Pengenalan OTA (*over-the-air*) *Provisioning* pada Aplikasi J2ME

Faisal Wiryasantika
faisal@winwinfaisal.info
http://faisal.winwinfaisal.info

Belakangan mungkin anda semakin sering mendengar terminologi OTA seiring dengan makin meluasnya telepon seluler yang *Java enabled*. Terminologi *over-the-air* (OTA) *provisioning* merupakan sebuah akronim yang berarti sebuah *device* memiliki kemampuan untuk men-*download* dan sekaligus meng-*install* sebuah aplikasi melalui suatu jaringan *wireless*. Jadi ketika berada di bandara atau di stasiun KA sambil menunggu keberangkatan pesawat/ KA, Anda bisa men-*download* sebuah games/ aplikasi bisnis MIDlet dari sebuah portal secara OTA, asyik bukan... Mudah-mudahan dengan adanya tulisan ini, Anda tidak bingung lagi jika mendengar terminologi OTA tersebut.

Pada tulisan ini akan dijelaskan tentang apa itu OTA, bagaimana cara kerjanya dan apa implikasinya bagi *user*. Sekilas pula kita tengok pula dari sisi *server* penyedia jasa OTA tersebut (bagaimana sebuah *web site* memiliki kapabilitas sebagai *download server*). Tulisan ini di fokuskan pada OTA *provisioning* untuk sebuah aplikasi MIDlet tetapi mungkin secara konsep dapat meng-cover juga untuk *provisioning* jenis aplikasi *wireless/ content* lainnya.

OTA *Overview*

Dari sisi *mobile client/ user*, konsep OTA dipandang sebagai suatu pencarian aplikasi yang diminati oleh *client* di dunia *web* dan melakukan inisiatif untuk melakukan proses *download* melalui jaringan *wireless*. Untuk lebih jelasnya silahkan perhatikan gambar 1.

Gambar 1. *Provisioning Achitecture*

Tampak bahwa ada empat buah *node* yang berpartisipasi dalam proses OTA ini :

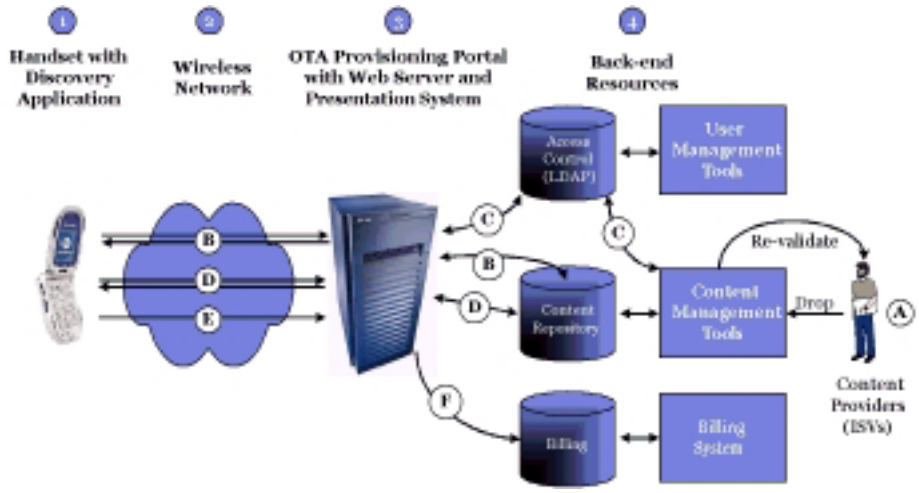
- ***Client Device with "Discovery Application"*** – *Client device* haruslah memiliki *software* yang memperbolehkan *user* untuk menemukan suatu aplikasi pada suatu *provisioning portal* dalam suatu jaringan dan dapat melakukan pemilihan aplikasi mana yang akan di-*download*. *Software* ini disebut sebagai *Discovery Application* (DA). DA bisa berupa *browser based* atau *native application*, sampai sejauh ini masih menggunakan *common provisioning protocol* yaitu HTTP.

- ***Wireless Network*** – Ini merupakan jaringan *wireless* maksudnya yang dapat berupa sebuah jaringan radio dan sebuah WAP *gateway*.

- ***Download Server*** – Juga disebut sebagai *provisioning portal* atau *server* atau *vending machine*. Secara tipikal *server* ini menjalankan sebuah *web server* dan memiliki akses pada suatu *content repository*. Dengan kata lain bahwa suatu *web site* dapat berlaku sebagai *provisioning portal* yang memiliki dua buah fungsi yaitu menyediakan menu dengan format khusus (biasanya ditulis dalam WML atau HTML) yang memiliki *list application* yang tersedia untuk di-*download* dan menyediakan akses pada aplikasi yang dimaksus. Sebuah *server* dapat melayani satu atau banyak OTA *protocol*.

- ***Content Repository*** – Sebagai tempat (*persistent storage*) dari seluruh aplikasi *descriptor* (jad file) dan aplikasinya itu sendiri (jar file).

Dalam kenyataannya OTA bukanlah hal yang sederhana. Sebuah OTA *provisioning system* secara tipikal mengutamakan pada publikasi dan manajemen *content*, *access control*, *installation* (*and upgrading*) dari suatu aplikasi, dan tracking dari suatu aplikasi (hal ini untuk keperluan *billing*). Gambar 2 akan coba menjelaskan hal yang tak sederhana tersebut.



Gambar 2. Detail sistem OTA *provisioning*

Seperti yang Anda lihat bahwa system OTA tak sesederhana Gambar 1 pada kenyataannya dimana proses dibelakang layar lebih banyak terlibat daripada yang dibayangkan dari perspektif *client* :

- ***Content Management*** – *Server side software* me-*manage repository*/ penyimpanan yang secara tipikal berupa sebuah basis data dan memiliki *content versioning* bahkan sangat mungkin memiliki kemampuan agar sebuah *content provider* menyimpan aplikasi secara remote dengan *restriction* tertentu (contoh *software content management* adalah **Appload Server** – buatan **Siemens** tampat pada Gambar 3).

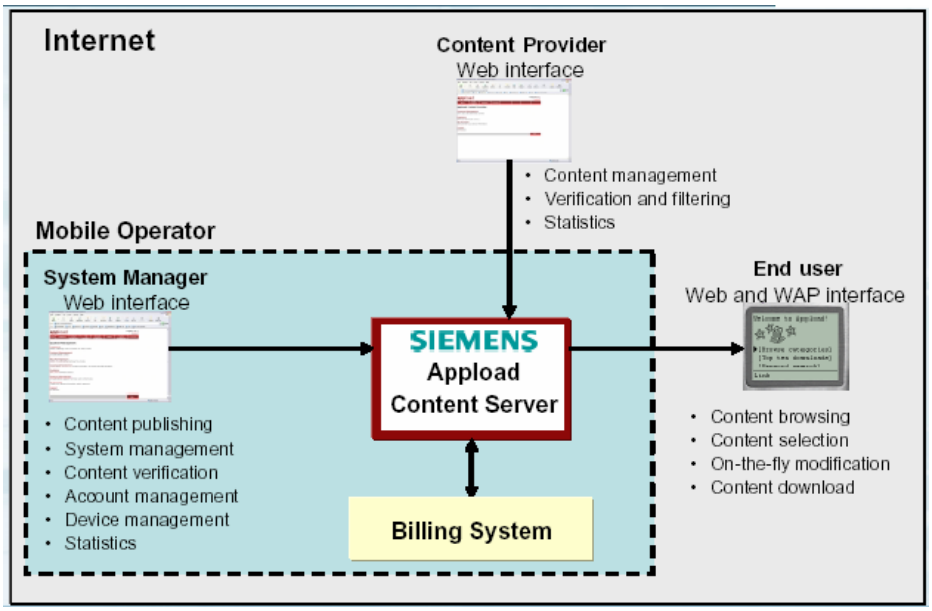
- ***Content Discovery*** – Seperti yang Anda lihat bahwa user mengarahkan DA-nya pada *download portal* yang memiliki akses pada *application repository* dan menyediakan menu aplikasi dan *content* apa saja yang tersedia.

- ***Authenticate*** – Jika *provisioning server* men-*support* sebuah modul otentikasi maka seorang *user* akan diberikan otentikasi sebelum dapat mengakses *repository*.

- ***Application Retrieval and Installation*** – Saat otentikasi telah selesai maka proses *download application* terbagi atas dua proses yang di *handle* oleh *Application Management Software* (AMS) sebagai manajer bagi proses *download, installation, execution*, dan *removal of applications*. Jika jad file ada maka AMS akan men-*download* aplikasi tersebut dari *server repository*. Berdasarkan jad file tersebut maka AMS otomatis akan men-*download* jar file (aplikasinya). Jika dibutuhkan *user* akan di re-otentikasi. Setelah itu bilamana aplikasi telah di-*download* seluruhnya maka dilanjutkan proses instalasi secara otomatis oleh *mobile device*.

- ***Confirmation*** – AMS akan mengirimkan konfirmasi yang mengindikasikan apakah proses instalasi berjalan sukses atau tidak.

- ***Tracking*** - Status konfirmasi dapat digunakan pula sebagai *tracking* pengguna suatu aplikasi (untuk keperluan *billing*). Sebuah *billing system* hampir selalu terintegrasi dalam *provisioning server*.



Gambar 3. *Siemens Appload Content Server*

Sekarang Anda telah memahami secara umum OTA *application provisioning*. Berikut kita akan turun pada hal yang lebih detail, pertama pada sisi *client* dimana kita akan melihat MIDlet *provisioning* secara spesifik dan kemudian dilanjutkan dengan *server side*.

Overview MIDP OTA Spesification

MIDP OTA *spesification* mendefinisikan bagaimana suatu MIDP devices berinteraksi dengan *provisioning server*, termasuk bagaimana *device* tersebut men-*download* aplikasi MIDP/ MIDlet tersebut. MIDP OTA mendefinisikan bahwa sebagai inisiator dari proses OTA adalah *user* dan bukanlah diinisiasi oleh *server* (*push* OTA *mechanism*).

Versi pertama dari MIDP OTA *spesification* mulai ada setelah MIDP 1.0 *spesification* muncul. MIDP OTA *spesification* versi pertama itu belumlah merupakan spesifikasi yang sebenarnya karena hanyalah berupa rekomendasi praktis saja tetapi hal itu sudah menyediakan pijakan yang baik bagi *protocol* OTA. Hingga saat ini telah banyak *device* yang *support* terhadap MIDP 1.0 *recommendation* itu.

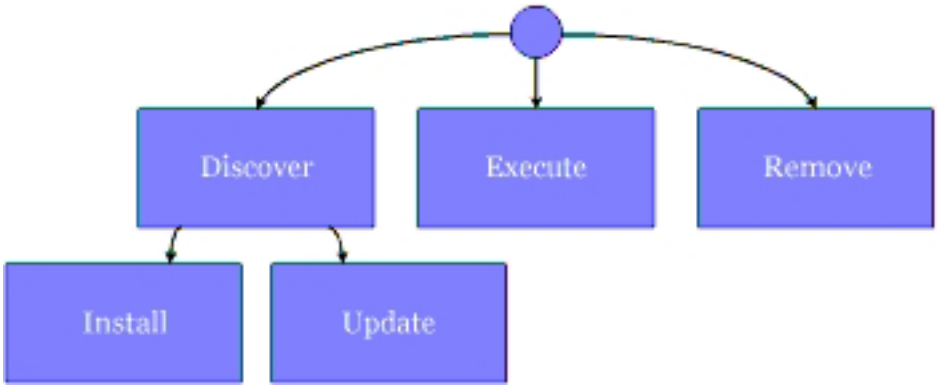
Sedangkan pada MIDP 2.0 dilakukan pengembangan dari MIDP 1.0 dimana MIDP 1.0 menjadi bagian dari versi 2.0 tersebut. Kini spesifikasi yang ada dalam MIDP 1.0 telah menjadi bagian standar dunia sehingga seluruh MIDP *device* yang akan muncul saat ini maupun dikemudian hari haruslah secara konsisten untuk memiliki kemampuan OTA secara konsisten, sebagaimana telah digariskan dalam standar. Dan ini merupakan keuntungan besar dimana kita dapat mendistribusikan aplikasi pada banyak jenis *mobile devices* dengan satu cara yang sama yaitu OTA.

Versi kedua dari MIDP juga relatif sama walau ada sedikit perbedaan yaitu pada *property* dari jad file dimana ada suatu nilai *property* digunakan oleh AMS sebagai *report* manakala MIDlet telah dihapus dan akan terjadi *stringent check*/ pemeriksaan yang seksama manakala terjadi *updating* MIDlet. Penggunaan *cookies* pun juga telah digantikan oleh URL *rewriting*. MIDP OTA spesification terbagi atas beberapa hal berikut :

- ***Expected device functionality (in support of OTA)*** – Terdapat sejumlah *functionality* tertentu pada MIDP-*compliant devices* yang harus untuk mendapatkan layanan OTA yaitu :

  - *Support* terhadap HTTP versi 1.1 atau sebuah *protocol* lainnya yang mengimplementasikan *functionality* HTTP 1.1, termasuk support terhadap HTTP Basic Authentication (RFC 2617).
  - Memiliki sebuah *Discovery Application* (DA).
  - Memiliki AMS. Nama lain untuk J2ME-*enabled devices* adalah *Java Application Manager* (JAM).

- ***The OTA provisioning life-cycle*** – Mari sekarang kita melihat lebih detail saat *discovery*, *installation, updated, execution*, dan *removal of application* seperti tampak dalam Gambar 3.



Gambar 4. Siklus OTA *Provisioning*

  - *Discovery* - Siklus hidup ini bermula ketika *user* menginstruksikan pada *discovery application* (DA) untuk mencari aplikasi yang disenangi pada sebuah *provisioning portal* dalam jaringan internet. Ketika ditemukan maka *user* akan memilih suatu aplikasi untuk di-*download* dan selanjutnya di-*install*. Setelah sukses meng-*install* maka aplikasi dapat di eksekusi ataupun di *update* atau jika bosan dapat di *remove* dari *device*. Seluruhnya hal ini dikendalikan oleh AMS. Secara keseluruhan siklus dari OTA ini dapat diamati oleh kita (*network protocol*, UI *detail*, DA, dan AMS *implementation*). Sedangkan hal yang tidak tampak oleh kita adalah seputar konfigurasi seperti terjadinya perubahan pada jad file, konfigurasi pada DA untuk menuju ke *server* yang dituju, dan konfigurasi *web server* untuk menyediakan *content* yang sesuai/ *appropriately*. Pada beberapa *devices* (termasuk didalamnya Nokia dan Siemens), fungsi DA diasosiakan dengan WAP *browser* tetapi bisa saja DA dilakukan oleh *native application*. Bagaimana sebuah DA dikonfigurasi untuk kepentingan OTA adalah merupakan hal yang *vendor-spesific*. DA diinisiasi dengan proses HTTP *request* untuk mendapatkan *list* dari aplikasi yang tersedia, sebagai contoh :

```
GET /ota/index.html HTTP/ 1.1
Host: www.j2medeveloper.com:80
:
:
```

    Respon dari sebuah *list* atau menu dari *available application* dengan sebuah *description* dan sebuah URL untuk setiap masing masingnya. Berikut adalah contoh respon dari berisi menu dalam WML :

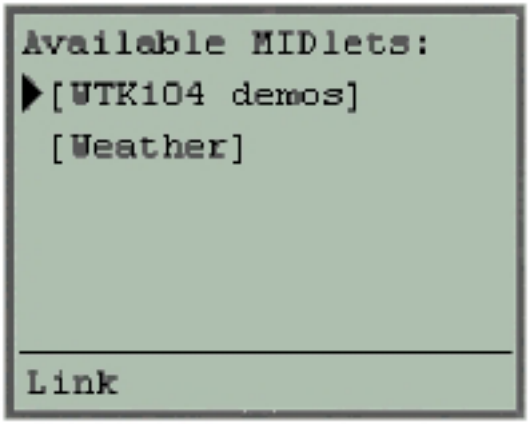
HTTP response with application menu in WML:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.wap.wml, text/x-wap.wml, text/vnd.wap.wml
Content-Type: application/vnd.wap.wml, text/x-wap.wml, text/vnd.wap.wml
Content-Length: .....
:
:
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML
1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta forua="true" http-equiv="Cache-Control"
content="must-revalidate, no-store"/>
</head>
<template>
<do type="prev" label="Back"><prev/></do>/
</template>
<card title="Available MIDlets" id="main">
<p>Available MIDlets:</p>
<p align="left">
<a href="<b>http://www.j2medeveloper.com/ota/demos.jad</b>">WTK104 demos</a></p>
<a href="<b>http://www.j2medeveloper.com/ota/weather.jar</b>">Weather</a></p>
</card>
</wml>
```

HTTP response with application menu in HTML :

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: .....
:
:
<html>
<head><title>J2MEDeveloper.com MIDlet Menu</title></head>
<body>
<p>Welcome to J2MEDeveloper.com</p>
<p>
Available MIDlets: <br>
<a href="www.j2medeveloper.com/ota/demos.jad">WTK104 demos.jad</a>
<a href="www.j2medeveloper.com/ota/weather.jar">Weather</a>
</p>
Enjoy!
</body>
</html>
```

Gambar 5 memperlihatkan menu yang dapat dipilih oleh *user*.



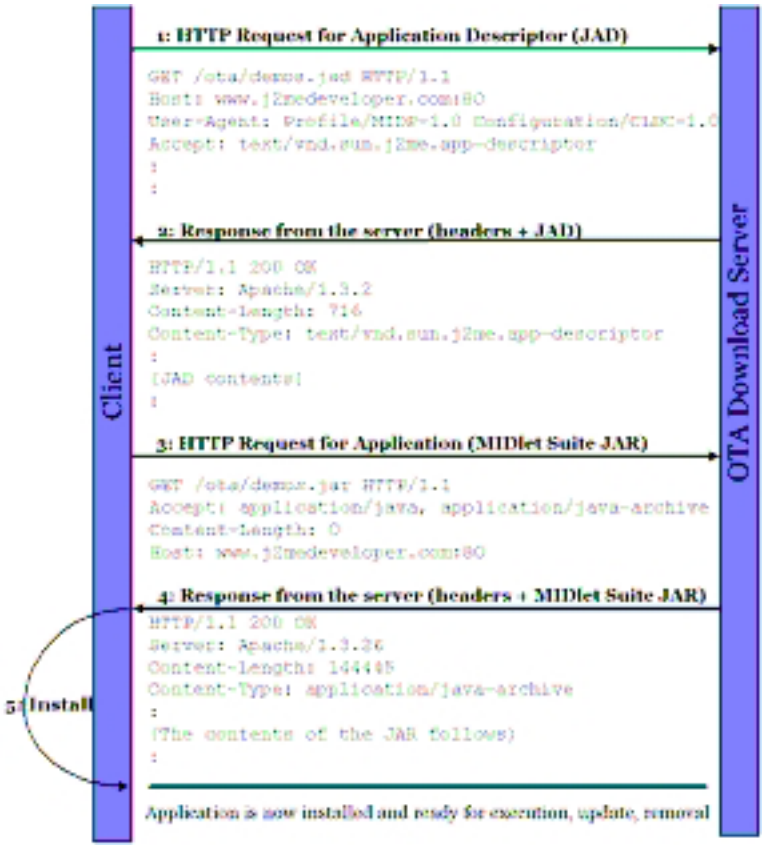
Gambar 5. *Menu of available Applications*.

Pada *source code* WML diatas URL dapat tertuju pada jad URL maupun pada jar URL tetapi perlu diingat bahwa MIDP OTA 1.0 *spesification* (maupun versi 1.0.4 dari WTK) hanyalah support untuk jad URL sedangkan pada MIDP 2.0 dapat support baik jad maupun jar URL. Perlu pula dicatat bahwa merkipun MIDP 2.0 *spesification* tidak mengharuskan *developer* untuk menyediakan jad file tetapi sangat direkomendasikan untuk menyediakan jad file tersebut karena jad tersebut membantu AMS mendeterminasikan apakah *device* memiliki *resources* yang cukup atau tidak untuk meng-*install* aplikasi tersebut.

- *Installation and Update* – Proses instalasi terdiri atas *download* MIDlet jar dan membuat jar file tersebut dapat dieksekusi. Sedangkan untuk proses *update*, AMS akan melakukan komparasi *version number* dan akan memberitahu user apakah yang ia miliki versi yang lebih tua, lebih baru, atau sejenis. Hal ini juga memperhitungkan isi dari *Recors Management System* (RMS) sebagai *persistent storage*. MIDP 2.0 menambahkan beberapa *security checks* untuk kepentingan konsistensi RMS tadi.

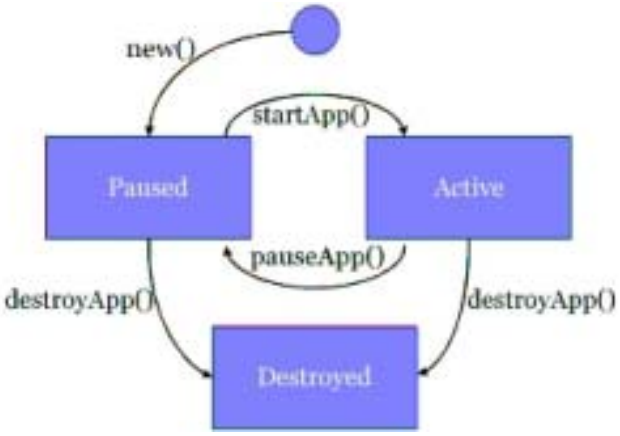
```
JAD File:
Created-By: 1.3.1 (Sun Microsystems Inc.)
MIDlet-1: Colors, /icons/ColorChooser.png, example.chooser.Color
MIDlet-2: Properties, /icons/App.png, example.PropExample
MIDlet-3: Http, , example.http.HttpTest
MIDlet-4: FontTestlet, , example.fonts.FontTestlet
MIDlet-5: Stock, /icons/Stock.png, example.stock.StockMIDlet
MIDlet-6: Tickets, /icons/Auction.png, example.auction.TicketAuction
MIDlet-7: ManyBalls, /icons/ManyBalls.png, example.manyballs.ManyBalls
MIDlet-Description: Technical demonstration programs for MIDP
MIDlet-Jar-Size: 144445
MIDlet-Jar-URL: http://www.j2medeveloper.com:80/ota/demos.jar
MIDlet-Name: SunSamples - Demos
MIDlet-Vendor: Sun Microsystems, Inc.
MIDlet-Version: 1.0.3
Manifest-Version: 1.0
```

Dengan asumsi file jar seperti diatas maka interaksi antara *client* dan *server* dalam proses *download* jad dan jar file akan tampak seperti Gambar 6 berikut :



Gambar 6. OTA *client-server interaction*

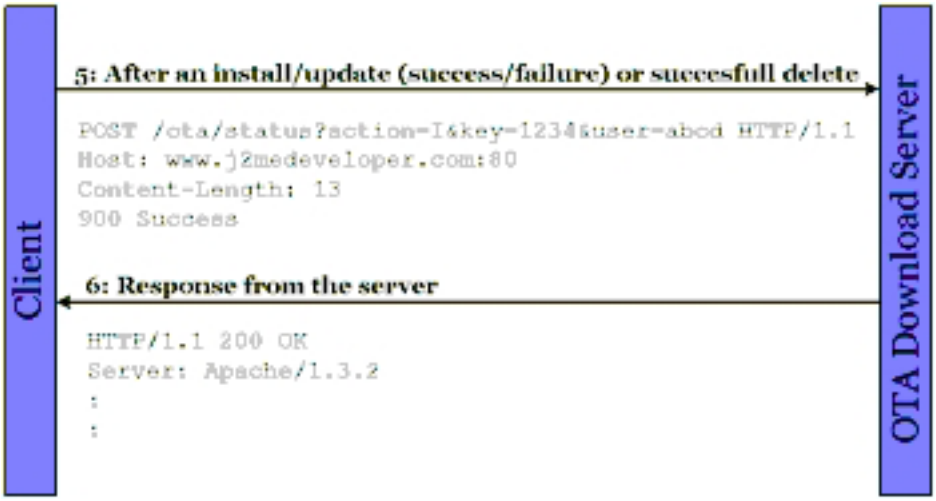
- *Execution* – Proses eksekusi terjadi setelah aplikasi MIDlet terinstal pada *device* Anda dan selanjutnya siap untuk dieksekusi (dari mulai *start application* hingga *destroy application*). Pada Gambar 6 Anda akan melihat siklus hidup dari sebuah MIDlet.



Gambar 7. Siklus hidup MIDlet.

- *Removal* – AMS memperbolehkan *user* untuk me-*remove* suatu aplikasi termasuk RMS yang berasosiasi dengan aplikasi tersebut. Setelah me-*remove* aplikasi, AMS akan melaporkan kode satus 912 kepada server (lihat *table status report*).

- *Status Report – Provisioning server* dapat mengambil manfaat dari status report ini untuk *tracking* pengguna aplikasi (baik untuk keperluan *billing* maupun untuk skala prioritas dari *content repository*). Jika status *reporting* dari jad *property* telah didefinisikan maka AMS akan melaportkan pada *provisioning server* via HTTP POST. *Notification* JAD *property* tampak pada table 2 (contoh : untuk instalasi atau *updated*, AMS menggunakan *value* dari JAD *entry* MIDlet-Install-Notify). Gambar 7 memperlihatkan interaksi yang terjadi ketika *client* melaporkan status :



Gambar 8. *Status Report Interaction*

Table 1 - *Status codes must supported by* AMS

Status Code	Description (Status Message)	MIDP 1.0	MIDP 2.0
900	Success	X	x
901	Insufficient Memory	X	x
902	User Cancelled	X	x
903	Loss of Service	x	x
904	JAR size mismatch	x	x
905	Attribute Mismatch	x	x
906	Invalid Descriptor	x	x
907	Invalid JAR		x
908	Incompatible Configuration or Profile		x
909	Application authentication failure		x
910	Application authorization failure		x
911	Push registration failure		x
912	Deletion Notification		x
913	Required package not supported by the device		x

- ***Changes to the Java Application Descriptor in support of* OTA** – Agar dapat mendukung OTA maka Anda harus melakukan sedikit perubahan pada jad file. Ada empat jad properties yang berasosiasi dengan OTA seperti yang terlihat pada table 2.

Table 2 - JAD *properties for* OTA

JAD property	Description (Status Message)
`MIDlet-Jar-URL`	This property indicates the location (URL) of the MIDlet suite (JAR file).
`MIDlet-Install-Notify`	This property indicates the location (URL) for posting installation (and update) status. You can use URL rewriting to encode tracking information for the MIDlet suite. The URL can't be longer than 256 UTF-8-encoded characters.
`MIDlet-Delete-Notify` (since MIDP 2.0)	This property indicates the location (URL) for posting removal status. You can use URL rewriting to encode tracking information for the MIDlet suite. The URL can't be longer than 256 UTF-8-encoded characters.
`MIDlet-Delete-Confirm`	The AMS uses this confirmation prompt when removing a MIDlet suite.

Mari kita lihat jad file berikut :

JAD File:
```
Created-By: 1.3.1 (Sun Microsystems Inc.)
MIDlet-1: Colors, /icons/ColorChooser.png, example.chooser.Color
MIDlet-2: Properties, /icons/App.png, example.PropExample
MIDlet-3: Http, , example.http.HttpTest
MIDlet-4: FontTestlet, , example.fonts.FontTestlet
MIDlet-5: Stock, /icons/Stock.png, example.stock.StockMIDlet
MIDlet-6: Tickets, /icons/Auction.png, example.auction.TicketAuction
MIDlet-7: ManyBalls, /icons/ManyBalls.png, example.manyballs.ManyBalls
MIDlet-Description: Technical demonstration programs for MIDP
MIDlet-Jar-Size: 144445
MIDlet-Jar-URL: http://www.j2medeveloper.com:80/ota/demos.jar
MIDlet-Install-Notify:
http://www.j2medeveloper.com:80/ota/status?action=I&key=1234&user=abcd
MIDlet-Delete-Notify:
http://www.j2medeveloper.com:80/ota/status?action=D&key=1234&user=abcd
MIDlet-Delete-Confirm: Are you sure you want to delete the WTK demos?
MIDlet-Name: SunSamples - Demos
MIDlet-Vendor: Sun Microsystems, Inc.
MIDlet-Version: 1.0.3
Manifest-Version: 1.0
```

Tampak bahwa pada *listing* diatas bahwa Anda dapat meng-*apply* URL *rewriting* pada properties MIDlet-Install-Notify dan MIDlet-Delete-Notify untuk meng-*encode* informasi yang dibutuhkan untuk *tracking application*. Dimana parameter dapat ditambahkan pada URL untuk membantu *provisioning server* melakukan *tracking application*. Pada contoh *listing* diatas : "action=I&key=1234&user=abcd", yang menandakan bahwa "*application installed*". Sedangkan *key* menunjukkan *user* yang menginstall aplikasi.

- ***Interactions between the* AMS *and the provisioning server*** – OTA dapat diimplementasikan dengan sangat sederhana menggunakan *web server* standar tetapi dengan platform J2EE menjadikan sebuah *server* dapat menjadi *complex provisioning server*.

  Dalam kesempatan ini hanya akan dibahas *minimal configuration* untuk *provisioning server* sedangkan untuk pembahasan *J2EE application server based configuration* akan dibahas pada kesempatan lain. Minimal sebuah OTA *configuration* membutuhkan *web server* yang

dikonfigurasi secara *properly*. Konfigurasi *server* terdiri atas *defining appropriate* MIME *types* untuk jad dan jar file seperti dalam table 3.

Table 3 - *Required* MIME *types for* OTA

File Extension	MIME Type
JAD	`text/vnd.sun.j2me.app-descriptor`
JAR	`Application/java-archive`

Download server mengidentifikasi file diatas dengan cara men-*setting* HTTP *Content-Type header* untuk jad menjadi text/vnd.sun.j2me.app-descriptor dan jar menjadi application/java-archieve. Sedangkan AMS menggunakan *Content-Type header* selama proses *application discovery* dan *installation phases*.

Bagaimana cara mengonfigurasi nilai ini pada *web server*? Pada Apache Tomcat, Anda dapat meng-*updated global configuration* file yang dapat ditemukan pada $CATALINA_HOME/conf/web.xml atau Anda dapat meng-*upadate* file web.xml dengan tambahan beberapa line berikut :

```
<!-- JAD file -->
<mime-mapping>
   <extension>jad</extension>
   <mime-type>text/vnd.sun.j2me.app-descriptor</mime-type>
</mime-mapping>

<!--JAR file -->
<mime-mapping>
   <extension>jar</extension>
   <mime-type>application/java-archive</mime-type>
</mime-mapping>
```