrate, control, and monitor IoT devices through the SmartThings Cloud with the SmartThings application. [ref. 14]

2.2.3. Behavior of Unsupported APIs

IoT device implementations MUST NOT omit any Native API listed in the Tizen Native API Reference for specific profile [ref. 12 and ref 13]. If some APIs are not supported on the Tizen devices by software or hardware limitations, they MUST return appropriate error code `[Uppercase module prefix]_ERROR_NOT_SUPPORTED` as specified in the Tizen Native API Reference.

2.3. Tizen .NET API

2.3.1. Namespace

IoT device implementations MUST provide all of the API namespaces and APIs listed in the TizenFX API Reference [ref. 10], including `Tizen.*`.

2.3.2. Tizen .NET API Categories

- **.NET Standard API 2.0**: implements the .NET Base Class library, allows you to use the well known C# language base class libraries and features (e.g. collections, threading, file I/O, and LINQ) as well as features like XML and JSON processing. [ref. 11]
- **TizenFX API**: allows you to access platform-specific features not covered by the generic .NET and Xamarin.Forms features, such as system information and status, battery status, sensor data, account, connectivity services.
  Some APIs may be indicated as “preview”. These APIs may change in their final official versions. They may be omitted from the scope of Tizen Compliance Tests.
- **Xamarin.Forms**: allows you to efficiently build a user interface from standard components in C# or XAML.
  Xamarin.Forms APIs may exist in application packages as commonly used for all Xamarin.Forms supported platforms.
  In this case, the providers of Xamarin.Forms NuGet packages for the Tizen devices MUST guarantee correct behavior of Xamarin.Forms APIs on compatible Tizen platform version and may be omitted from the scope of Tizen Compliance Tests.

2.3.3. Behavior of Unsupported APIs

IoT device implementations MUST NOT omit any .NET API listed in the TizenFX API. Specially, .NET APIs are designed to be common to all Tizen profiles. If some APIs are not supported on the Tizen devices by software or hardware limitations, they MUST throw appropriate `System.NotSupportedException` exception, as specified in the TizenFX specification.

2.4. Application Control

The application control interface in the Tizen .NET 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 IoT 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. 5].

2.5. Platform Attributes

IoT device implementations MUST provide accurate platform attributes via the appropriate interfaces in the .NET API for System Information.

Platform attributes include but are not limited to the following:

- Device capabilities (see section 2.6)
- Information about data storage devices
- Display information
- Information about the device orientation
- Locale information
- Network information

2.6. 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 IoT device requirement such as higher processing power or memory requirements (See section 3.1 for minimum hardware requirements).

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

2.6.1. Tizen Native API

See section 2.2.3. If an IoT device implementation does not include the corresponding hardware or software feature, the APIs MUST return `[Uppercase module prefix]_ERROR_NOT_SUPPORTED` error code and the IoT device implementation MUST accurately report the availability of these underlying features through the Tizen Native API System Information API.

2.6.2. Tizen .NET API

See section 2.3.3. If an IoT device implementation does not include the corresponding hardware or software feature, the APIs MUST throw `System.NotSupportedException` exception and the IoT device implementation MUST accurately report the availability of these underlying features through the Tizen .NET API System Information API.

2.7. Privilege

Certain APIs have access to privacy-sensitive information (for example the Location API can be used to track user location) or have security or stability implications. If an application uses such APIs, then corresponding privileges MUST be declared in the application's `config.xml` file or `tizen-manifest.xml` file.

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.

2.7.1. Tizen Native API

If a Native application does not declare a required privilege in the `tizen-manifest.xml` file, the corresponding API MUST return following error code `[Uppercase module prefix]_ERROR_PERMISSION_DENIED`.

2.7.2. Tizen .NET API

If a .NET application does not declare a required privilege in the `tizen-manifest.xml` file, access to the corresponding API MUST return errors according to TizenFX specification (usually `System.UnauthorizedAccessException`).

2.8. Application Packaging Compatibility

Tizen defines a mandatory application packaging format. IoT device implementations MUST correctly process packages in this format. They MUST NOT extend the packaging format in a way that would prevent packages generated on the implementation from running on other conforming IoT device implementations.

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

IoT device implementations MUST be able to install, remove, list, and update Native and .NET application packages in the .tpk format.

2.9. Chromium

The WebView implementation on Headed IoT device implementations SHOULD be based on Chromium version 47 or higher. This is strongly recommended for maintaining application compatibility across Tizen IoT devices. Any customizations made by device implementations SHOULD NOT alter the original web exposed behavior from the Chromium version used.

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

Mozilla/5.0 (`DEVICE TYPE`; Linux; Tizen `PLATFORM VER`; `MODEL`) AppleWebKit/`537.36` (KHTML, like Gecko) `APP NAME`/`APP VER` Chrome/`47.0.2526.69` IoT Safari/`537.36`

- The value of the `DEVICE TYPE` string SHOULD be the same as the type of the device.
- The value of the `PLATFORM VER` string MUST be "6.0".
- 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 `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.
- IoT device implementations MAY omit the word “IoT” from the user agent string.

2.10. .NET Runtime

IoT device implementations MUST support all mandatory requirements of .NET Core Runtime version 2.0 or newer.

2.11. Keys

Hardware keys are not mandatory for an IoT device. The Headed IoT implementations MUST embed Hardware keys or Softkey container that supports following events:

- **Task Manager** – opens task manager allowing to switch to or close opened applications.
- **Home** – switches to the start screen.
- **Back** – used to navigate to previous view in the applications. Sends a ‘`tizenhwkey`’ event with `keyName == “back”`.

The implementation MUST deliver **Back** event to a listening application.

2.12. 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 except for the case of external storage such as USB mass storage.
- Secure execution environment:
    - There SHOULD NOT be any set-user-ID binaries in the device.
- Smack supported modules:
    - The device SHOULD contain coreutils or equivalent, d-bus, and udev 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.

2.13. Multimedia

The following tables list media formats/codecs for TV device implementations and whether they are REQUIRED or OPTIONAL. 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	Requirement
Audio Codec (Decoder)	AAC LC	REQUIRED
	HE-AAC	OPTIONAL
	AC3	REQUIRED
	EAC3	OPTIONAL
	MPEG	REQUIRED
	DTS (Core)	OPTIONAL
	WMA v7/v8/v9 (Standard v1/v2/v3)	OPTIONAL
	Vorbis	REQUIRED
	PCM (raw PCM)	REQUIRED

Video Codec (Decoder)	MPEG1	OPTIONAL
	MPEG2 (SP/MP)	REQUIRED
	MPEG4 (SP/ASP)	REQUIRED
	H.264	REQUIRED
	Divx 3.11/4/5/6	OPTIONAL
	VP6/8	OPTIONAL
	Theora	OPTIONAL
Image Decoder (Decoder)	JPEG	REQUIRED
	BMP	OPTIONAL
	PNG	OPTIONAL

Type	File Type / Container format	Requirement
Audio	MPEG (*.mp3)	REQUIRED
	MPEG4 (*.m4a, *.mpa)	REQUIRED
	OGG (*.ogg)	REQUIRED
	WMA (*.wma)	OPTIONAL
	WAV (.wav)	OPTIONAL
Video	AVI (*.avi, *.divx)	REQUIRED
	MKV (*.mkv)	REQUIRED
	MP4 (*.mp4)	REQUIRED
	PS (*.mpg. *.mpeg)	OPTIONAL
	TS (*.ts, *.tp, *.trp, *.m2ts, *.mts)	REQUIRED
Image	JPEG (*.jpeg, *.jpg)	REQUIRED
	PNG (*.png)	OPTIONAL
	BMP (*.bmp)	OPTIONAL

2.14. Developer Tools

IoT 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. 3] 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. 2] function to retrieve the Tizen platform log (dlog).

In addition, if the implementation supports SDB it MUST support either USB 2.0 or later or another data networking technology listed in section 3.1.3.

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.

2.15. Software Update

IoT device implementations SHOULD provide a mechanism for updating system software. If provided, user data, application private data, and application shared data SHOULD be preserved. Upgrade process is described in detail in [Ref. 9].

2.16. Tizen Compliance Tests

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

2.16.1. Satisfying TCT preconditions

IoT device implementations MUST satisfy preconditions to pass TCT.
The list of TCT preconditions that MUST be satisfied is available in the Tizen 6.0 TCT for IoT Profile.

3. IoT 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 IoT device implementation and any corresponding API must be fully implemented.

3.1.1. Memory Storage

A Tizen IoT device MUST have at least 512MB of RAM to run applications smoothly.

IoT device implementations MUST have at least 1 GB of internal storage.

IoT device implementations SHOULD allow an external device to access files in the shared media folder on the device. The precise method is unspecified.

3.1.2. Sound

IoT device implementations MAY support audio output.

3.1.3. Connectivity / Networking

IoT device implementations MUST support at least one form of data networking capable of accessing the Internet. Examples of acceptable data networking technologies include Wi-Fi, Ethernet, etc. Implementations MAY omit any individual mechanism, as long as at least one method is supported.

3.1.4. Display

A IoT device implementation, whether it is an integrated device with display or a controller-type device without its own display, MUST support driving a display with a minimum screen resolution of 1280x720 (HD). It is strongly recommended to use a display resolution of 1280x720 (HD) for an IoT Headed device implementation.

IoT device implementations SHOULD support a 32-bit frame buffer.

3.1.5. Input Devices

IoT device implementations MUST provide applications a means of receiving keyboard input from users.

- Implementations MAY omit a full hardware keyboard.
- If no hardware keyboard is available, a soft keyboard MUST be provided.
- A soft keyboard or an input method setup MUST 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.

3.2. Optional Hardware Requirements

If an IoT device implementation reports that it includes an optional hardware component that has a corresponding optional API, the implementation MUST fully implement that API, as described in this specification. IoT 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

An IoT device implementation MAY omit camera devices.
If an IoT 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 format
UYVY	The UYVY pixel format
H.264	MPEG-4 AVC video compression format

3.2.2. HDMI Input

An IoT device implementation MAY omit HDMI input hardware.

3.2.3. Wi-Fi

An IoT 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.

3.2.4. Audio Input Devices

An IoT device implementation MAY omit a microphone. IoT device implementations MUST accurately report the presence or absence of a microphone.

3.2.5. USB

IoT device implementations SHOULD provide USB client functionality.

IoT device implementations MUST accurately report the presence or absence of USB client functionality.

3.2.6. Graphics

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

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

3.2.7. Bluetooth

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

4. IoT 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

.NET applications MUST use only the APIs defined in the Tizen .NET API specifications when making calls external to the application. Additional libraries included in the application are considered internal to the application. The Tizen .NET API include API allowing access to various features such as application common, network, sytem, etc.

4.2. Application Packaging

Tizen Native and .NET applications MUST be packaged into a .tpk file format. See section 2.8.

4.3. Namespace

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

4.4. 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. 7]. Further details on how to implement this requirement are provided in the developer documentation [ref. 4 and ref. 8].

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.

In application manifest file (`tizen-manifest.xml`):

```
<feature name="http://tizen.org/feature/iot.ocf">true</feature>
<privileges>
  <privilege name="http://tizen.org/privilege/filesystem.write"/>
</privileges>
```

4.5. 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 IoT, the following declaration style is used.

In application manifest file (`tizen-manifest.xml`):

```
<manifest xmlns="http://tizen.org/ns/packages" api version="6" ...>
    <profile name="" />
```

For the Headed IoT profile name is `iot-headed`.
For Headless IoT profile name is `iot-headless`.

4.6. Tizen Native UI Framework

A Tizen IoT application MUST use one of the following Native UI frameworks:

- EFL
  The EFL framework is a part of Native API for Headed IoT Profile. This set of libraries are provided by the system and not included in an application package.
- DALi
  Dynamic Animation Library is a part of Native API for Headed IoT Profile. This set of libraries are provided by the system and not included in an application package.

4.7. Tizen .NET UI Framework

A Tizen IoT application MUST use one of the following .NET UI frameworks:

- Xamarin.Forms
  To ensure Xamarin.Forms support, system MUST be compatible with Xamarin.Forms libraries version 4.3 or newer built for `tizen40` .NET target framework. The libraries are usually included in an application package.
- Tizen.NUI
  The Tizen.NUI framework is a part of the TizenFX API, and as such is subject of the requirements described in the section 2.3 Tizen .NET API. This set of libraries are provided by the system and not included in an application package.