Caixa Mágica

Aptoide XMLSpecification

Date:	*22/10/2010*
Pages:	*11*
Issue:	*Aptoide - XML Specification*
State:	*Working Document*
Access:	*Public*
Reference:	*Aptoide - XML Specification*

Approved Version: RR			
Name	**Function**	**Signature**	**Date**

Authors and Contributors:			
Name	**Contact**	**Description**	**Date**
Roberto Jacinto	Roberto.Jacinto@caixamagica.pt	Developer	22-10-09
Pedro Fragoso	Pedro.Fragoso@caixamagica.pt	Developer	22-10-09
Paulo Trezentos	Paulo.Trezentos@caixamagica.pt	Technical director	22-10-09

Access List:
Internal Access
External Access

Revision History:			
Version	**Date**	**Description**	**Author**
0.5	15-10-09	First draft of the document	RJ
0.51	22-10-09	Revision	PT
0.52	23-10-09	Revision	RJ

Índice

[AD 1]Introduction

In this document, we are going to give an overview of the XML documents used by *Aptoide* in its communications with a remote server (repository).

We will start by presenting all the mandatory tags and the the optional ones.

You can see a full example of a XML document in the end of this document – section “Full Example”.

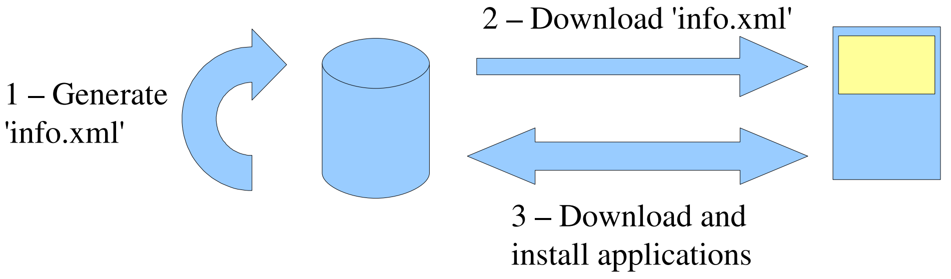
[AD 2]What XML file?

The major communication in *Aptoide* client-server platform is between the client and the remote repositories. Every info from every package present in a given repository must be sent to the client side, as fast and as accurate possible.

This channel of communication is, in *Aptoide*, based on the '**info.xml**' file. This file, which must be present in any repository, is a simple XML file, with the below tags used in accordance to the packages provided by the repository.

Let's first star by understanding how *Aptoide* uses '**info.xml**' file.

[AD 1]Basic working mechanism



The diagram above, shows how a repository normally would behave.

First, we must generate the XML file with all the information about the applications in the repository. Next, we publish that XML file – info.xml – to be retrieved by any *Aptoide* device. Finlay, *Aptoide* uses the information it gathered to download and install applications.

[AD 2]Configuration

For better understanding, we are going to use a imaginary repository. Let's say we have a web accessible server, with the Internet address “http://my.repo.com”.

Accessing that address, will put us in our main directory, in this example *htdocs*. Now, to create a *Aptoide* repository, you just need to put the **info.xml** file in the directory you will give away as your repository address.

If we create the directory '*arepo*' and place it in our main directory – *htdocs* – we would put it in there. The address that should be supplied to access your repository would be: 'http://my.repo.com/arepo'.

[AD 3]Generating the XML file

There are a number of ways one might create the info.xml file. From asking directly to the user o supplies the application, to more scripted ways, were information is gathered from the *.apk file itself.

That's not the scoped of this document, and the only consideration here, is that the XML file must be placed in the same directory as all applications apk's.

[AD 4]Downloading XML file

The first thing *Aptoide* must do to use a new repository, is to retrieve the info.xml file from it. All the information needed to download and install applications is there.

As said above, the file must be in the same directory that the Internet address of it access.

After downloading the file, *Aptoide* will parse all the information on it, and generate a list with all the applications information. Be aware that if an application have an icon, *Aptoide* will connect to the repository again to retrieve it.

[AD 5]Downloading applications

Wen a user tries to download an application, *Aptoide* tries to get it based on the information included in the XML file. The applications can be anywhere as long as they are inside the same repository – in other words, they can be in a different directory from the **info.xml** file. To understand better how it works, read the **<path>** tag below.

[AD 3]XML tags overview

The XML file, as most, starts with the standard informations header. Just a basic one is needed as *Aptoide* don't parse any of this information.

Next, the opening tag is used: **<apklst>**. This is the enclosed tag for the entire file.

Next we must delimit every application. We use the tag **<package>** for that purpose.

As an example, imagine we have two applications in the imaginary repository X, the 'info.xml' file must have the below skeleton:

```
<?xml version="1.0" encoding="UTF-8"?>
<apklst>
  <package>
    /* all information about application 1 goes here*/
  </package>
  <package>
    /* all information about application 2 goes here*/
  </package>
</apklst>
```

[AD 1] Mandatory tags

The only information *Aptoide* needs to work is the application package - which is unique to every application, and therefore identifies uniquely the application – and the path to the application *.apk in the repository.

- **Application package:** This string identifies the application and is unique. It's enclosing tag is **<apkid>**.

- **Application file path:** This is the relative path where the *.apk file for this application is in the remote repository. It's enclosing tag is **<path>**.

[AD 2] Optional tags

This tags are used to transmit additional information about applications. Although they are not mandatory – *Aptoide* can operate with just the tags presented above – they are very useful for improving the user experience while operating *Aptoide*.

- **Application name:** The truth is, *Aptoide* don't really need to know the application name. A application is identified by it's package name. Even so, it's simpler to the user to have a more human-readable way to identify them. The enclosing tag for this property is **<name>**. When no value is passed, *Aptoide* presents the package name as the application name.

- **Version:** The version of the application present in the repository is useful for upgrading issues. If no version is specified, *Aptoide* declares it as version 0.0. The enclosing tag is **<ver>**.

- **Application icon:** Not every application has an icon. If it has, the path to the icon file in the server must be passed. If there is no icon, or that information is not passed, *Aptoide* uses the standard Android icon. The enclosing tag is **<icon>**.

- **Category:** Applications can be joined in categories. For example, an application can be a game, an office application or other things. This information is used to modify the way the list of applications appear on *Aptoide*. If no category information is available, it will be put on “other” category. The tag for this is **<catg>**.

- **Date:** As category, date refers to the date where the application first became available in the repository. In the same manner as above, this information might be used for user interface only, were the user might want to display applications in a “most recent first” order. The enclosing tag is **<date>**.

[AD 4]Example of info.xml

For a better understandment we provide an overview example.

Let's say we have the repository 'http://aptoide.repo.com”. In there, we have two applications, “App1” and “App2”. We also have a directory called “icons” where icons are stored.

General applications information:

	Application 1	Application 2
Package	ex.app2.com	ex.app2.com
Path	App1.apk	App2.apk
Name	- - -	App 2
Version	- - -	'2.1.1
Icon	- - -	/icons/app2.ico
Category	- - -	Game
Date	- - -	21-01-10

As shown in the table above, application 1 only have the basic information available. Has for application 2, there is every thing we might want to know. The 'info.xml' file for this repository would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<apklst>
  <package>
         <apkid>ex.app1.com</apkid>
         <path>App1.apk</path>
  </package>
  <package>
         <apkid>ex.app2.com</apkid>
         <path>App2.apk</path>
         <name>App 2</name>
         <ver>2.1.1</ver>
         <icon>/icons/app2.ico</icon>
         <catg>Game</catg>
         <date>21-01-10</date>
  </package>
</apklst>
```

[AD 5]Quick reference guide

	Mandatory?	Default Value	Information
Package	yes	- - -	The package that identifies the application
Path	yes	- - -	The path to the *.apk
Name	No	Equal to package	The human-readable application name
Version	No	0.0	Application version
Icon	No	Standard Android icon	The path to the icon file
Category	No	Others	The category of the application
Date	No	- - -	The date the application entered the repository