Application Framework

1. Overview

The definition of Application framework can be difficult to agree upon. For the moment we start with a Wikipedia definition:

“Application framework consists of a software framework used by software developers to implement the standard structure of an application”

Where software framework is defined like this:

“A software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code,thus providing application-specific software. […] Software frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution”

The GENIVI Reference Architecture works with two different approaches to developing applications, namely the **Managed** and the **Native** application type.

Most people associate Application Framework mostly with the solution for the Managed application category, which includes a well-defined and limited set of application APIs and usually a constrained (“sandboxed”) execution environment.

Refer latest "Reference Architecture" document at the below link for more information on Applications FW , Native and Managed Applications, Application Life cycle

Reference Architecture (access requires Genivi Member Login)

1.1 App FW Scope and Concept

In the following review comments can be added to the App FW scope and concept document.

App FW Scope and Concept - Review

1.2 Application Manager

The Application Manager component is a part of the Application Framework set of components, aimed at supporting Applications, "Apps", in a GENIVI setting.

An Application manager component manages the overall responsibilities of the application framework infrastructure. This may include launching applications, restarting applications upon failure or when updated, controlling the privileges associated with the applications, keeping track of application states (in focus / background, speech context and access to audio).

Below is the comparison study of API's belonging to **Application Manager** in **Tizen** and **Apertis**.

Sl No.	Tizen Web API	Description	Tizen Native	Description	Apertis (formerly eCORE)	Description	Qt Automotive Suite Application Manager	Description	Comments / Remarks
1	getCurrentApplication	Gets the Application object defining the current application.			CurrentActiveApp	Property update Get the current active application on top of the application stack		Defined by the System UI, can support multiple active applications at once.	
2	kill	Kills an application with the specified application context ID.			Apps are killed based on their states by app manager.	Apps are killed based on their activity state. There is no exposed API using which system UI or managed app can request another app to be killed.	ApplicationManager:: stopApplication(string id, bool forceKill).	Only available to System UI and via System DBus	
3	launch	Launches an application with the given application ID.			LaunchNewApp ()	Launch an application from any other application	ApplicationManager:: startApplication ApplicationManager::openUrl	Only available to System UI and via System DBus. Apps themselves can only launch other apps through mime-types.	
4	launchAppControl	Launches an application with the specified application control.			OpenURI()	This method is used to launch an application which handles the MIME type of the arguments.	For apps: [http://doc.qt.io/qt-5/qml-qtqml-qt.html#openUrlExternally-method] For System UI: ApplicationManager:: startApplication	Applications uses standard Qt interfaces, i. e. they are independent of Application Manager.	

5	findAppControl	Finds application information can be launched with the given application control.			AppLaunchDatabase	Returns a list of apps and the MIMEs that they can handle. NOTE: the name will be corrected.	The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.	Only available to System UI and via System DBus.	
6	getAppsContext	Gets a list of application contexts for applications that are currently running on a device.				Not clear about the API		Not sure what a context is.	
7	getAppContext	Gets the application context for the specified application context ID.	app_manager_get_app_context	Gets the application context for the given ID of the application.		Not clear about the API		Not sure what a context is.	
8	getAppsInfo	Gets the list of installed application's information on a device.			Not yet implemented	Assuming that this API should provide a complete list of apps that are installed on the device. NOTE: there are API for system UI like launcher to get this info. But not in general for managed apps.	The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.	Only available to System UI and via System DBus.	
9	getAppInfo	Gets application information for a specified application ID.			GetApplicationInfo	Returns manifest info to the calling process	The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.	Only available to System UI and via System DBus.	
10	getAppSharedURI	Gets URI of read-only shared directory of application for a specified application ID.				Not yet implemented			
11	getAppMetaData	Gets application meta data array for a specified application ID.				Not clear if this is different from GetAppInfo	The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.	Only available to System UI and via System DBus.	
12	addAppInfoEventListener	Adds a listener for receiving any notification for changes in the list of the installed applications on a device.				The application entries use the freedesktop desktop entry format and method. [https://developer.gnome.org/desktop-entry-spec/]	Available through the object returned from ApplicationManager::get.	Only available to System UI and via System DBus.	
13	removeAppInfoEvent Listener	Removes the listener to stop receiving notifications for changes on the list of installed applications on a device.				As above	Yes, through Qt signals/slots.	Only available to System UI and via System DBus.	
14					GetGlobalSearchApps	Global search is distributed over all apps. This API gives a list of apps that are supporting global search NOTE: This is a product specific feature	This would be implemented as a tag in the manifest, e.g. another capability.	Global search is not a part of the Application Manager but would have to be supported through some other service.	

15					RegisterMyApp	All applications register to application manager The registration is needed as app manager is a dbus service.	The ApplicationManager itself is not tied to a specific appstore implementation - as long as the package can be downloaded via HTTP, HTTPS or FTP; it can also be provided as a local file or via a UNIX socket connection. QtAS comes with a PoC appstore implementation (server side: django, client side: QML). See ApplicationInstaller singleton (system-UI and system DBus)	Not sure if this refers installation or during run-time. Installation is initiated through an Application Manager interface, so it works automatically. Apps are launched through Application Manager, so again, it works automatically.	
16					SetUninstalledApplication	DEPRECATED API ! AppStore :Set the application manifest name which isuninstalled.		Application Manager handles uninstall, no need to call.	
17					InsertNewEntry	DEPRECATED! AppStore : Launcher displays categories of applications and the list of applications in each category.		Application Manager handles install, no need to call. AppStore is integrated through a custom plugin, so anything apart from the reference store needs to be integrated by writing a C++ plugin.	
18					SetInstalledApplicationManifest	DEPRECATED API ! AppStore :Set the manifest file of the installed application.		Application Manager handles install, no need to call.	
20									
21					"signal: AppLaunchDatabaseUpdate "	This signal indicates an update of the app database		Application Manager owns the database. No need to call. System UI is notified through the QAbstractItemModel interface of the ApplicationManager singleton object.	
24			app_manager_foreach_app_context	Retrieves all application contexts of running applications.				Not sure what a context is.	
25			app_manager_foreach_app_info	Retrieves all installed applications information.			The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.		
26			int app_manager_get_app_id	Gets the ID of the application for the given process ID.			The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.		
27			app_manager_get_external_shared_data_path	Gets the absolute path to the shared data directory of the application specified with an application ID.			ApplicationInstaller:: getInstallationLocation	There's no real "shared" directory, but the path to the app's private data directory is available. Only available to System UI and via System DBus.	
28			app_manager_get_shared_resource_path				ApplicationInstaller:: getInstallationLocation	There's no real "shared" directory, but the path to the app's installation directory is available. Only available to System UI and via System DBus.	
29			int app_manager_is_running	Checks whether the application with the given package name is running.			The ApplicationManager singleton presents a model with the metadata regarding all installed apps and information regarding if the application is running or not.		

30			int app_manager_resume_app	Resumes the application.			ApplicationManager::startApplication ApplicationManager::openUrl	The "docs" just state "resumes app". If this means bringing a background app to the foreground, then the ApplicationManager will take care of that transparently.	
31			Many API's wrt Application context	https://developer.tizen.org/dev-guide/2.3.0/org.tizen.mobile.native.apireference/group_CAPIAPP_CONTEXT__MODULE.html					
32			Many Api's wrt Application information	https://developer.tizen.org/dev-guide/2.3.0/org.tizen.mobile.native.apireference/group_CAPI_APP_INFO_MODULE.html					

1.3 App Manifest

Manifest contains App meta data. some of the information will help to determine is this App compatible/installable on a particular device.

It contains App info, permission, license, dependencies, services that are mandatory for the proper operation of the App, mime types , version and so on.

App manifest comparative study of Tizen and Apertis.

Sl No	Apertis (formerly eCORE)	Description	Tizen	Description	Comments / Remarks
1	app-name	Audio-Player'	ui-application appid		
2	working-directory	/usr/Applications/AudioPlayer/'			
3	exec-path	/usr/Applications/AudioPlayer/bin/mrs_audio_player'	ui-application exec	Application executable file path.	
4	background-state	killed'			
5	exec-type	application'	ui-application	service app/ui app	
6	category	M U S I C'	ui-application		
7	category-icon	/icon_music_AC.png'	ui-application		
8	application-entry-names	'A R T I S T S','A L B U M S','S O N G S'			
9	application-entry-icons	'file:///usr/Applications/Launcher/share/icon_music_artists_AC.png','file:///usr/Applications/Launcher/share/icon_music_albums_AC.png','file:///usr/Applications/Launcher/share/icon_music_songs_AC.png'			
10	tile-thumbnails	'### UNKNOWN ###','### UNKNOWN ###','### UNKNOWN ###'			
11	exec-args	('app-name','Audio-Player'),('menu-entry','A R T I S T S'),('url',' ')			
12	env-key-value-pair	'key1','value1'	metadata		
13	mime-type	[]	app-control	mime type	
14	mime-list	[]			
15	audio-channel-name	mrs_audio_service'			
16	audio-resource-owner	Audio-Agent-Service'			
17	audio-resource-type	music'			
18			datacontrol access		
19			account	account provider, icon , lang, capability,	
20			Previleges		

21			Feature		
22			ui-application		
23			multiple		
24			nodisplay		
25			taskmanager		
26			type		
27			auto-restart		
28			on-boot		