http://dx.doi.org/10.3745/KTSDE.2013.2.11.779

A Reference Architecture and Manifest Standard Suggestions for Interworking Open Web Store

Taejun Ryu† · Changjun Kim† · Jonghong Jeon†† · Seungyoon Lee††† · Sangwon Park††††

ABSTRACT

With a wide dissemination of smartphones, the number of native applications developed and sold freely by anyone is growing now. The application market activated by Apple's App Store is spreading more rapidly with Google's Google Play. But due to platform-dependent of native application's attribute, developers are programming at each platform. As a result, development cost is increasing compared to earnings. To solve a dependency problem, people focused on web application developed by web-based language. However, stores at each browser are requiring a web application to follow manifest format. And this causes browser-dependent problem. Those problems can be solved by installing a certain browser, but this can make an application useless on the other browser of a store. Dependency problem can narrow not only user's application variation, but also concentration on some specific store. OWS(Open Web Store) is a standard store that supports various web environments. It overcomes browser or platform dependency problems by interworking applications between stores. Also customers are able to choose a large number of applications. In this paper, related to OWS, I would like to suggest manifest standards and store's reference architecture. An interworking scenario is going to be proposed as well.

Keywords : Web Application, Web Application Store, Standard Store, Web Application Manifest

OWS(Open Web Store) 연동을 위한 참조 모델 및 Manifest 표준 제안

류 태 준† · 김 창 준† · 전 종 홍†† · 이 승 윤††† · 박 상 원††††

요 약

스마트폰 보급화와 함께 누구나 자유롭게 애플리케이션을 개발하여 판매할 수 있는 네이티브 애플리케이션 시장이 성장하고 있다. 애플의 앱스토어를 시작으로 활발해진 애플리케이션 시장은 구글이 구글플레이를 개방하면서 더욱 빠르게 확대되고 있다. 하지만 네이티브 애플리케이션의 플랫폼 종속적인 면으로 인해 개발자들은 각 플랫폼에 맞게 개발해야함에 따라 수익 대비 개발비용이 증가하는 현상이 발생하고 있다. 네이티브 애플리케이션의 종속적인 문제를 해결하기 위해 차세대 웹 기반 언어로 개발된 웹 애플리케이션이 주목을 받고 있지만 각 브라우저의 스토어가 요구하는 매니페스트(Manifest) 형식에 따라 웹 애플리케이션을 제작하기 때문에 브라우저 종속적인 문제를 가진다. 이러한 웹 애플리케이션의 문제는 웹 애플리케이션을 사용하기 위해서 반드시 해당 브라우저를 설치해야만 하며, 타 브라우저의 스토어에 있는 애플리케이션은 사용하지 못하는 현상을 발생시킨다. 종속적인 문제는 사용자의 애플리케이션 선택 범위를 좁히고, 일부 스토어의 편중 현상을 불러올 수 있다. 개방형 웹 애플리케이션 스토어(OWS, Open Web Store)는 다양한 환경을 지원하는 표준 스토어로서, 스토어간 애플리케이션 연동을 통해 플랫폼이나 브라우저에 종속적인 문제점을 해결하고, 소비자의 폭 넓은 선택을 가능하게 한다. 본 논문에서는 OWS와 관련하여 연동을 위한 애플리케이션 Manifest 표준과 스토어의 표준 구조를 제안하며, 이를 기반으로 한 연동 시나리오를 제안한다.

키워드 : 웹 애플리케이션, 웹 애플리케이션 스토어, 표준 스토어, 웹 애플리케이션 Manifest

1. 서 론

최근 스마트폰 보급화와 함께 구글의 구글플레이(Google Play)나 애플의 앱스토어(App Store)를 비롯하여 여러 애플리케이션 스토어들이 등장하기 시작했다. 누구나 자유롭게 애플리케이션을 개발하여 판매할 수 있는 구조로 이루어진 애플리케이션 스토어의 특성 때문에 많은 개발자가 애플리

※ 본 연구는 방통통신위원회의 지원을 받는 방송통신표준개발지원사업의 연구 결과로 수행함.
† 준 회 원 : 한국외국어대학교 정보통신공학과 학사과정
†† 종신회원 : ETRI 표준연구센터 서비스융합표준연구팀 책임연구원
††† 정 회 원 : ETRI 표준연구센터 서비스융합표준연구 팀장
†††† 종신회원 : 한국외국어대학교 정보통신공학과 교수
논문접수 : 2013년 8월 9일
수 정 일 : 1차 2013년 9월 14일
심사완료 : 2013년 9월 27일
* Corresponding Author : Taejun Ryu(tjryu@hufs.ac.kr)

케이션 개발에 참여하였고, 이에 따라 네이티브 애플리케이션 시장이 성장하게 되었다. 하지만 약 30개의 스토어가 운영 중인 현재, 네이티브 애플리케이션 기반의 스토어 시장은 포화상태에 이르고 있고, 구글의 안드로이드와 애플의 iOS 등 특정 플랫폼에 대한 편중 현상이 심화되고 있다[1]. 특정 플랫폼의 모바일 시장 독점과 플랫폼 종속적인 문제를 해결하기 위해 모든 모바일 플랫폼에서 동작할 수 있는 크로스 플랫폼의 필요성이 부각되었고, 이에 따라 SKT와 KT, AT&T 등 전 세계 24개의 주요 통신사들과 삼성전자, LG전자, Sony Ericsson 등의 제조업체가 모여 WAC (Wholesale App Community)[2]를 결성하였다.

플랫폼에 맞춰 개발자가 애플리케이션을 개발해야 하는 모바일 플랫폼의 현실적인 문제점 속에서 차세대 웹 기반 언어(HTML5[3], CSS3, JavaScript)로 개발된 웹 애플리케이션이 주목을 받고 있다. 웹 애플리케이션은 특정 OS나 플랫폼에 종속을 받지 않는 크로스 플랫폼 특징을 가지는데[4], 현재 대표적으로 구글의 크롬 웹 스토어[5]와 파이어폭스의 마켓플레이스[6]이 있다.

하지만 웹 애플리케이션 또한 특정 브라우저에 종속적인 특징을 가진다. 웹 기반에서 구동되는 웹 애플리케이션은 해당 브라우저의 이용자에 한해서만 애플리케이션 이용이 가능한데 애플리케이션을 제작할 때 기술하는 매니페스트(Manifest)에 대해서 각 스토어의 형식과 요구사항이 다르기 때문이다. 매니페스트는 애플리케이션 구동하는데 필요한 이름, 허가 설정(Permission) 등의 정보를 포함하는 기본 설정 파일로, 각 스토어가 정의한 항목이나 속성값이 다를 경우 호환이 불가능하다. 이것은 웹 애플리케이션이 브라우저에 종속적인 주요 원인이며, 이를 효율적으로 해결하기 위해서는 다양한 환경을 지원할 수 있도록 애플리케이션 매니페스트 표준과 스토어의 표준화가 필요하다.

OWS(Open Web Store)는 개방형 웹 애플리케이션 스토어로서, 특정 OS나 통신사, 단말기 등의 어떠한 조건에도 종속적이지 않은 웹 애플리케이션 스토어를 말한다[7]. 따라서 OWS는 애플리케이션 스토어 간에 연동이 가능할 수 있도록 표준화된 구조를 가진다. 본 논문에서는 OWS에 관한 연구로서 연동이 가능한 애플리케이션의 매니페스트 표준과 표준화된 애플리케이션을 지원하는 애플리케이션 스토어의 표준 구조를 제안하여 모바일 시장에서 일부 회사의 독점 구조를 해결하고, 사용자에게 폭넓은 애플리케이션 서비스를 제공하고자 한다.

본 논문의 2장은 웹 애플리케이션 스토어 및 OWS 관련 연구 사항에 대해서 설명하고, 3장에서는 매니페스트 표준 제안을 위한 사전 연구에 대해 설명한다. 3장에서 나온 비교 분석 결과를 통해 도출된 매니페스트 표준을 4장에서 제안하며, 5장에서는 스토어의 표준 구조인 참조 모델(Reference Architecture)를 제안한다. 6장에서 참조 모델을 기반으로 한 연동 시나리오에 대해서 기술하며, 7장에서 결론 및 향후 연구에 관하여 기술한다.

2. 관련 연구

네이티브 애플리케이션은 모바일 기기에서 최적화된 언어로 개발되는 애플리케이션이다. 대표적인 예로 구글 안드로이드를 기반으로 하는 네이티브 애플리케이션은 안드로이드 SDK를 이용하여 자바 언어로 개발이 되며, 애플 iOS 기반의 네이티브 애플리케이션은 iOS SDK를 이용하여 Objective-C 언어로 개발된다. 이러한 네이티브 애플리케이션은 속도가 빠르고, 단말기의 기능을 효과적으로 사용할 수 있지만, 각 플랫폼을 지원하는 API를 사용하여 별도로 개발해야 하는 단점이 있다.[8] 이러한 네이티브 애플리케이션의 종속적인 특성은 개발비용에 대비하여 수익 저하 현상을 일으키는 문제점을 가져왔다.

웹 애플리케이션은 차세대 웹 언어인 HTML5와 CSS3, JavaScript를 이용하여 개발이 되며 브라우저를 기반으로 하기 때문에 플랫폼 종속적인 네이티브 애플리케이션의 주된 문제점을 해결했다. 하지만 애플리케이션 성능이 브라우저의 성능에 따라 달라지며, 고성능의 게임이나 섬세한 UI 표현에는 한계를 가지고 있다. 또한 단말기의 하드웨어를 제어할 수 없다는 치명적인 문제를 가지고 있고, 단말기에 저장된 일정 정보나 주소록 등의 정보를 활용할 수 없어서 모바일 플랫폼에서 많은 제약을 가지고 있다. 이러한 문제점 해결을 위해 최근 하드웨어 제어를 위한 Device API 표준화 연구가 활발히 진행되고 있다[8].

크롬 웹 스토어는 가장 활발하게 운영되고 있는 웹 스토어로 2010년 12월에 출시되었다. 인터넷 익스플로러(IE)와 비교하여 간단한 UI(User Interface)와 빠른 속도가 특징인 크롬 브라우저는 현재 약 7억 5000만명이 사용하고 있다. 이러한 크롬 브라우저를 기반으로 운영되고 있는 크롬 웹 스토어는 각 국의 사용자들을 위해 한국어를 포함한 약 40여개의 언어를 지원하며, 게임이나 교육, 뉴스 등의 다양한 컨텐츠를 지원하고 있다. 하지만 현재 크롬 브라우저를 제외한 브라우저는 지원을 하지 않고 모바일 플랫폼 또한 지원하지 않고 있어서 스마트폰 환경에서는 사용이 불가능하다.

파이어폭스 브라우저를 기반으로 한 마켓플레이스는 2012년 6월에 출시되어 현재 운영 중에 있다. 크롬보다 늦게 출시되었지만, 출시 이전부터 별도의 개발자 허브를 구성하여 여러 개발자들이 개발에 참여할 수 있도록 했다. 현재 마켓플레이스 는 크롬의 웹 스토어와 마찬가지로 교육, 게임 등의 컨텐츠를 지원하고 있으며, 파이어폭스 OS 또는 파이어폭스 브라우저 환경에서만 구동이 가능하다. 안드로이드 환경에서는 파이어폭스 브라우저를 설치한 후 웹 애플리케이션을 다운받아 이용할 수 있다.

WAC(Wholesale App Community)는 스마트폰 애플리케이션을 공유할 수 있는 도매시장으로, SKT와 KT, AT&T 등의 전세계 24개 주요 통신사와 삼성전자, LG전자 등의 제조업체가 참여하여 결성하였다. WAC의 목적은 현재 모바일 시장에서 구글과 애플의 콘텐츠 의존도를 낮추고 통합형 시장을 구축하기 위한 것으로 이동 통신사나 운영체제, 단

말기와 관계없이 모든 애플리케이션 스토어에 판매할 수 있도록 하는 플랫폼이다. 하지만 참여한 많은 회사들을 주도하는 주체가 없어서 처음 출범 당시의 계획에 비해 뚜렷한 성과를 나타내지 못했다. WAC은 2012년 7월 세계이동통신사업자연합회(GSMA)[9]가 인수하여 총괄 운영 중에 있지만, 지난 3월 WAC 사업이 실패했음을 공식적으로 선언했다. 8WAC 사업은 실패했지만 통합형 웹 스토어를 통해서 표준화의 초석을 마련한 것에 의미가 있다.

한국형 통합 웹 스토어인 Korea-WAC이 이름을 바꾼 후 2011년 9월 재출범한 것이 Korea-Apps[10]이다. Korea-Apps는 기존의 Korea-WAC과 연동되는 독자적인 플랫폼으로 SK텔레콤, KT, LG 유플러스 등의 이동 통신사와 삼성전자, LG전자, 한국무선인터넷산업연합회가 참여했다. 현재 HTML5 기반의 애플리케이션을 판매할 수 있도록 지원하여 기존의 모든 모바일 OS에서 구동이 가능한 '원소스 멀티플랫폼'이 가능하다. 하지만 기존 WAC과 마찬가지로 구글과 애플이 장악한 모바일 시장에서 개발자가 자발적으로 참여할만한 장점이 없고, 시장이 작아서 크게 성장하지 못하고 있다.

Fig. 1. Korea-Apps Procedure

3. W3C, 파이어폭스 및 크롬의 매니페스트 비교

현재 대표적인 웹 애플리케이션 스토어인 파이어폭스 마켓플레이스, 크롬 웹 스토어의 매니페스트와 W3C에서 정의한 위젯(Widget)의 매니페스트 비교를 통해서 상호간에 필수 요구사항을 파악 및 분석할 수 있다. 이러한 비교 분석 과정을 통해서 파악된 매니페스트 요구사항은 현재 많은 웹 애플리케이션과 사용자를 보유한 스토어의 항목이기 때문에 신뢰성이 보장된다고 할 수 있다.

각 스토어의 매니페스트를 기본정보, 언어, 개발자 등 9개의 카테고리로 분류하고, 같은 역할을 수행하는 항목을 동일한 행(row)에 나열했다. 본 논문에서는 비교의 예시로 3가지 카테고리에 대해서 설명하며, 이를 포함한 9개의 모든 비교표 및 분석은 OWS 기술문서인 참고문헌 [11]에 설명되어 있다.

3.1 기본 정보 비교

매니페스트 비교를 위해 나눈 카테고리 중 기본 정보는 웹 애플리케이션의 이름이나 설명 등의 기본적인 매니페스트 항목을 나열했다. name은 웹 애플리케이션의 이름을 쓰는 항목으로, 각 스토어 모두 필수적으로 요구하는 사항이다. description은 웹 애플리케이션에 대한 간략한 설명이다. 파이어폭스는 description 항목을 필수적으로 사용하도록 규정하고 있지만, 크롬은 필수적이 아닌 개발자의 선택에 따라 명시하도록 규정하고 있다. 파이어폭스의 version과 크롬의 manifest_version은 매니페스트의 버전을 나타내는 문자열로, 파이어폭스는 임의값을 입력하여 사용한다. 이에 비해 크롬은 브라우저의 버전에 따라 지원이 가능한 웹 애플리케이션을 구분하기 위한 방법으로 사용한다. manifest_version 1은 차츰 중단할 예정이며, 브라우저 버전이 17 또는 그 이하인 경우 manifest_version 2를 권장하지 않고 있다. 크롬은 매니페스트 버전 이외에 애플리케이션의 버전을 명시할 수 있도록 하는 version과 애플리케이션, 테마 등이 요구하는 최소 크롬 버전을 명시하는 minimum_chrome_version 항목이 있다.

Table 1. Basic Information Comparison

W3C	Firefox	Chrome
name	name	name
description	description	description
-	version	manifest_version
-	-	version
-	-	minimum_chrome_version

3.2 언어 비교

Table 2. Language Comparison

W3C	Firefox	Chrome
span	default_locale	default_locale
-	locales	-

언어 비교항목에는 기본 Locale 항목인 default_locale을 크롬과 파이어폭스가 모두 필수적으로 사용할 것을 규정하고 있다. W3C 위젯의 경우에는 default_locale과 비슷한 항목으로 span이 있다. 또한 파이어폭스에는 default_locale 이 외에 다른 언어에 대한 지원을 위해 locales라는 항목을 별도로 두고 개발자의 선택에 따라 쓰도록 한다.

3.3 개발자 비교

Table 3. Developer Comparison

W3C	Firefox	Chrome
author	developer	-

파이어폭스는 개발자 관련 매니페스트 항목으로 developer가 있는 반면, 크롬은 개발자와 관련된 항목이 존재하지 않는다. 파이어폭스의 developer 항목은 개발자의 이름과 url을 내부 속성으로 정의하는데, url은 사용자가 애플리케이션 개발자에 대한 자세한 정보를 얻을 수 있는 사이트의 주소를 말한다. developer는 필수 항목이 아닌 개발자 선택에 따라 작성하도록 규정하고 있고, 이와 비슷한 항목으로 W3C에는 author가 있다.

4. 매니페스트 표준 제안

브라우저를 기반으로 한 웹 애플리케이션이 네이티브 애플리케이션의 플랫폼 종속적인 문제를 해결했으나, 각 스토어간에 연동이 되지 않는 주요 원인은 매니페스트에 있다. 매니페스트는 애플리케이션이 구동하는데 필요한 정보를 포함하는 기본 설정 파일로, 애플리케이션의 명칭과 설명을 기술한 JSON 파일이다. 스토어에 따라 JSON이 아닌 XML 형식의 매니페스트도 존재하며, 아이콘이나 필요한 허가 설정 등을 포함하고 있다.

```
{
   "name": "TaeJun's Homepage",
   "description": "Web application test",
   "manifest_version": 2,
   "version": "0.0.1",
   "app": {
      "launch": {
      "local_path": "Home.html"
      }
   },
   "icons": {
      "16":"icon_16.png",
      "128": "icon_128.png"
   },
   "permissions": [
      "unlimited_storage",
      "notifications"
   ]
}
```

(a) Chrome Manifest

```
{
   "name": "TaeJun's Homepage",
   "description": "Web application test",
   "launch_path": "/Home.html",
   "icons": {
      "128": "icon_128.png"
   },
   "developer": {
      "name": "TaeJun",
      "url": "http://facebook.com/taejun.ryu"
   },
   "default_locale": "en"
}
```

(b) Firefox Manifest

Fig. 2. Chrome and Firefox Manifest Example

Fig. 2의 (a)와 (b)는 동일한 웹 애플리케이션을 개발할 때 사용된 매니페스트로, (a)는 크롬의 매니페스트, (b)는 파이어폭스의 매니페스트 예시이다. 두 매니페스트에는 많은 차이가 있는데, 먼저 크롬은 매니페스트의 파일명으로 manifest.json을 요구하며, 파이어폭스는 manifest.webapp을 요구한다. 각 스토어가 요구하는 파일명으로 매니페스트를 생성하지 않으면 웹 애플리케이션을 업로드 할 때 매니페스트를 찾을 수 없기 때문에 파일명 또한 매니페스트 표준화할 때 함께 정의해야 한다. 따라서 본 논문의 매니페스트 표준안에서는 파일명으로 파이어폭스에서 사용중인 manifest.webapp이 가장 적합하다고 생각하여 이를 제안한다.

크롬의 경우 `manifest_version`과 `version`으로 나누어 필수 항목으로 명시하도록 되어 있지만 파이어폭스의 경우 `version`은 개발자의 선택에 따라 입력하도록 되어있다. 또한 크롬과 파이어폭스 모두 `Permissions`을 개발자의 선택에 따라 입력하지만 각 스토어가 정의한 내부 속성이 달라서 서로의 매니페스트를 인식할 수 없다.

현재 운영 중인 크롬의 웹 스토어와 파이어폭스의 마켓플레이스의 매니페스트 요구사항을 분석하고 W3C Widget Standard의 Recommendation 문서인 Packaging and XML Configuration[12]에서 정의한 위젯 매니페스트를 비교하여 표준안을 도출하였다. 표준 매니페스트는 기본 정보, 언어, 개발자 등 총 8개의 카테고리로 나뉘어 있으며, 총 16개의 각 항목은 개발자가 필수로 입력해야 하는 필수항목

Fig. 3. Manifest Test Error Message

(required)과 선택에 따라 입력할 수 있는 선택항목(optional) 특성을 가지며 일부는 내부 속성을 포함하고 있다[13].

4.1 기본 정보 표준안

Table 4. Basic Information Standard Suggestions

name	No attribute	required
description	No attribute	required
version	No attribute	required

`name`, `description`, `version`은 웹 애플리케이션의 기본 정보 항목으로, 모두 필수로 입력해야 하는 필수 항목(required)이며 내부 속성이 없다. `name`은 애플리케이션의 이름이고, `description`은 애플리케이션에 대한 간략한 설명이다. `name`과 `description`은 사용자가 애플리케이션을 구별하는 최소의 정보이기 때문에 필수로 입력한다. `version`은 애플리케이션의 버전을 뜻하는데, `version`을 명시함으로써 개발자에게는 이전의 소스코드와 수정된 코드를 분리하여 버전 관리를 용이하도록 하고, 사용자는 애플리케이션이 업데이트 된 것을 알 수 있다.

4.2 언어 표준안

Table 5. Language Standard Suggestion

locales	default, others [name, description]	required

언어에 관련된 매니페스트 항목은 `locales`가 있다. `locales`는 사용자가 기본으로 사용하는 언어에 따라 웹 애플리케이션을 사용할 수 있도록 하는 언어 설정 항목이다. 개발자는 `default` 내부 속성으로 배포할 때의 기본 언어를 설정하고, `others` 내부 속성을 이용하여 각 언어에 대한 애플리케이션의 `name`과 `description`을 정의할 수 있다. `locales`는 필수적으로 입력하는 필수 항목이다.

4.3 개발자 표준안

개발자의 정보를 명시할 수 있도록 한 `developer` 항목은 필수이다. 파이어폭스 마켓플레이스는 `developer`를 개

Table 6. Developer Standard Suggestion

developer	name, url	required

발자의 선택에 따라 정의할 수 있도록 선택 항목(optional)으로 규정되어있지만 표준안에서 developer가 필수인 이유는 스토어 간 웹 애플리케이션 연동 시 개발자의 개인 정보를 함께 연동할 필요 없이 최소의 정보만을 전달하기 위해서이다. 개발자의 이름과 url를 매니페스트에 명시하고 연동함으로써 RemoteStore에서 개발자를 구분하고, 블랙리스트 개발자를 관리하는데 용이하도록 했다.

4.4 그래픽 표준안

Table 7. Graphic Standard Suggestion

icons	16, 32, 48, 64, 128, 256	required

icons는 사용자가 특정 애플리케이션을 다른 애플리케이션들과 시각적으로 구별할 수 있도록 하는 이미지이다. 현재 파이어폭스의 경우 icons가 선택이고 크롬의 경우는 필수인데, 표준안에서는 필수 항목으로 제정하였다. icons는 다양한 크기가 내부 속성으로 정의되어 있는데, 각 스토어마다 요구하는 아이콘의 이미지 크기가 다르기 때문에 연동 시 호환을 위해 다양한 크기를 정의하도록 했다. 내부 속성에서 16, 32, 48 등의 숫자는 16x16, 32x32, 48x48과 같은 아이콘의 크기를 의미하며, 크기의 기준은 현재 스토어를 운영 중인 크롬과 파이어폭스에서 사용하는 최적화된 아이콘 크기를 모두 포함시켰다.

4.5 뷰 표준안

Table 8. View Standard Suggestions

orientation	portrait, landscape	optional
fullscreen	true, false	optional

모바일 환경에서 애플리케이션은 필요에 따라 가로 모드나 세로 모드로 고정해야 할 경우가 있는데, orientation은 이러한 스크린 방향 모드를 정의하기 위한 항목이다. 개발자의 판단에 따라 최적화된 스크린 모드를 결정하기 때문에 orientation은 선택 항목으로 정의되어 있다. 세로 속성인 portrait와 가로 속성인 landscape를 내부 속성으로 사용가능하며 orientation은 모바일 환경에서만 가능한 항목이다. 불린 값으로 정의되는 fullscreen은 애플리케이션이 전체 화면을 사용할 것인지에 대한 항목이다. fullscreen 또한 orientation과 마찬가지로 개발자의 선택에 따라 정의하는 선택항목이다.

4.6 승인 표준안

resources는 애플리케이션이 사용하는 리소스에 대해서 미리 명시하여 불필요한 리소스의 낭비를 막을 수 있도록

Table 9. Approval Standard Suggestions

resources	No attribute	required
key	No attribute	optional
offline_mode	true, false	optional
permissions	defined attributes by each store	required

한다. 개발자는 resources 항목을 통해 이미지나 영상 등 애플리케이션 구동에 필요한 리소스를 명시하며, 매니페스트에 명시하지 않은 리소스가 존재할 경우 업로드가불가능하기 때문에 resources는 필수이다.

key 태그는 연동과정에서 애플리케이션을 구별하는 고유식별 라이센스로서 개발자가 애플리케이션을 등록한 후 검수에서 통과하면 스토어로부터 발급받는다. key는 선택 항목으로서 개발자의 선택에 따라 매니페스트에 명시하지만, 발급받은 키를 적지 않을 경우 다른 스토어로 해당 애플리케이션을 연동시킬 수 없다. 따라서 선택이지만 연동을 하기 위해서는 매니페스트에 꼭 적어야 한다. 현재 제안하는 key 태그는 알파벳과 숫자로 구성된 10자리 문자열이며, 앞의 3자리는 브라우저 코드, 뒤의 7자리는 애플리케이션 코드로 사용한다. key 태그가 적힌 애플리케이션이 연동될 경우 RemoteStore는 브라우저 코드를 이용하여 해당 애플리케이션이 어디로부터 연동되어 왔는지 파악할 수 있다. 또한 연동된 애플리케이션의 코드를 모두 보관하여, 연동이 아닌 불법 파일을 이용한 애플리케이션의 설치를 방지할 수 있다.

B	1	C	2	3	G	9	8	D	H
Browser code			Given application code from the browser						

Fig. 4. Proposal of Key Example

offline_mode는 오프라인 환경에서 애플리케이션 사용 가능 여부를 정의하는 항목으로 불린 값을 내부 속성으로 사용한다. 애플리케이션이 네트워크 환경의 영향을 받지 않는다면 오프라인에서도 사용이 가능하기 때문에 개발자는 참(true)을 설정할 수 있다. offline_mode는 선택 항목으로 개발자가 별도로 명시하지 않으면 거짓(false)가 되어 오프라인 환경에서 애플리케이션을 사용할 수 없다. permissions는 애플리케이션이 특정 자원이나 하드웨어, 네트워크에 접근하기 위한 권한 설정으로 필수 항목이다. 애플리케이션을 연동할 경우 각 브라우저가 정의한 내부 속성을 정의해야 하는데, 예를 들어, 파이어폭스 개발자가 크롬에 애플리케이션을 연동한다면, Fig. 5와 같이 파이어폭스와 크롬에서 정의한 permissions의 내부 속성을 모두 명시한다.

4.7 웹 정보 표준안

웹 정보에 관한 매니페스트 항목으로 경로설정에 대한 launch_path와 appcache_path가 있고, 옵션 설정에 관한 options이 있다. launch_path는 웹 애플리케이션을 실행

```
"permissions" : {
   "firefox" : {
      "power",
      "contacts"
   }
   "chrome" : {
      "storage",
      "geolocation"
   }
}
```

Fig. 5. Proposal of Permissions Example

Table 10. Web Information Standard Suggestions

launch_path	No attribute	required
appcache_path	No attribute	required
options	No attribute	optional

하기 위한 시작 경로를 명시하는 것이고, appcache_path는 애플리케이션의 캐시 경로를 정의하여, 서버의 부하를 줄이고 더 빠르게 애플리케이션이 구동될 수 있도록 한다. launch_path와 appcache_path 모두 필수 항목으로 개발자가 매니페스트에 반드시 적어야 하는 항목이다. options는 사용자에게 편의를 제공하기 위해 옵션 기능을 부여하는 것으로, 개발자의 선택에 따라 정의할 수 있는 선택 항목이다.

4.8 정책 표준안

Table 11. Policy Standard Suggestion

csp	privileged, certified	optional

csp(Content Security Policy)는 애플리케이션 인증에 관련된 정책을 포함하고 있다. 내부 속성으로 privileged와 certified를 사용하는데, privileged는 특별한 검수 과정을 통해서 승인이 되고, 이에 따라서 사용자에게 높은 안전성을 제공한다. certified는 보안에 취약점을 가진 애플리케이션이 높은 보안성을 유지할 수 있도록 하는 것으로, 주로 시스템 기능을 이용하는 애플리케이션에 대해서 인증한다. csp는 크로스 사이트 스크립팅 또는 이와 관련된 공격을 방지하기 위한 컴퓨터 보안 개념에 속한다.

5. 참조 모델(Reference Architecture) 제안

Extended Storefront[14]는 모바일 관련 표준화 단체인 OMA[15]에서 제안한 TAS의 애플리케이션 스토어 구조를 확장한 형태이다. 클라이언트에게 직접적인 서비스를 제공하는 Storefront[1]를 연동을 위한 SAM(Shared App Management)과 정책에 관련된 Policy 모듈, Policy Provider로 구성되어있다. 참조 모델[16]은 Storefront, SAM, Policy 각각의 모듈을 추가된 연동 요구사항을 포함하여 내부를 상세히 설계한 구조이다.

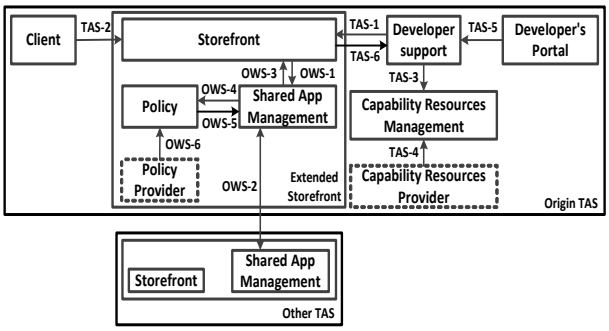


Fig. 6. TAS Architecture including Extended Storefront

5.1 Storefront

Storefront 모듈의 중앙 처리 내부 모듈로, 클라이언트 또는 개발자로부터 요청 받은 정보를 애플리케이션 관리자(App Manager)나 계정 관리자(Account Manager)로 전달한다. 또한, 애플리케이션 연동 정보 전달을 위해 SAM에게 정보를 전달한다. 애플리케이션의 추가, 삭제, 업데이트, 검색 연산을 수행하는 애플리케이션 관리자는 애플리케이션을 카테고리에 따라 분류한다. 계정 관리자는 클라이언트의 개인정보, 장바구니 리스트, 즐겨 찾기 리스트와 같은 정보를 사용자 계정 저장소(User Account Repository)에 추가, 삭제, 변경 등의 연산을 수행한다. 이 뿐만 아니라 클라이언트가 애플리케이션을 구매할 때 발생하는 요금이나 연동된 애플리케이션으로부터 발생하는 수수료에 관한 처리를 수행한다.

애플리케이션 저장소(App Repository)는 OriginStore에서 개발되었거나 RemoteStore에서 연동되어온 애플리케이션을 저장하고 있는 저장소이며, 사용자 계정 저장소는 개발자를 포함한 사용자의 계정 정보를 보관하는 저장소이다.

5.2 Shared App Management

SAM(Shared App Management)은 RemoteStore 커넥터(Connector) 모듈과 저장소 2개로 이루어져있다.

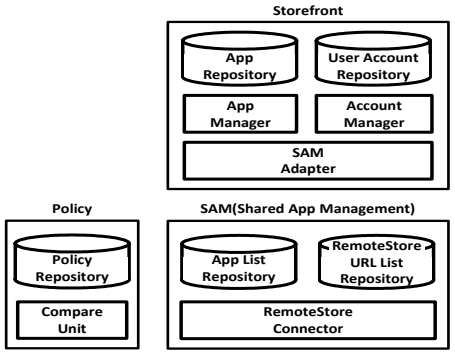


Fig. 7. OWS Reference Architecture

1) Storefront, SAM, Policy 명칭은 참고문헌 [14]와 [15]에서 정의된 것으로 본 논문에서는 해당 명칭을 유지하여 기술함

RemoteStore 커넥터는 Storefront의 SAM 어댑터로부터 요청을 받아 연동이 약속되어 있는 다른 스토어로 통신을 하는 부분이다. 애플리케이션이나 애플리케이션 리스트에 대한 수신, 발신을 모두 담당하며, Policy 모듈에 검수를 요청할 수 있다. RemoteStore 커넥터는 통신에 관련된 부분 이외에도 저장소의 값을 추가, 삭제, 변경하는 연산도 함께 수행한다.

애플리케이션 리스트 저장소(App List Repository)는 OriginStore에서 개발되었거나 RemoteStore에서 연동되어온 애플리케이션 리스트를 저장하는 저장소이며, Storefront는 SAM의 애플리케이션 리스트 저장소로부터 애플리케이션 ID를 참조 받아 다른 스토어에게 연동을 요청한다. 애플리케이션 리스트 저장소에는 Policy 모듈로부터 연동이 미승인된 애플리케이션의 리스트가 포함되어 있다. 또한 SAM에는 연동 요청을 위해서 연동이 약속된 다른 애플리케이션 스토어의 URL를 보관하는 RemoteStore URL 리스트 저장소(RemoteStore URL List Repository)를 가지고 있다.

5.3 Policy

애플리케이션 검수를 수행하는 Policy 모듈은 SAM으로부터 애플리케이션 ID를 전달받아 검수를 실시하는 검수 모듈(Compare Unit)과 정책을 보관하고 있는 정책 저장소(Policy Repository)를 가지고 있다. 검수 모듈에서는 연동과 관련하여 다른 스토어로 발신이 가능한가, 수신한 애플리케이션이 등록 가능한가 등에 대해서 검수를 실시하고 이에 따른 승인 및 미승인 정보를 SAM에게 전달한다.

6. 연동 시나리오 예시

본 논문에서 제안한 참조모델을 기반으로 스토어 간 애플리케이션 연동 시나리오를 설명한다. 연동 시나리오는 웹 애플리케이션 리스트나 사용자에 의한 악의적인 애플리케이션 레포트, 버그 레포트 등을 포함하여 연동 요구사항을 적용한 9개의 시나리오에 대해서 설명한다.

6.1 애플리케이션 리스트 연동 과정

애플리케이션 연동을 위해서 평상시에 각 스토어는 애플리케이션 리스트를 연동한다. OriginStore에서 개발된 애플리케이션은 SAM에 있는 애플리케이션 리스트 저장소에 리스트를 보관하고 있는데, RemoteStore 커넥터가 RemoteStore URL 리스트 저장소로부터 연동이 약속된 스토어의 URL를 참조하여 각 스토어로 리스트를 연동한다. RemoteStore SAM 내부에 있는 RemoteStore 커넥터는 애플리케이션 리스트를 전달받아 애플리케이션 리스트 저장소에 저장한다.

6.2 애플리케이션 연동 과정

RemoteStore로 애플리케이션 리스트 연동이 완료되면 Remote Storefront는 SAM을 통해서 연동하고자 하는 애플리케이션을 OriginStore로 요청한다. 연동을 요청받은 OriginStore는 해당 애플리케이션을 발신하기 전에 Policy 모듈에게 검수를 요청한다. 이 때 Policy 모듈은 애플리케이션이 연동에 적합한 지에 대해서 버전이나 상태 등을 검수한다. 검수에서 통과된 애플리케이션은 RemoteStore로 전송을 하고, 이를 전달받은 RemoteStore의 SAM은 연동된 애플리케이션을 Policy로 검수를 요청한다. 모든 검수가 완료된 애플리케이션은 Storefront의 애플리케이션 저장소에 저장되면서 연동을 완료한다. 만약 검수에서 통과하지 못하면 SAM의 애플리케이션 리스트 저장소에 등록이 거부된 애플리케이션 목록을 저장한다.

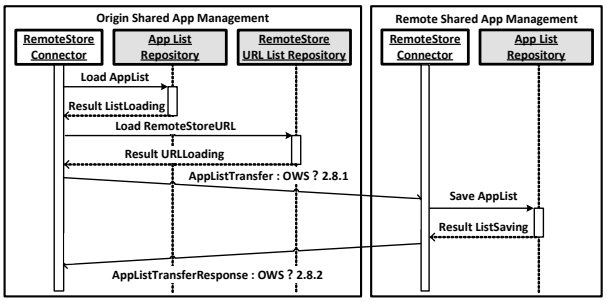


Fig. 8. Application List Shared Sequence Diagram

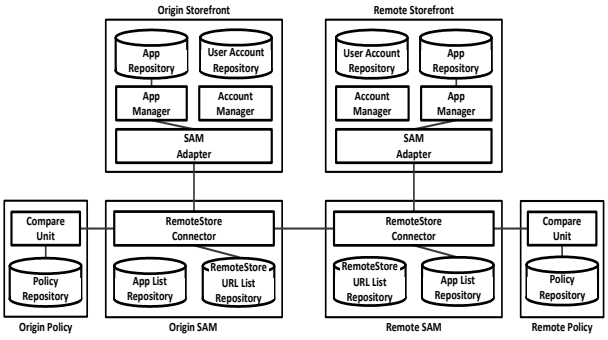


Fig. 9. Application Shared Procedure

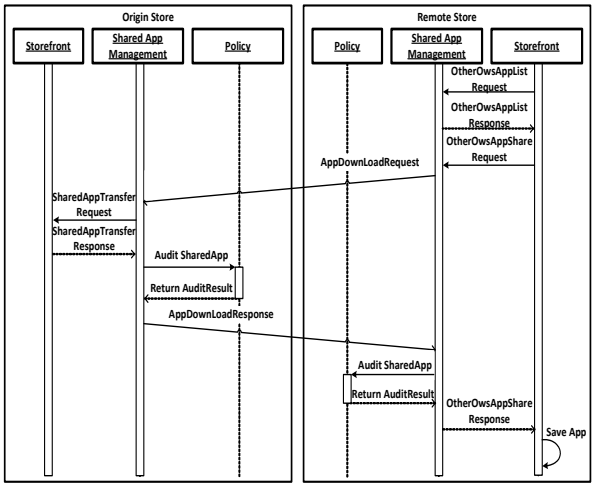


Fig. 10. Application Shared Sequence Diagram

6.3 애플리케이션 재검수 후 결과 전달과정

애플리케이션 연동 과정에서 RemoteStore는 애플리케이션을 수신한 후 Policy 모듈에 검수를 요청한다. 검수는 Policy 내부에 있는 검수 모듈에서 수행하는데, Policy에 적합하지 않을 경우 Storefront에 애플리케이션 등록이 거부될 수 있다. 거부된 애플리케이션은 SAM에 있는 애플리케이션 리스트 저장소에 저장되며, RemoteStore 커넥터가 주기적으로 Policy 모듈에게 재검수를 의뢰한다. 재검수 후에 애플리케이션 등록 허가 또는 거부의 결과를 OriginStore로 전달하고, Developer Support를 통해서 개발자에게 최종적으로 전달한다.

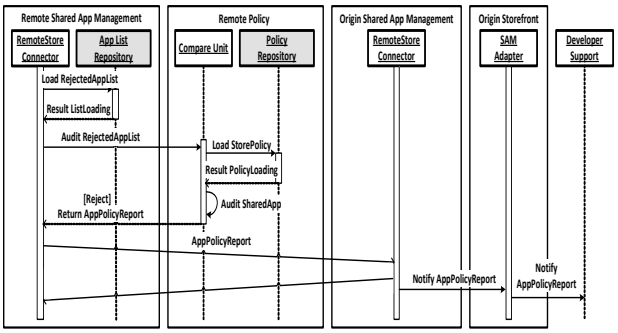


Fig. 11. Result of Reaudition Transfer Sequence Diagram

6.4 애플리케이션 재검수 후 등록 과정

RemoteStore에서 애플리케이션을 재검수한 후 등록되는 과정이다. OriginStore로부터 연동 후 등록이 거부되었던 애플리케이션에 대해서 RemoteStore는 주기적으로 Policy 모듈에게 재검수를 요청한다. RemoteStore의 정책이 변경되거나 애플리케이션의 부적합했던 요소가 수정되면 재검수에서 등록 허가가 될 수 있는데, 이 때 SAM은 Storefront로 정보를 전달하여 해당 애플리케이션을 등록하고 OriginStore에게 등록이 완료되었음을 전달한다. OriginStore는 Developer Support를 이용하여 개발자에게 결과를 전달한다.

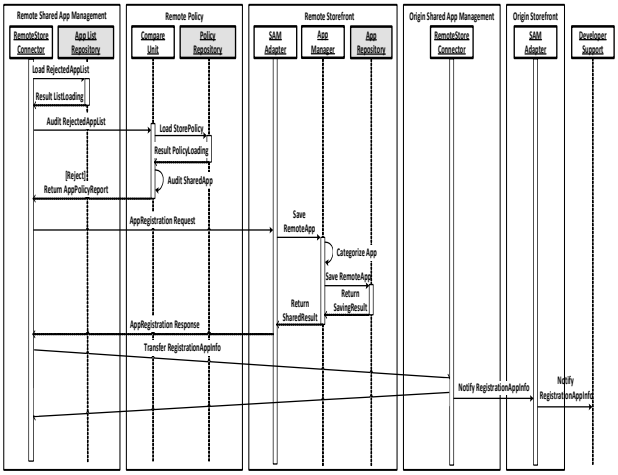


Fig. 12. Reaudit Shared Success Sequence Diagram

6.5 애플리케이션 구매에 따른 과금 절차

RemoteStore에서 애플리케이션 구매에 따른 과금 과정이다. OriginStore에서 RemoteStore로 애플리케이션이 연동된 후에 RemoteStore의 클라이언트가 애플리케이션을 구매할 경우 계정 관리자에서 구매 금액 중 일부 수수료를 제외한다. 그 후 SAM 어댑터는 나머지 금액을 SAM에게 전달하고, SAM은 OriginStore에게 전달한다. 이 때 전달하는 방식은 클라이언트가 애플리케이션을 구매할 때마다 전달할 수도 있고, 일정 기간 동안 금액을 모은 뒤 주기적으로 전달할 수도 있는데, 이것은 해당 스토어의 정책에 따라 달라질 수 있다. 금액이 OriginStore로 전달되면 Developer Support를 통해서 개발자에게 금액을 전달한다.

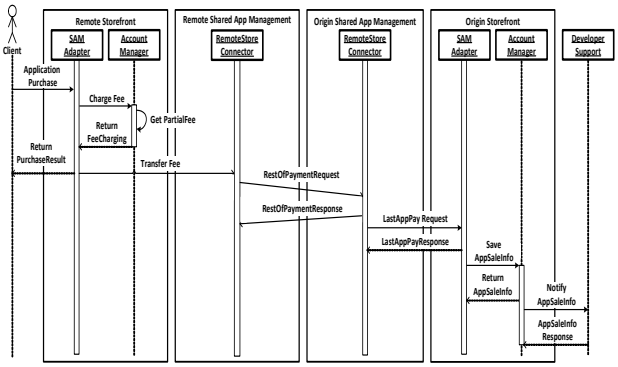


Fig. 13. Application Charging Sequence Diagram

6.6 연동된 애플리케이션 폐기 과정

개발자가 OriginStore에 애플리케이션 폐기 요청을 보내면 RemoteStore에 연동되어 있는 애플리케이션을 폐기하는 절차이다. 먼저, 개발자는 Developer Support를 통해서 Storefront로 삭제 요청을 하면 SAM 어댑터는 애플리케이션 관리자를 통해 저장소에서 애플리케이션을 폐기하고 SAM은 애플리케이션 리스트 저장소에서 애플리케이션 정보를 삭제한 뒤 RemoteStore로 폐기 요청을 보낸다. RemoteStore는 삭제 요청에 따라 OriginStore와 마찬가지로 애플리케이션에 대한 정보를 모두 폐기한다.

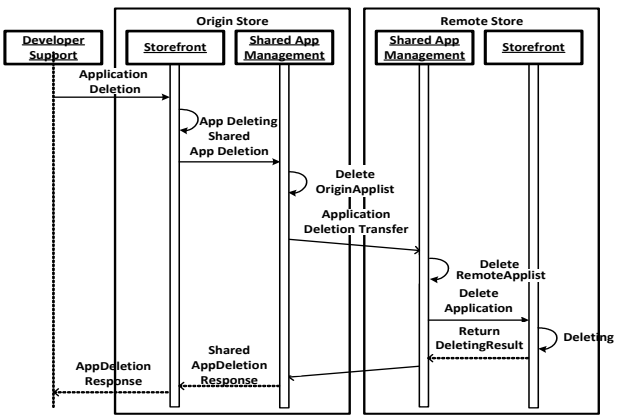


Fig. 14. Application Deletion Sequence Diagram

6.7 OriginStore에서 Malicious&Bug Report 전달과정

애플리케이션이 악의적인 요소나 오류를 포함하고 있을 경우 클라이언트는 Storefront에게 악의적인 애플리케이션 레포트 또는 버그 레포트를 전달할 수 있다. 본 과정은 OriginStore의 클라이언트가 악의적인 애플리케이션에 대해서 Storefront로 악의적인 애플리케이션 레포트를 보내면 SAM 어댑터는 개발자에게 해당 정보를 전달하고 애플리케이션이 연동되어 있는 RemoteStore로 각각 전달한다. 단순한 오류일 경우 개발자에게 전달하는 것으로 과정이 완료될 수 있지만, 클라이언트에게 심각한 영향을 줄 수 있거나, 정책에 어긋나는 악의적인 요소가 포함되어 있을 경우 본 과정 후에 애플리케이션이 게시 중단 될 수 있다.

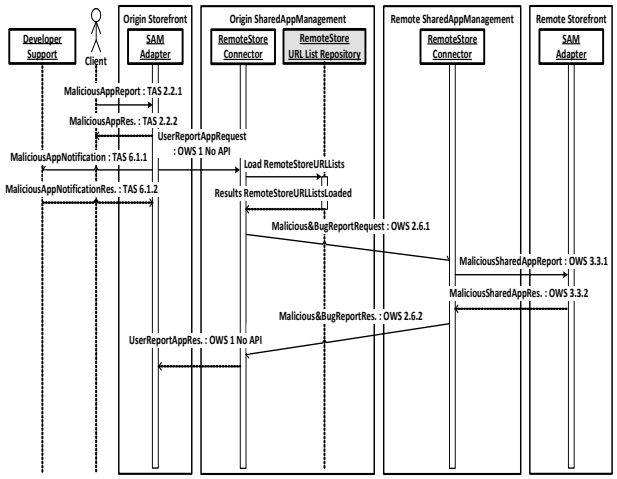


Fig. 15. Transfer Malicious&Bug Report From OriginStore Sequence Diagram

6.8 RemoteStore에서 Malicious&Bug Report 전달과정

OriginStore로부터 연동되어온 애플리케이션에 오류나 악의적인 요소가 포함되어 있을 경우 RemoteStore의 클라이언트는 Storefront에게 악의적인 애플리케이션 레포트 또는 버그 레포트를 보낼 수 있다. 정보를 전달받은 RemoteStore는 OriginStore로 전달하고 OriginStore는 해당 애플리케이션의 개발자에게 레포트를 전달한다. 또한 OriginStore는 SAM을 통해서 연동이 약속된 각각의 RemoteStore로 악의적인 애플리케이션 레포트 또는 버그 레포트를 전달한다.

6.9 애플리케이션 업데이트 과정

개발자가 애플리케이션을 OriginStore에 처음 등록하면 해당 애플리케이션은 연동 절차를 통해서 RemoteStore에 등록된다. 등록 후 개발자가 Developer Support를 통해서 애플리케이션을 업데이트하면 OriginStore 뿐만 아니라 RemoteStore에 등록된 애플리케이션도 함께 업데이트되어야 한다. 개발자가 업데이트를 완료하면 먼저 OriginStore에서 SAM 어댑터가 애플리케이션 관리자를 통해서 애플리케이션을 갱신한다. 그 후 업데이트 정보를 SAM을 통해서 RemoteStore로 전달하고, RemoteStore의 SAM은 Store

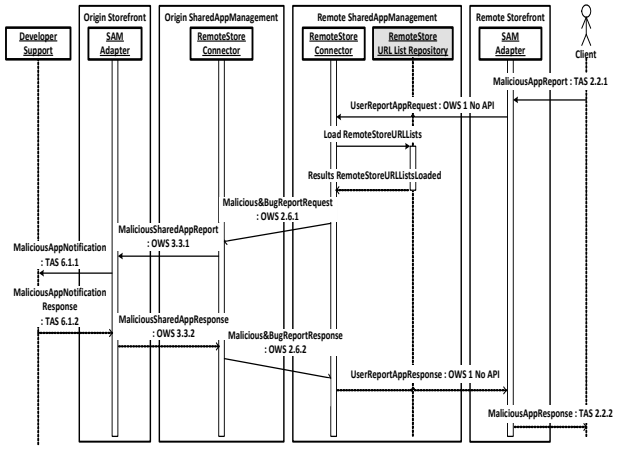


Fig. 16. Transfer Malicious&Bug Report From RemoteStore

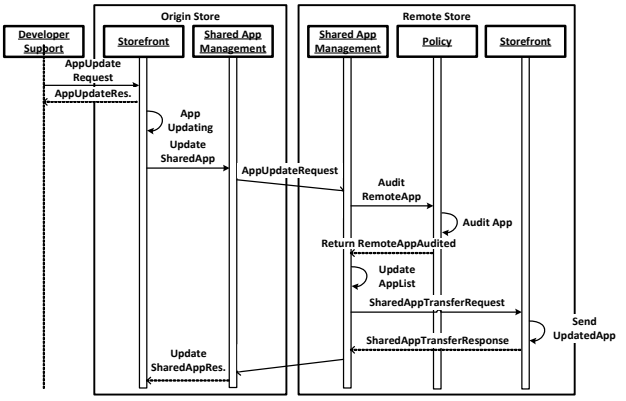


Fig. 17. Application Update Sequence Diagram

front의 SAM 어댑터에게 정보를 전달하여 OriginStore와 마찬가지로 저장소에 있는 애플리케이션을 갱신한다.

7. 결론 및 향후 연구 과제

모바일의 플랫폼이 발전하고 이에 따라 많은 응용 애플리케이션이 등장하고 있지만 종속적인 네이티브 애플리케이션으로 특성으로 인해 웹 애플리케이션이 주목을 받고 있다. 크로스 플랫폼을 지원하는 웹 애플리케이션 또한 해당 브라우저가 요구하는 매니페스트의 형식에 따라 웹 애플리케이션을 개발하기 때문에 브라우저에 종속되는 단점을 가진다. 따라서 이러한 종속적인 문제를 해결하여 사용자에게 더 넓은 서비스를 제공하고, 개발 비용 대비 수익을 향상시킬 수 있도록 모바일 플랫폼 및 스토어 구조에 대한 표준화 연구가 필요하다.

본 논문에서는 플랫폼이나 통신사, 스마트폰 기종에 관계없이 모든 플랫폼을 아우를 수 있는 개방형 웹 스토어의 참조모델과 애플리케이션 매니페스트 표준안을 제안한다. 현재 애플리케이션 스토어를 운영 중인 크롬과 파이어폭스의 매니페스트를 분석하고 연동과 관련하여 표준 항목들을 도출한 매니페스트 표준안은 개발자가 필수로 입력해야 하는

필수 항목과 선택에 따라 입력할 수 있는 선택 항목을 포함하여 총 16개로 이루어져있다. 참조모델은 TAS에서 제안한 애플리케이션 스토어의 구조를 연동 요구사항에 맞게 설계한 Extended Storefront를 기반으로 상세히 설계하였다.

향후 연구로는 Sysapp이나 Open webOS, OpenSocial Gadget의 매니페스트 요구사항을 분석하여 본 논문에서 제안한 매니페스트 표준안을 보완하고 참조 모델과 함께 표준안으로 제안할 예정이다.

참 고 문 헌

[1] 이승윤, “앱스토어 표준화 전략”, TTA, TTA Journal No.131, 2010.11

[2] WAC(GSMA), http://www.gsma.com/oneapi/

[3] W3C, Working Draft, "HTML5",http://www.w3.org/TR/html5

[4] 전종홍, 이승윤, “차세대 모바일 웹 애플리케이션 표준화 동향, ETRI, 전자통신동향 분석 제 25권 제 1호, 2010.

[5] Chrome Web Store, https://chrome.google.com/ webstore

[6] Firefox Marketplace, https://marketplace.firefox.com/

[7] J. H. Kim, J. H. Baek, Y. W. Nam, J. H. Jeon, S. Y. Lee, & S. W. Park, “Prototype Implementation of OWS (Open Web Store) interworking scenario”, Korea Computer Congress, Vol.39, No.1, pp.137-139, 2012.

[8] 이강찬, “모바일웹 플랫폼과 Device API 표준”, TTA,TTA Journal No.128, 2010. 3

[9] GSMA, http://www.gsma.com/

[10] Korea-Apps, http://www.koreaapps.net/

[11] C. J. Kim, T. J. Ryu, & S. W. Park, “DISLab_OWS_TR_20130722_R2”, http://dislab.hufs.ac.kr/wiki4lab/img_auth.php/9/99/DISLab_OWS_TR_20130721_R2.pdf, 2013.7

[12] Manifest for Web Applications, http://www.w3.org/2012/sysapps/manifest/

[13] C. J. Kim, T. J. Ryu, J. H. Jeon, S. Y. Lee, & S. W. Park, “A Manifest Standard Suggestions for Interworking OWS (Open Web Store)”, Korea Computer Congress, pp.475-477, 2013.

[14] J. H. Baek, J. H. Kim, Y. W. Nam, J. H. Jeon, S. Y. Lee, & S. W. Park, “A Protocol for interworking Open Web Application Store”, Korea Computer Congress, Vol.39, No.1, pp.70-72, 2012.

[15] Open Mobile Alliance, http://openmobilealliance.org/

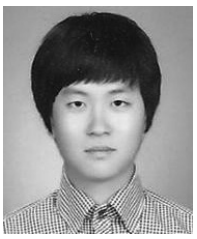
[16] T. J. Ryu, C. J. Kim, J. H. Jeon, S. Y. Lee, & S. W. Park, “A Reference Architecture for Interworking Open Web Application Store”, Korea Computer Congress, pp.478-480, 2013.

류 태 준

e-mail : tjryu@hufs.ac.kr

2008년～현 재 한국외국어대학교 정보통신공학과 학사과정

관심분야 : Database, Web Application, Mobile Application



김 창 준

e-mail : changjun@hufs.ac.kr

2008년～현 재 한국외국어대학교 정보통신공학과 학사과정

관심분야 : Database, Web Application, Mobile Application

전 종 홍

e-mail : hollobit@etri.re.kr

1996년～1999년 한국정보시스템 기술개발연구소 주임 연구원

1999년～현 재 ETRI 표준연구센터 서비스융합표준연구팀 책임연구원

2006년～현 재 TTA국제표준전문가

2008년～현 재 TTA 모바일 웹 실무반 (WG6051) 의장

2011년～현 재 정보과학회 CG&I 소사이어티 운영이사

2012년～현 재 TTA휴대폰 촬영음 실무반(WG7037) 의장

2012년～현 재 모바일 웹 포럼 표준기술위원장

2012년～현 재 W3C Web Application Store CG Chair

2012년～현 재 방통위 HTML5 리더스캠프 코디네이터

관심분야 : Mobile Web, Web Application Technology, Augmented Browsing, Web OS & Future Web Technology Standardization

이 승 윤

e-mail : syl@etri.re.kr

1999년～현 재 ETRI 표준연구센터 책임연구원

2003년～현 재 ETRI 표준연구센터 서비스융합표준연구 팀장

2004년～현 재 TTA국제표준전문가

2004년～현 재 ITU-T SG13 Editor

관심분야 : Next-generation Web Standards, Ubiquitous Web Services Standards, Mobile Web Standards, Web 2.0 Standards

박 상 원

e-mail : swpark@hufs.ac.kr

1994년 서울대학교 컴퓨터공학과(학사)

1997년 서울대학교 컴퓨터공학과(석사)

2002년 서울대학교 컴퓨터공학과(박사)

2002년～2003년 세종사이버대학교 디지털콘텐츠학과 전임강사

2003년～현 재 한국외국어대학교 정보통신공학과 교수

관심분야 : Flash Memory, Embedded Database, Mobile Computing