An Overview of Tizen Application Core Framework

Youngjoo Park
Contents

• Introduction
• Architecture
• Application Core
• Application Utility Library
• Application Service
• Package Manager
• Application Information Library
• Application Use History
Introduction

• **Application core framework** provides infrastructure in which each application works seamlessly in the platform.
  – Application life cycle management
  – Application launch service
  – System event handlers
  – Application launch history
  – Application configuration data management
  – Application installation / uninstallation
Framework Overview

Tizen Application (Application / Runtime)

Application core Framework

- AUL
- Launch Pad (AUL daemon)
- VCONF
- AppCore
- RUA
- Configuration DB
- RUA DB

*RUA : recently used application
*AUL : application utility library

Database
Daemon
Library
Application Core: Appcore

• Appcore is the application core handling various important events each of which application should be aware of.

• Using appcore, developers can
  – Manage application life-cycle
    • Create, Reset, Pause, Resume, Terminate
  – Handle System Events
    • Low Memory
    • Low Battery
    • Screen orientation Change
    • Language & Region Change
### Appcore Internal Callbacks

**Operation** | **Description**
--- | ---
**CREATE** | Called once before the main loop. Initialize the application such as window creation, data structure allocation, and etc.
**RESET** | Called at the launching and every re-launch request.
**PAUSE** | Called when the window of the application becomes invisible. Recommend to suspend the application’s tasks.
**RESUME** | Called when the window of application becomes visible again. Resume the paused actions.
**TERMINATE** | Called once after the main loop. Release the resources.

---

**Diagram:**
- App. launch → Create
- Main loop → Reset → Pause → Resume → Terminate
- App. terminated
Application Life Cycle: States and Transitions

<table>
<thead>
<tr>
<th>States</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>READY</td>
<td>The application is launched.</td>
</tr>
<tr>
<td>CREATED</td>
<td>The application initialized its state.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>The application is running in the foreground and is receiving input events.</td>
</tr>
<tr>
<td>PAUSED</td>
<td>The application is in the background.</td>
</tr>
<tr>
<td>TERMINATED</td>
<td>The application was terminated.</td>
</tr>
</tbody>
</table>
Application Overall Structure

- App Utility Library
- Window Manager
- Sensor Framework
- System Server

Appcore
- Internal IPC
- X Client Message
- Sensor
- Vconf
- Default Behavior

Main loop

Launch
- Create
- Reset
- Pause
- Resume
- Screen Orientation
- Low Memory
- Low Battery
- Lang Change
- Terminate

Terminated

Appcore operation

System Event
How to launch an application?
Application Utility Library: AUL

- **Application Utility Library (AUL) provides the following features**
  - Launching/terminating applications
  - Providing running application information

- **AUL consists of the following sub-components:**
  - AUL library: Sending/receiving requests for launching and terminating
  - AUL daemon (a.k.a. launch pad): Handling the requests
AUL Overall Architecture

Launch, Resume, Terminate request

Result if necessary

Launch / Reset, Resume, Terminate

Fetch default configuration with APN

AUL daemon

App A

AUL

App B

AUL

Application Information DB

/proc filesystem
Application Data Exchange

• The actual Application Data Exchange (ADE) occurs as an argument between the caller and callee, using a bundle.
• Bundle is a type of dictionary abstract data, in which information is stored as key-value pairs.
• Bundle contains information regarding the state the app should prepare.
AUL: Launch

In case of single-instance application
- If app is not running, launch application
- If app is already running, send reset event to the running application

In case of multi-instance application
- launch an app

![Diagram of Application Life Cycle]

- Launch request
- Launch PAD

**AppCore**
- Create
- Reset
- Resume
- Terminate

**AUL handler**
- Reset
- Resume
- Terminate

**Event loop**
- Create
- Reset
- Resume
- Pause
- Terminate

**App. launch**
- App. terminated
Putting All Together

Menu Screen

1. User request received for new app launch

2. aul_launch_app (pkname, bundle);

3. create

Launch PAD

App Utility library

Window manager

Sensor framework

System manager

Legend

Process  operation  Module

flow  event

Web Application

main(argc, argv)

4. Register user-defined callback functions for lifecycle transition

appcore_efl_main(...)

5. Call ‘reset’ callback
   (check if this launch is for service)

6. - WRT initialization
   - Creating new webview
   - Rendering it
   - Find the service page URL
   - Removing existing webview
   - Rendering service page

7. start event loop

Event Loop

- Call "reset" callback
- Call "pause" callback
- Call "resume" callback
- Call "terminate" callback
- Call system event callback (e.g. language change)

Clear resources (callbacks)

- Send invisibility event to page
- Send visibility event to page
- Handle each system events
- Remove existing webview
- WRT deinitialization

Rendering

- WRT initialization
- Creating new webview
- Rendering it

Rendering

- WRT deinitialization
- Handling each system events
- Removing existing webview
- WRT deinitialization

App Utility library

Application Name

This is an empty template of Tizen Web Application. Users will support multiple device categories:

- smartphones, tablets and smart TVs
- automobiles, in-vehicle infotainment systems

This is a basic section of a document.

The following button displays a timer using JavaScript.

WRT initialization
Creating new webview
Rendering it

Putting All Together

Put All Together

Event Loop
High Level Application Launch Service: App Service
App Service

- App service exposes general service terms, such as view, create, call, and so forth, to developers in case of launching an application with a specific feature
  - More desirable asking image view service without knowing what image viewer apps are available.
- Each service can be determined by given operation, URI and MIME type.
  - Operation: expected action for the request (e.g., view, edit, call, send..)
  - URI: URI information for requested operation (e.g., http://... file://...)
  - MIME type: MIME type information for requested operation (e.g., image/jpeg)
  - Data: Extra data to launched service application (contained in a bundle packet)
App Service Overall Flow
App Service Request

Op: http://tizen.org/appsvc/operation/send
Uri: file:///opt/media/Downloads/test.jpg
(MIME: image/jpg)

Check corresponding apps

Application Information DB

App A
App B
http://tizen.org/appsvc/operation/view
Mime: image/jpeg
http://tizen.org/appsvc/operation/send
Mime: image/jpeg

App C
http://tizen.org/appsvc/operation/view
Uri_scheme: http
http://tizen.org/appsvc/operation/search

App D
http://tizen.org/appsvc/operation/send
Mime: image/jpeg
http://tizen.org/appsvc/operation/pick
Mime: image/jpeg
## Operation List of App service

<table>
<thead>
<tr>
<th>Standard Operation Type (Followed Web Intents Format)</th>
<th>Related Data Key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://tizen.org/appsvc/operation/default">http://tizen.org/appsvc/operation/default</a></td>
<td>N/A</td>
<td>Do default action of each application. It is used to launch application explicitly</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/edit">http://tizen.org/appsvc/operation/edit</a></td>
<td>N/A</td>
<td>Provide explicit editable access to the give data. URI should contain the path to edit</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/view">http://tizen.org/appsvc/operation/view</a></td>
<td>N/A</td>
<td>Display the specific data. Proper action should be performed by requested data. URI should contain the path to perform</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/pick">http://tizen.org/appsvc/operation/pick</a></td>
<td><a href="http://tizen.org/appsvc/data/selected">http://tizen.org/appsvc/data/selected</a></td>
<td>Pick items and return the path what was selected. The key of return value should be set to &quot;<a href="http://tizen.org/appsvc/data/selected">http://tizen.org/appsvc/data/selected</a>&quot;</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/create_content">http://tizen.org/appsvc/operation/create_content</a></td>
<td><a href="http://tizen.org/appsvc/data/selected">http://tizen.org/appsvc/data/selected</a></td>
<td>Create content and return the path what was created. The key of return value should be set to &quot;<a href="http://tizen.org/appsvc/data/selected">http://tizen.org/appsvc/data/selected</a>&quot;</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/call">http://tizen.org/appsvc/operation/call</a></td>
<td>N/A</td>
<td>Make a phone call to a specific phone number. URI should composite with &quot;tel:&quot; scheme (spec RFC 3966)</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/dial">http://tizen.org/appsvc/operation/dial</a></td>
<td>N/A</td>
<td>Dial a number as a specific phone number. Dialer UI should be shown with given specific phone number. URI should composite with &quot;tel:&quot; scheme (spec RFC 3966) or set to NULL. If URI is NULL then dialer UI except phone number should be shown</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/send">http://tizen.org/appsvc/operation/send</a></td>
<td>N/A</td>
<td>Deliver some data to someone else. URI should contain the path to deliver</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/send_text">http://tizen.org/appsvc/operation/send_text</a></td>
<td><a href="http://tizen.org/appsvc/data/text">http://tizen.org/appsvc/data/text</a></td>
<td>Deliver some text data to someone else</td>
</tr>
<tr>
<td><a href="http://tizen.org/appsvc/operation/search">http://tizen.org/appsvc/operation/search</a></td>
<td><a href="http://tizen.org/appsvc/data/keyword">http://tizen.org/appsvc/data/keyword</a></td>
<td>Perform a search. The searching keywords should contain to data with key &quot;<a href="http://tizen.org/appsvc/data/keyword">http://tizen.org/appsvc/data/keyword</a>&quot;</td>
</tr>
</tbody>
</table>
Application Core Framework

Application Installation
Package Manager

• Package manager is responsible for installing, upgrading and uninstalling of applications and storing their information.

• Expandable structure to support various types of applications
  – Designated installation modules can be added to the manager
    • Web app, native app, java app, and so forth
Package Manager

Installation flow

Application (e.g. Market)

Call API

Pkg-Mgr Client Library

Send msg

Dbus daemon

Get cookie

Auto activation

Package Manager Server

Launch

Corresponding backend

Check privilege

Security Server

Installer backends

native
Java
WRT

Parse desktop file & save to App Info DB

Install

Status broadcast by DBus signal

Storage

desktop

App Info DB
Application Core Framework

Application Information Management
Application Information Library: AIL

- AIL is the library providing functionalities to execute application information-related tasks.
- AIL provides the following features
  - Adding, updating and removing application information.
  - Managing application’s information – application name, type, icon path, exec path, etc.
  - Retrieving an application list which meets a given filter.
AIL: Adding Application Information

1. Save package
2. Install
3. Call Backend installer
4. Install
5. Read desktop file (using inotify)
6. Parse desktop file & add app info to AIL DB
Application Use History

• Recently Used Application (RUA) logs application use history.
• When an application is launched, launch pad updates the history and task manager can either get or clear the history.
• RUA also provides below information regarding launching:
  – package name
  – launch time
  – application path
  – application launch argument
Questions?