Diplomarbeit

Sicheres Applikationsmanagement für mobile Endgeräte

ausgeführt zum Zwecke der Erlangung des akademischen
Grades eines Diplom-Ingenieurs
unter der Leitung von

o. Univ. Prof. Dipl.-Ing. Dr.techn. Dietmar Dietrich
und
Dipl.-Ing. Albert Treytl

am
Institut für Computertechnik
der
Technischen Universität Wien

von
Gfader Peter
Matr. Nr.: 9826521
Schönburgstrasse 2/1, 1040 Wien

Datum

Unterschrift

Für meine Eltern.

Kurzfassung

Im Rahmen dieser Diplomarbeit wurde eine Methode bzw. ein System zur sicheren und autorisierten Distribution von Applikationen für Java-2-Micro-Edition-fähige (J2ME) mobile Endgeräte entwickelt. Dabei wurde nur die Applikationsdistribution über das Mobilfunknetzwerk betrachtet. Dieser Vorgang wird als „Over The Air (OTA) User Initiated Provisioning" bezeichnet. Diese standardisierte Verteilung von Applikationen auf die entsprechenden Endgeräte bringt einige Sicherheitsmängel mit sich, die diese Arbeit zu beseitigen versucht.

Ziel des entwickelten Systems, war die Integrität der Applikationen und die Vertraulichkeit der Transaktionen zu schützen. Deshalb darf kein unbefugter Benutzer Zugriff auf die übertragenen Applikationen erhalten. Ein weiterer wichtiger Aspekt ist die Authentifizierung der verschiedenen Teilnehmer im System, die eine Authentizität innerhalb des Systems gewährleistet. Weiters soll die Beschaffung von Applikationen sowohl unverbindlich anonym, als auch bindend für registrierte Benutzer möglich sein.

Für die Entwicklung des Projekts zu dieser Arbeit wurde zum Teil die JSR124-Referenzimplementierung verwendet. Die gesamte Implementierung wurde auf der Java-2-Enterprise-Edition-Plattform durchgeführt, auf der die verschiedenen Server für Applikationsdistribution, Bearbeitung von J2ME-Notifications und Authentifizierung entworfen, erstellt und getestet wurden.

In dieser Arbeit wurde ein verteiltes System realisiert, das die grundlegenden Anforderungen, der Authentifizierung der Teilnehmer und der Authentizität des Systems gewährleistet. Sämtliche Sicherheitsbedürfnisse konnten nicht erfüllt werden, da ein Eingreifen und Ändern des Standards für Applikationsdistribution nicht erwünscht und möglich war. Bei der Entwicklung für ein spezielles Endgerät wäre zusätzlich die Haupteigenschaft von Java (Applikationen geräteunabhängig ausführen) nicht mehr gegeben gewesen. In den nächsten Versionen von J2ME sollten diese noch bestehenden Probleme beseitigt sein, da entsprechende Sicherheitsmaßnahmen von den Standardisierungsgremien geplant sind.

Schlagwörter

JAVA, J2ME, J2EE, JSR124, Java 2 Micro Edition, Over the Air, MIDP, CLDC, Mobile Kommunikation, Sicheres Applikationsmanagement

Abstract

This work presents a system for authorized and secure distribution of applications for Java 2 Micro Edition (J2ME) enabled mobile phones. Due to the lack of security measures in the standardized J2ME distribution mechanisms – Over the Air (OTA) User Initiated Provisioning –, the aim of this work is to implement and analyze a secure distribution system.

The main target of the developed system is to protect the integrity of the applications and to keep transactions confidential within the whole system. No unauthorized user should get access to the applications or to the transferred data. Another important issue is the authentication of the different participants, which guarantees authenticity within the system. Additionally the download of applications should be allowed for anonymous, noncommittal or binding distribution including user registration.

In the development of the project the JSR124 reference implementation was integrated. The whole system was implemented on the Java-2-Enterprise-Edition-Platform, which forms the base for the different servers responsible for application distribution, processing of J2ME notifications and authentication.

The result of this work is a distributed system, which fulfills the basic requirements of authentication of participants and system authenticity. It was not possible to fulfill all security needs, because this would require to change the Standard OTA Download, which was not an option. However, in the next versions of J2ME these security problems should be eliminated, because corresponding security procedures are already planned to be included in the J2ME standards.

Keywords

JAVA, J2ME, J2EE, JSR124, Java 2 Micro Edition, Over the Air, MIDP, CLDC, Mobile Communication, Secure Application Distribution

Inhaltsverzeichnis

1 Einleitung

Der Anfang ist die Hälfte des Ganzen.
Aristoteles

Lothar Roitner, Vorstandssprecher des Forums Mobile Kommunikation, sagte im März 2002 folgendes über den österreichischen Mobilfunkmarkt: *„In Österreich sehen 55% der Menschen für sich eine steigende Bedeutung der Mobilkommunikation. Da fast drei von vier ÖsterreicherInnen diese Technologie verwenden, geben sie damit ein klar positives Signal für den Nutzen und die Qualität der Mobiltelefonie."* [1]

In den vergangenen Jahren erlebten die Hersteller mobiler Kommunikationsgeräte einen buchstäblichen Boom, der auf die zunehmenden Mobilitätsbedürfnisse der Menschen und der immer fortschreitenden Miniaturisierung von Halbleiterchips zurückzuführen ist. Die Palette der mobilen Geräte, erstreckt sich von einfachen Ein-Weg-Pagern bis zu multifunktionalen Mobiltelefonen, wobei die Menge der möglichen Features immer mehr zunimmt. Mit der rasanten Entwicklung der Hardware der mobilen Endgeräte, wurden auch die Anforderungen an die Software immer größer. Wenn die Software die Funktionen des Geräts bzw. der Hardware nicht nutzt, ist das Gerät wenig nutzbar, und verliert somit an Notwendigkeit und Akzeptanz.

Das Problem, das diese Entwicklung mit sich gebracht hat, ist, dass die verschiedenen Gerätehersteller keine standardisierte Plattform hatten und haben, auf der sie aufbauen konnten. Das heißt jeder Hersteller musste seine Software spezifisch für seine Geräte entwickeln. Am Beispiel des Taschenrechners am mobilen Endgerät wird diese Problematik am deutlichsten. Beinahe jedes Gerät besitzt einen Taschenrechner zum schnellen Berechnen von einfachen mathematischen Standardfunktionen wie zum Beispiel Addition, Subtraktion, Division und Multiplikation. Jeder Gerätehersteller muss für seine Hardware- und Betriebssystemplattform diesen Taschenrechnern implementieren. Von den Kosten für Entwicklung abgesehen, entstehen bei dieser Hardware- und Softwareentwicklung, für den Hersteller eine Menge von zusätzlichen Nachteilen:

[1] Zitat aus: Forum Mobile Kommunikation [FMK01]

So ist mit jeder Hardwaremodifikation eine Softwaremodifikation nötig. Weiters ist nach dem Verkauf des Geräts, ein Update der Software nicht mehr ohne weiteres möglich, und das Hinzufügen von neuen Applikationen zu den bestehenden Applikationen auf dem Gerät ist im Vorhinein nicht vorgesehen.

„Write once, run anywhere": „Software schreiben und überall, auf jeder möglichen Hardwareplattform, ausführen". Mit diesem Ausspruch erobert Suns Programmiersprache Java seit Mitte der 90er Jahre erfolgreich die Landschaft etablierter Programmiersprachen. Hinter der Java Technologie steckt die Idee, Software plattformunabhängig zu entwickeln und ausführen zu können.

Die Java Technologie existiert bereits für zahlreiche Hardware-Plattformen und mit der „Java 2 Micro Edition" (J2ME) wird Java auf bislang nicht unterstützten Plattformen – wie Mobiltelefonen, Smartphones, PDAs, Spielkonsolen oder Set-Top-Boxen – einheitlich verfügbar gemacht. Die Vorteile der gesamten Java Technologie gelten auch für J2ME: Applikationen sind unabhängig vom Gerätehersteller, vom mobilen Gerät selber und vom Netzbetreiber. Das heißt eine J2ME-Applikation kann auf jedem J2ME-fähigen Endgerät ausgeführt werden.

Im Jahre 2002 wurde die Anzahl der benutzten J2ME-fähigen Kleingeräte auf 68 Millionen weltweit geschätzt[2], und bis zum Jahre 2007 sollen bis zu 100% der mobilen Kleingeräte J2ME-fähig sein[3]. Im Februar 2003 standen bereits 100 verschiedene J2ME-fähige Geräte von über 20 verschiedenen Herstellern zum Verkauf bereit[4]. An diesen Zahlen erkennt man den rasanten Einzug der Java-Technologie in die Welt der mobilen Endgeräte.

Die Anwendungen von J2ME, die diese Java-Edition so interessant und wichtig für die Zukunft machen, kann man in folgende Kategorien zusammenfassen:

- Spiele: Netzwerk-Spiele für mehrere Benutzer und Einzelplatzspiele
- Nachrichten und Messaging Services: Emailprogramme, Sofortmitteilungen und Chatprogramme
- Entfernter Datenzugriff (Remote directory access); z.B.: Online Telefonbücher, Gelbe Seiten oder sonstige Datenbanken
- Standortbasierte Anwendungen (Location-based services): z.B. Landkarten, Routenplaner usw.

[2] John Jackson, Yankee Group, Februar 2003 [CLDCSTAT]

[3] ARC Group, "Wireless Java Report", Juni 2002 [CLDCSTAT]

[4] Sun, Februar 2003

- Finanzapplikationen: z.B. Onlinebanking oder Aktien- oder Depot-Verwaltung
- Automation: Kundenbeziehungsmanagementsoftware (Customer Relationship Management; CRM), Unternehmensapplikationen (z.B. Supply Chain Management SCM), Fernmesstechnik
- Konsumapplikationen: Online Auktionen, News Services, Online Reservierungssystem
- Hybride Applikationen

Bei dieser Aufzählung von Anwendungsgebieten stellt sich sogleich die Frage, wie eine Applikation auf ein J2ME-fähiges Endgerät übertragen wird, da diese meist nachträglich nach den Bedürfnissen des Kunden geladen werden müssen. Dafür wurden in J2ME mehrere Möglichkeiten vorgesehen. Zum einen die direkte Übermittlung von einem PC in das mobile Gerät, mittels serieller Verbindung, USB Docking Station, Bluetooth, Infrarot oder Speicherkarte, und zum anderen die Datenübertragung über das Mobilfunknetzwerk, auch Over The Air (OTA) Provisioning genannt.

Diese Diplomarbeit beschäftigt sich mit der Implementierung einer sicheren Applikationsdistribution von J2ME-Applikationen über das Mobilfunknetzwerk, und fand als Teil eines Industrieprojekts, mit der Bezeichnung: Secure Third Generation Information System (S3GIS) statt. In der Kooperation mit einem österreichischen Unternehmen wurde am Institut für Computertechnik dieses Forschungsprojekt ausgeführt.

Diese Diplomarbeit ist folgendermaßen gegliedert:

Eine kurze Einführung in die Java-Technologie und der Java-2-Micro-Edition (J2ME) wird in Kapitel 2 präsentiert, wobei in die für diese Diplomarbeit wichtigen Aspekte detaillierter eingegangen wird. In Kapitel 3 wird der Stand der Technik erläutert der als Vorraussetzung für das Verständnis dieser Arbeit wichtig ist. Kapitel 4 zeigt die Aufgabenstellung und deren Anforderungen und in Kapitel 5 wird die entwickelte Lösung präsentiert. Während in Kapitel 6 der Kommunikationsablauf detaillierter beschrieben wird, beschäftigt sich Kapitel 7 mit der Implementierung der Projektlösung. Schließlich findet in Kapitel 8 eine Zusammenfassung und Diskussion zu dieser Diplomarbeit statt.

2 Java 2 Micro Edition

Ein Pfad, dem nur wenige folgen, führt zu höherem,
als eine Straße, die Tausende gehen.

Thomas Edward Lawrence

In diesem Kapitel wird eine kurze Einführung in die verschiedenen Java-Technologien präsentiert. Anschließend wird die Java 2 Micro Edition (J2ME) genauer durchleuchtet, wobei vorwiegend die für die Arbeit relevanten Aspekte beschrieben werden.

2.1 Java Plattformen

Jede Edition von Java zielt auf eine eigene Hardwareplattform ab, für die sie konzipiert ist. Es gibt dabei folgende vier verschiedene Java Editionen[5] (siehe auch Abbildung 1):

- J2SE: Die Java 2 Standard Edition (J2SE) wurde für Desktopcomputer und deren Anwendungen entwickelt [J2SE01]
- J2EE: Die Java 2 Enterprise Edition (J2EE) baut auf der J2SE auf und wurde für Serveranwendungen in Unternehmen mit vielen Benutzen entwickelt [J2EE01]
- J2ME: Die Java 2 Micro Edition (J2ME) ist eine Zusammenfassung von Technologien und Spezifikationen für mobile Kleingeräte, wie zum Beispiel Mobiltelefone, PDAs, Smartphones [J2ME01]
- JavaCard: Die Edition für Smartcards ist zugeschnitten auf die Ressourcen von Smartcards und deren Anwendungsbereich und weicht am meisten von der Standardedition J2SE ab [JCARD01]

[5] Entwicklungsstand November 2003

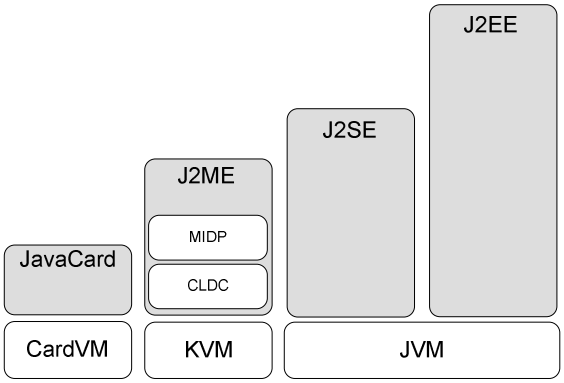


Abbildung 1: Java Editionen und deren Virtual Machine

Für das erfolgreiche Umsetzen des „Write once run anywhere"-Prinzips ist in jeder Java Edition die Virtual Machine (virtuelle Maschine) [VMSpec] verantwortlich. Wie aus der Grafik ersichtlich, verwenden die Standard Edition und die Enterprise Edition aufgrund der ähnlichen Hardware (Desktopcomputer und Unternehmensrechner) die diesen Editionen zugrunde liegt, dieselbe Virtual Machine zum Ausführen der Java Applikationen. Die JavaCard Edition besitzt eine für Smartcards optimierte Virtual Machine, die Card Virtual Machine (CardVM), die eine kleinere Menge an Klassen und Objekten zur Verfügung stellt und eigentlich aus zwei Virtual Machines für Laden, Linken, Verifizieren und Ausführen besteht. Details dazu sind der Spezifikation von JavaCard zu entnehmen [JCARD01SP].

Schließlich existiert J2ME für mobile Endgeräte, bestehend aus einer Untermenge der J2SE-Komponenten. Zusätzlich sind in J2ME andere, auf die mobilen Endgeräte zugeschnittene Objekte und Methoden vorhanden. Die J2ME besteht aus einer schlankeren virtuellen Maschine und einer kleineren Menge an Programmbibliotheken.

2.2 Architektur der J2ME Plattform

In diesem Kapitel geht es um die Architektur der Plattform, auf welcher J2ME-Applikationen ausgeführt werden können. Applikationen für die Java 2 Micro Edition werden auch MIDlets genannt. MID steht für Mobile Information Device und bezeichnet ein J2ME-fähiges mobiles Endgerät und „let" leitet sich von den Java-Applets (Java-Programme für Webbrowser) ab.

Abbildung 2 stellt die J2ME-Architektur dar:

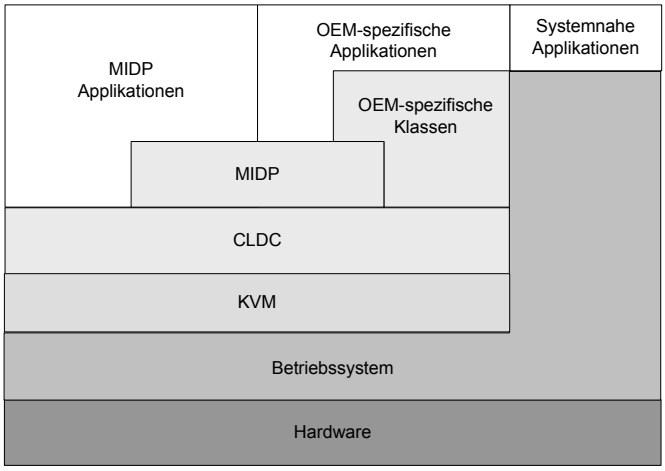


Abbildung 2: J2ME Architektur

Auf der jeweiligen Gerätehardware läuft ein Betriebssystem (Native System Software), das die wichtigsten Hardware-Eigenschaften kapselt. Auf diesem Betriebssystem setzt dann die Kilobyte Virtual Machine (KVM, [KVM]) auf. Die KVM übernimmt das Abbilden von Java-Bytecode-Instruktionen auf das Betriebssystem und dient als Ausführungseinheit von Java Applikationen. Die KVM ist somit für das „Write once, run anywhere"-Prinzip zuständig.

Unmittelbar auf der KVM setzt die Connected Limited Device Configuration (CLDC, [CLDC1]) mit der CLDC-API[6] auf, welche grundlegende Java-Funktionalität für Kleingeräte mit Netzwerkzugang (z.B. Mobiltelefone) zur Verfügung stellt. Diese Konfiguration ist für eine bestimmte Klasse von mobilen Endgeräten mit folgenden Eigenschaften konzipiert:

- 16 Bit oder 32 Bit Prozessor mit 8 bis 32 MHz
- 160 bis 512KB Speicheranforderungen für die gesamte J2ME (d.h. für die Virtual Machine und J2ME-Anwendungen)
- mindestens 128KB nichtflüchtigen Speicher für KVM und CLDC-Bibliotheken
- mindestens 32KB für die Laufzeitumgebung und neue erzeugte Objekte
- Anschluss an einem Netzwerk mit begrenzter Bandbreite (z.B. GSM-Handy mit 9600 Bit/s).

[6] API: Application Programm Interface; Programmbibliothek

Auf der CLDC-Konfiguration baut das Mobile Information Device Profile (MIDP, [MIDPW]) und dessen MIDP-API auf und erweitert die CLDC-Konfiguration. Das MIDP-API verschafft dem Entwickler eine Reihe von Funktionen, die in der gleichnamigen Spezifikation definiert sind, wie zum Beispiel Netzwerkfunktionen und Funktionen zur grafischen Oberfläche. Eine MIDP-Applikation, die nur auf dem CLDC-API bzw. MIDP-API aufsetzt, ist theoretisch plattformunabhängig.

Neben dem MIDP-API ist es dem Entwickler des Gerätes selber überlassen weitere Funktionen zu implementieren, welche möglicherweise auf ein spezifisches Gerät zugeschnitten sind (in Abbildung 2 als *OEM spezifische Klassen* bezeichnet). Die Applikationen, die von solchen Funktionen Gebrauch machen reizen vielleicht die spezifischen Dienste eines Gerätes besser aus, verlieren aber mit Sicherheit die Plattformunabhängigkeit.

Es existiert eine weitere Konfiguration für Endgeräte, die Connected Device Configuration (CDC) [CDCWP], die für Gerätetypen mit folgenden Eigenschaften in Frage kommt:

- permanente Netzwerkverbindung
- schneller Prozessor (32Bit)
- mindestens 512KB Speicher (ROM/Flash) für die Ausführung von J2ME-Anwendungen
- mindestens 256KB Speicher (RAM)
- schnelle Netzverbindung (z.B. LAN, 56KB Modem bis hin zu DSL)

Diese Konfiguration läuft auf der klassischen Virtual Machine (JVM) und ist sozusagen für kleine Geräte mit mehr Speicherplatz und Rechenleistung wie "Screen Phones" und "Set Top Boxen " gedacht.

Im weiteren Verlauf dieser Arbeit wird mit der Bezeichnung *MIDP2.0* die Technologie J2ME, CLDC und MIDP2.0 zusammengefasst. Mit der Bezeichnung MIDP2.0 wird das Mobile-Information-Device-Profile in der Version 2.0 genannt. Eine Beschreibung von J2ME kann in [J2ME01DC] gefunden werden.

2.3 MIDlet Lebenszyklus

MIDlets werden durch die Application Management Software (AMS), auch unter dem Begriff Java Application Manager (JAM) bekannt, kontrolliert. Die AMS übernimmt das Management der MIDlets, d.h. sie installiert, startet, stoppt und deinstalliert MIDlets.

Bei der Applikationsdistribution hat der Entwickler bzw. Hersteller der Applikationen die Möglichkeit mehrere MIDlets in ein Archiv zu verpacken und dieses als Einheit zu verteilen. Dieses Archiv nennt man MIDlet-Suite. Bei der Installation und Deinstallation wird eine MIDlet-Suite immer als Einheit betrachtet und es ist nicht möglich, gezielt einzelne MIDlets aus der MIDlet-Suite zu installieren bzw. deinstallieren.

Eine MIDlet-Suite hat den in der Abbildung 3 abgebildeten Lebenszyklus, der mit der Applikationssuche beginnt und der Deinstallation endet.

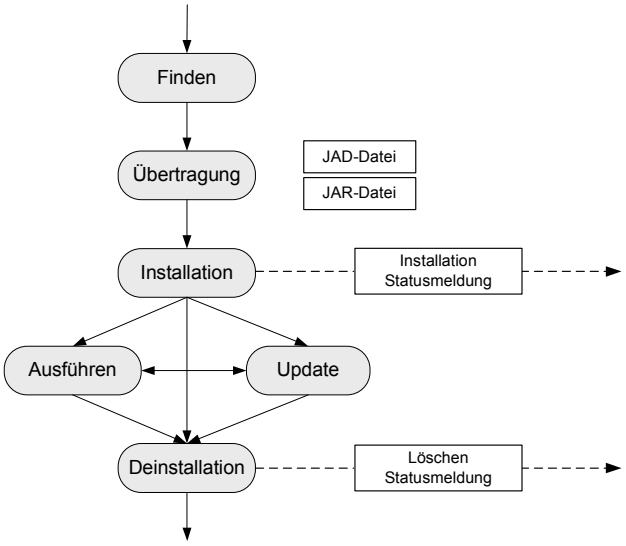


Abbildung 3: MIDlet Lebenszyklus

Durch Auffinden der Applikation durch den Benutzer wird die Applikationsdistribution initiiert. J2ME-fähige Geräte müssen laut MIDP-Spezifikation eine Möglichkeit bieten, MIDlet-Suites zu finden (Discovery). Dies kann entweder durch einen am Gerät befindenden Browser oder eine Applikation geschehen. Eine Applikation die es ermöglicht MIDlet-Suites zu finden bzw. zu entdecken und somit den Download zu starten wird Discovery Application (DA) genannt.

Danach findet die Übertragung der MIDlet-Suite auf das mobile Endgerät statt. Diese Übertragung kann entweder über ein Mobilfunknetzwerk erfolgen (Over The Air Provisioning) genannt, oder über eine direkte Verbindung, wie z.B. eine serielle Verbindung, USB Docking Station, Bluetooth, Infrarot oder Speicherkarte. Die Übertragung der MIDlet-Suite erfolgt in Form einer Archivdatei (JAR-Datei) und kann einer Applikationsbeschreibungsdatei (JAD-Datei) nachfolgen. Den Inhalt der

Archiv-Datei und der Applikationsbeschreibungsdatei ist im Abschnitt 2.4 zu finden.

Nach der Übertragung wird die MIDlet-Suite am mobilen Endgerät installiert, und ab diesem Zeitpunkt stehen die MIDlets aus der MIDlet-Suite dem Benutzer zur Ausführung zur Verfügung.

Optional kann der Benutzer ein Update der MIDlet-Suite durchführen. Wurde die MIDlet-Suite über das „Over The Air Provisioning" übertragen, kann der Benutzer die Application Management Software (AMS) des mobilen Endgerätes veranlassen, von der Downloadadresse der MIDlet-Suite eine neuere Version der MIDlet-Suite herunterzuladen. Wurde die MIDlet-Suite hingegen nicht über das „Over The Air Provisioning" übertragen, ist ein solches Update nicht möglich, und die MIDlet Suite muss neu installiert werden. Dazu muss der Benutzer die alte MIDlet-Suite deinstallieren und die neue installieren.

Schließlich endet der Lebenszyklus der MIDlet-Suite indem die MIDlet-Suite wieder vom Gerät entfernt (Deinstallation) wird. Dabei ist es wiederum nicht möglich einzelne MIDlets der MIDlet-Suite zu entfernen, die MIDlet-Suite wird immer als eine Einheit behandelt.

Bei der Installation und Deinstallation können die in Abbildung 3 gezeigten Statusmeldungen über den Erfolg der Installation oder des Löschens verschickt. Diese Statusmeldungen werden Notifications genannt und im Abschnitt 2.4.2 genauer beschrieben.

2.4 Applikationspackaging für Distribution

Ein oder mehrere MIDlets werden jeweils in einer MIDlet-Suite ausgeliefert. Das dazu verwendete Java-Archiv (JAR-Datei) enthält:

- Eine oder mehrere MIDlet-Klassen (Applikationen) inklusive Hilfsklassen
- Eine Manifest-Datei, die den Inhalt der Suite beschreibt
- Zusatzdateien wie Grafiken (PNG-Format[7]), Textdateien etc.

Die Manifest-Datei dient als Übersicht und Beschreibung des Inhalts der MIDlet-Suite und enthält enthalten die in Tabelle 1 und Tabelle 2 angegebenen Attribute.

[7] Grafiken im PNG-Format [PNGF01] müssen von MIDP unterstützt werden. Die Unterstützung anderer Formate ist nicht geregelt und den Herstellern überlassen. (Stand November 2003)

Tabelle 1: Mainfest-Attribute

Zwingend vorgeschrieben	**Optional**
MIDlet-Name	MIDlet-Description
MIDlet-Version	MIDlet-Icon
MIDlet-Vendor	MIDlet-Info-URL
MIDlet-X für jedes MIDlet	MIDlet-Data-size
Microedition-Profile	
Microedition-Configuration	

Das Manifest kann zusätzliche herstellerspezifische Attribute enthalten. Alle Attribute müssen jedoch den Konventionen folgen, die in der MIDP Spezifikation [MIDP20] angegeben sind. Jedes Attribut kann von allen MIDlets innerhalb der gleichen Suite, über die Methode *MIDlet.getAppProperty* angesprochen werden und wird häufig für Konfigurationseinstellungen genutzt.

Optional kann zur gelieferten JAR-Datei eine Applikationsbeschreibungsdatei (Java Application Descriptor) existieren. Diese Text-Datei hat zwingend die Dateiendung *.jad* und stellt Zusatzinformationen zur MIDlet-Suite ähnlich dem Manifest bereit.

Durch die JAD-Datei kann die AMS vor der eigentlichen Installation der MIDlet-Suite feststellen, ob die MIDlet-Suite auf dem Gerät überhaupt lauffähig ist, - d.h. ob z.B. genügend Ressourcen oder geforderte Eigenschaften vorhanden sind- und präsentiert dem Benutzer den Inhalt der JAD-Datei. Gibt der Benutzer sein Einverständnis so wird die eigentliche Applikation in Form der JAR-Datei heruntergeladen.

Kommt ein Attribut in der JAD-Datei und im Manifest vor so müssen sie identisch sein. Ist dies nicht der Fall so dominieren die Werte der JAD-Datei bei untrusted MIDlet-Suites und die Werte des Manifests bei trusted MIDlet-Suites[8]. Für nähere Details ist auf Abschnitt 2.5 verwiesen.

2.4.1 Attribute eines MIDlets

In der MIDP2.0 Spezifikation [MIDP20] sind eine Reihe von Attributen definiert. Je nach Attribut gilt folgender Zusammenhang zur JAD-Datei und dem Manifest:

- Das Attribut muss sowohl im JAD-Datei als auch im Manifest vorkommen

[8] Die Attribute MIDlet-Name, MIDlet-Version und MIDlet-Vendor müssen immer identisch sein, ansonsten darf das MIDlet vom AMS nicht installiert werden.

- Das Attribut muss entweder im Manifest oder in der JAD-Datei vorkommen
- Das Attribut kann im Manifest bzw. in der JAD-Datei vorkommen.

Alle in MIDP 2.0 definierten Attribute sind in Tabelle 2 aufgelistet. Dabei bedeutet ein „+“, dass das Attribut vorkommen muss, ein „-“ das es nicht Vorkommen darf und ein „°“ bedeutet, dass im Falle des Vorkommens dieses Attribut in der JAD-Datei eingetragen sein sollte. Die anderen Attribute sind in beiden Dateien optional.

Tabelle 2: Attribute eines MIDlets

Attribute Name	**JAD-Datei**	**Manifest**
MIDlet-Name	+	+
MIDlet-Version	+	+
MIDlet-Vendor	+	+
MIDlet-Icon		
MIDlet-Description		
MIDlet-Info-URL		
MIDlet-<n>	+	
MIDlet-Jar-URL	+	-
MIDlet-Jar-Size	+	-
MIDlet-Data-Size	°	
MicroEdition-Profile	+	+
MicroEdition-Configuration	+	+
MIDlet-Permissions		
MIDlet-Permissions-Opt		
MIDlet-Push-<n>	°	
MIDlet-Install-Notify	+	-
MIDlet-Delete-Notify	+	-
MIDlet-Delete-Confirm		

2.4.2 Statusmeldungen

Die AMS versendet Statusmeldungen über den Erfolg, bei einer De- oder Installation einer Applikation (vergleiche Abbildung 3). Dieser Status Report einer MIDlet-Suite erfolgt durch eine HTTP-POST-Anweisung an eine URL, die in der Beschreibungsdatei der MIDlet-Suite (JAD-Datei) eingetragen werden kann. Diese URL befindet sich für eine Installation im Attribut *MIDlet-Install-Notify* und für das Löschen einer MIDlet-Suite im Attribut *MIDlet-Delete-Notify*. Das einzige Protokoll

zur Übertragung, das unterstützt werden muss ist das HTTP-Protokoll (Hypertext Transfer Protokoll [HTTP10]).

Im Inhalt der POST Anfrage muss der Statuscode und die Statusmeldung enthalten sein. Tabelle 3 zeigt ein Beispiel für eine solche HTTP-POST-Anweisung einer erfolgreichen Installation.

Tabelle 3: HTTP Post Statusmeldung

```
POST http://foo.bar.com/status HTTP/1.1
Host: foo.bar.com
Content-Length: 13
900 Success
```

Sämtliche Statuscodes und deren Statusmeldungen sind in Tabelle 4 aufgeführt.

Tabelle 4: Installation/Löschen Statusmeldungen

Statuscode	**Statusmeldung**
Status 900	Installation erfolgreich abgeschlossen
Status 901	Speicherplatz des Gerätes nicht ausreichend
Status 902	Abbruch durch Benutzer
Status 903	Unerwartetes Übertragungsende
Status 904	Fehlerhafte Größe der JAR-Datei
Status 905	Falsche oder fehlende Attribute
Status 906	Ungültiger Java Application Descriptor (JAD-Datei)
Status 907	Ungültige JAR-Datei
Status 908	Inkompatibilitäten bei der Konfiguration oder dem Profil
Status 909	Ungültige Anmeldung bei der Anwendung
Status 910	Ungenügende Rechte für die Anwendung
Status 911	Push Registrierung fehlgeschlagen
Status 912	Bestätigung über die Löschung des MIDlets

Diese Statusmeldungen müssen bei der ersten Installation oder beim ersten Löschen einer MIDlet-Suite verschickt werden. Ist der Server aber nicht erreichbar, oder kann keine Funkverbindung aufgebaut werden bzw. tritt ein Fehler höherer Gewalt auf, sollte die Statusmeldung zu einem späteren Zeitpunkt nochmals verschickt werden. Ob diese Statusmeldung jedoch nochmals versendet wird und wie oft der Vorgang bis zum Erfolg wiederholt wird, hängt von der Implementierung des Herstellers ab und ist in der MIDP 2.0 Spezifikation nicht weiter festgehalten. Es sollte lediglich die Anzahl der Sendewiederholungen klein

gehalten werden, da der Benutzer meist für die Netzwerkverbindungen bezahlen muss.

2.5 Sicherheit

J2ME verfolgt wie jede Java-Edition mehrere Sicherheitsziele, die für die Benutzer und Geräte der jeweiligen Technologie wichtig sind. Wichtig dabei sind der Schutz der Gerätehardware vor Schädigung, der Schutz der Daten vor Manipulation, und der Schutz des Benutzers vor ungeplanten Kosten durch Funktionsaufrufe, wie z.B. dem Aufbau von Netzwerkverbindungen oder dem Versand von SMS-Nachrichten.

2.5.1 Sicherheit in J2ME

Sämtliche MIDlets werden in einer Java-Virtual-Machine-Sandbox ausgeführt. Das ist eine spezielle Umgebung, die die Applikationen von der äußeren Hard- und Software abschottet. Diese Sandbox gewährleistet, dass Applikationen nur Klassen der Konfiguration, der Profile und der geladenen MIDlet-Suite ausführen können. Außerdem können zur Laufzeit keine weiteren Klassen hinzugefügt werden und Systemklassen können nicht überschrieben werden.

In der J2SE-Umgebung werden Klassen vor der Nutzung von der virtuellen Maschine auf ihre Gültigkeit überprüft[9]. Die CLDC-Konfiguration bei J2ME sieht im Gegensatz dazu vor, dass diese Überprüfung nicht von der virtuellen Maschine (KVM) des Gerätes, sondern vor dem Laden der Applikation auf das Gerät von einem Preverifier extern durchgeführt werden muss, da der virtuellen Maschine des mobilen Endgerätes nicht so viele Ressourcen wie in J2SE-Umgebungen zur Verfügung stehen.

Dieser Prozess der Überprüfung (Verifying) ist bei J2ME zweigeteilt. Zum einem findet ein externes Preverifiying statt und ein zweites Verifying am Gerät selber.

Im ersten Schritt werden die kompilierten Java-Bytecode-Dateien nach dem externen Kompilieren vor-verifiziert, d.h. in einer JAR-Datei[10] gespeichert und mit einem StackMap-Attribut versehen. Die Vorverifizierung erweitert eine Applikationsklasse um das StackMap-Attribut, das Informationen über lokale und Stackvariablen und deren Datentypen enthält. Nach diesem Schritt, der unabhängig vom mobilen Endgerät ist, kann die Applikation (JAR-Datei) auf das mobile

[9] Der Bytecode der ausführenden Dateien muss konsistent sein.

[10] JAR-Datei ist eine Java-Archiv-Datei; Details zum Inhalt siehe Abschnitt 2.4.

Endgerät kopiert werden. Vor dem Ausführen am mobilen Endgerät wird dann der zweite Teil der Prüfung von der virtuellen Maschine am Gerät vorgenommen. Bei dieser Überprüfung wird anhand der StackMap-Attribute kontrolliert, ob es sich um eine gültige J2ME Applikation handelt. Für die genauen Gültigkeitsregeln über eine J2ME-Applikation sei auf die Spezifikation in [CLDC1] verwiesen.

Diese zwei Schritte gewährleisten dass keine ungültigen, ungeprüften Applikationen in der virtuellen Maschine ausgeführt werden.

2.5.2 Mobile Information Device Profile

Das Mobile Information Device Profile in der Version 1.0 (MIDP 1.0, [MIDP1.0]) erlaubt den ausführenden MIDlets Zugriff auf den RMS-Speicher[11], Netzwerk und sämtliche weitere Funktionen die die virtuelle Maschine oder die CLDC Konfiguration oder das MID Profil zur Verfügung stellt. Die Sicherheit bei MIDP1.0 konformen Geräten besteht also lediglich aus der Sandbox Sicherheit.

Eine Applikation kann also Netzwerkverbindungen aufbauen oder auf den RMS-Speicher zugreifen. Dieses Vorgehen kann dem Benutzer Geld kosten und stellt daher ein Sicherheitsproblem dar. Deshalb wurden im MIDP 2.0 vertrauenswürdige Applikationen, Zugriffsberechtigungen und Codesignatur eingeführt, die den Benutzer warnen, wenn nicht vertrauenswürdige Applikationen solche kritischen Funktionen aufrufen.

Das Mobile Information Device Profile in der Version 2.0 (MIDP 2.0, [MIDP20]) ist seit Ende des Jahres 2002 verfügbar und seit Anfang 2003 stehen auch schon die ersten mobilen Endgeräte, die dieses Profil enthalten, zur Verfügung. Eine Liste von Geräten, die das MIDP2.0 Profil unterstützen, ist unter [DEVLST] zu finden.

Mit MIDP 2.0 haben neue Konzepte zur Sicherung der Daten Einzug gehalten: vertrauenswürdiger (trusted) und nicht vertrauenswürdiger (untrusted) Code sowie Zugriffsberechtigungen:

- Ein nicht vertrauenswürdiger Code kann nicht beliebig Verbindungen nach außen aufbauen und ist weiters in seiner Funktionalität stark eingeschränkt. Für jede sicherheitsrelevante Funktion benötigt die Anwendung die explizite Erlaubnis des Anwenders

[11] RMS (Record Management Store) bietet den MIDlets einen „abstrakten" Speicher, der im Wesentlichen eine geräteunabhängige Speicherung in Feldform (Array) ermöglicht.

- Ein Hersteller oder Vertreiber kann aber eine Applikation digital signieren, um sie als vertrauenswürdig einzustufen. Das Endgerät bzw. die Application Management Software (AMS), die die Applikationen am mobilen Endgerät verwaltet, überprüft bei der Installation die Signatur bzw. Unterschrift. Ist diese Verifikation erfolgreich erhält die Applikation den Status einer vertrauenswürdigen Applikation und kann dann sicherheitsrelevante Funktionen aufrufen. Mit Hilfe von speziellen Einträgen in der Manifest-Datei können MIDlet-Suites[12] angeben, welche Zugriffsrechte sie benötigen, um korrekt ablaufen zu können, und so das AMS bei der Installation unterstützen
- Zugriffsrechte: Jeder Applikation können vom Benutzer Zugriffsrechte auf Funktionen gegeben werden. Ein Zugriffsrecht kann dabei nur einmalig für die Dauer der Ausführung der Applikation oder für immer, d.h. solange die Applikation am Gerät installiert ist, vergeben werden
- Alle MIDP1.0 konformen MIDlet-Suites werden auf MIDP2.0-Geräten als "untrusted" (nicht-vertrauenswürdig) ausgeführt, weil ihre JAR-Datei keine Sicherheitserlaubnis enthält

2.5.3 Signatur

Durch das Signieren von MIDlet-Suites ist es möglich, den Unterzeichner (Hersteller) der Signatur zu authentifizieren und somit die Integrität einer MIDlet-Suite zu verifizieren. Dazu wird vor der Installation, die Signatur der MIDlet-Suite laut MIDP2.0 Spezifikation vom AMS verifiziert und bei erfolgreicher Verifikation die Installation durchgeführt. Bei fehlgeschlagener Signaturverifikation erhält der Benutzer eine Fehlermeldung, und kann die betroffene MIDlet-Suite nicht installieren.

Durch das Entfernen der entsprechenden Attribute aus der JAD-Datei und Entfernen der Signatur der MIDlet-Suite, kann der MIDlet-Suite die Eigenschaften einer MIDP1.0-MIDlet-Suite gegeben werden und diese als nicht-vertrauenswürdige (untrusted) Applikation am Gerät installiert werden. Dies muss der Benutzer aber explizit durchführen, und ist nur mit entsprechenden technischen Kenntnissen zu bewerkstelligen. Im Sinne einer Sicherheitsanalyse hat der Benutzer aber trotzdem die Möglichkeit diese MIDlets auszuführen.

[12] MIDlet-Suites: Einzelne MIDlets (MID-Applikationen) können in ein Archiv verpackt werden und werden dann als ganze Einheit betrachtet. Dieses Archiv nennt man MIDlet-Suite.

Die Signierung und Verifikation der MIDlet-Suites wird anhand einer X.509-Public-Key-Infrastructure (PKI [RFC2510]) durchgeführt. Der genaue Ablauf des Signierens und die Verifikation der Schlüssel und Zertifikate sind in [MIDP2] festgelegt.

2.6 Fazit

Der im Kapitel 2.2 aufgeführte Technologie-Stack von MIDP, ermöglicht eine Geräteunabhängigkeit der Applikationen, und bildet somit die Grundlage für das Prinzip „Write once, run anywhere“ für J2ME-Geräte.

Bei J2ME-fähigen Endgeräten können die Applikationen direkt per Datenkabel oder Infrarot oder über ein Mobilfunknetzwerk (Over The Air) ins Gerät übertragen werden. Bei Nicht-J2ME-fähigen Geräten werden die Applikationen vom Hersteller in das mobile Gerät abgespeichert bzw. in den Speicher eingebrannt und nur der Hersteller kann bestimmen welche Applikationen beim Ausliefern des Gerätes vorhanden sind. Nach dem Verkauf der Geräte sind benutzerinitiierte Applikationsdownloads meist nicht mehr vorgesehen. Der klare Vorteil von J2ME-Geräten ist dieser benutzerinitiierte Applikationsdownload durch die Unabhängigkeit der Applikationen vom Gerätehersteller, Netzbetreiber und vom Gerät.

Zur Sicherheit ist zu sagen, dass die Abschottung der Applikationen durch die Java-Sandbox und die Verifikation von Applikationen vor deren Ausführung, nur zur Sicherheit der Virtual Machine beiträgt und nur indirekt zur Sicherheit des Gerätes und des Benutzers. Denn durch diese Abschottung und Verifikation ist es lediglich nicht möglich ungültige bzw. „fehlerhafte“ Applikationen am Gerät auszuführen.

Erst durch das Mobile Information Device Profile in der Version 2.0 (MIDP2.0) ist eine weitere wichtige Sicherheitsstufe durch vertrauenswürdige Applikationen (Trusted-Applications) und Zugriffsberechtigungen hinzugefügt worden. Vertrauenswürdige Applikationen erhalten dabei das Recht bestimmte Funktionen, Objekte und deren Methoden (APIs) zu benutzen, die unsichere Applikationen nur nach expliziter Erlaubnis des Benutzers erhalten. Zum Beispiel kann eine unsichere Applikation keine HTTP-Verbindung aufbauen, ohne dass der Benutzer dies genehmigt.

3 Stand der Technik

Wir lernen aus Erfahrung, dass die Menschen nichts aus Erfahrung lernen.
George Bernhard Shaw

In diesem Kapitel werden sämtliche Komponenten und wichtige Voraussetzungen beschrieben, auf denen diese Diplomarbeit aufbaut. Dies ist die standardisierte Applikationsdistribution für J2ME-Geräte, die an dieser Stelle beschrieben und deren Vor- und Nachteile durchleuchtet wird. Weiters wird eine bereits existente Serverimplementierung analysiert.

3.1 Applikationsdistribution

Unter der Applikationsdistribution „Over The Air User Initiated Provisioning" versteht man den Download einer Applikation über ein Mobilfunknetzwerk, typischerweise auf Wunsch des Benutzers. Diese vom Benutzer initiierte Applikationsdistribution ist das zentrale Thema dieser Arbeit.

3.1.1 OTA Provisioning MIDP1.0

In Abbildung 4 wird der OTA-Download nach MIDP1.0 [MIDP1.0] grafisch dargestellt. In der MIDP1.0-Spezifikation ist diese Art der Applikationsdistribution nur „Best Practice"[13]. Das heißt sie ist nur eine Empfehlung und muss vom MIDP1.0-Endgerät nicht unterstützt werden, obwohl viele Endgeräte das Verfahren unterstützen.

[13] optimale Verfahrensweise

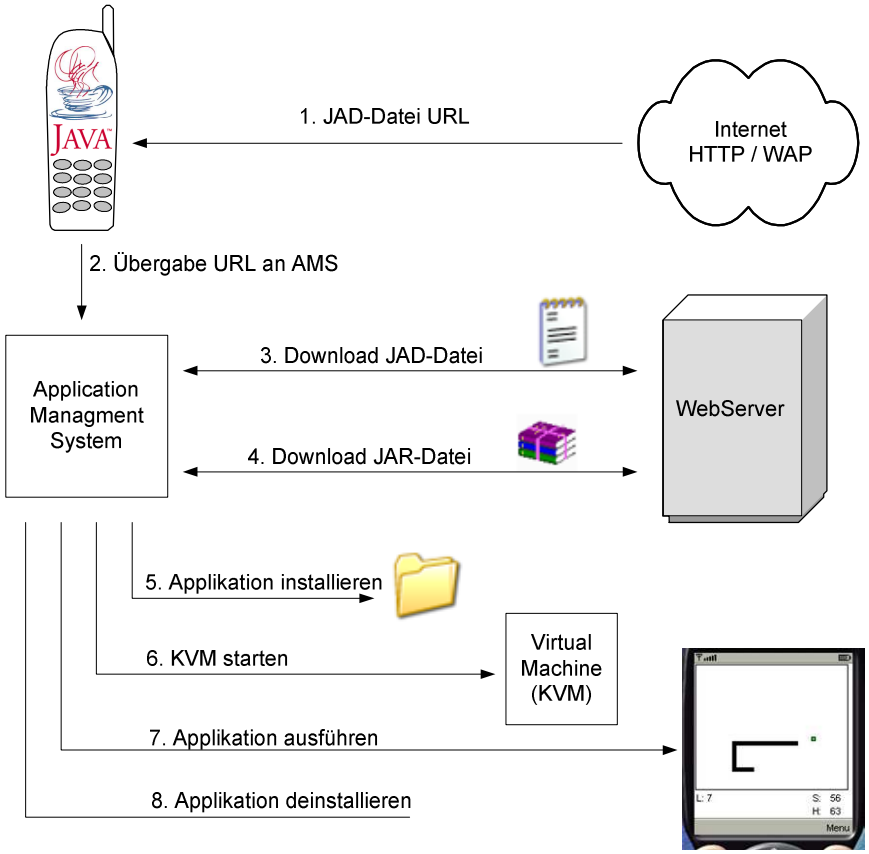


Abbildung 4: Over-The-Air Provisioning nach MIDP1.0

Der OTA-Download gliedert sich, wie in Abbildung 4 dargestellt, in folgende Schritte:

1. Der Benutzer steuert eine JAD-Datei-Download-URL an. Entweder durch direkte Eingabe der URL oder durch Browsen am Gerät per WAP (Wireless Application Protocol [WAP]) oder bekommt diese per SMS zugeschickt
2. Die URL wird an die Application Management Software (AMS) übergegeben
3. Die AMS lädt die JAD-Datei vom entsprechenden Webserver
4. Der Benutzer bekommt die Informationen aus der Beschreibungsdatei (JAD-Datei) präsentiert und erhält die Möglichkeit die Applikation zu installieren oder zu verwerfen. Vorher überprüft die AMS, ob die Applikation überhaupt am Gerät lauffähig ist. Das kann sie anhand der Minimalvoraussetzungen (z.B. ausreichend freier Speicher, Konfigurations- und Profilversion höher oder gleich einer evtl. schon installierten Version)

aus der Beschreibungsdatei erkennen. Hat der Benutzer seine Zustimmung gegeben, lädt die AMS die JAR-Datei, welche die Applikationen beinhaltet, vom entsprechenden Webserver herunter

5. Die Applikation wird am mobilen Endgerät abgespeichert, und steht ab dem Moment dem Benutzer zur Ausführung zur Verfügung. Enthält die JAD-Datei eine URL zur Benachrichtigung bei Installation, so wird an diese URL das Ergebnis der Installation geschickt. Die verschiedenen Werte die eine Installation zurückliefern kann, können in der Tabelle 4 gefunden werden
6. Die Applikation wird gestartet. Dazu wird die virtuelle Maschine (KVM [KVM]) gestartet und ihr dann die Applikation übergeben
7. Die Applikation wird ausgeführt
8. Wird die Applikation zu einem späteren Zeitpunkt gelöscht, wird sie aus dem Speicher des Gerätes entfernt

Die Anfragen für die JAD- bzw. JAR-Datei an den Webserver müssen dem HTTP-Protokoll[14] entsprechen und haben den in Tabelle 6 und Tabelle 7 dargestellten Aufbau.

3.1.2 OTA Provisioning MIDP2.0

In der MIDP2.0-Spezifikation [MIDP20] ist der Over The Air User Initiated Provisioning-Prozess, nicht mehr „Best Practice" sondern als Standard definiert.

Der OTA-Download nach MIDP2.0 ist dem OTA-Download nach MIDP1.0 sehr ähnlich. Hinzugekommen ist die mögliche Authentifizierung per Login (JARLogin[15]) und Passwort beim JAR-Datei Download und die Verifikation der Applikationen, wie im Abschnitt 2.5.2 beschrieben wurde.

Entfernt wurde das „Cookie Support Requirement". Das heißt, dass die AMS keine Cookies[16] mehr akzeptieren darf und muss. Cookies wurden meist dazu verwendet, um einen Zusammenhang über mehrere HTTP-Anfragen zu ermöglichen, da das HTTP-Protokoll selber zustandslos ist. Mittels der URL-Rewriting-Methode

[14] HTTP-Protokoll: Hypertext Transfer Protokoll [HTTP10]

[15] In diesem Dokument wird der Begriff „JARLogin" als Synonym für die Authentifizierung des JAR-Datei-Downloads verwendet.

[16] Cookies: Benutzerbezogene Dateneinheit die beim HTTP Protokoll mitgeschickt wird.

können in MIDP2.0 Zusammenhänge zwischen HTTP-Anfragen (bzw. Sessions) in der URL abgespeichert werden und so die Cookies ersetzt werden.

Die notwendigen MIDP2.0-Fähigkeiten werden an dieser Stelle aufgeführt, da sie erst in der MIDP2.0 Spezifikation standardisiert sind:

- Browsen und Auffinden von MIDlet-Suites Applikationsbeschreibungen (JAD-Dateien) im Internet
- Übertragen einer MIDlet-Suite und der MIDlet-Suite-Beschreibung vom Server zum Gerät anhand des HTTP1.1[17] Protokolls
- Korrekte Handhabung von 401- (Unauthorized) oder 407- (Proxy Authentication Required) Antworten nach einer HTTP Anfrage, sodass der Benutzer nach JARLogin und Passwort gefragt wird, und danach die HTTP Anfrage nochmals mit diesen Daten gesendet wird. Das Gerät muss dafür zumindest den Bestimmungen des RFC2617 „Basic Authentication Scheme" [RFC2617] befolgen
- MIDlet-Suite am Gerät installieren
- MIDlets ausführen
- MIDlets wieder vom mobilen Endgerät löschen. Einzelne MIDlets können nicht vom Gerät gelöscht werden, da die MIDlet-Suite immer als ganze Einheit bei der Installation und Transfer gehandhabt wird
- Statusmeldung „Installation": Bei Installation einer MIDlet-Suite muss die AMS eine Statusmeldung verschicken. Diese in Abschnitt 2.4.2 beschriebenen Statusmeldungen sind in der MIDP2.0-Spezifikation standardisiert worden
- Statusmeldungen „Löschen": Die Statusmeldungen Löschen sind erst in der MIDP2.0 Spezifikation eingeführt worden und waren in der vorherigen MIDP1.0 nicht enthalten

In Abbildung 5 ist der erfolgreiche OTA-Download nach der MIDP2.0 Spezifikation dargestellt.

[17] HTTP1.1: Hypertext Transfer Protokoll in der Version 1.1 [HTTP11]

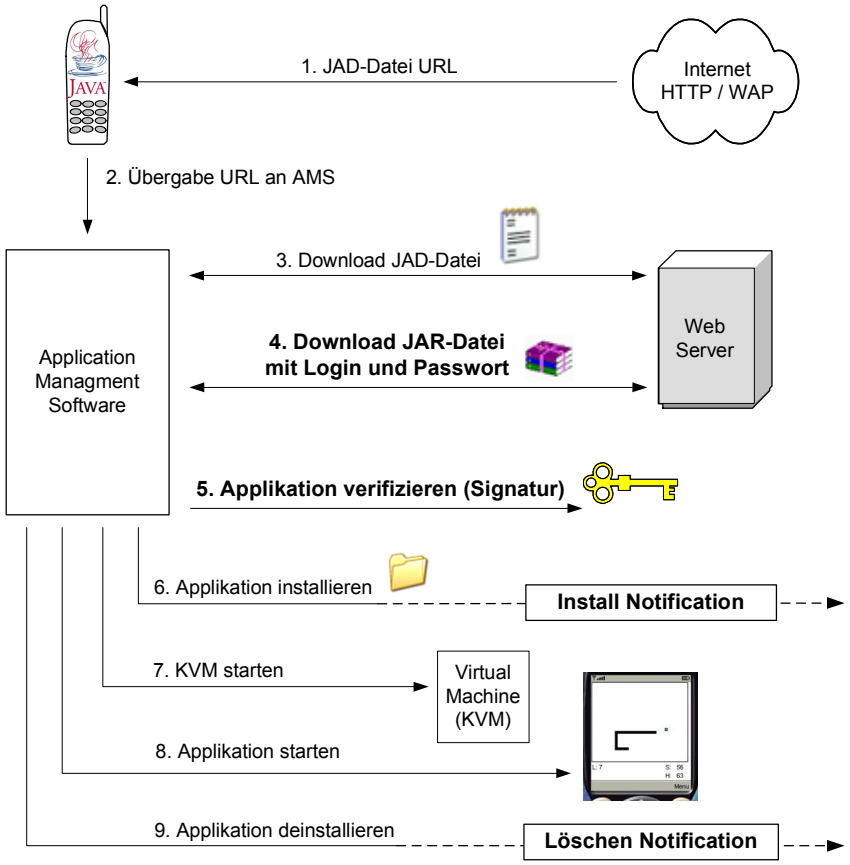


Abbildung 5: Over-The-Air Provisioning nach MIDP2.0

Die grundlegenden Unterschiede zum OTA-Download nach MIDP1.0 sind in der Abbildung 5 hervorgehoben, und betreffen den Download der JAR-Datei und die Verifikation der Applikationssignatur. Der Download mit JARLogin und Passwort in Schritt 4 wird nur durchgeführt wenn der Webserver auch ein JARLogin und Passwort zum Download anfordert und die Verifikation der Applikationssignatur findet nur dann statt, wenn das MIDlet entsprechend signiert ist. Sollte keine Signatur des MIDlets vorhanden sein entfällt auch der Schritt 5 in Abbildung 5.

Außerdem werden in der MIDP2.0-Spezifikation Statusmeldungen definiert, die die AMS übertragen muss. Diese Statusmeldungen sind in Abschnitt 2.4.2 beschrieben.

3.1.3 Sicherheitsprobleme

Die ersten offensichtlichen Sicherheitsprobleme beim OTA-Provisioning erkennt man schon daran, dass die gesamte Kommunikation über eine unverschlüsselte Verbindung per HTTP-Protokoll durchgeführt wird. Das heißt sämtliche Daten, die verschickt und gesendet werden, werden im Klartext über das jeweilige

Übertragungsmedium geschickt. Jeder der entlang der Kommunikationsstrecke auf den Datenverkehr zugreifen kann, kann daher die Applikationen abfangen, und somit die Applikationen auch verändern. Die Gefahr bringt nicht nur die Manipulation der Applikationen mit sich, sondern auch die Möglichkeit des Weiterverkaufs der abgefangenen Applikationen, was zu erheblichen Schaden für den Hersteller, Betreiber und schließlich auch für den Benutzer führt. Diese Angriffe nennt man „Man-in-the-Middle" Attacken.

Weiters erhält der Benutzer bei MIDP1.0-konformen Geräten keinen Aufschluss darüber welche Funktionen und Methoden die Applikation aufruft, zum Beispiel, ob die Applikation Netzwerkverbindungen aufbaut, auf den Speicher für Benutzerdaten zugreift, usw. Das Problem dabei ist, dass besonders bei Netzwerkverbindungen Kosten für den Benutzer entstehen, die ihm nicht notwendigerweise angezeigt werden und sich daher seiner Kontrolle entziehen.

Außerdem gibt es beim Standard Applikationsdownload keine Möglichkeit der Authentifizierung vor dem Downloadvorgang, im Sinne eines Zugriffschutzes, bzw. eines Vertragabschlusses. Es wird weder der Benutzer gegenüber dem Applikationsvertreiber noch der Applikationsvertreiber gegenüber dem Benutzer identifiziert. Solch eine Authentifizierung würde eine Autorisierung nur für gewisse Benutzergruppen ermöglichen oder aber zum Beispiel den möglichen Zugriff von Jugendlichen auf jugendgefährdende Inhalte verbieten.

3.2 Applikationsdistributionssysteme

Die Java-Gemeinde hinter dem „Java Community Process" (JCP) ist eine offene Organisation von Softwareentwicklern und Lizenznehmern die Spezifikationen für die Java-Technologie entwickeln. Diese öffentliche Gruppe bringt neue Ideen in die Java-Technologie ein und spezifiziert diese. Die Java-Technologie und der JCP wurden ursprünglich von Sun Microsystems gegründet, aber die Java-Gemeinde hinter JCP wird von Repräsentanten aus mehreren unabhängigen Organisationen geleitet und überwacht und bildet somit einen formalen von Sun unabhängigen Prozess.

Ein Java-Specification-Request (JSR) ist eine Spezifikation, die von einer JCP-Gruppe erstellt wurde und eine neue Idee oder Technologie beinhaltet. Diese Technologie wird von der jeweiligen JCP-Gruppe aufgegriffen, öffentlich diskutiert,

spezifiziert und eventuell eine Referenzimplementierung[18] entwickelt. Der genaue Ablauf dieser Spezifikationsentwicklung kann unter [JCP] gefunden werden und ist ähnlich der Standardisierung.

3.2.1 Java Specification Request 124

Der JSR-124 [JSR124V03], die J2EE Client Provisioning Specification, liefert ein Gerüst und APIs, um Applikationen auf einem J2EE Server zur Verfügung zu stellen. Abbildung 6 stellt dieses Konzept schematisch dar.

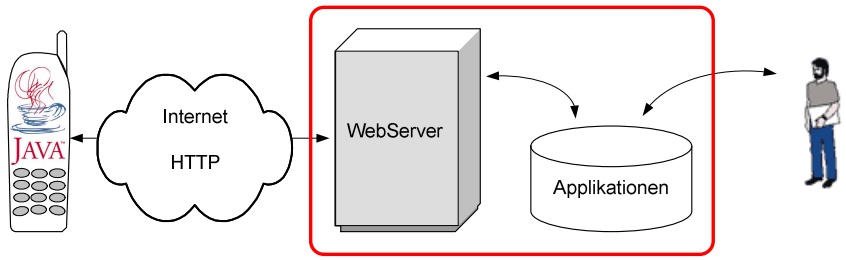


Abbildung 6: JSR 124 Überblick

Provisioning Server werden oft mit Imbissautomaten verglichen. So wie Imbissautomaten geben Provisioning Server dem Client (bzw. Benutzer) die Möglichkeit Applikationen zu suchen, auszuwählen und diese Applikationen herunterzuladen. Dabei besitzt aber der Provisioning Server Eigenschaften und Funktionen von denen ein Imbiss-Automat nur träumen kann. Zum Beispiel liefert der Provisioning Server jedem Client eine andere Liste von Applikationen, die genau auf ihn zugeschnitten und auf dem jeweiligen Client ausführbar sind. Dazu besitzt der Server eine Liste von unterstützten Plattformen, deren Leistungsspektrum und Eigenschaften. Diese Eigenschaften können von Anzeigengröße und Bittiefe des Displays bis zu möglichen Netzwerkprotokollen und APIs reichen. Anstatt jedem Client alle Softwareapplikationen anzubieten, filtert der Server die Liste von verfügbaren Wahlmöglichkeiten anhand der Eigenschaften des Client. Der Provisioning Server ist wie ein Imbissautomat, der weiß, dass ein Benutzer (Mensch) gegen Nüsse allergisch ist, und somit dem Benutzer keine Imbisse anbietet, die Nüsse enthalten.

Abbildung 7 illustriert die grundlegenden Konzepte und APIs, die die J2EE Client Provisioning Specification definiert.

[18] Referenzimplementierung: Implementierung einer Spezifikation, die als Referenz zum Entwickeln eigener Software dient und helfen soll Punkte der Spezifikation zu lösen, die sich als mehrdeutig herausstellen.

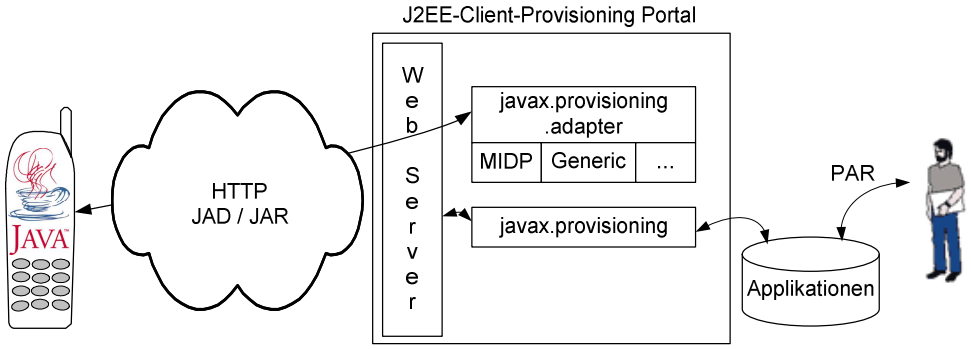


Abbildung 7: Client Provisioning Architektur

Die J2EE-Client-Provisioning-Benutzeroberflächen ermöglichen es, die Applikationen zu durchforsten, nach bestimmten Applikationen zu suchen und die Applikationen am Server zu verwalten. Das Administrations-Package *javax.provisioning* stellt für diese Funktionalität notwendigen Klassen und Objekte zur Verfügung. Das Provisioning-Package *javax.provisioning.adapter,* auf der anderen Seite, ist für das eigentliche Provisioning für die verschiedenen Geräte und Technologien (MIDP 1.0/2.0, Webapplets,…) verantwortlich. Sämtliche Packages der JSR124-API sind auf den Webserver der J2EE Edition angewiesen über den sie ihre Funktionalität zur Verfügung stellen.

Die JSR124-API inkludiert zahlreiche XML-Dateien [XMLW], die zum Konfigurieren des Frameworks benötigt werden. Die Spezifikation definiert auch ein eigenes Dateiformat, das Provisioning-Archiv (PAR-Datei[19]), das benutzt wird um Applikationen unabhängig von der Distributionsart auf den Server zu stellen (Deploying[20]).

Applikationspakete, die vom J2EE-Client-Provisioning-Server verwaltet werden, werden „Bundle“ genannt. Ein Bundle kann eine oder mehrere Applikationen (MIDlets) enthalten.

3.2.2 JSR-124 Referenzimplementierung

Für die JSR124-Spezifikation stand ab März 2003 eine Referenzimplementierung in der Version 1.0Beta [JSR124RI] zur Verfügung, die aufgegriffen wurde und an die Projektziele angepasst bzw. erweitert wurde. Diese Referenzimplementierung spielt

[19] Archivdatei, die eine Menge von MIDlet-Suites und eine Beschreibungsdatei (Descriptor) des PAR-Datei Inhaltes enthält. Genauere Details dazu siehe [JSR124V03] Kapitel 9.2.

[20] Deploying: Hochladen einer Applikation und zur Verfügung stellen der Applikation.

eine wichtige Rolle in dieser Arbeit, denn sie diente als Ausgangsbasis für die Entwicklung des Systems.

Das Suchen nach Applikationen ist bei der JSR124-Referenzimplementierung schon vollständig integriert. Dabei besitzt der Benutzer die Möglichkeit die Liste sämtlicher Applikationen (MIDlet-Suites) zu durchstöbern oder gezielt nach Applikationen in der Applikationsdatenbank zu suchen. Die Applikationsdatenbank wird beim JSR124 auch „Repository" genannt und stellt viele verschiedene Funktionen zum Zugriff auf die Applikationen zur Verfügung. Dies ist die vollständige Verwaltung der Applikationen, also MIDlet-Suites bereitstellen, löschen und auflisten.

Die JSR124-Referenzimplementierung beinhaltet weiters Funktionen um die Installationsmeldungen und Löschmeldungen der mobilen Endgeräte zu empfangen und zu verwalten.

3.3 Zusammenfassung

Die Over-The-Air-Applikationsdistribution, die in diesem Kapitel beschrieben wurde, enthält zahlreiche Sicherheitsprobleme, die in Abschnitt 3.1.3 behandelt wurden. Zu diesen Problemen kommt noch hinzu, dass bei MIDP1.0-konformen Geräten der Benutzer keinen Aufschluss darüber erhält, welche Funktionen die Applikation auf seinem Gerät ausführt. Der Benutzer muss also der Applikation sein Vertrauen schenken und darauf hoffen, dass die Applikation keine unvorteilhaften Funktionen durchführt. Weiters erhält der Benutzer beim Over-The-Air-Download einer Applikation keine Gewissheit darüber, dass eine Applikation nicht manipuliert worden ist bzw. dass die Applikation wirklich diejenige ist, die der Benutzer angefordert hat.

Letztere zwei Probleme werden mit den vertrauenswürdigen Applikationen (Trusted-Applications) in der MIDP2.0-Spezifikation behoben. Wenn der Benutzer eine Applikation mit Signatur heruntergeladen hat, erhält er eine Fehlermeldung sollte die Applikation manipuliert worden sein, und weiters erhält der Benutzer Aufschluss darüber von welchem Hersteller die Applikation stammt (anhand der Signatur und deren Prüfung).

Die Referenzimplementierung des JSR-124, die in Abschnitt 3.2 vorgestellt wurde, bietet zahlreiche grundlegende Funktionalität, um einen Provisioning Server zu erstellen. Um den Provisioning Server individuell anzupassen, bietet die Referenzimplementierung die Konfiguration über XML-Dateien. Jedoch muss gesagt werden, dass diese Anpassung nur bis zu einem bestimmten Punkt möglich

ist. Die in dieser Arbeit geforderte Funktionalität, wie zum Beispiel die Download-URL umzugestalten, können nicht konfiguriert werden. Im Rahmen der Arbeit musste daher ein Reengineering der JSR124-Referenzimplementierung durchgeführt werden (siehe Abschnitt 7.1.1). Für einen genaueren Einblick in die Java-Service-Request-124-Referenzimplementierung sollte [JSR124V03] konsultiert werden. Weiters findet man dort auch eine detaillierte technische Dokumentation zum Aufbau der Klassen und deren Funktionalität.

4 Aufgabenstellung

Nichtstun macht nur dann Spaß
wenn man eigentlich viel zu tun hätte.

Noël Coward

Das Ziel meiner Diplomarbeit war ein sicheres Applikationsmanagementsystem auf Basis von Java 2 Micro Edition, Connected Limited Device Configuration und Mobile Information Device Profile (J2ME, CDLC, MIDP) aufzubauen.

Dieses Kapitel behandelt die Projektziele, Rahmenbedingungen und die sich daraus ergebenden Aufgaben.

4.1 Ziele

Ein grundlegendes Ziel dieser Arbeit war eine Authentifizierung der Benutzer gegenüber dem Hersteller zu schaffen. Damit besitzt der Hersteller bzw. Applikationsvertreiber die Möglichkeit einen Zugriffsschutz auf seine Applikationen zu installieren und kann somit nur ausgewählten Benutzern den Zugriff erlauben. Zusätzlich ist es dadurch möglich einen Kaufvertrag für den Erwerb der Applikation abzuschließen. Dies ist vor allem für Mobilfunkbetreiber interessant, denn diese können den Kauf direkt in ihrem Verrechnungssystem abbuchen.

Ein weiteres wichtiges Ziel war die Authentifizierung des Herstellers gegenüber dem Benutzer. Damit hat der Benutzer die Sicherheit, dass die Applikation von einem gewissen Anbieter stammt. Diese Funktionalität stellt somit eine Art Gütesiegel für die Applikationen dar[21].

Wichtig für diese Arbeit war weiters der Schutz der Applikationen vor Manipulationen. Da beim OTA unverschlüsselte HTTP-Verbindungen genutzt werden, kann ein versierter Angreifer den Datenverkehr abhören und auch gegebenenfalls manipulieren. Auch dieses Manko des OTA sollte beseitigt werden.

[21] Es wird dabei angenommen, dass die Qualität der Applikationen durch den Hersteller überprüft wird.

Ein weiteres Ziel, das nicht direkt das sichere AMS betrifft, war eine Profilbildung über die verschiedenen Benutzer zu ermöglichen. Dabei geht es prinzipiell um Versionskontrolle, also welche Applikation welcher Benutzer besitzt, und zusätzlich um Präferenzen bestimmter Benutzer ausfindig zu machen. Es muss aber trotzdem auch ein anonymer Vertrieb möglich sein. Das letzte Ziel liegt besonders im Interesse des Mobilfunkbetreibers, denn dieser hat dadurch die Möglichkeit Präferenzen seiner Benutzer zu bestimmen, aber trotzdem die Privatsphäre der Benutzer nicht zu verletzen. Diese Profilbildung wird durch das Sammeln der Statusmeldungen von Seiten des mobilen Endgeräts erleichtert und stellt somit ein weiteres Ziel dieses Projektes dar.

Das System soll außerdem eine leichte Einbindung eines externen Abrechnungssystems ermöglichen, um dem Mobilfunkbetreiber die Möglichkeit zu geben direkt den Kauf der Applikationen in seinem Abrechnungssystem zu verrechnen.

4.2 Überlegungen und Rahmenbedingungen

In diesem Abschnitt werden Überlegungen zu den Zielen angestellt und verschiedene Lösungsmöglichkeiten durchleuchtet. Dabei werden die Rahmenbedingungen erläutert und die daraus resultierenden Lösungen diskutiert.

4.2.1 Endgerät

Um die gewünschte Funktionalität zu implementieren bestehen die Lösungsansätze auf Geräteseite in der Modifikation des AMS bzw. in der Erstellung eines AMS auf Applikationsebene. Beide Lösungsansätze sind aus den folgenden Gründen nicht durchführbar:

Für die Modifikation des AMS sind die Probleme wirtschaftlicher und politischer Natur. Das Mobile-Information-Device-Profile (MIDP) stellt nur geringe Anforderungen an die AMS, damit möglichst große Interoperabilität gewährleistet ist und die Endgerätehersteller viele Freiheiten in der Implementierung besitzen. Um ein sicheres Applikationsmanagement zu erreichen, müssten aber sowohl das herstellerspezifische AMS jedes Endgerätes modifiziert werden und an entscheidender Stelle auch in den MIDP- Standard eingegriffen werden. Beide Ziele waren im Rahmen dieses Projektes nicht umsetzbar, da der Aufwand und die Kosten für eine solche Entwicklung den Rahmen einer Diplomarbeit und eines Diplomarbeitprojektes weit überragen würden und hätten zusätzlich die Interoperabilität stark eingeschränkt.

Die Implementierung eines sicheren Applikationsmanagementsystems auf Applikationsebene andererseits wird sowohl in MIDP 1.0 als auch MIDP 2.0 durch die folgenden Eigenschaften von MIDP verhindert:

- Applikationen bzw. MIDlets können nur auf Daten der eigenen MIDlet-Suite zugreifen. Auf den Programmcode kann ausschließlich die AMS zugreifen. Somit kann ein MIDlet nicht über andere MIDlets verfügen bzw. diese verwalten
- Programmcode kann nicht dynamisch geladen und ausgeführt werden. Das bedeutet, dass keine Applikation ein anderes Programm laden und starten kann
- Ein MIDlet hat keinen Zugriff auf die Applikationsmanagementsoftware (AMS). Funktionen wie die Überprüfung der Signatur können nicht angestoßen werden

Im Rahmen dieses Projektes bestehen auf Endgeräteseite somit wenige Möglichkeiten Adaptionen vorzunehmen. Die AMS ist durch den MIDP-Standard definiert und darf nicht verändert werden, da ansonsten das Sicherheitskonzept von J2ME nicht gewährleistet werden kann. Auch ist der Eingriff in die herstellerabhängigen Funktionen des Endgerätes im Sinne eines allgemein gültigen Ansatzes und im Sinne der Interoperabilität zwischen den Geräten nicht zielführend.

4.2.2 SIM-Karte

Eine weitere Möglichkeit ist die Integration von Zusatzfunktionalität auf der SIM-Karte des mobilen Endgerätes. Hat eine SIM-Karte bzw. ein Endgerät mit zwei Kartenlesern für eine zusätzliche Kryptographiechipkarte die Möglichkeit Nachrichten zu ver- und entschlüsseln bzw. zu signieren, so ist es möglich dem Besitzer des Endgerätes einen elektronischen Kaufvertrag unterschreiben zu lassen, und ein einmalgültiges Benutzername/Passwort Paar sicher zuzusenden aber auch eine Authentifizierung anhand einer PKI durchzuführen. Diese Funktionalität wurde im WASA-Projekt [Schwaiger01] implementiert.

Dieses Konzept hat den Vorteil, eine sichere Authentifizierung zu ermöglichen, da es folgende Mängel in *MIDP2.0* beseitigt:

- Es ist keine sichere Aufbewahrung von geheimen Schlüsseln oder Passwörtern am Gerät möglich, da ein Benutzer auf seinen Gerätespeicher vollen Zugriff besitzt

- Die AMS kann auch nach MIDP2.0 keine sichere (Benutzer-) Authentifizierung anhand einer PKI durchführen

Es gibt bereits Ansätze zur Einführung eines Standards, der diese Probleme beheben soll. Im JSR-177 [JSR177] unter dem Titel „Security and Trust Services API for J2ME“ wurde im April 2002 eine Initiative gestartet, die an einer Spezifikation für Sicherheitsservices für J2ME-fähige Endgeräte arbeitet. Diese APIs nach der JSR-177-Spezifikation werden es ermöglichen Sicherheitsmechanismen für J2ME-fähige Endgeräte zu implementieren, wie es in diesem Projekt erforderlich ist. Diese Spezifikation befindet sich im jetzigen Zeitpunkt in der Überarbeitungsphase und hat noch immer nicht den Veröffentlichungsstatus erreicht, obwohl der im 2. Quartal 2003 geplant war. Momentan ist deshalb noch kein Zugriff auf die SIM-Karte möglich.

4.2.3 Übertragungsprotokolle

Die MIDP2.0-Spezifikation [MIDP20] legt fest, dass alle kompatiblen Geräte für die Kommunikation von Benutzerapplikationen sowohl HTTP als auch HTTP über SSL (HTTPS[22]) unterstützen müssen. Jedem Hersteller steht es offen noch weitere Verbindungsmöglichkeiten, wie z.B. Sockets, oder serielle Verbindungen zu implementieren, die aber nicht verpflichtend sind und damit für einen interoperablen Einsatz nicht standardmäßig verfügbar sind.

Die AMS hingegen, die den OTA-Download durchführt, muss lediglich eine Kommunikation mit HTTP unterstützen. Die Kommunikation über andere Kanäle ist nicht vorgeschrieben und wird daher meist herstellerspezifisch angelegt. Aus diesem Grund musste das Übertragungsprotokoll für diese Arbeit auf das unverschlüsselte HTTP festgelegt werden. Dies hat jedoch zur Konsequenz, dass die Übertragung der MIDlets nicht gesichert ist und von Dritten abgehört werden kann.

4.2.4 Signatur

Um dem Benutzer die Sicherheit zu geben, dass die Applikation authentisch ist, gibt es in MIDP2.0 die Signatur für eine MIDlet-Suite wie in Abschnitt 2.5.3 beschrieben wurde. Mit der Signaturüberprüfung, kann festgestellt werden, ob die Applikation manipuliert wurde und welcher Hersteller sie signiert hat.

Dies gibt dem Benutzer die Sicherheit, dass die Applikation ordnungsgemäß übertragen wurde und mit dem Vertrauen zum Hersteller ausgeführt werden kann.

[22] HTTPS: HTTP über eine sichere Verbindung: Secure Socket Layer [SSL]

Obwohl MIDP2.0 diese Signatur zum sicheren Applikationsmanagement anbietet, ist dieser Sicherheitsbausteine benutzerorientiert. Das heißt der Benutzer erhält die Gewissheit, dass eine Applikation nicht manipuliert worden ist, und der Benutzer muss entscheiden, ob er dem Hersteller vertraut.

Die Applikationsdistribution ist mit dieser Signaturverifizierung aber nicht abgesichert gegen Abhören des Downloads und Weiterverkaufs einer Applikation. Denn ein bösartiger Dritter kann immer noch den unverschlüsselten Downloadkanal abhören und eine signierte Applikation abfangen. Im nächsten Schritt entfernt er nun die Signatur von der Applikation, und kann nun die Applikation trotzdem wieder auf jedem Endgerät installieren und verändern. Denn Applikationen ohne Signatur werden vom AMS nicht mehr überprüft[23] und in den „unsicheren" Bereich (Untrusted Area) installiert.

4.3 Konklusion

Aufgrund der Überlegungen und Rahmenbedingungen aus Abschnitt 4.2 kann ein sicheres Applikationsmanagementsystem höchstens gerätespezifisch umgesetzt werden. Da aber ein bedeutendes Ziel von Java und J2ME die Geräteunabhängigkeit ist, wurden die Projektziele und Anforderungen geändert und an die Rahmenbedingungen angepasst:

Jedes Endgerät muss ein den *MIDP2.0*-Spezifikationen entsprechendes AMS für die Applikationsdistribution besitzen, um eine Applikationsmanipulation zu erkennen (siehe Abschnitt 4.2.4). Diese Geräte standen zum Zeitpunkt der Projektentwicklung noch nicht zur Verfügung. Um aber das Projekt trotzdem entwickeln und testen zu können wurde der WTK2.0-Simulator [WTK20] verwendet, der ein MIDP2.0-fähiges Gerät am PC emuliert.

Um eine sichere Authentifizierung zu implementieren wird, auf die Funktionalität einer Kryptographie-Chipkarte zugegriffen, denn nur diese Methode kann zum jetzigen Zeitpunkt eine sichere Authentifizierung gewährleisten (siehe Abschnitt 4.2.2). Um eine solche Kryptographiechipkarte zu verwenden, muss ein Endgerät, die Möglichkeit bieten gleichzeitig eine Kryptographiechipkarte und eine SIM-Karte ansprechen zu können. Ein solches Gerät steht zwar am Institut für Computertechnik der TU Wien zur Verfügung ist aber nicht J2ME-fähig. Deshalb

[23] Prinzipiell besteht auch die Möglichkeit ein MIDlet über einen Simulator, der meist vom Hersteller gratis zu beziehen ist, auf jedem PC auszuführen. In diesem Fall kann die AMS nicht mehr als manipulationsgeschützte Umgebung angesehen werden.

wurde in der Testphase eine weitere Authentifizierungsmöglichkeit (MSISDN[24] oder Benutzername und Passwort, siehe Abschnitt 5.1.2) implementiert, die nicht so sicher ist wie die Authentifizierung anhand von Zertifikaten, aber trotzdem den heutigen Sicherheitsbedürfnissen der Hersteller genügt.

Zusammenfassend kann gesagt werden, dass die im Projekt entwickelte serverseitige verteilte Infrastruktur die bestehenden Eigenschaften des AMS ausnützt und in den nicht spezifizierten Bereichen erweitert. Durch die Authentifizierung anhand einer Kryptographie-Chipkarte auf dem Endgerät wird eine sichere Authentifizierung gewährleistet. Dieses serverseitige System wird in J2ME Provisioningsystem genannt. Dieser Ausdruck wird auch in diesem Dokument im Weiteren verwendet.

[24] Mobilfunknetz spezifische Kundennummer, im Normalfall die mobile Rufnummer.

5 Systemkonzept

Verantwortlich ist man nicht nur für das, was man tut,
sondern auch für das, was man nicht tut.

Laotse, chin. Philosoph

In diesem Kapitel wird die entwickelte Lösung präsentiert. Dabei wird zu Beginn das grundlegende Konzept aufgezeigt, in Abschnitt 5.1 Überlegungen und Begründungen dazu angestellt und schließlich in Abschnitt 5.2 auf die Systemkomponenten eingegangen.

Abbildung 8 stellt die Hauptkomponenten des Systems dar, die im Folgenden näher spezifiziert werden. Eine Ausnahme stellt hierbei das externe Abrechnungssystem (Billing System) dar, zu dem lediglich eine Schnittstelle entwickelt wurde, um dieses einfach einbinden und ansprechen zu können.

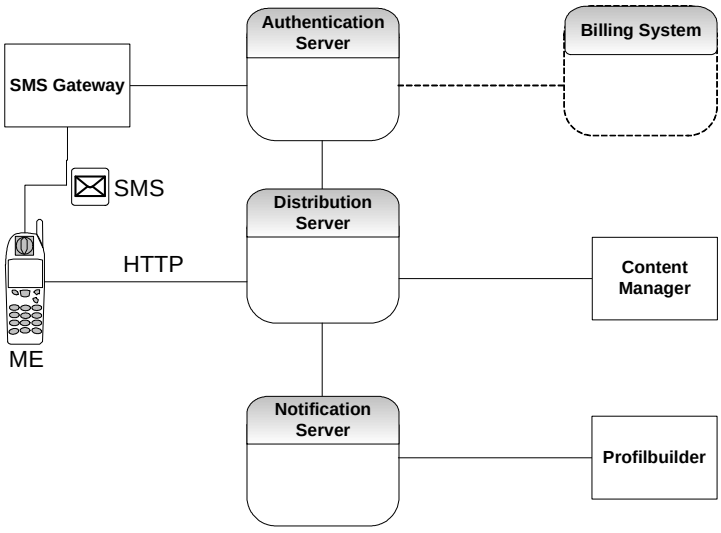


Abbildung 8: Überblick Systemaufbau

Die Aufgaben der einzelnen Komponenten sind:

- Der Distribution Server [DISTS03] kommuniziert mit dem Benutzer bzw. dem mobilen Endgerät und stellt diesem die Applikationen zur Verfügung. Der Distribution Server besitzt weiters einen Content-Manager, mit dem der Server-Administrator die Applikationen am Server verwalten kann
- Der Authentication Server [AUTHS03] übernimmt die Authentifizierung der Benutzer, anhand von verschlüsselten SMS-Nachrichten die über das SMS-Gateway geschickt werden
- Der Notification Server [NOTS03] sammelt sämtliche Statusmeldungen und verwaltet diese. Zusätzlich ist über den Notification Server eine Profilbildung über die Benutzer möglich
- Das externe Abrechnungssystem (Billing System) dient zur Abrechnung der Applikationen

5.1 Überlegungen

In diesem Abschnitt werden die Überlegungen zu dem entwickelten Konzept aufgeführt, die zu dieser Lösung geführt haben. Dabei werden das verteilte System, die Authentifizierung, die Downloadmethode und die Bearbeitung der Statusmeldungen behandelt.

5.1.1 Verteiltes System

Die Entscheidung ein verteiltes System zu entwickeln, lag in der Zuordnung der Funktionalität zu bestimmten Benutzergruppen. Jeder Server ist einer Rolle bzw. Benutzergruppe innerhalb des Systems zugedacht:

- Der Authentication Server ist dem Mobilfunkbetreiber zugeordnet. Da der Mobilfunkbetreiber über seine Benutzer (Kunden) sämtliche Informationen hat, und weiters die Möglichkeit besitzt Rechnungen auszustellen. Der Authentication Server stellt eine Vertrauensrolle im System dar
- Der Distribution Server ist dem Applikationshersteller und dem Applikationsvertreiber zugeordnet. Diese Vertreiben ihre Applikationen über ihren Distribution Server und greifen für eine Authentifizierung auf den Authentication Server bzw. dem Mobilfunkbetreiber zurück
- Der Notification Server bzw. mehrere kaskadierte Notification Server (siehe Abschnitt 7.4.4) sind dem Infrastrukturbetreiber, dem Hersteller und dem

Vertreiber zugeordnet. Sie alle bekommen die Möglichkeit Informationen über Benutzerpräferenzen und eine Versionskontrolle zu erhalten

Durch dieses verteilte System ist die Gewährleistung der Anonymität der Benutzer möglich. So ist in diesem Systemkonzept der Authentication Server für die Authentifizierung der Benutzer zuständig und nur der Authentication Server weiß auch über deren Identität Bescheid. Dies ist damit gerechtfertigt, da der Mobilfunkbetreiber sowieso über die Benutzerdaten seiner Kunden verfügen kann, und somit deren Identität bestimmen kann.

5.1.2 Authentifizierung

Für die Authentifizierung des Benutzers gegenüber dem Authentication Server stehen prinzipiell zwei verschiedene Varianten zur Verfügung:

1. MSISDN[25] oder Benutzername mit zugehörigem Passwort
2. Zertifikate und asymmetrische Kryptographie

Der Vorteil der Methode mit MSISDN (oder Benutzername) und Passwort ist der relativ kleine technische Aufwand für die Implementierung. Eine Benutzername-Passwort–Kombination ist im Allgemeinen aber nicht sehr benutzerfreundlich, da um wirkliche Sicherheit zu garantieren, gewisse Regeln eingehalten werden sollten. Das Passwort sollte regelmäßig geändert werden und für jemand anderes schwer zu erraten sein, aber trotzdem sollte das Passwort für den Benutzer leicht zu merken sein.

Aus den oben angeführten Gründen wird für dieses Projekt die Variante mit einem Zertifikat bevorzugt. Zu Testzwecken wurde aber ebenso die Authentifizierungsmethode anhand MSISDN (oder Benutzername) und Passwort implementiert.

Es gibt verschiedene Möglichkeiten, wie sich ein Benutzer mit seinem Zertifikat authentifizieren kann:

1. Browser (Discovery Application) der HTTPS-Client-Authentication unterstützt
2. MIDlet, das die Authentifizierung übernimmt
3. Signierte (verschlüsselte) SMS mit Kryptographie-Chipkarte

[25] MSISDN: Mobile Subscriber-Integrated Service Digital Network.

Für die erste Variante wird ein Browser vorausgesetzt, der eine Verschlüsselung auf der Transportschicht (TSL/SSL [SSL] oder WTLS [WTLS]) ermöglicht, und somit die Authentifizierung des Clients übernimmt. Weder die gängigen Browser für den PC, noch die WAP-Browser, noch die AMS der mobilen Endgeräte unterstützen derzeit diese Option.

Die zweite Variante, die Authentifizierung mit einem MIDlet, kann zurzeit ebenfalls nicht verwendet werden, da es noch keine Möglichkeit gibt, per MIDlet auf eine Chipkarte zuzugreifen. Und nur auf einer Chipkarte kann das Zertifikat und die dazugehörigen Schlüssel sicher verwahrt werden. Die dafür notwendigen Voraussetzungen werden im Java-Service-Request-177 [JSR177] spezifiziert. Die Veröffentlichung dieser Spezifikation (JSR177) war für das 2. Quartal 2003 geplant, ist aber nochmals verschoben worden und ist zum jetzigen Zeitpunkt nicht erfolgt.

Bei der dritten Variante, die Signierung des Inhalts einer SMS mit Hilfe einer Kryptographie-Chipkarte, wird dem Benutzer eine SMS zugesandt, die er signiert und dem Authentication Server zurücksendet. Dieser kann dann die Signatur überprüfen und feststellen, von welcher Person diese Nachricht signiert wurde und so die Identität feststellen.

In diesem Projekt wurde als Authentifizierung des Benutzers die signierte (verschlüsselte) SMS-Nachricht ausgewählt, da sie die einzige sichere Variante ist, die auf heute verfügbaren mobilen Endgeräten implementierbar ist, und da diese Funktionalität im WASA Projekt [Schwaiger01] schon vorhanden ist. Zusätzlich wurde in der Entwicklungsphase die Authentifizierungsmethode durch die MSISDN (oder Benutzername) und Passwort implementiert, um das System auch ohne Kryptographie-Chipkarte testen zu können, da zum Zeitpunkt der Entwicklung des Systems kein Endgerät mit J2ME und Dual-Slot-Kryptographiechipkarte am Markt verfügbar war. Letztere Authentifizierungsmethode ist nicht so sicher wie die Authentifizierung anhand von signierten (verschlüsselten) SMS, genügt aber den heutigen Sicherheitsbedürfnissen der Mobilfunkbetreiber.

5.1.3 Downloadmethode

Für die Installation eines MIDlets ist das Application-Management-System (AMS) des mobilen Endgerätes zuständig. Unabhängig davon wie die JAR-Datei auf das Gerät gelangt (z.B. OTA) kann eine JAD-Datei zur Verfügung gestellt werden (siehe Abschnitt 3.1.1). Es besteht dazu aber keine Verpflichtung, deshalb ist mit folgenden drei Downloadmethoden zu rechnen:

- Download mit JAD-Datei

- Download ohne JAD-Datei
- Download mit einer Java-Applikation

Download mit JAD-Datei

Bei dieser Variante wird zuerst die JAD-Datei auf das mobile Endgerät geladen. Die AMS kann anhand der in der Datei enthaltenen Attribute eine Vorentscheidung treffen, ob das MIDlet installiert werden kann oder nicht (siehe Abschnitt 2.4). Nachdem der Benutzer zusätzlich bestätigt hat, dass das MIDlet installiert werden soll, lädt die AMS automatisch die JAR-Datei von dem im MIDlet-Jar-URL angegebenen Attribut auf das mobile Endgerät.

Die Vorteile und Nachteile dieser Variante sind:

+ Der Benutzer muss nicht das gesamte MIDlet auf sein Gerät laden um festzustellen, ob eine Installation möglich ist

+ Die Notification-URLs können in der JAD-Datei dynamisch eingetragen werden und somit ist keine Änderung am Manifest in der JAR-Datei notwendig. Man muss daher eine MIDlet-Suite (JAR-Datei) nur einmal signieren und kann diese dann ohne Bedenken zur Verfügung stellen. Diese Notification-URLs werden für jeden Download dynamisch erzeugt und müssen deshalb dynamisch in die JAD-Datei eingetragen werden, siehe Abschnitt 6.1

− Die JAD-Datei wird vom AMS über eine unverschlüsselte HTTP-Verbindung übertragen und ist daher nicht sicher zum Gerät übermittelbar

− Die URL für den Download des MIDlets ist in der JAD-Datei enthalten und kann von Dritten mitgelesen werden

− Auch die JAR-Datei wird vom AMS mit HTTP übertragen und ist daher nicht sicher zum Gerät übertragbar

Download ohne JAD-Datei

Der AMS wird direkt die URL der JAR-Datei von der Discovery-Application (DA) mitgeteilt. Der Download der JAD-Datei laut Abschnitt 2.3 wird sozusagen übersprungen. Nach der Bestätigung des Benutzers startet die AMS den Download der JAR-Datei.

Der Vorteil dieser Methode ist:

+ Der Download kann von einer einmal gültigen URL erfolgen, die dem Benutzer auf sicherem Weg mitgeteilt werden kann

Die Nachteile sind:

- Es besteht die Möglichkeit, dass eine heruntergeladene und bezahlte MIDlet-Suite am mobilen Endgerät des Benutzers nicht installiert werden kann. Zum Beispiel aufgrund zu geringen Ressourcen oder durch einen Downloadabbruch durch höhere Gewalt
- Die Notification-URLs müssen im Manifest gespeichert und geändert werden. Deshalb muss bei jeder Änderung der Notification-URLs das MIDlet neu signiert werden. Dies ist aber nicht wünschenswert, da somit ein Mehraufwand für den Distribution Server Betreiber entsteht
- Die JAR-Datei wird vom AMS über eine unverschlüsselte HTTP-Verbindung übertragen und ist daher nicht sicher zum Gerät übermittelbar

Download mit einer Java-Applikation

Eine weitere Möglichkeit der Installation ist der Download der MIDlet-Suite mit einem speziellen Download-MIDlet, das sowohl die JAD-Datei als auch die JAR-Datei herunterlädt. Dazu kann nach MIDP 2.0 das verschlüsselte SSL Verbindungsprotokoll verwendet werden. Anschließend kann das herunter geladene MIDlet manuell vom Benutzer per AMS installiert werden. Dieses MIDlet bildet sozusagen die AMS-Funktionalität nach.

Die Vorteile sind:

+ Kein Mithören der Kommunikation von Dritten, da die Kommunikation zwischen mobilen Endgerät und Server auf der Transportschicht mit SSL gesichert werden kann. Somit wird die JAR-Datei über verschlüsselte Verbindung übertragen
+ Identifizierung des Servers (mit Zertifikat), welche im SSL-Protokoll notwendig ist
+ Änderungen der Notification-URLs in der JAD-Datei sind möglich

Die Nachteile sind:

- Doppelter Speicherbedarf am Endgerät, da das MIDlet zuerst abgespeichert werden muss und dann erst vom AMS installiert werden kann
- Für die Identifizierung des Servers ist ein Zertifikat notwendig, das am mobilen Endgerät nicht sicher aufbewahrt werden kann
- Die Installation aufgrund der Beschränkungen von *MIDP* (siehe Abschnitt 4.2.1) ist nur proprietär möglich und nicht benutzerfreundlich

Auswahl der Downloadmethode

Da der Erfolg eines Systems zu einem großen Teil von der Akzeptanz der Benutzer abhängt, wurde die Methode mit dem *Download der JAD-Datei* gewählt. Sie ist die benutzerfreundlichste Methode, da dem Benutzer so bald als möglich –bereits nach Download der JAD-Datei, die eine sehr geringe Größe aufweist- mitgeteilt werden kann, ob eine Installation am mobilen Endgerät möglich ist. Der Download ohne JAD-Datei bietet diese Möglichkeit nicht und scheidet auch wegen der Notwendigkeit der Generierung der dynamischen Notification-URLs und des damit verbundenen Zertifizierungsaufwandes aus.

Der Download mit einer Java-Applikation wurde aus den Gründen in Abschnitt 4.2.1 und aus den Nachteilen aus diesem Abschnitt nicht gewählt. Der immense Vorteil dieser Methode wäre die sichere verschlüsselte Übertragung, der aber gravierende Nachteile gegenüberstehen.

In einer der nächsten Versionen des MIDP wird voraussichtlich der Download durch die AMS auch mit SSL-Verschlüsselung ermöglicht werden. Dies wird von zahlreichen Herstellern schon propagiert, ein Erscheinungstermin ist aber nicht abzusehen. Durch die SSL-Verschlüsselung werden die Nachteile dieser Downloadmethode eliminiert. Die in dieser Arbeit implementierte Infrastruktur hält daher an den OTA-Download-Standard fest, damit sie bei zukünftige Versionen des OTA-Downloads mit SSL-Verschlüsselung nur mit geringfügigen Änderungen weiterverwendbar bleibt.

5.1.4 AMS-Statusmeldungen

Vom Provisioningsystem kann durch das HTTP-Protokoll nur mit Sicherheit festgestellt werden, dass eine JAR- bzw. eine JAD-Datei vollständig übertragen wurde. Im Sinne einer vollständigen Informationskette wird aber auch die Information über die korrekte Installation und Deinstallation benötigt. Diese Information wird über die AMS-Statusmeldungen übertragen.

Um diese Statusmeldungen von der AMS zu erhalten, gibt es in MIDP1.0 das *install_confirm_URL*-Attribut in der JAD-Datei. An die dort angegebene URL wird vom AMS das Ergebnis des Installationsvorganges zurückgemeldet. MIDP2.0 hat dieses Konzept um das *delete_confirm_URL*-Attribut erweitert, an das analog zum erstgenannten Attribut von der AMS eine Benachrichtigung über die Löschung der Applikation gesendet werden kann (siehe Abschnitt 2.4.2).

Im Sinne eines sicheren Applikationsmanagements können diese Nachrichten nur als informativ betrachtet werden, da:

- Gemäß MIDP-Spezifikation bei fehlender oder unterdrückter Netzverbindung zwar Wiederholungen empfohlen werden aber das Absenden der Nachrichten nicht garantiert wird
- Applikationen über andere Kanäle gelöscht bzw. installiert werden können. Zum Beispiel kann beim mobilen Gerät: Siemens-SL45i [SL45i] der Applikationsspeicher direkt per Datenkabel und PC verwaltet werden, also Applikationen gelöscht und installiert werden
- Die Einträge können jederzeit aus der JAD-Datei gelöscht werden, da diese Datei vom Benutzer im Klartext (human-readable) einsehbar ist und gegen Manipulation nicht gesichert ist

Im S3GIS-Provisioningsystem werden die vom AMS versendeten Statusmeldungen vom Notification Server (siehe Kapitel 5.2.3) aufgezeichnet. Um zu demonstrieren mit welcher Qualität und in welcher Quantität Informationen über MIDP gesammelt werden können, werden so viele Daten wie möglich gesammelt. Ob Daten im Sinne des Datenschutzes überhaupt erhoben werden dürfen, soll für diese Arbeit kein Grund zu Beschränkungen geben, auch wenn in der Realität die Privatsphäre in den letzten Jahren immer wichtiger geworden ist.

5.2 Systemkomponenten

In Abbildung 9 ist der Systemaufbau detailliert dargestellt, wobei die wichtigste Funktionalität des jeweiligen Systemteils skizziert ist. In diesem Abschnitt werden diese einzelnen Systemkomponenten und deren Aufgabenbereich aufgezeigt.

5.2.1 Distribution Server

Der Distribution Server [DISTS03] ist das zentrale Element, das Java Applikationen (MIDlet-Suites) dem mobilen Endgerät zur Verfügung stellt. Der Distribution Server dient zum Suchen und Herunterladen der Applikationen. Außerdem besitzt er eine Schnittstelle, die es dem Applikationsvertreiber ermöglicht neue Applikationen auf den Server zu legen und diese dort zu verwalten (Content Manager). Weiters steht der Distribution Server in Kommunikationsverbindung zum Notification Server und leitet sämtliche Statusmeldungen an diesen weiter.

Das S3GIS-Projekt ist so konzipiert, dass jeder Applikationsvertreiber über einen Distribution Server verfügt. Natürlich ist es auch möglich, dass mehrere Applikationsvertreiber einen Distribution Server gemeinsam nutzen, dabei müssen sie sich auch alle Daten und Informationen teilen. Grundsätzlich ist es so

konzipiert, dass einem Applikationsvertreiber ein Distribution Server zum Vertreiben seiner Applikationen eindeutig zugeordnet ist.

Daraus ergibt sich die Möglichkeit, dass mehrere verschiedene Distribution Server mit einem Authentication Server kommunizieren. Der Authentication Server identifiziert die einzelnen Distribution Server anhand deren eindeutigen ID die ihnen vom Infrastrukturbetreiber gegeben wurde. Details zur Kommunikation sind in [XMLKOM] zu finden.

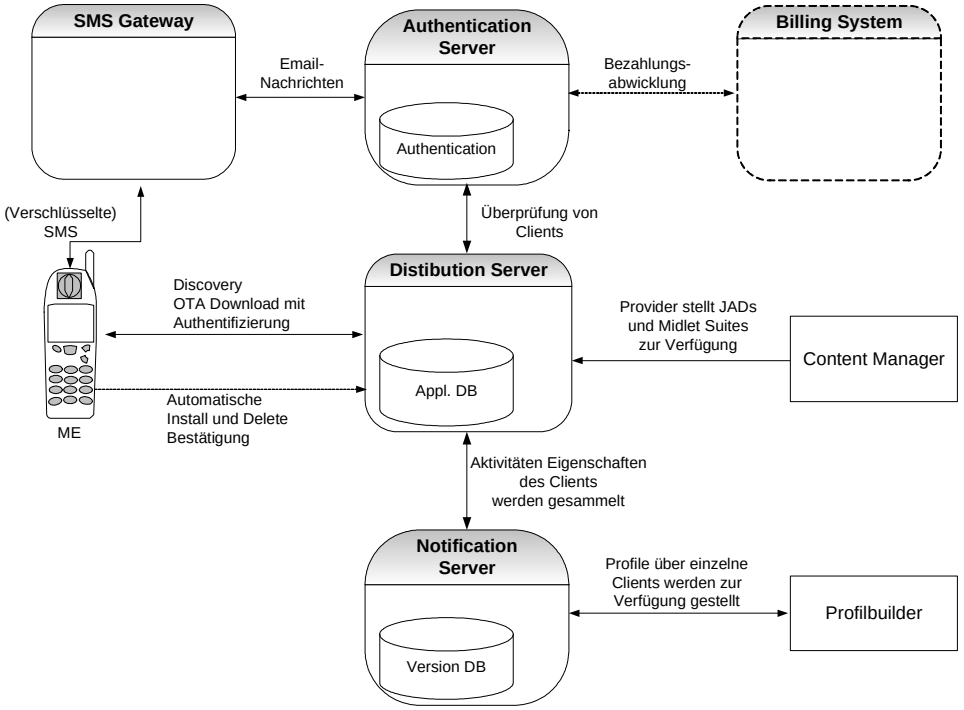


Abbildung 9: Systemüberblick (detailliert)

Der Distribution Server bietet folgende Funktionalität:

Suchen von Applikationen

Beim Suchen von Applikationen hat der Benutzer die Möglichkeit die gesamten am Server befindlichen MIDlet-Suites anhand gewisser Kriterien zu filtern und auch nach gewissen Stichwörtern innerhalb der MIDlet-Beschreibung zu suchen.

Nach der Angabe der Filter (z.B. Applikationstyp, Kategorie oder Sprache) und Eingabe von Suchwörtern kann die Anfrage an den Distribution Server abgeschickt werden und dieser liefert als Ergebnis eine Liste von passenden Applikationen. Im nächsten Schritt kann dann der Download gestartet werden.

Herunterladen von Applikationen

Hat der Benutzer eine Applikation aus der Ergebnisliste selektiert oder wählt er eine Applikation aus der gesamten MIDlet-Suite-Liste, wird der Download der Applikation gestartet. Bevor aber der eigentliche OTA-Download gestartet wird, muss sich der Benutzer authentifizieren und wird dazu zum Authentication Server weitergeleitet.

Nach erfolgreicher Authentifizierung kann die Applikation heruntergeladen werden.

Meldungen an Notification Server

Der Distribution Server erhält als zentrales Element sämtliche Nachrichten und Statusmeldungen im Provisioningsystem[26], schickt aber sämtliche Meldungen an den bzw. die Notification Server weiter. Durch das Empfangen der Statusmeldungen hat der Distribution Server als zentrales Element die Möglichkeit Nachrichten zu filtern und weiterzuleiten, und dient damit als Vorstufe zu weiteren Notification Servern.

Content Manager

Der Content Manager ist eine der Hauptbenutzerschnittstellen des Distribution Servers und bietet folgende Funktionen an:

- Anzeige von Applikationen auf dem Distribution Server
- Einspielen einer neuen Applikation bzw. einer neuen Version auf dem Distribution Server (Update)
- Löschen einer Version auf dem Distribution Server

Die Schnittstelle für diese Aufgabe ist als Web-Interface konzipiert, da dieses keine Installation auf dem Rechner des Applikationsvertreibers erfordert und mit Standardsoftware (Browser) betrieben werden kann.

5.2.2 Authentication Server

Der Authentication Server [AUTHS03] dient als Trusted-Third-Party[27]. Er überprüft die Identität des Händlers und Endbenutzers, verschlüsselt bei Bedarf die Kommunikation und kann mit Hilfe von einem externen Abrechnungssystem

[26] Statusmeldungen kommen vom mobilen Endgerät des Benutzers und vom Authentication Server.

[27] Unabhängige dritte Instanz der (a priori) Vertrauen entgegengebracht wird.

(Billing System) überprüfen, ob die beteiligten Geschäftspartner berechtigt und in der Lage sind den miteinander eingegangenen Vertrag zu erfüllen.

Der Authentication Server stellt weiters sicher, dass keine sensiblen Benutzerdaten weitergegeben werden, und verschlüsselt bzw. signiert die Nachrichten, die mit den Benutzern ausgetauscht werden.

Optional kann der Authentication Server auch die Kommunikation mit einem Abrechnungssystem übernehmen. Dies wird durch die implementierte Schnittstelle ermöglicht, ist aber außerhalb des Projektes.

Der Authentication Server ist verantwortlich für:

- Authentifizierung des Händlers
- Authentifizierung des Benutzers
- Überprüfung der Zugriffsrechte auf ein MIDlet-Suite
- Überprüfung und Abwicklung der Bezahlung des Downloads mit einem externen Abrechnungssystem (Billing System). Hierzu wurde eine einfache leistungsfähige Schnittstelle implementiert
- Anonymisierung des Benutzers
- Verschlüsselung der JAD-Datei-Download-URL zum Versand per SMS
- Überprüfung der Signaturen

Überprüfung und Abwicklung der Bezahlung des Downloads

Der Authentication Server kann die Liquidität des Benutzers durch eine Anfrage an ein externes Abrechnungssystem (Billing System) überprüfen. Für diese Kommunikation mit dem externen Abrechnungssystem wurde in dieser Arbeit eine Schnittstelle geschaffen, mit der es leicht möglich ist, ein externes Abrechnungssystem einzubinden. Die Information über den Abschluss eines Downloads wird auch an das externe Abrechnungssystem vom Distribution Server über den Authentication Server weitergeleitet.

Überprüfung der Signatur

Um eine sichere Authentifizierung des Benutzers am Authentication Server zu realisieren, wird eine Authentifizierung anhand von Zertifikaten verwendet, so wie es im Abschnitt 5.1.2 erläutert wurde. Für diese Authentifizierung muss der Authentication Server imstande sein, Zertifikate zu überprüfen und zu verifizieren.

Diese Funktionalität wurde bereits im WASA-Projekt entwickelt. Diese Überprüfung der Signaturen wird nach folgendem Ablauf eingesetzt:

1. Das Angebot des Distribution Servers wird vom Authentication Server signiert, die Server vertrauen sich untereinander, da sie über verschlüsselte Verbindungen und Authentifizierung miteinander kommunizieren (siehe Abschnitt 6.1.1)
2. Der Authentication Server sendet die signierte Nachricht an den Benutzer
3. Der Benutzer, bzw. die JavaCard-Applikation des Benutzers verifiziert die Signatur des Authentication Servers und präsentiert dem Benutzer die Daten aus der Nachricht
4. Nimmt der Benutzer das Angebot an, signiert die JavaCard-Applikation das Angebot ihrerseits und schickt es dem Authentication Server zurück
5. Der Authentication Server verifiziert die Signatur des Benutzers und teilt bei Erfolg dem Distribution Server mit, dass die Authentifizierung erfolgreich war

Der Authentication Server verwendet für die Überprüfung der Zertifikate die Dienste des OCSP-Servers aus dem WASA-Projekt [Schöberl01].

Mit diesem Verfahren muss der Benutzer lediglich in der Lage sein das Zertifikat des Authentication Servers zu überprüfen. Ebenso muss der Distribution Server (Händler) nur dem Authentication Server vertrauen. Es liegt also eine vereinfachte PKI vor und der Authentication Server tritt für Benutzer und Händler gleichermaßen als TTP (Trusted-Third-Party) auf.

Anonymisierung des Benutzers

Für die Identifikation des Benutzers bei der Kommunikation zwischen Authentication Server und Distribution Server wird eine vom Benutzer unabhängige Kundennummer verwendet. Damit ist es nicht notwendig sensible Daten des Benutzers zwischen Händler und Authentication Server auszutauschen. Die Anonymisierung erfolgt in der Art, dass der Authentication Server keine Daten über das Kaufverhalten des Benutzers in Erfahrung bringen kann und der Distribution Server auch keine Möglichkeit hat persönliche Daten über den Benutzer vom Authentication Server zu erhalten. Die Kundennummern, die aber eine Identifizierung des Benutzers systemweit ermöglichen, werden in Abschnitt 7.3.4 aufgezeigt und erklärt.

Verschlüsselung der JAD-Datei-Download-URL

Die Aufgabe der Verschlüsselung der JAD-Datei-URL übernimmt der Authentication Server. Der Benutzer bekommt diese URL per verschlüsselte SMS-Nachricht zugeschickt, entschlüsselt diese und erhält somit die JAD-Datei-URL

Der Sicherheitszuwachs durch diese Maßnahme muss aber als sehr gering eingestuft werden, da der Download der JAD-Datei, die wiederum den JAR-URL enthält laut MIDP2.0 unverschlüsselt über das HTTP-Protokoll erfolgen muss. Sinnvolle Anwendungen dieses Verfahren bestünden nur dann, wenn die Zeit zwischen JAD-Datei-Download-URL und JAD-Datei-Download sehr groß ist, was im Normalfall nicht der Fall ist. Diese Verschlüsselung der JAD-Datei-Download-URL ist daher lediglich ein positiver Seiteneffekt.

5.2.3 Notification Server

Der Notification Server [NOTS03] ist zuständig für das Sammeln der J2ME-Notifications. Diese Statusmeldungen vom Endgerät und die Statusmeldungen vom Authentication Server werden im gesamten Dokument „Notifications“ genannt (siehe auch Abschnitt 2.3). Der Notification Server dient somit als Datensammelstelle. Ein Notification Server empfängt diese Benachrichtigungen eines Distribution Servers und speichert diese ab. Hat ein Notification Server die zusätzliche Aufgabe Nachrichten weiterzuleiten, so schickt er die Benachrichtigungen an andere Notification Server weiter. Somit ist es möglich im Sinne einer Reproduktion oder einer Konzentration der Daten redundante Systeme zu bilden.

Profilbildung

Die auf der Datensammlung aufbauende Profilbildung ist als Webapplikation implementiert[28] und bietet dem Administrator des Notification Servers Zugriff auf die Datenbank des Notification Server. Da die Abfragen sehr anwendungsorientiert und von den aktuellen Fragestellungen des Betreibers bzw. seiner Benutzer abhängig ist, wird hier ein direkter Zugriff auf die SQL-Datenbank erlaubt und keine gesonderte Schnittstelle definiert.

Im Rahmen dieser Diplomarbeit sind folgende Abfragen als Beispiel implementiert:

- Anzahl der Benutzer
- Im System verwendete und vorhandene Applikationen

[28] Die Profilbildung ist Teil des Administrationsbereichs des Notification Servers.

- Anzahl der installierten Applikationen nach Benutzer, Applikation und Gesamtanzahl
- Benutzer, die eine bestimmte Applikation installiert haben
- Datum der Installation/Löschung einer Applikation eines Benutzers
- Anzahl der Downloadversuche für eine Applikation

Push Service

Um dem Applikationsvertreiber die Möglichkeit zu geben den Benutzer über neue Versionen von Applikation zu informieren, ist ein Push-Service implementiert. Um dieses Service zu nutzen sendet der Notification Server an den Authentication Server die Kundennummer sowie den vom Applikationsvertreiber eingegebenen SMS-Text (vorzugsweise mit Download-URL). Diese Informationen werden dann an den Benutzer per SMS zugeschickt und dieser kann bei Bedarf die Applikation herunterladen.

Auch wenn der Ausdruck *Push*[29] annehmen lässt, dass es sich hierbei um eine betreiberinitiierte Technik handelt, muss klar gesagt werden, das der eigentliche Download des MIDlets wie auch beim benutzerinitiierten Standard-OTA-Provisioning abläuft. Der Unterschied ist, dass im Falle von OTA-Push kein Link auf einer WAP bzw. HTML Seite oder ein Bookmark gesucht werden muss, sondern der URL der JAD-Datei in einer SMS versandt wird und damit vorselektiert wird. Da SMS eine Push-Technologie ist, wird dieses Verfahren OTA-Push genannt, obwohl sich der eigentliche Download nicht vom OTA-Pull-Download unterscheidet.

Für eine OTA-Push-SMS [SL45i], [WAP] ist folgende Randbedingung zu beachten: Das mobile Gerät sucht in der SMS nach den Schlüsselworten `http:`, `Http:`, `www.`, `WWW.`, `wap.`, `Wap`, um einen Link auf externe Inhalte zu erkennen. Findet das Gerät einen Link in der SMS, wird er optisch hervorgehoben und kann vom Benutzer ausgewählt werden. Drückt der Anwender dann die Senden-Taste, wird die URL aufgerufen und im Falle einer JAD, oder JAR Download URL wird die URL der AMS übergeben und die Applikationsinstallation startet.

5.2.4 SMS Gateway

Das verwendete SMS-Gateway hat lediglich die Aufgabe eingehende Email-Nachrichten per SMS an das Endgerät weiterzuleiten, und umgekehrt Nachrichten vom Endgerät zu empfangen.

[29] Pushtechnik: ermöglicht das Senden von Informationen an den Benutzer ohne dass dieser die Information aktiv abgerufen hat.

Diese Funktionalität wird vom Authentication Server benutzt um verschlüsselte und signierte Daten per SMS an den Benutzer zu schicken und diese wieder vom Gateway zu erhalten. Konkret wird dieser Dienst für folgende Aufgaben genutzt:

- Passwortübermittlung
- Authentifizierung

Als SMS-Gateway soll der schon bestehende und getestete WASA-Message-Forwarder (WMF [Schöberl01]) verwendet werden. Alternativ können auch E-Mail-SMS-Gateways verwendet werden.

5.3 Applikationsdistribution

Für die Applikationsdistribution (Application Provisioning) steht eine Menge von unterschiedlichen Wegen zur Verfügung, die vom direkten proprietären Kopieren von Applikationen in das Dateisystem des Endgerätes bis zum standardisierten Download über das Mobilfunknetzwerk reichen.

Die Applikationsdistribution dieser Arbeit sieht folgende drei Downloadmöglichkeiten vor:

1. User-initiated OTA-Provisioning: Vom Benutzer initiiertes Herunterladen über das vorhandene Mobilfunknetzwerk (siehe Abschnitt 3.1)
2. HTTP-Provisioning: Herunterladen der Applikationen per PC und manuelles Kopieren der Applikation auf das mobile Endgerät
3. Push-Services: Benachrichtigung per SMS-Nachricht mit Download-URL. Danach findet ein OTA-Download über die übermittelte Download-URL statt

Im folgenden Abschnitt werden nun die vorgestellten Downloadmöglichkeiten genauer durchleuchtet.

5.3.1 User-initiated OTA-Provisioning

Bei dieser manchmal auch als *Pull Service* bezeichneten Distributionsart sucht der Benutzer aktiv nach Applikationen. Das Wireless Application Protocol (WAP [WAP]) ist hierbei der meist verbreiteten „Suchdienst“, der auf fast allen europäischen Endgeräten vorhanden ist. In dieser Diplomarbeit wurde der Standard-OTA-Download erweitert und inkludiert die folgenden Schritte:

1. Der Benutzer wählt die gewünschte Applikation am Distribution Server durch aktives Suchen oder Auswahl aus den möglichen Applikationen aus
2. Der Benutzer wird zum Authentication Server weitergeleitet wo er sich anhand eines Benutzernamens oder MSISDN[30] identifizieren muss. Nun gibt es zwei Möglichkeiten der Authentifizierung des Benutzers: (siehe Abschnitt 5.1.2)
    a. Anhand des Benutzernamens bzw. der MSISDN schickt der Authentication Server dem mobilen Endgerät eine signierte SMS. Der Benutzer prüft deren Inhalt und bestätigt den Erhalt der SMS-Nachricht durch Rücksenden einer mit seinem privaten Schlüssel verschlüsselten SMS. Danach erhält der Benutzer eine verschlüsselte SMS mit der URL der JAD-Datei und dem einmaligen JARLogin[31]/Passwort-Paar zum Download der JAR-Datei. Der Benutzer entschlüsselt diese wiederum und erhält somit die Zugangsdaten zum Download
    b. Der Benutzer gibt zusätzlich zum Benutzernamen (bzw. MSISDN) sein Passwort ein, anhand dessen er authentifiziert wird. Wird die Authentifizierung vom Authentication Server stattgegeben, hat der Benutzer die Möglichkeit die Zugangsdaten zum Download (JAD-Datei-Download-URL und das einmalige JARLogin/Passwort-Paar zum Download der JAR-Datei) auf einer Webseite abzuholen
3. Der WAP-Browser lädt über die erhaltene URL die angegebene JAD-Datei und übergibt sie der AMS
4. Die AMS überprüft, ob die MIDlet-Suite schon installiert ist und genügend Platz für die Installation vorhanden ist. Der Benutzer muss schließlich die Installation der MIDlet-Suite[32] bestätigen
5. Die AMS entnimmt der JAD-Datei die Referenz auf die JAR-Datei und lädt diese über den angegebenen URL mittels HTTP. Sollte dabei der Distribution Server ein JARLogin und Passwort anfordern, so muss die

[30] Rufnummer eines mobilen Endgerätes.

[31] Loginbezeichnung für den Download der JAR-Datei.

[32] Die AMS zeigt als Hilfestellung den Namen, die Größe, die Versionsnummer, den Hersteller der MIDlet-Suite an und präsentiert außerdem die Beschreibung aus der JAD-Datei und die JAR-Datei-Download-URL.

AMS dem Benutzer einen Login-Dialog präsentieren und dem Benutzer die Möglichkeit geben JARLogin und Passwort in zwei Textfelder sicher einzugeben, und diese Daten abzusenden. Die eingegebenen Daten werden dann an den Distribution Server geschickt und dieser verifiziert diese. Sollten die JAR Logindaten mit denen in der Datenbank des Distribution Servers übereinstimmen, sendet der Distribution Server der AMS die angeforderte JAR-Datei. Diese Benutzer- Authentifizierung ist im RFC2617 [RFC2617] geregelt und Teil des HTTP-Protokolls

6. Ist die MIDlet-Suite signiert wird vom AMS die Signatur vor der Installation verifiziert. Sollte diese Verifizierung erfolgreich sein, wird die MIDlet-Suite installiert und die MIDlets aus der MIDlet-Suite stehen ab diesem Zeitpunkt zur Ausführung zur Verfügung

Nach diesem Ablauf ist die MIDlet-Suite am mobilen Endgerät installiert und die darin enthaltenen Applikationen können über die entsprechenden Menüpunkte gestartet werden.

Wichtig ist bei dieser Downloadmöglichkeit, dass die Schritte 1 und 2 zwar nicht Teil des Standard-OTA-Downloads sind, aber diesen konfliktfrei um die Authentifizierung erweitern. Die Schritte 3-6 stellen dann den Standard-OTA-Download dar.

5.3.2 Provisioning über einen PC

Die dritte Downloadmöglichkeit ist der Download mittels eines konventionellen Browsers auf einem PC und die anschließende direkte Installation auf das Endgerät. Besonders für größere Applikationen ist dies in Anbetracht der geringeren Bandbreiten und Kosten in Mobilfunknetzwerken eine Alternative. Da der Provisioning-Vorgang in MIDP, HTTP-Protokoll basierend ist, können JAD- und JAR-Datei auch über einen Standardbrowser geladen werden. Einzig die Extraktion der JAR-Datei-Download-URL aus der JAD-Datei muss händisch oder über ein spezielles Plug-In erfolgen.

Die direkte Installation der JAD- und JAR-Dateien auf dem Endgerät, auf die bei der PC basierenden Methode zurückgegriffen werden muss, ist herstellerspezifisch und deshalb außerhalb dieser Arbeit. Weiters ist anzumerken, dass in diesem Fall aufgrund der direkten Installation von JAD- und/oder JAR-Datei eine Benachrichtigung des Notification Servers unterbleiben könnte, da die AMS dies nur bei einem OTA-Download durchführen muss (siehe Abschnitt 2.4.2).

Beim Download über einen PC, kann die Authentifizierung des Benutzers nur per Benutzernamen (bzw. MSISDN) und Passwort durchgeführt werden, da ein

konventioneller Browser, die sichere Authentifizierungsmethode durch Zertifikate und verschlüsselten SMS-Nachrichten nicht unterstützt.

5.3.3 Push Service

In Kapitel 5.2.3 wurde das Push-Service bereits beschrieben. Der Unterschied zu OTA aus 5.3.1 ist, dass sobald der Benutzer eine SMS-Nachricht mit einer Download-URL erhält, er direkt aus der SMS-Nachricht heraus den Benutzerinitiierten OTA-Download zu starten. Es erfolgt also wiederum ein benutzerinitiiertes OTA-Provisioning, nur wurde das Auffinden (Discovery) der Applikation durch die direkte Information vereinfacht.

5.4 Fazit

In diesem Kapitel wurden das System und dessen Komponenten vorgestellt und Überlegungen angestellt wie es zu dieser Lösung gekommen ist.

Das System besteht primär aus Authentication, Distribution und Notification Server. Der Distribution Server stellt den Benutzern die Applikationen zur Verfügung, bzw. bietet die Möglichkeit nach diesen zu suchen und diese herunterzuladen. Mit dem Content Manager des Distribution Servers hat man die Möglichkeit von Herstellerseite das Angebot an Applikationen zu verwalten. Der Authentication Server übernimmt die Authentifizierung der Benutzer, und schließlich dient der Notification Server als Datensammelstelle zur Profilbildung.

Die zwei in Abschnitt 5.1.2 präsentierten Authentifizierungsmethoden stellen zwei Möglichkeiten dar, wie sich Benutzer gegenüber dem Authentication Server authentifizieren können. Die Authentifizierung über den Benutzernamen (bzw. MSISDN) und Passwort wurde zu Testzwecken implementiert, um das System auch ohne Kryptographie-Chipkarte testen zu können. Diese Authentifizierungsmethode bietet eine geringere Sicherheit als die Authentifizierung über Zertifikate und verschlüsselten SMS-Nachrichten, genügt aber den derzeitigen Sicherheitsanforderungen der Mobilfunkbetreiber. Die Authentifizierungsmethode über Zertifikate und asymmetrische Kryptographie bietet eine sichere und zuverlässige Methode um Benutzer zu authentifizieren, greift dazu aber auf die Funktionalität einer Kryptographie-Chipkarte zurück (WASA-Projekt [Schöberl01]).

6 Kommunikationsablauf

Wenn man Dummheiten macht,
müssen sie wenigstens gelingen.

Napoleon I. Bonaparte

In diesem Kapitel wird der Kommunikationsablauf zwischen den in diesem Projekt entwickelten Authentication, Distribution und Notification Server sowie dem Benutzer beschrieben.

6.1 Allgemeine Überlegungen

Im diesem Abschnitt werden Überlegungen zum Kommunikationsablauf angestellt, die die Rahmenbedingungen aufzeigen und beschreiben wie es zur Realisierung gekommen ist.

6.1.1 Kommunikation der Server

Die Server: Authentication-, Distribution- und Notification Server kommunizieren untereinander über verschlüsselte HTTPS Verbindungen. HTTPS steht für das HTTP-Protokoll [HTTP11] über eine sichere „Secure Socket Layer"-Verbindung (SSL, [SSL]).

Eine Verbindung zwischen zwei Servern wird in dieser Arbeit folgendermaßen aufgebaut: Beide Teilnehmer müssen sich ihrem Gegenüber wechselseitig authentifizieren, dabei fungiert ein Server in diesem Verbindungsaufbau als Client und einer als Server.

Der Server wird immer anhand des SSL-Verbindungsaufbau authentifiziert und der Client authentifiziert sich daraufhin anhand einer Basic Authentication [RFC2617] über die verschlüsselte SSL-Verbindung. Im SSL-Protokoll wäre eine Client-Authentifizierung auch vorgesehen, ist aber an der benutzten Webserver-Software [S3SERV03] gescheitert, denn diese unterstützt diese Authentifizierungsmethode nicht richtig. Sie hält sich nicht an das SSL-Protokoll und hat einen Fehler in der Clientzertifikatsanforderung. Der Fehler liegt darin, dass an der vorgesehenen Stelle kein Clientzertifikat angefordert wird und so die verschlüsselte Verbindung nicht

aufgebaut werden kann. Darum wurde die Authentifizierung des Clients durch die Basic Authentication nach RFC2617 gewählt, die über eine verschlüsselte Verbindung kein Sicherheitsproblem darstellt, sofern die Authentizität des Servers garantiert ist.

Der Verbindungsaufbau läuft folgendermaßen ab:

1. Durch die SSL-Server-Authentifizierung kommt eine verschlüsselte SSL Verbindung zustande. Dabei versucht der Client den Server anhand seiner gespeicherten Zertifikate im TrustedStore zu authentifizieren [SSLSICH]
2. Über die verschlüsselte SSL-Verbindung authentifiziert sich der Client am Server anhand der Basic Authentication [RFC2617]

Nach diesem Verbindungsaufbau, haben sich beide Server gegenüber authentifiziert und die eigentliche Kommunikation zwischen den Server findet über die verschlüsselte SSL-Verbindung statt.

Die Server kommunizieren untereinander anhand von XML-Dateien [XMLW] in denen Nachrichten abgelegt sind. Dazu werden Nachrichten-Objekte (Instanzen einer Java-Klasse) nach XML konvertiert und diese erzeugte XML-Datei dann versendet. Dazu muss jedes dieser Nachrichten-Objekte eine Methode implementieren die es ermöglicht eine XML-Datei aus dem Nachrichten-Objekt, bzw. aus dessen Informationen, zu generieren. Diese XML-Dateien ermöglichen eine leichte Verarbeitung der Nachrichten und eine leichte Modifikation und Ergänzung der Nachrichteninformationen im Fall einer Änderung oder Erweiterung. Für eine detaillierte Einsicht in die Kommunikation und das Format der XML-Dateien ist auf die Dokumentation [XMLKOM] der entwickelten Serverschnittstellen verwiesen.

6.1.2 Generierung von JARLogin und Passwort

Wie in Abschnitt 3.1.2 erklärt, erlaubt die AMS eines J2ME-fähigen Endgeräts nach MIDP 2.0, eine Authentifizierung vor dem JAR-Datei-Download nach RFC2617 (Basic Authentication [RFC2617]) mit JARLogin und Passwort. Dabei werden JARLogin und Passwort im Klartext base64 kodiert [RFC2045] übertragen. Somit kann das Passwort, nachdem es einmal vom Benutzer benutzt wurde, nicht mehr als sicher angesehen werden. Deshalb wird in diesem Projekt dem Benutzer für jeden Download ein neues, einmalig gültiges JARLogin/Passwort Paar übermittelt.

Folgende Punkte sind bei der Generierung des JARLogins und des Passworts für den Download der JAR-Datei zu berücksichtigen:

1. Für die Generierung des JARLogins und des Passworts ist der Distribution Server verantwortlich. Die Gründe dafür sind, dass ein Distribution Server (es gibt mehrere, siehe Abschnitt 5.2.1) ein bei ihm eindeutiges JARLogin und Passwort generieren muss, und der Download der JAR Datei schließlich über den Distribution Server abgewickelt wird

2. Für die Gestaltung des JARLogins und des Passworts musste ein Kompromiss zwischen Benutzerfreundlichkeit und Sicherheit eingegangen werden. Da diese zwei Zeichenfolgen auf dem mobilen Endgerät mit einer 10-Tasten-Tastatur eingegeben werden müssen, ist die Entscheidung dahingehend gefallen, dass das JARLogin aus sechs Zahlen und das Passwort aus vier Zahlen bestehen soll, und diese durch einen „sicheren" Zufallszahlengenerator erzeugt werden

3. Das JARLogin und das Passwort, werden für die Authentifizierung beim Download der JAR-Datei, im Klartext übertragen und dürfen daher nur einmal verwendet werden

Die JARLogin und Passwort Generierung am Distribution Server erfolgt durch die SecureRandom Java-Klasse [RAND01]. Diese Java-Klasse liefert dem Entwickler „sichere" Zufallszahlen. Diese Zufallszahlen werden im Gegensatz zu anderen Zufallszahlen durch einen wirklich zufälligen Initialisierungswert (Seed) initialisiert. Es ist bekannt, dass sich mit deterministischen Maschinen keine echten Zufallszahlen erzeugen lassen. Eigentlich müssten wir daher von Pseudo-Zufallszahlen sprechen, um darauf hinzuweisen, dass unsere Zufallszahlengeneratoren stets deterministische Zahlenfolgen erzeugen. Mit der zusätzlichen Forderung kryptographischer Zufallszahlen (praktisch unvorhersagbare Zahlenfolgen) zu generieren wird diese Unterscheidung an dieser Stelle unbedeutend. Tatsächlich besteht der wichtigste Unterschied zu "echten" Zufallsgeneratoren nur noch darin, dass deren Folgen nicht zuverlässig reproduziert werden können. Dies ist bei den Pseudo-Zufallszahlen Klasse SecureRandom aber trotzdem der Fall.

6.1.3 Transaktionen

Eine Transaktion identifiziert einen Applikationsdownload eines Benutzers. Das heißt eine Transaktion kapselt die Aufgabe des Übertragens einer Applikation auf Seiten der Server.

Eine Transaktion wird am Distribution Server generiert, sobald ein Benutzer den Download einer Applikation initiiert. Diese Transaktion wird anhand einer eindeutigen Transaktions-ID identifiziert, und den anderen Servern innerhalb der

Kommunikation über XML-Dateien (Abschnitt 6.1.1) mitgeteilt. Die Transaktions-ID ist innerhalb eines Distribution Servers eindeutig. Systemweit werden die Transaktionen anhand der Kombination ihrer Transaktions-ID und der ID des Distribution Servers identifiziert. Somit ist jede Transaktion systemweit eindeutig festgelegt.

6.1.4 Download-URL für die JAD-Datei

Die Download-URL für die JAD-Datei wird dem Benutzer bzw. dem mobilen Endgerät, wie in Abschnitt 5.3.1 beschrieben, per verschlüsselte SMS-Nachricht zugesandt. Diese Download URL wird vom Distribution Server erstellt und enthält die 6 Zeichen lange Transaktions-ID zur Identifikation der Transaktion und eine 8 Zeichen lange Zufallszahl zur sicheren Erkennung der Transaktion.

Diese Download-URL (allgemeiner Aufbau einer URL, siehe [RFC1738]) wird gemäß *Abbildung 10* aufgebaut:

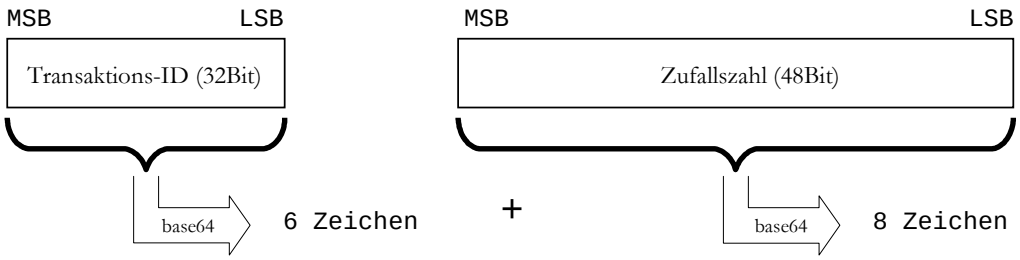


Abbildung 10: Dynamischer Teil der JAD-URL

Die Transaktions-ID ist eine 32Bit lange Zahl die nach base64 [RFC2045] codiert wird und ergibt somit 6 Zeichen. Die 48Bit lange Zufallszahl ergibt nach base64 codiert 8 Zeichen. Die „xxxxxx" stehen für die 6 base64 codierten Zeichen lange Transaktions-ID (zur Identifikation der Transaktion) und die „yyyyyyyy" stehen für die 48Bit lange Zufallszahl. Die Zeichenkette „xxxxxxyyyyyyyy" wird in diesem Dokument als Zufalls-Nounce bezeichnet, da der Zufallsteil („yyyyyyyy") zufällig generiert wird, und somit nicht vorhersehbar ist.

Die JAD-Datei-Download-URL enthält den base64 codierten Zufallsteil (per SecureRandom generiert) und ist somit „unmöglich" vorherzusagen, bzw. zu erraten. Als zusätzliche Sicherheitsstufe wird in der JAD-Datei nur die relative JAR-Datei-Download-URL eingetragen. Diese URL enthält nur den Namen der MIDlet-Suite in der Form: *name.jar*.

6.1.5 Notification Codes

Wie in Abschnitt 5.2.1 erläutert, übermittelt der Distribution Server den Notification Server sämtliche Statusmeldungen sowohl vom System als auch von Seiten der Application Management Software (AMS). Der Typ der Statusmeldung wird durch die in Tabelle 5 angegebenen Status Codes codiert. Der Notification Code ist also ein Kürzel, der den Typ einer Benachrichtigung, bzw. Statusmeldung kennzeichnet.

In diesem Dokument wird der Begriff *Notification* für die Benachrichtigungen vom Distribution Server an den Notification Server verwendet.

In Tabelle 5 werden die Notifications und deren Notification-Code in deren zeitlichen Reihenfolge aufgelistet.

Tabelle 5: Notification Codes

Notification Codes	**Bedeutung**
New	Eine neue Transaktion wurde beim Distribution Server angelegt.
authNew	Eine neue Autorisierung (Transaktion) wurde im Authentication Server angelegt. Jetzt kann und muss sich der Benutzer noch authentifizieren.
authOK	Der Benutzer wurde vom Authentication Server erfolgreich authentifiziert.
authShow	• Authentifizierung per Zertifikate: Dem Benutzer wurden die Downloaddaten (JAD-Datei-Download-URL und JARLogin/Passwort) zugeschickt. • Authentifizierung per MSISDN (oder Benutzername) und Passwort: Der Benutzer hat sich diese Downloaddaten über die HTML/WAP Seite abgeholt
jad	Der Download der JAD-Datei hat stattgefunden.
jar	Der Download der JAR-Datei hat stattgefunden.
in [codeNr.]	Installationsbenachrichtigung von Seiten des AMS: Eine Applikation bzw. MIDlet-Suite wurde vom Benutzer installiert, und der Notification Server erhält diese Benachrichtigung mit dem Installationserfolg Für [codeNr.] steht der dreistellige Code, der vom mobilen Endgerät gesendet wurde und der den Erfolgsstatus identifiziert. Diese Installations-Codes sind in Tabelle 4 zu finden.
dn [codeNr.]	Löschen Statusmeldung von Seiten des AMS: Eine Applikation bzw. MIDlet-Suite wurde vom Benutzer vom

	Gerät gelöscht und der Notification Server erhält eine Benachrichtigung über deren Erfolg. Für [codeNr.] steht wiederum der dreistellige Code, der vom mobilen Endgerät gesendet wurde und der den Erfolgsstatus identifiziert.

6.2 Kommunikationsabläufe

In den folgenden Abschnitten wird der Kommunikationsablauf erläutert. Die Kommunikation ist dabei in vier Abschnitte gegliedert, die auch der zeitlichen Abfolge entsprechen:

1. Authentifizierung und Übertragung der JAD-Datei-Download-URL an den Benutzer
2. Installation der MIDlet-Suite
3. Abschluss des Downloads
4. Übertragung der Statusmeldungen vom mobilen Endgerät (Notifications)

6.2.1 Authentifizierung und Übertragung

Wie schon in Abschnitt 5.1.2 beschrieben gibt es aus Testzwecken, zwei Möglichkeiten wie sich der Benutzer authentifiziert und seine Downloaddaten (JAD-Datei-Download-URL, JARLogin und Passwort) erhält.

In diesem Abschnitt wird zuerst die sichere Authentifizierungsmethode und Übertragung der JAD-Datei-Download-URL per Zertifikate beschrieben, und darauf folgend, wird die Authentifizierung per MSISDN (oder Benutzername) und Passwort erläutert.

Authentifizierung per Zertifikat

Der Authentifizierungs- und Übertragungsvorgang der JAD-Datei-Download-URL ist in *Abbildung 11* anhand eines Sequenzdiagramms dargestellt.

Die Kommunikation zwischen Distribution Server, Authentication Server und Notification Server ist dabei, wie in Abschnitt 6.1.1 erklärt, durch Secure Socket Layer (SSL, [SSL]) gesichert.

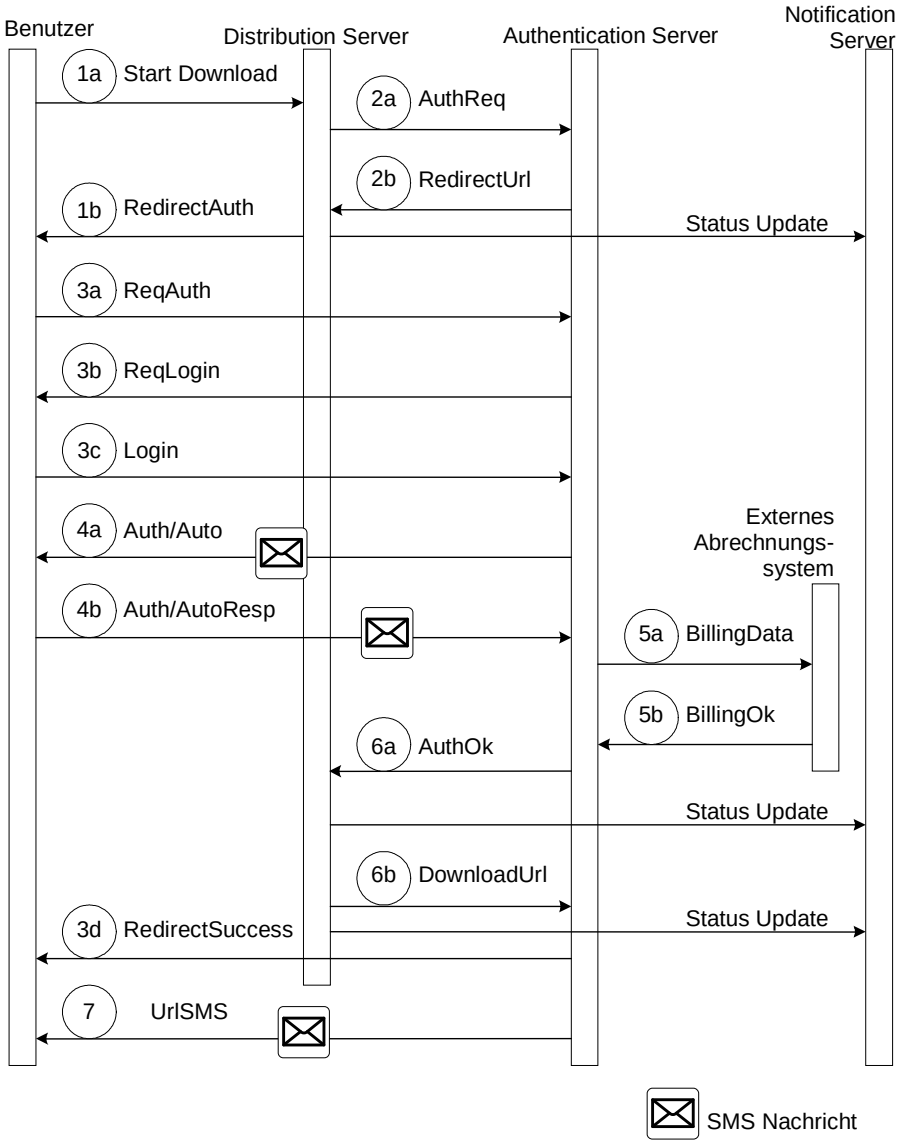


Abbildung 11: Erzeugung und Übertragung der Download URL

Die Kommunikation läuft folgendermaßen ab:

1. Der Benutzer startet den Download indem er auf der Seite des Distribution Servers den entsprechenden Link für den Download selektiert
2. Der Distribution Server erstellt eine Transaktion und generiert eine eindeutige Transaktions-ID, anhand derer später der gewünschten Download wieder identifizieren werden kann. Diese schickt er an den Authentication Server (2a). Optional werden in diesem Schritt auch Daten,

die für die Bezahlung relevant sind, für die spätere Bezahlung mitgeschickt. Der Authentication Server antwortet mit einer *Redirect-URL* für den Benutzer (2b). Der Distribution Server schickt daraufhin dem Notification Server ein Status Update über die angelegte Transaktion (Notification Code: New) und über den Status der Erzeugung der Transaktion (Authentication, siehe Abschnitt 7.3.2) am Authentication Server (Notification Code: authNew)

3. Der Benutzer erhält mit *Redirect Auth* (1b) diese URL
4. Der Benutzer wird zum Authentication Server (3a) weitergeleitet, worauf dieser ihn auffordert seinen Benutzernamen (oder MSISDN) einzugeben (3b)
5. Der Benutzer übermittelt seinen Benutzername oder alternativ seine MSISDN (3c)
6. Wird dem Authentication Server eine gültige MSISDN bzw. ein gültiger Benutzername vom Benutzer übermittelt, sendet der Server an die MSISDN eine SMS-Nachricht (4a), die die Rechnungsdaten, sowie die Signatur des Authentication Servers enthält. Der Authentication Server bestätigt damit, dass er dem Händler (Distribution Server) vertraut, und dass die Nachricht vom Händler generiert worden ist
7. Der Benutzer überprüft die Signatur des Authentication Servers (JavaCard Applikation aus WASA-Projekt [Schöberl01]) und signiert seinerseits den Inhalt der SMS-Nachricht und sendet diese an den Authentication Server zurück (4b)
8. Der Authentication Server überprüft die Signatur und überprüft die Identität des Benutzers. Ist dieser Vorgang erfolgreich beendet, sendet der Authentication Server, die für die Verrechnung relevanten Daten an ein externes Abrechnungssystem (Billing System) (5a), das die Bonität des Benutzers überprüft und mit *BillingOk* (5b) dem Authentication Server das Ergebnis dieser Überprüfung mitteilt
9. Der Authentication Server teilt dem Distribution Server nun mit, dass die Authentifizierung erfolgreich war (6a). Diese Information wird mit einem *Status Update* an den Notification Server gemeldet (Notification Code: authOK)
10. Der Distribution Server generiert nun eine URL für den Download der JAD-Datei, die die Form aus Abschnitt 6.1.4 besitzt. Analog dazu werden

die *Install-Notification-URL* und die *Delete-Notification-URL* erzeugt, die alle dieselbe Zufalls-Nounce enthalten. Die URL für den Download der JAR-Datei enthält nur den Namen der MIDlet-Suite, damit die URL nicht direkt aus der JAD-Datei auslesbar ist, nicht verwertet werden kann. Diese drei URLs werden in die JAD-Datei eingetragen. Zusätzlich generiert der Distribution Server ein JARLogin/Passwort Paar, das eindeutig sein muss. Dieses Paar dient beim Download der JAR-Datei als Zugriffskontrolle. Die JAD-Datei-Download-URL und JARLogin/Passwort werden dem Authentication Server zur Weiterleitung zugesandt (6b)

11. Der Authentication Server verschlüsselt die SMS mit dem Public Key des Benutzers und schickt diese an die MSISDN des Benutzers (7). Diese Information wird mit einem *Status Update* ebenfalls an den Notification Server gemeldet (Notification Code: authShow)

Die JAR-Datei-Download-URL erstellt das AMS anhand der absoluten JAD-Datei-Download-URL und dem Namen der MIDlet-Suite, der in der JAD-Datei eingetragen ist. Dies hat den Vorteil dass ein Angreifer die JAR-Datei-Download-URL nicht durch Ausspähen des JAD-Datei-Inhaltes erhalten kann.

Authentifizierung per MSISDN (oder Benutzername) und Passwort

In diesem Abschnitt werden die Authentifizierung des Benutzers per MSISDN (oder Benutzername) und Passwort und die alternative Übertragung der JAD-Datei-Download-URL beschrieben.

Die wesentlichen Unterschiede zur sicheren Authentifizierung durch signierte SMS-Nachrichten sind, dass die Authentifizierung, anhand eines Benutzernamen (oder MSISDN) und Passwort erfolgt. Und die anschließende Übertragung der JAD-Datei-Download-URL einfach per Präsentation einer entsprechenden WAP/HTML-Seite geschieht, auf der JAD-Datei-Download-URL und das JARLogin/Passwort Paar dargestellt werden.

An dieser Stelle werden nur mehr die Schritte präsentiert, die von der sicheren Übertragung der JAD-Datei-Download-URL verschieden sind. Der Ablauf der Authentifizierung beginnt bei der Anmeldung am Authentication Server.

- Der Authentication Server fordert den Benutzer zur Eingabe seines Benutzername (oder MSISDN) und Passwort auf (3b)
- Der Benutzer übermittelt seinen Benutzername (oder seine MSISDN) und Passwort (3c)

- Wird dem Authentication Server ein gültiger Benutzername (oder MSISDN) samt korrektem Passwort übermittelt, teilt er dem Distribution Server mit, dass die Authentifizierung erfolgreich war (6a). Diese Information wird mit einem *Status Update* an den Notification Server gemeldet (Notification Code: authOK)

- Der Distribution Server generiert wiederum eine URL für den Download der JAD- und der JAR-Datei, eine *Install-Notification-URL* und eine *Delete-Notification-URL*. Die letzten drei URLs werden in die JAD-Datei eingetragen. Zusätzlich generiert der Distribution Server das JARLogin/Passwort Paar. Die JAD-Datei-Download-URL und JARLogin/Passwort werden dem Authentication Server zugesandt, und der Authentication Server leitet bei dieser Authentifizierung den Benutzer zum Distribution Server weiter (3d)

- Der Distribution Server zeigt dem weitergeleiteten Benutzer die JAD-Datei-Download-URL samt JARLogin/Passwort an. Der Notification Server wird darüber (Notification Code: authShow) informiert. Abbildung 12 zeigt die HTML-Seite mit den Downloaddaten. Nach Selektion der URL kann der Download der JAD-Datei gestartet werden, bzw. direkt die JAR-Datei geladen werden

Abbildung 12: Downloaddaten bei unsicherer Authentifizerung

Wie schon in Abschnitt 5.1.2 beschrieben, wurde dieser Download implementiert um das System auch ohne Kryptographie-Chipkarte am PC zu testen. Deswegen wurden auch die beiden *Abschicken* Buttons im Formular aus Abbildung 12

eingefügt, die es dem Tester ermöglichen die Install- bzw. Delete-Notifications abzuschicken.

6.2.2 Download der MIDlet-Suite

Nachdem der Benutzer die Downloaddaten, d.h. die JAD-Datei-Download-URL, das JARLogin und das Passwort erhalten hat, kann er den Download auf das mobile Endgerät beginnen, indem er die URL aufruft und damit die JAD-Datei-Download-URL an die AMS übergibt.

Alternativ dazu, kann der Benutzer auch die JAD- und JAR-Datei mit einem PC herunterladen, wie es in Abschnitt 5.3.2 beschrieben wurde.

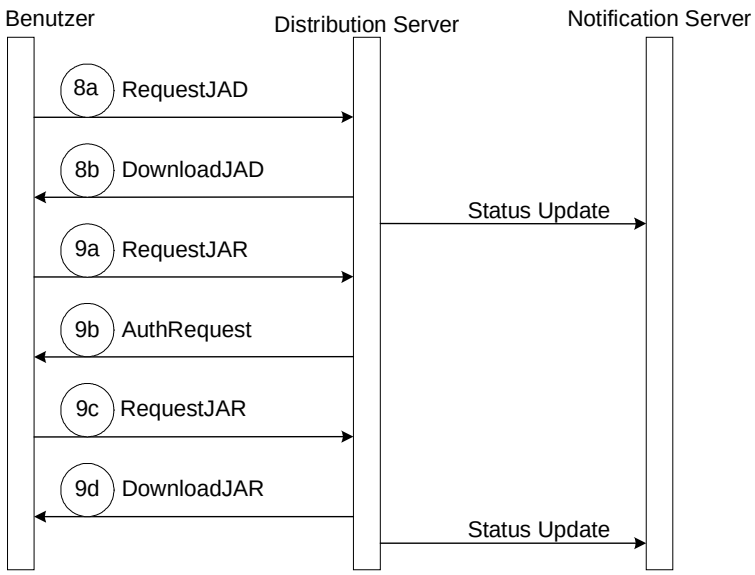


Abbildung 13: Download der MIDlet-Suite

Die Downloadschritte beinhalten wie in Abschnitt 3.1.1 bereits erklärt, den Download der JAD- und JAR-Datei und sind der Vollständigkeit halber in *Abbildung 13* genau dargestellt und folgend erklärt:

1. Die AMS startet den Download der JAD-Datei mit einem Request (8a) an den Distribution Server. Als Beispiel für die dabei übertragenen Informationen siehe Tabelle 6

2. Der Distribution Server sendet die gewünschte JAD-Datei (8b). Anschließend wird ein *Status Update* an den Notification Server gesendet (Notification Code: jad)

3. Nach einer Überprüfung der JAD-Datei durch die AMS und einer Bestätigung des Benutzers wird der Download der JAR-Datei gestartet (9a). (Dazu wird die in der JAD-Datei angegebene URL verwendet)

4. Der Distribution Server verlangt eine Authentifizierung nach RFC2617 [RFC2617] (9b). Daraufhin fordert die AMS den Benutzer auf, JARLogin und Passwort einzugeben und sendet den Request für die JAR-Datei zusammen mit JARLogin und Passwort abermals an den Server (9c)

5. Sollten die Logindaten (einmaliges JARLogin/Pass) korrekt sein, sendet der Distribution Server die JAR-Datei an den Benutzer (9d). Anschließend wird ein *Status Update* an den Notification Server gesendet (Notification Code: jar)

In Tabelle 6 ist ein JAD-Datei Request angegeben, der zeigen soll, welche Informationen an den Distribution Server geschickt werden. Der Distribution Server empfängt nur diese Information vom mobilen Endgerät und leitet somit auch nur diese Information an den Notification Server weiter:

- User-Agent: Beinhaltet Informationen über das mobile Endgerät und dessen J2ME-Eigenschaften. Dem Hersteller ist es überlassen, ob und welche Information versendet wird. Üblicherweise enthält dieser Parameter den Namen des Endgerätes bzw. den Hersteller, die Version des unterstützten MIDP-Profils und die Version der unterstützten CLDC-Konfiguration
- Language: Entspricht der gewünschten Sprache des Benutzers, in der die Antwort (HTTP-Response) auf den JAD-Datei Request verfasst sein sollte. Normalerweise ist das die Sprache, die am mobilen Endgerät für das Menü konfiguriert ist
- Charset enthält den gewünschten Charaktersatz des mobilen Endgerätes

Tabelle 6: JAD-Datei Request

```
GET http://www.merchant.at/AAAAJgsI6VV6zE/name.jad HTTP/1.1
Host: www.merchant.at
Accept: text/vnd.sun.j2me.app-descriptor
User-Agent: CoolPhone/1.4 Profile/MIDP-2.0 Configuration/CLDC-1.0
Accept-Language: en-US
Accept-Charset: utf-8
```

Ein Beispiel der übertragenen Informationen beim JAR-Datei-Request ist in Tabelle 7 zu finden. Es handelt sich hierbei um eine standardmäßige HTTP-GET-Anfrage. Aus dieser HTTP-GET-Anfrage wird wiederum die Information über User-Agent, Language und Charset ausgewertet, außer diese wurde nicht übermittelt so wie es in diesem durchaus typischen Beispiel der Fall ist.

Tabelle 7: JAR-Datei Request

```
GET http://www.merchant.at/AAAAJgsI6VV6zE/name.jar HTTP/1.1
Host: www.merchant.at
Accept: application/java, application/java-archive
```

6.2.3 Clearing

Nachdem die JAR-Datei vollständig vom Distribution Server heruntergeladen wurde, informiert der Distribution Server den Authentication Server über den Abschluss der Transaktion. Der Authentication Server veranlasst damit aber auch den Abschluss der Transaktion durch das externe Abrechnungssystem(Clearing genannt).

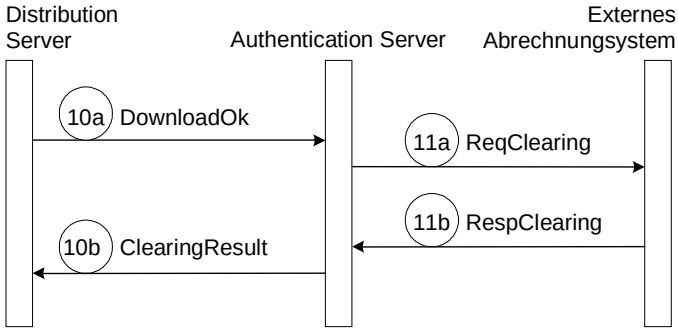


Abbildung 14: Clearing

Der Kommunikationsablauf ist in *Abbildung 14* graphisch dargestellt und gliedert sich in folgende drei Schritte:

1. Der Distribution Server benachrichtigt den Authentication Server, dass der Download und somit die Transaktion beendet ist und fordert damit den Authentication Server auf, das Clearing zu starten (10a)
2. Das Ergebnis des Clearings wird dem Distribution Server in (10b) mitgeteilt
3. Der Authentication Server beantragt das Clearing der Transaktion mit *ReqClearing* (11a). Das Abrechnungssystem (Billing System) bearbeitet die Anfrage und bestätigt den Erhalt (11b).

Den Clearing Request der Transaktion an das externe Abrechnungssystem könnte auch der Notification Server nach Erhalt der Status Updates durchführen. Dies hat allerdings den Nachteil, dass das Abrechnungssystem dann mit dem Authentication Server (Überprüfung der Bonität und Reservierung des Betrages) und dem Notification Server (Clearing) kommunizieren müsste. Außerdem ist der Mobilfunkbetreiber laut Abschnitt 5.1.1 für die Abrechnung am besten geeignet und somit wird die Kommunikation mit dem Abrechnungssystem vom Authentication Server durchgeführt.

6.2.4 Notification

Wenn der Benutzer die MIDlet-Suite am Endgerät installiert bzw. deinstalliert (löscht) sendet die AMS eine *Install-Notification* bzw. ein *Delete-Notification* an den Distribution Server (siehe Abschnitt 5.1.4). Diese Nachrichten werden vom Distribution Server an den Notification Server weitergeleitet.

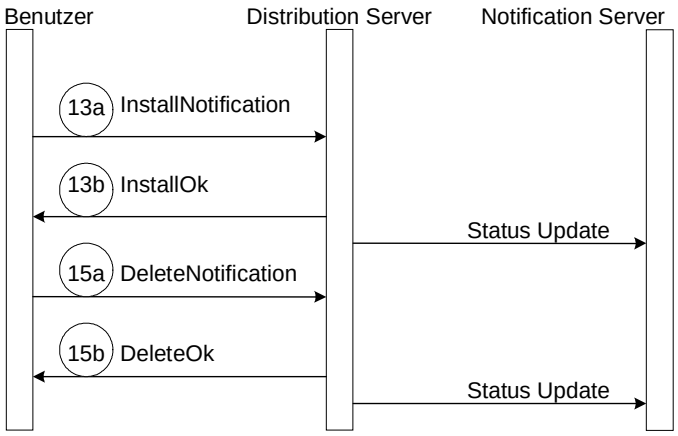


Abbildung 15: Benachrichtigungen Installation/Deinstallation

Der Kommunikationsablauf für Install und Delete Notifications ist in *Abbildung 15* dargestellt, und umfasst folgende Schritte:

1. Nach der Installation der MIDlet-Suite (unabhängig ob fehlerhaft oder erfolgreich) sendet die AMS an die in der JAD-Datei angegebenen *MIDlet-Install-Notify-URL* den Installationsstatus Report (13a)

2. Der Distribution Server antwortet darauf mit *OK* (13b) und leitet die Nachricht an den Notification Server weiter (Notification Code: in [codeNr.])

3. Wenn das MIDlet wieder vom mobilen Endgerät deinstalliert (gelöscht) wird, sendet die AMS analog an die in *MIDlet-Delete-Notify-URL* eine *Delete-Notification* (15a)

4. Der Distribution Server antwortet wiederum mit *OK* (15b) und leitet die Nachricht dem Notification Server weiter (Notification Code: dn [codeNr.])

Die Installations-Codes die vom mobilen Endgerät (bzw. AMS) übermittelt werden sind in Tabelle 4 (Kapitel 2) zu finden.

6.3 Diskussion

In diesem Kapitel wurde die Kommunikation zwischen den Systemkomponenten und dem Benutzer erläutert. Die aus Abschnitt 6.1 angestellten Überlegungen dazu, betreffen weitestgehend die Sicherheit und führen auf wie die Realisierung des Kommunikationsablaufs zustande gekommen ist.

Da die Kommunikation zwischen Benutzer und Distribution Server nur über eine unverschlüsselte Verbindung per HTTP (siehe Abschnitt 4.2.3) erfolgen kann, wird die JAD- und JAR-Datei im Klartext übertragen. Da das JARLogin/Passwort Paar auch im Klartext übertragen wird, dürfen diese Logindaten nur einmal gültig sein. Damit wird verhindert, dass einem Benutzer Schaden durch Dritte entsteht, indem ein Angreifer in irgendeiner Weise ein JARLogin und Passwort ausforscht und damit Applikationen herunterlädt. Zusätzlich müsste ein Angreifer aber auch noch die richtige JAD-Datei-Download-URL bzw. die richtige JAR-Datei-Download-URL besitzen, um einen Download durchführen zu können. Aber die JAD-Datei-Download-URL wird ja im Normalfall (siehe Abschnitt 6.2.1) per verschlüsselte SMS-Nachricht übertragen und die JAR-Datei-Download-URL alleine ist unbrauchbar, da diese relativ zur JAD-Datei-Download-URL vergeben wird. Somit hat ein Angreifer „keine“ Chance an die Download-URL der JAR-Datei (verweist auf die Applikationen) anhand der JAD-Datei zu gelangen. Die AMS hingegen erhält die JAR-Datei-Download-URL durch die relative Pfadangabe zur JAD-Datei-Download-URL automatisch.

7 Systemimplementierung

Make it as simple as possible, but not simpler.
Albert Einstein

Dieses Kapitel beschreibt die entwickelten Distribution, Authentication und Notification Server. Dabei werden die grundlegende Funktionalität des Systems, und deren Komponenten von der technischen Seite dargestellt.

7.1 Einführung

In diesem Abschnitt werden die verwendeten Technologien beschrieben und deren Zweck in dieser Arbeit erläutert.

7.1.1 JSR124-Referenzimplementierung

Der entwickelte Distribution Server baut wie schon in Abschnitt 3.2.2 beschrieben auf der J2EE-Client-Provisioning-Referenzimplementierung (JSR124-RI [JSR124V03]) auf. Diese Implementierung stellt eine „Provisioning-Plattform“ zur Verfügung und wurde für den Distribution Server erweitert und angepasst.

Das Provisioning, also das Versorgen der Benutzer mit Applikationen geschieht grundsätzlich in zwei Schritten:

- Finden und Bereitstellen von Applikationen (Discovery)
- Übertragen von Applikationen (Download)

Das Bereitstellen von Applikationen endet bei der JSR124-RI mit der Produktion von URLs, die der Benutzer anfordern kann, um eine bestimmte Applikation herunterzuladen. Diese URL wird durch Aufrufen der Methode *getDeliveryURI()* des FulfillmentTask-Objekts erzeugt. Das FulfillmentTask-Objekt kapselt die Aufgabe des Übertragens der Applikation in einer Transaktion ab (Transaktionen, siehe Abschnitt 6.1.3). Um jetzt einzelne Transaktionen am Server zu identifizieren bekommen diese eine eindeutige Identifikationsnummer, die so genannte FullfillmentID.

Die JSR124-RI arbeitet mit diesen FullfillmentIDs anhand derer sie Transaktionen (FullfillmentTasks) erkennt und bearbeitet. Die FullfillmentIDs werden anhand einer fortlaufenden Nummer vergeben und sind auch Bestandteile der URLs, anhand derer später die Applikationen heruntergeladen werden können. Das heißt also, dass die URLs keine zufälligen Bestandteile enthalten. Dies wird aber genau in dieser Arbeit gefordert wurde (siehe Abschnitt 6.1.4).

Um dieses Manko zu beseitigen werden in dieser Arbeit eigene Transaktion-IDs und Transaktionen verwendet und der Distribution Server koppelt die externen Transaktionen mit den internen Transaktionen (FullfillmentTasks) der JSR124-RI. Dazu findet man in der Distribution Server Datenbank das Feld *fft* in der Tabelle *TransactionTable* wo die internen JSR124 FullfillmentIDs abgespeichert sind, die zur jeweiligen Transaktion gehören. Siehe EER Diagramme im Anhang.

Das heißt also zu einer Transaktion am Distribution Server existiert genau ein FulfillmentTask-Objekt der JSR124-RI. Und ein FulfillmentTask-Objekt der JSR124-RI besitzt genau eine Transaktion am Distribution Server. Der Distribution Server kommuniziert mit den anderen Servern und dem Benutzer (bzw. AMS) nur über die eigenen Transaktionen. Intern aber reicht er die Anforderungen an die JSR124-RI-FullfillmentTask weiter.

Eine Beschreibung der weiteren Aufgaben der FullfillmentTasks und deren Objekte finden sich in [JSR124V03].

7.1.2 Koppelung JSR124 und Distribution Server

Der Distribution Server besteht grundsätzlich, wie alle drei Server, aus einem Webserver und einer Datenbank. Der Webserver nimmt Anfragen entgegen und übergibt sie einem Servlet[33], das die Funktion des Distribution Servers implementiert. Dieses Servlet kommuniziert mit der Datenbank im Hintergrund.

Für die Entwicklung des Distribution-Server-Servlet musste der Quellcode der JSR124-Referenzimplementierung dekompiliert werden und die Methoden die in Verbindung zum Benutzer stehen abgeändert werden.

Die Dekompilierung war deshalb nötig, da die JSR124-Referenzimplementierung zwar über Konfigurationsdateien anpassbar ist, aber wie schon im vorherigen Abschnitt beschrieben ist es nicht möglich die URLs, die die JSR124-Referenzimplementierung (JSR124-RI) pro Transaktion generiert, zu verändern oder anzupassen. Da diese URLs der JSR124-RI aus einer fortlaufenden Nummer

[33] Servlet Technologie siehe [Hall01] und Servlet Spezifikation [SERVSP01].

bestehen, konnten diese URLs nicht verwendet werden, und wurden deshalb durch die URLs mit einem Zufallsteil ersetzt, die dem Format aus Abschnitt 6.1.3 entsprechen.

7.1.3 Administration der Server

Die drei Server (Distribution, Authentication und Notification Server) können vollständig über ein HTML-Benutzerinterface [HTML01], das über Webbrowser per HTTP- bzw. HTTPS-Protokoll erreichbar ist, konfiguriert und administriert werden. Weiters erhält der Server-Administrator über dieses Benutzerinterface Informationen über den aktuellen Status des Servers und den Inhalt der Datenbank des Servers.

Diese Informationsdarstellung und Server-Administration erfolgt im Administrationsbereich des jeweiligen Servers, in den der Administrator anhand eines Logins und Passworts kommt. Login und Passwort des Administrators werden in den jeweiligen Datenbanken der jeweiligen Server abgespeichert. In den Tabellen: *AdminUserTable_Dist* beim Distribution Server, *AdminUserTable_Aut* beim Authentication Server und *AdminUserTable_Not* beim Notification Server werden Login und Passwort des Administrators abgespeichert. Siehe EER Diagramme im Anhang.

Die Administrationsseiten sind als Java Server Pages (JSP) ausgeführt. Dies ist eine Technologie der Java-2-Enterprise-Edition (J2EE, siehe [Hall01]). Die jeweiligen Server bestehen aus einem Servlet, das die Funktionalität des Servers erfüllt. Die Servlets kommunizieren über GET- und POST-Anfragen des HTTP-Protokolls mit dem Benutzer, bzw. anderen Servern. Die Servlet Technologie ist wiederum Bestandteil der Java-2-Enterprise-Edition (J2EE) [Roß01].

Für das JSP-Benutzerinterface, wurde im Rahmen dieser Diplomarbeit zusätzlich eine Library mit Utility- und HTML-Funktionen erstellt. Diese Library stellt Methoden für die Formatierung von Strings, Darstellung von HTML-Tabellen, Sortierung von Datenbanktabellen, Scrolling von Tabellen, DHTML[34]-Funktionen und vieles mehr zur Verfügung.

7.1.4 Distribution-Server-Servlet

Wie schon im vorherigen Abschnitt beschrieben, bedient sich das Distribution-Server-Servlet der J2EE-Servlet-Technologie. Das Distribution-Server-Servlet

[34] DHTML: Dynamisches HTML [DHTML]

besitzt drei grundlegende Methoden die, die Hauptaufgaben des Servlets durchführen.

Methode Init()

Diese Methode wird bei der Initialisierung des Servlets aufgerufen. Hier werden neben Initialisierungsschritten auch die Parameter aus der Konfigurationsdatei web.xml gelesen. In dieser Konfigurationsdatei sind alle Parameter festgelegt die für den Betrieb des Servlets und somit des Server notwendig sind (siehe allgemeine Serverdokumentation [S3SERV03]).

doPost()-Methode

Diese Methode wird aufgerufen wenn eine Install- oder Delete-Notification Statusmeldung von der AMS an den Distribution Server geschickt wird oder wenn eine Meldung (in Form einer XML-Datei) vom Authentication Server über eine erfolgreiche Authentifizierung eines Kunden empfangen wird.

doGet()-Methode

Alle Anforderungen, wie das Präsentieren der Liste der MIDlet-Suites, den Start des Downloads, das Präsentieren der Downloaddaten und der eigentliche JAD- bzw. JAR-Datei-Download übernimmt diese Methode am Distribution Server.

Sollte ein genauerer Einblick in das Servlet des Distribution Servers gewünscht sein, sollte die JSR124-RI Dokumentation [JSR124V03] und die Distribution Server Dokumentation [DISTS03] konsultiert werden.

7.1.5 Externes Abrechnungssystem

Für das externe Abrechnungssystem wurde eine allgemein gültige Schnittstelle implementiert, anhand derer es ein Leichtes ist, ein Abrechnungssystem einzubinden. Dazu wurde die Klasse *Billingobject* im Package *at.ac.tuwien.ict.s3gis* geschaffen, die die Methoden enthält, die für die Kommunikation mit einem Abrechnungssystem benötigt werden. Diese Klasse wurde bewusst einfach gehalten, um das Handling des Billing-Objekts bzw. Abrechnungssystems nicht auf einen bestimmten Typ von Abrechnungssystem festzulegen.

Sollte ein Abrechnungssystem eingebunden werden, sind folgende Methoden entsprechend zu erweitern:

- sendBillingData: Sendet sämtliche für die Abrechnung relevanten Daten an das externe Abrechnungssystem und liefert dessen Antwort zurück

- sendReqClearing: sendet den Clearingrequest an das externe Abrechnungssystem und liefert dessen Antwort zurück (siehe Abschnitt 6.2.3 in Kapitel 6)

Die Informationen, die in diesen Methoden versendet und empfangen werden, sind noch nicht festgelegt, da im Rahmen des Projektes kein Billingsystem definiert wurde. Die Klasse *Billingobject* wurde dabei so konzipiert, dass diese Daten per s*et()*- und g*et()*-Methoden gesetzt und ausgelesen werden, um einen gleich bleibenden Methodenaufruf der Methoden sendBillingData und sendReqClearing zu ermöglichen. Momentan wird im Projekt ein leeres *Billingobject* verwendet, das keine Funktion ausführt. Als Beispiel für zu übertragene Daten wurden der Nettopreis und die Währung im Sourcecode der *Billingobject*-Klasse eingefügt, um zu veranschaulichen wie diese Vorgehensweise konzipiert ist.

7.2 Distribution Server

Unter der URL `http://<Distribution_Server_Host>/distro` wird der Startbereich des Distribution Servers aufgerufen (Abbildung 16 zeigt diese Startseite).

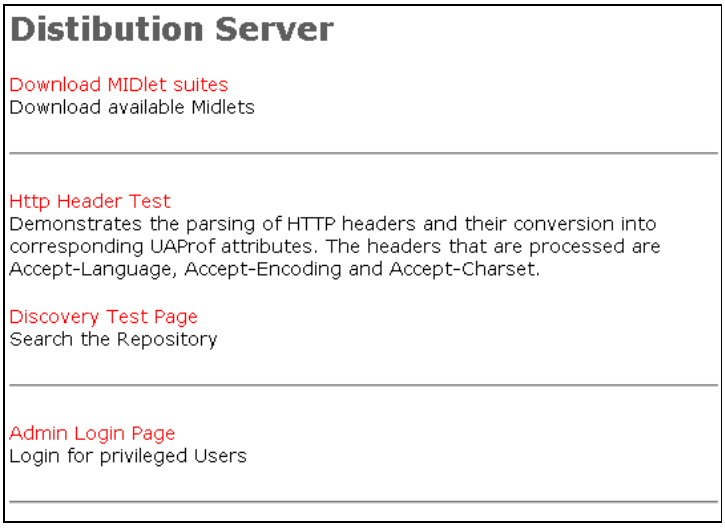


Abbildung 16: Startseite des Distribution Server

Von der Startseite können die folgenden Funktionen aufgerufen werden:

- Unter dem Punkt „*Download MIDlet-Suites*“ bekommt der Benutzer eine Liste der MIDlets, die vom Distribution Server zur Verfügung gestellt werden, und die der Benutzer im nächsten Schritt herunterladen kann

- Hinter dem Link „*HTTP Header Test*“ verbirgt sich eine Testseite, die sämtliche HTTP-Header Felder anzeigt, die der Browser bzw. die AMS an den Server schickt
- Die „*Discovery Test Page*“ dient dazu, um Applikationen innerhalb des Servers zu suchen
- Schließlich führt der Link „*Admin-Login-Page*“ in den Administrationsbereich des Distribution Servers

7.2.1 Administrationsbereich

Nach Eingabe des Administrator Logins und Passwortes, das in der Tabelle *AdminUserTable_Dist* gespeichert ist, erreicht der Server-Administrator den Administrationsbereich. Hier stehen die Bereiche: Transaktionen, Bundles verwalten, sowie Bundles und Devices zur Verfügung, in denen Einstellungen geändert und Informationen über den Status des Distribution Servers erhalten werden können. Diese Bereiche werden nun in den folgenden Unterabschnitten 7.2.2 bis 7.2.3 erklärt.

7.2.2 Transaktionsbereich

In Abbildung 17 wird die Transaktionstabelle aus dem Transaktionsbereich dargestellt, die standardmäßig nach dem Login in den Administrationsbereich, angezeigt wird bzw. über das Menü *Transaktionen verwalten* erreicht werden kann.

ID	CUST	RANDOMD	DOWNUID	PASS	CREATET	EXPIRE	CLEARENCE	EXP	JAD	JAR	PASSF	MAXP	BUNDLEID	
2		U0/y6WIT	095211	8592	2003-10-20 22:46:46.02	2003-10-20 23:31:46.02	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
3	1	hfAoYOSA	174904	0192	2003-10-20 22:51:02.329	2003-10-20 23:36:02.329	1	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
4		9QVmh8wM	455431	5796	2003-10-20 22:51:22.978	2003-10-20 23:36:22.978	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
5		KgUVAlF+	153525	6208	2003-10-21 09:49:11.631	2003-10-21 10:34:11.631	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
6		NH/pPYfn	084542	3267	2003-10-21 09:55:53.108	2003-10-21 10:40:53.108	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
7		DLeU8U9b	418792	4467	2003-10-21 09:58:26.499	2003-10-21 10:43:26.499	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
8		rKxmdcX/	060486	0041	2003-10-21 10:19:30.176	2003-10-21 11:04:30.176	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
9		RjD2HvMt	538154	6192	2003-10-21 10:22:49.913	2003-10-21 11:07:49.913	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid
10		FiM4kE1k	342205	5882	2003-10-21 10:31:45.303	2003-10-21 11:16:45.303	0	0	0	0	-1	9	2.1	Freigeben Sperren Expired Valid

Abbildung 17: Distribution Server Administration Transaktionen

In dieser Tabelle werden sämtliche Transaktionen aufgelistet. Im Detail enthält diese Auflistung:

- ID: Identifikationsnummer des Datensatzes innerhalb dieser Tabelle
- CUST: (CustomerID) Identifikationsnummer des Benutzers, der diese Transaktion gestartet hat. Diese Identifikationsnummer wird erst im Verlauf

des Downloads bekannt und wird erst bei Benachrichtigung des Authentication Servers in die Datenbank eingetragen. Beim Start der Transaktion ist dieser Eintrag leer bzw. enthält den Wert NULL

- RANDOMD: Der zur Transaktion gehörende, zufällig generierte Teil der URL. Diese URL enthält diese Zufalls-Nounce wie es im Abschnitt 6.1.3 beschrieben ist
- DOWNUID: Das einmal gültige JARLogin für den JAR-Datei-Download
- PASS: Das einmal gültige Passwort für den JAR-Datei-Download
- CREATET: Datum- und Zeitstempel, des Erstellens der Transaktion
- EXPIRE: Datum- und Zeitstempel, wann die Transaktion ungültig wird. Ab diesem Zeitpunkt ist auch mit gültigem JARLogin und Passwort kein Download mehr möglich
- CLEARENCE: Attribut das den Freigabestatus der Transaktion anzeigt
- EXP: (Expired) Attribut das anzeigt, ob die Transaktion verfallen und somit ungültig ist, wobei der Wert 0 für Falsch (false) und 1 für Wahr (true) steht
- JAD: Anzahl der JAD-Datei-Downloads
- JAR: Anzahl der JAR-Datei-Downloads
- PASSF: Anzahl wie oft das Passwort für den JAR-Datei-Download falsch eingegeben wurde. Dieser Wert startet mit -1, da der Browser das erste Mal immer versucht die JAR-Datei ohne Passwort zu holen und dabei eine „Passwort Falsch“ Meldung erzeugt. Dieses Verhalten wird von allen Browsern hervorgerufen, bevor die eigentliche Authentifizierung anhand Basic Authentication [RFC2617] startet
- MAXP: Hier wird die Anzahl angezeigt wie oft der Benutzer das Passwort falsch eingeben darf, bis die Transaktion ungültig wird
- BUNDLEID: Die zugehörige BundleID[35], die die MIDlet-Suite identifiziert
- In der letzten Spalte bekommt der Administrator vier Hyperlinks präsentiert, um eine Transaktion freizugeben, zu sperren, oder auf den Status abgelaufen bzw. gültig zu setzen

Eine Transaktion (siehe Abschnitt 6.1.3) wird vom Distribution Server freigegeben, wenn die Authentifizierung des Benutzers gegenüber dem Authentication Server erfolgreich war. Diese Transaktion kann auch vom Administrator manuell freigegeben werden.

[35] Applikationspakete, die vom J2EE-Client-Provisioning-Server verwaltet werden, werden „Bundle“ genannt (siehe Abschnitt 3.2.1).

Eine Transaktion ist gültig wenn sie sich innerhalb der Ablaufzeit befindet. Also zwischen dem Erzeugungszeitpunkt (CREATET) und der Ablaufzeitpunkt (EXPIRE).

Eine Transaktion muss freigegeben und gültig sein damit der entsprechende Download gestartet werden kann.

7.2.3 Bundleverwaltung

In diesem Bereich hat der Administrator die Möglichkeit die Bundles (bzw. MIDlet-Suites), die am Distribution Server zur Verfügung stehen zu verwalten. Dieser Bereich stellt somit den Content Manager zur Verfügung und gliedert sich in drei Unterbereiche:

- Unter „*Repository Contents"* kann der Administrator in das Repository (Lager, in dem sich die MIDlet-Suites befinden) hineinschauen und sämtliche Informationen darüber erhalten. Durch Auswahl der Bundles kann der Administrator eine PAR-Datei (siehe Ende dieses Abschnitts) öffnen, und deren Inhalt mit all seinen MIDlet-Suites anzeigen lassen. Durch Klick auf *Remove* kann der Server-Administrator letztendlich diese PAR-Datei mit samt seinen MIDlet-Suites wieder aus dem Repository entfernen.

- Im Bereich „*Applikationen für Distribution Server bereitstellen"* können Applikationen auf den Distribution Server anhand einer PAR-Datei gelegt werden. Diese hochgeladenen MIDlet-Suites aus der PAR-Datei, können dann im nächsten Schritt von den Benutzern heruntergeladen werden.

- Im Bereich *Empty Repository* kann der Server-Administrator alle Bundles (MIDlet-Suites), die sich auf dem Server befinden, in einem Schritt vom Server löschen.

- In Bundle Bereich werden alle Bundles (MIDlet-Suites) aufgelistet, die vom Server-Administrator am Distribution Server hochgeladen wurden. Zusätzlich kann der Server-Administrator weitere detaillierte Informationen über einzelne MIDlet-Suites erhalten, wenn er diese selektiert.

- Devices: Hier werden alle Geräte (Devices) aufgelistet, die der JSR124 Referenzimplementierung (JSR124-RI) bekannt sind. Diese Geräteliste ist für die Referenzimplementierung konzipiert, damit diese entscheiden kann welche MIDP-Version welches Gerät unterstützt und z.B. so Anfragen vorselektieren kann. Die verwendete Referenzimplementierung Version 1.0Beta verwendet diese Geräteliste aber noch nicht. Somit dient diese Liste lediglich als reine Information und ist für diese Arbeit uninteressant.

Eine PAR-Datei ist nichts anderes als eine ZIP-Datei[36], die eine Menge von Applikationen (in diesem Fall MIDlet-Suites) und einen PAR-Datei-Descriptor enthält (Beschreibung des Inhaltes der PAR-Datei). Diese PAR-Datei wird Bundle genannt und dient dem Server-Administrator als Verpackungseinheit für das Hochladen der Applikationen. Der JSR124 [JSR124V03] ist wie in Abschnitt 3.2.1 beschrieben nicht auf das Bereitstellen von MIDlet-Suites beschränkt, sondern ermöglicht allgemein das Bereitstellen von Informationen (Dateien in verschiedenen Formaten) und Applikation, deshalb existiert diese allgemeine Verpackungsdatei. Diese Datei enthält im Falle von MIDlet-Suites, die JAD- und JAR-Datei der MIDlet-Suite, die von der JSR124-Referenzimplementierung beim Hochladen aus der PAR-Datei geholt werden. Details zum Aufbau einer PAR-Datei sind in [JSR124V03 Kapitel 9.2] zu finden.

7.3 Authentication Server

Der Authentication Server Administrationsbereich kann unter der URL `http://<Authentication_Server_Host>/authent` aufgerufen werden und ist wie in Abschnitt 7.1.3 beschrieben, nur mit gültigem Administrator Login zugänglich.

Im nachfolgenden Abschnitt werden die einzelnen Bereiche des Authentication Server Administrationsbereichs beschrieben.

7.3.1 Logins für die Authentifizierung

In Abbildung 18 sieht man den Bereich *„Zulässige Logins“*. In diesem Bereich werden alle MSISDN-Nummern (bzw. der dazugehörige frei wählbare Login) aufgelistet, anhand derer der Authentication Server Benutzer authentifizieren kann.

[36] ZIP-Datei: Archivdatei im ZIP-Format [ZIPF]

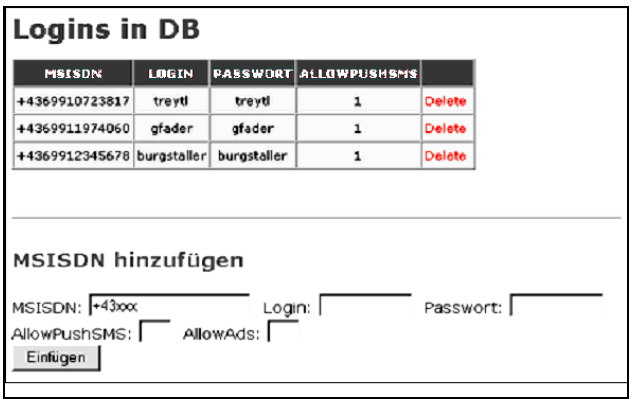


Abbildung 18: Authentication Server Administration Logins in Datenbank

In diesem Bereich hat der Administrator des Authentication Servers die Möglichkeit neue Benutzer mit ihrer MSISDN hinzuzufügen und zusätzlich zur MSISDN einen Benutzername zu vergeben, der alternativ zur MSISDN bei der Authentifizierung verwendet werden kann. Weiters muss für jeden Benutzer angegeben werden, ob er PushSMS erlaubt. Das Flag für PushSMS regelt also das Versenden von SMS-Nachrichten zur Information über eine Applikationsupdate (siehe Abschnitt 5.2.3), und ist im Zusammenhang mit dem Verbot unaufgeforderter elektronischer Werbung (E-Commerce Gesetz) integriert worden.

7.3.2 Transaktionsbereich

Im Transaktionsbereich werden sämtliche Transaktionen des Authentication Servers angezeigt. Eine Transaktion im Authentication Server ist ein Eintrag in der Datenbank, der besagt, dass ein Benutzer sich bei diesem Authentication Server authentifiziert hat oder sich noch authentifizieren muss.

ID	MERCHANTID	TRANSACTIONID	STATUS	MSISDN	CREATETIME	EXPIREDTIME
2	1111	3	2	+4369911974060	2003-10-20 22:51:02.839	2003-10-20 23:41:02.839
3	1111	4	1		2003-10-20 22:51:23.469	2003-10-20 23:41:23.469
4	1111	5	1		2003-10-21 09:49:13.123	2003-10-21 10:39:13.123
5	1111	45	1		2003-10-22 14:37:35.749	2003-10-22 15:27:35.749
6	1111	46	1		2003-10-22 14:39:00.281	2003-10-22 15:29:00.281
7	1111	47	2	+4369911974060	2003-10-22 14:41:41.573	2003-10-22 15:31:41.573
8	1111	48	1		2003-10-22 14:48:41.947	2003-10-22 15:38:41.947

Abbildung 19: Authentication Server Administration Transaktionen

In Abbildung 19 sieht man die Transaktionstabelle aus dem Administrationsbereich. Hier werden sämtliche Transaktionen aufgelistet, die dem Authentication Server bekannt sind. Dabei wird zu jeder Transaktion ihr Status angegeben: angelegt (Feldwert 1), autorisiert (Feldwert 2) oder abgeschlossen (Feldwert 3).

Felder, die in diesem Bereich angezeigt werden, sind:

- ID: Primärschlüssel für den Transaktion Datensatz in der Datenbank
- MERCHANTID: Die ID des Händlers bzw. des Distribution Servers
- TRANSACTIONID: Die Transaktionsnummer, die vom Distribution Server übermittelt wurde
- STATUS: Status der Transaktion
    - 1: angelegt: Transaktion wurde am Authentication Server erzeugt und der Benutzer hat ab diesem Zeitpunkt die Möglichkeit sich für die angegebene Transaktion zu authentifizieren
    - 2: autorisiert: Benutzer wurde erfolgreich authentifiziert, und die Freigabe des Downloads wurde an den Distribution Server gemeldet
    - 3: abgeschlossen: Der Benutzer hat die Applikation bezahlt und heruntergeladen und der Distribution Server hat die Benachrichtigung an die anderen Server geschickt dass die gesamte Transaktion abgeschlossen ist
- MSISDN: Die MSISDN anhand derer der Benutzer identifiziert wird
- CREATETIME: Datum- und Zeitstempel, wann die Transaktion erzeugt wurde
- EXPIREDTIME: Datum- und Zeitstempel, wann die Transaktion abläuft

7.3.3 Authentifizierung von Distribution Servern

Bevor ein Distribution Server mit einem Authentication Server kommunizieren kann, muss sich der Distribution Server mittels Basic Authentication [RFC2617] gegenüber dem Authentication Server authentifizieren. Dazu muss der Distribution Server im Bereich *„Zulässige Distribution Server“* beim Authentication Server eingetragen werden. Ansonsten werden Anfragen von diesem Distribution Server abgelehnt. Dazu wird einfach die ID und Passwort des Distribution Servers und optionale Adressdaten eingegeben. Für die erfolgreiche Authentifizierung nach Basic Authentication des Distribution Servers ist die ID und das Passwort relevant (siehe Abschnitt 6.1.1). Als Username wird für die Basic Authentication die ID des Distribution Servers verwendet.

7.3.4 Benutzer pro Distribution Server

In diesem Bereich *„Benutzer pro Distribution Server„* werden die Identifikationsnummern (CustomerIDs) der Benutzer, die Transaktionen auf dem Distribution Server durchführen bzw. durchgeführt haben, pro Distribution Server angezeigt.

Wie in Abschnitt 5.2.1 beschrieben, kann ein Authentication Server mehrere Distribution Server bedienen, und legt für jeden Benutzer pro Distribution Server eine eigene CustomerID an, um die verschiedenen Benutzer zu identifizieren.

Die CustomerIDs werden vom Authentication Server automatisch vergeben. Wenn ein Benutzer bei einer Anfrage eines Distribution Servers noch keine CustomerID für einen Distribution Server besitzt. Diese CustomerID (Benutzer-Identifikationsnummer) ist von Distribution Server zu Distribution Server verschieden und wird pro Distribution Server immer fortlaufend vergeben.

Zu den Benutzer-Identifikationsnummern (CustomerIDs) ist besonders hervorzuheben, dass diese vom Authentication Server generiert werden, und nur der Authentication Server einen Zusammenhang zwischen CustomerID und MSISDN herstellen kann. Der Distribution und der Notification Server erkennen die einzelnen Benutzer nur anhand deren CustomerID und können von sich aus auf keine weiteren Daten schließen. Somit ist die Anonymität des Benutzers gegenüber den verschiedenen Servern garantiert.

7.4 Notification Server

Der Notification Server Administrationsbereich gliedert sich, wie die Administrationsbereiche der anderen Server auch in verschiedene Unterbereiche, die im folgenden Abschnitt erklärt werden. Der Notification Server Administrationsbereich kann unter der URL `http://<Notification_Server_Host>/notify` nur mit gültigem Administrator Login verwendet werden, wie es im Abschnitt 7.1.3 beschrieben ist.

7.4.1 Benachrichtigungen

In Abbildung 20 ist der Bereich Benachrichtigungen dargestellt. Hier werden alle Notifications, das sind Benachrichtigungen und Statusmeldungen von den anderen Servern gebündelt nach der zugehörigen Transaktion dargestellt (siehe Abschnitt 6.1.3). Unter dem Hyperlink *Detail* in der letzten Spalte in der Abbildung kann der Server-Administrator die einzelnen Notifications zu einer Transaktion abrufen.

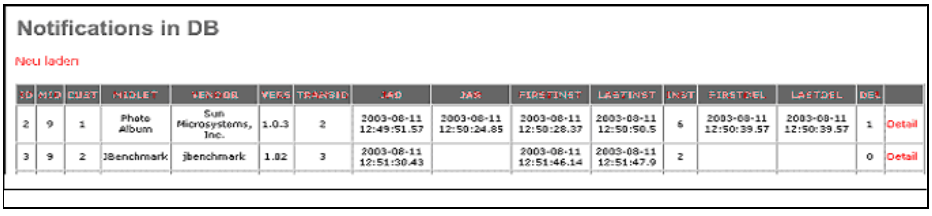


Abbildung 20: Notification Server Administration Benachrichtigungen

Die Felder, die in diesem Bereich angezeigt werden beinhalten:

- ID: Primärschlüssel des Datensatzes in der Datenbank
- MID: Die ID des Händlers (Merchant ID) beziehungsweise des Distribution Server
- CUST: Die CustomerID (Benutzer-Identifikationsnummer); die den Benutzer identifiziert der zu dieser Transaktion gehört. Wenn dieser Wert *„-1"* ist, heißt das, dass der Benutzer noch nicht bekannt ist. Diese Werte werden vom Authentication Server vergeben und werden bei Bekanntgabe eingetragen. Wie in Abschnitt 7.3.4 erklärt sind diese CustomerIDs eindeutig pro Benutzer
- MIDLET: Name der MIDlet-Suite, die zu dieser Transaktion gehört
- VENDOR: Hersteller der MIDlet-Suite die zu dieser Transaktion gehört
- VERS: Version der MIDlet-Suite die zu dieser Transaktion gehört
- TRANSID: Eindeutige Identifikationsnummer der Transaktion, die vom Distribution Server erhalten wird
- JAD: Datum- und Zeitstempel, an dem das letzte Mal die JAD Datei heruntergeladen wurde
- JAR: Datum- und Zeitstempel, an dem das letzte Mal die JAR Datei heruntergeladen wurde
- FIRSTINST: Datum- und Zeitstempel der ersten Installation der MIDlet-Suite
- LASTINST: Datum- und Zeitstempel der letzten Installation der MIDlet-Suite
- INST: Anzahl der erfolgreichen Installationen am mobilen Endgerät zu dieser Transaktion
- FIRSTDEL: Datum- und Zeitstempel, zu dem das erste Mal die MIDlet-Suite gelöscht wurde
- LASTDEL: Datum- und Zeitstempel, zu dem das letzte Mal die MIDlet-Suite gelöscht wurde
- DEL: Anzahl, wie oft diese MIDlet-Suite vom mobilen Endgerät gelöscht wurde

7.4.2 Abfragenbereich

Im Bereich Abfragen, hat der Server-Administrator die Möglichkeit Abfragen an die Datenbank abzuschicken. Neben den vordefinierten Anfragen aus Abschnitt 5.2.3 für:

- Anzahl der Benutzer
- Anzahl Applikationen

- Gelöschte Applikationen
- Zur Zeit installierte Applikationen
- Anzahl der Applikationsinstallation, d.h. wie oft wurde eine gewisse Applikation installiert
- Anzahl Applikationslöschen, d.h. wie oft wurde eine gewisse Applikation gelöscht
- Anzahl Applikationen Versuch Installation und wirkliche Installation

besteht auch noch die Möglichkeit benutzerdefinierte Kommandos abzusetzen.

Unter *Manuelle Abfrage* kann der Administrator ein beliebiges SQL-konformes Kommando eingeben und dieses dann an die Notification Datenbank abschicken. Unter dem Eingabefeld für die SQL-Abfrage wird als Hilfestellung die Struktur der Tabellen mit den Notifications dargestellt. Die Ausgabe der abgeschickten Abfrage wird dann als Tabelle angezeigt. Diese manuelle Abfrage hat den Vorteil, dass jede beliebige Abfrage an den Notification Server gestellt werden kann.

Diese SQL-Abfrage ist aus zwei Gründen sehr wichtig: Erstens erlaubte sie es zum Zeitpunkt der Projektdefinition nicht berücksichtigte Anfragen zu inkludieren und zweitens ist mit SQL ein mächtiges Standardinterface zu einer Vielzahl von Auswertungswerkzeugen gegeben.

7.4.3 Push Service

Durch das Push Service hat der Administrator des Notification Servers die Möglichkeit einen SMS-Text an eine Menge von Benutzern zu schicken.

Dazu gibt es zwei Möglichkeiten Benutzer zu bestimmen und zu selektieren, die im Rahmen dieser Arbeit als Demonstration entwickelt wurden:

- Nach Benutzer: Listet die Benutzer pro Distribution Server auf
- Nach Applikationen: Listet die Benutzer auf, die eine Applikationen installiert haben. Weiters werden hier Informationen angezeigt, wann die Applikation das erste und letzte Mal installiert wurden

Hat der Server-Administrator die Benutzer ausgewählt, an die er eine SMS-Nachricht versenden will (*SELECTED* Checkbox), gibt er den SMS-Text ein, der an diese Benutzer geschickt werden soll, und bestätigt mit dem *Submit*-Button. Der SMS-Text sollte sinnvollerweise die URL für den Start des Downloadvorgangs aus Abschnitt 5.3 enthalten, damit der Benutzer den entsprechenden Applikationsdownload starten kann.

Der Notification Server schickt dann an den Authentication Server die Anfrage für das Absenden der SMS-Nachrichten an diese Benutzer. Der Authentication Server

prüft daraufhin die Anfrage, kontrolliert, ob die Benutzer PushSMS erlauben, und schickt dann eine Liste mit Benutzer zurück an denen das Verschicken der SMS-Nachricht erfolgreich war.

7.4.4 Kaskadierung

Um die Informationsverarbeitung durch den Notification Server auch hierarchisch gestalten zu können, wurde die Möglichkeit geschaffen einzelne Notification Server miteinander zu vernetzen.

Ausgehend von der ersten Stufe, die ihre Daten vom Distribution Server empfängt, können Nachrichten automatisch an die weiteren Stufen weitergeleitet werden. Dabei sind prinzipiell sowohl ein Konzentrator- als auch eine Reproduktionsfunktion möglich (siehe Abbildung 21).

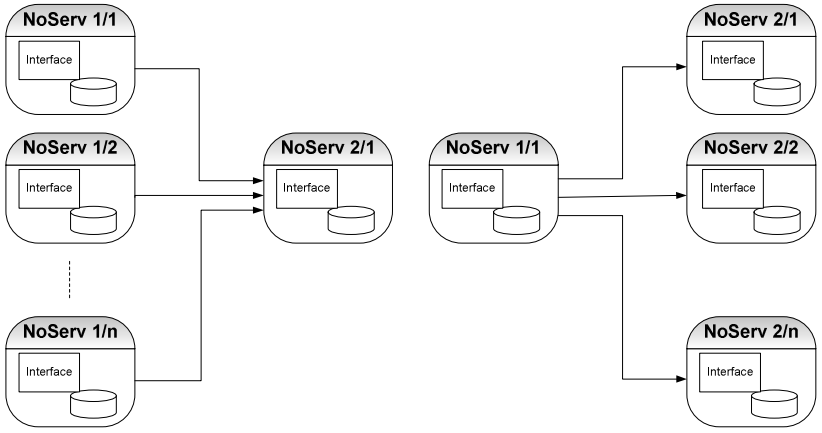


Abbildung 21: Vernetzung von Notification Servern (li. Konzentrator, re. Reproduktion)

Im Administrationsbereich „*Kaskadierung*" werden diese Kaskadierungen angezeigt und der Administrator kann diese Kaskadierungen von Notification Servern verwalten.

Die Kaskadierungen werden als Liste angezeigt, die aus den Bedingungen (für die Weiterleitung einer Notification) und deren URLs für den adressierten Notification Server besteht. Bekommt der empfangende Notification Server eine Notification, die den Bedingungen entspricht, wird diese Notification an die eingegebene Notification Server URL weitergeschickt.

Damit eine Bedingung erfüllt ist muss der Inhalt der Notification in den drei Feldern, *MIDLETVENDOR*, *MIDLETNAME* und *MIDLETVERSION* übereinstimmen. Dabei hat man auch die Möglichkeit über so genannte Wildcards

die Entsprechungen zu verallgemeinern. Das Prozentzeichen „%“ steht hierbei für eine beliebige Zeichenkette und der UnderScore „_“ steht für ein einzelnes beliebiges Zeichen.

Diese Kaskadierung von Notification Servern ermöglicht dem Administrator die Notifications von mehreren Servern an zentraler Stelle zu sammeln (Konzentrator) oder die Notifications auf mehreren Servern zu verteilen (Reproduktion). Die Anwendung dieser Mechanismen dient der gezielten Weiterleitung von Statusmeldungen. So können z.B. MIDlet-Suites eines gewissen Herstellers nur an diesen gewissen Hersteller weitergeleitet werden, oder der Mobilfunkbetreiber kann alle Notifications sammeln, um Informationen über die in seinem Netz verwendeten MIDlets zu erhalten.

7.5 Zusammenfassung

In diesem Kapitel wurden die einzelnen Server, Distribution-, Authentication- und Notification Server und deren technologischer Hintergrund beschrieben.

Die JSR124-Referenzimplementierung (JSR124-RI) aus Abschnitt 7.1.1 war Grundlage für die Implementierung des Distribution Servers. Auch wenn die Anpassung und Erweiterung der JSR124-RI vom technologischen Aufbau her vorgesehen wäre, so sind diese Erweiterungen und Anpassungen nur bis zu einem bestimmten Punkt möglich gewesen. Aus den Gründen aus Abschnitt 7.1.2 musste die JSR124-RI dekompiliert werden und der daraus entstandene Quellcode der Klassen weiterverarbeitet werden.

Durch den Distribution Server, der laut Kapitel 5 dem Applikationshersteller und dem Applikationsvertreiber zugeordnet ist, werden die Applikationen vertrieben. Demnach stellt der Distribution Server die eigentliche Applikationsdistribution zur Verfügung. Für jede Applikationsdistribution wird wie in Abschnitt 6.1.3 beschrieben eine Transaktion angelegt, die die Aufgabe des Übertragens einer Applikation abkapselt.

Für die Authentifizierung der Benutzer und die Anonymität der Benutzer innerhalb des Gesamtsystems, ist der Authentication Server verantwortlich. Den Status des Authentication Servers und dessen Authentications (Transaktionen) kann im Administrationsbereich eingesehen werden. In diesem Bereich kann der Authentication Server auch administriert werden und Einsicht in die Datenbank des Servers genommen werden.

Im Notification Server wurde die Möglichkeit geschaffen, eine Vernetzung von mehreren Notification Servern einzurichten, sodass es möglich ist, dass mehrere unabhängige Notification Server untereinander zu verbinden und so ein Tool für Profilbildung und Leistungsanalyse zur Verfügung zu stellen.

In der Systemspezifikation [S3SPEC03] und in der von mir geschriebenen Serverdokumentation [S3SERV03] findet man weiterführende technische Implementierungsdetails, die den Rahmen dieser Ausarbeitung sprengen würden.

Zusätzlich wurden von mir die drei Serverdokumentationen erstellt, in denen Entwicklungsdetails zu den einzelnen Servern gefunden werden kann: Distribution Server [DISTS03], Authentication Server [AUTHS03] und Notification Server [NOTS03].

8 Zusammenfassung und Ausblick

Die Historiker verfälschen die Vergangenheit,
die Ideologen die Zukunft.

Zarko Petan, slowenischer Schriftsteller

Ziel dieser Arbeit war es, ein sicheres Applikationsmanagementsystem zu entwickeln, das aber aus den Gründen aus Kapitel 4.2 gewissen Einschränkungen unterworfen ist. Deshalb wurde in dieser Diplomarbeit ein verteiltes System realisiert, das die grundlegenden Anforderungen, der Authentifizierung der Teilnehmer und der Authentizität des Systems, verwirklicht.

Erreichte Ziele

Durch die Verteilung der Aufgaben auf Distribution-, Authentication- und Notification Server ist der anonyme Vertrieb der Applikationen möglich, und die Privatsphäre der Benutzer wird garantiert. Der Distribution Server ist für die Applikationsdistribution zuständig, der Authentication Server übernimmt die Authentifizierung und der Notification Server dient als Datensammelstelle für die Profilbildung.

Die Authentifizierung des Benutzers gegenüber dem Hersteller und umgekehrt erfolgt über signierte SMS-Nachrichten oder alternativ dazu, aber weniger sicher über MSISDN (oder Benutzername) und Passwort. Die verschlüsselten SMS-Nachrichten, dienen dazu die Authentizität von den beteiligten Parteien sicherzustellen und einen Zugriffsschutz auf die Applikationen zu ermöglichen. Somit ist nur autorisierten Benutzern der Download von Applikationen erlaubt.

Weiters liefert die MIDP-Authentifizierung des Herstellers die Sicherheit für den Benutzer, dass die Applikation von einem gewissen Anbieter stammt, und stellt somit ein Gütemerkmal für die Applikation dar.

Das Ziel, die Applikationen vor Abhören und Manipulation zu schützen, wurde nicht ganz erreicht, da die AMS in der MIDP2.0-Spezifikation nur einen Download der JAR-Datei über eine ungesicherten HTTP Verbindung erlaubt. Deshalb ist es im Rahmen dieser Arbeit nicht zu verhindern, dass Dritte die JAD- oder JAR-Datei beim Download kopieren („Man-in-the-Middle"-Attacke). Das entwickelte und

implementierte Authentifizierungs- und Übertragungsverfahren (JAD-Datei-Download-URL) stellen lediglich sicher, dass zumindest der Benutzer, der die JAR-Datei erhält, der Benutzer ist, der sich authentifiziert hat. Auch durch die sichere Übertragung der JAD-Datei-Download-URL kann nicht garantiert werden, dass die JAR-Datei, die die Applikationen enthält, beim Download kopiert wird. Gewiss wird aber in einer der nächsten *MIDP*-Versionen der OTA-Download über HTTPS ermöglicht. Entsprechende Tendenzen sind in der Standardisierung erkennbar.

Das Kopieren und die Weitergabe von Applikationen direkt vom Endgerät ist mit der MIDP2.0-Spezifikation auch nicht zu verhindern, da jede Applikation über eine direkte Datenübertragung auf einen PC heruntergeladen werden kann und von dort weiterverbreitet werden kann.

Die stark gesicherte Authentifizierungsmethode konnte leider nicht umfassend getestet werden, da kein mobiles J2ME-fähiges Gerät zur Verfügung steht, das über eine Kryptographiechipkarte verfügt. Die Kombination eines Endgerätes mit SIM- und Kryptographiechipkarte (Dual Slot), die für das WASA-Projekt [Schöberl01] benötigt wird, und J2ME in einem Endgerät ist zum jetzigen Zeitpunkt noch nicht erhältlich. Somit wurde nur die weniger sichere Authentifizierung und Übertragung der JAD-Datei-Download-URL getestet. Die Funktionsfähigkeit des Systems mit Kryptographie-Chipkarte wurde im WASA-Projekt entwickelt, getestet und bewiesen.

Ausblick

Eine interessante Verbesserung des implementierten Systems wäre die Bezahlung und Autorisierung des Downloads erst nach dem Download der JAD-Datei durchzuführen. Denn nach dem JAD-Datei-Download könnten der Benutzer und die AMS zuvor noch genauer entscheiden, ob die MIDlet-Suite für sie interessant und lauffähig ist.

In den nächsten Versionen des Mobile-Information-Device-Profile ist weiters zu erwarten dass der OTA-Download auch über eine HTTPS-Verbindung stattfinden kann. Auch werden von den Herstellern, den Entwicklern und den Netzbetreibern, Sicherheitsvorkehrungen gewünscht, die das unerwünschte Kopieren von Applikationen über eine direkte Übertragung verhindern. Die beiden letztgenannten Erweiterungen könnten ohne Probleme in das realisierte System Einzug finden und würden das System um die fehlenden Sicherheitsstufen komplettieren.

Anhang

It is simplicity that is difficult to make.

Bertholdt Brecht

A Glossar

AMS	Application Management Software
API	Application Program Interfaces
CA	Certification Authority
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
DA	Discovery Application
EER	Enhanced Entity Relationship
GSM	Global System for Mobile Communication
HTML	Hypertext Mark-up Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transport Protocol Secure
IMSI	International Mobile Subscriber Identity
IOP	Infrastructure Operator
IP	Internet Protocol
ISDN	Integrated Service Digital Network
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition

J2SE	Java 2 Standard Edition
JAD	Java Application Descriptor
JAR	Java ARchive
JSP	Java Server Pages
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine
ME	Mobiles Endgerät
MExE	Mobile Station Application Execution Environment
MIDlet	Mobile Information Device Profile Applet
MIDlet-Suite	MIDlet Package
MIDP	Mobile Information Device Profile
MIDP	Wird in dieser Arbeit als Technologie: J2ME+ CDLC1.0+ MIDP 1.0 oder 2.0 verwendet
MIME	Multipurpose Internet Mail Extensions
MS	Mobile Station (Mobiles Endgerät + SIM Karte)
MSISDN	Mobile Subscriber-Integrated Service Digital Network
OOA	Object-Orientated Analysis
OOD	Object-Orientated Design
OOP	Object-Orientated Programming
OTA	Over The Air
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
RFC	Requests for Comments
S3GIS	Projektbezeichnung
SDL	Specification and Description Language

SIM	Subscriber Identification Modul
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Trusted Third Party
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunication System
URL	Uniform Resource Locators
VM	Virtual Machine
WAP	Wireless Application Protocol
WASA	Wireless Applications Security Architecture
WTK	Wireless Toolkit
XML	Extensible Mark-up Language

B Literaturverzeichnis

[Ammel02]	Ammelburger, Dirk: „XML mit Java“, Markt-und-Technik-Verlag, München, 2002
[AUTHS03]	Gfader Peter: „Authentication Server technische Dokumentation“, Internes Projektdokument S3GIS, Austria Card/ICT, 2003
[Bien01]	Bien, Adam: „Enterprise Java Frameworks“, Addison-Wesley, München, 2001
[Bodoff02]	Stephanie Bodoff: „The J2EE tutorial“, Addison-Wesley, 2002
[Boo94]	Booch, G.: “Object-orientated Analysis and Design with Applications.”, Benjamin/Cummings, Redwood, CA, 1994
[Bud02]	Budd, T. A.: “An Introduction to Object Oriented Programming (3rd Ed)”, Addison-Wesley, 2002
[Burke03]	Burke, Eric M.: „Java extreme programming cookbook“, O'Reilly, 2003
[CLDC1]	Connected, Limited Device Configuration Specification Version 1.0a, Micro Edition
[DISTS03]	Gfader Peter: „Distribution Server technische Dokumentation“, Internes Projektdokument S3GIS, Austria Card/ICT, 2003
[Esser02]	Esser, Friedrich : „Java 2 : Java Essentials für Fortgeschrittene, Patterns und Idiome“, Galileo Press, Bonn, 2002
[Fla97]	Flanagan, D.: “Java in a Nutshell”, O’Reilly & Associates, Sebastopol, CA, 1997
[Hall01]	Hall, M.: “Servlets and Java Server Pages”, Markt + Technik, München, 2001
[Hall02]	Hall, M.: „Core Servlets und JavaServer Pages“, Markt + Technik, München, 2001
[Hall03]	Hall, M.: „More Servlets und JavaServer Pages“, Markt + Technik, München, 2002
[Haw02]	Hawlitzek, Florian: „Java 2“, Addison-Wesley, München, 2002

[HTTP10]	T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996
[HTTP11]	R. Fiedling et al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997
[MIDP10]	JSR 37 Expert Group: "Mobile Information Device Profile (JSR-37) JCP Specification Java 2 Platform, Micro Edition, 1.0a", Sun Microsystems, December 2000
[MIDP20]	JSR 118 Expert Group: "Mobile Information Device Profile for Java™ 2 Micro Edition Version 2.0", Sun Microsystems, Release: November 5, 2002
[Muchow01]	Muchow John W.: "Core J2ME technology & MIDP", Prentic Hall PTR, 2002
[NOTS03]	Gfader Peter: „Notification Server technische Dokumentation", Internes Projektdokument S3GIS, Austria Card/ICT, 2003
[Piro01]	Piroumian Vartan: „Wireless J2ME platform programming", Sun Microsystems Press, 2002
[Roß01]	Roßbach Peter: „Java-Server und Servlets", aktualisierte Aufl., Addison-Wesley, 2000
[S3SPEC03]	Wolfgang Burgstaller, Peter Gfader, Albert Treytl: „S3GIS-Projekt Systemspezifikation", Internes Projektdokument S3GIS, Austria Card/ICT, 2003
[S3SERV03]	Gfader Peter: „S3GIS-Projekt Allgemeine technische Serverdokumentation", Internes Projektdokument S3GIS, Austria Card/ICT, 2003
[Schwaiger01]	Schwaiger C.: "Projekt WASA Top-Level-Spezifikation", internal WASA Project-document, ICT TU Wien, 2001
[Schöberl01]	Schöberl C.: Diplomarbeit, „Secure Mobile Payment using Public Key Infrastructures", ICT TU Wien 2003
[SSL]	A. O. Freier, P. Karlton, Paul C. Kocher, "The SSL Protocol -- Version 3.0", draft-ietf-tls-ssl-version3-00.txt, November 18, 1996
[TCP]	Postel, J., "Transmission Control Protocol," STD 7, RFC 793, September 1981

[Tur01]	Turau V.: "Java Server Pages und J2EE.", dpunkt.verlag, Heidelberg, 2001
[Wobst01]	Wobst Reinhard: „Abenteuer Kryptologie", Addison-Wesley, 2001
[WTLS]	Wireless Application Forum. „WAP WTLS - Wireless Application, Protocol Wireless Transport Layer Security Specification", February 1999
[XMLKOM]	Gfader Peter, Burgstaller Wolfgang: S3GIS-Projekt, „XML-Schnittstellen zwischen den S3GIS-Serverkomponenten", Internes Projektdokument S3GIS, Austria Card/ICT, 2003

Literaturverzeichnis von Internetquellen.

Quellen sind auf beigelegter CD zu finden.

[ANT03]	Apache Ant: Java-based build tool. `http://ant.apache.org`
[BC03]	Bouncy Castle Crypto APIs `http://www.bouncycastle.org`
[CDCWP]	"Connected Device Configuration and the Foundation Profile", White Paper, Sun Microsystems Inc., 2001 `http://java.sun.com/products/cdc/wp/CDCwp.pdf`
[CLDCSTAT]	The CLDC HotSpot™ Implementation Virtual Machine, Sun Microsystems Inc., May 2003 http://java.sun.com/products/cldc/wp/CLDC_HotSpot_WhitePaper.pdf

[DEVLST]	MicroDevNet devices list
	`http://www.microjava.com/devices`
	SUN Wireless device list
	`http://wireless.java.sun.com/device/`
	MIDP 2.0 Phones and PDAs
	`http://www.benhui.net/midp2phonelist.html`

[DHTML]	Dynamisches HTML `http://selfhtml.teamone.de/dhtml/`

[EJBSpec]	"Enterprise Java Beans Specification Version2.0", Sun Microsystems Inc. http://java.sun.com/products/ejb/docs.html
[ERST01]	http://www.embarcadero.com/products/erstudio/index.asp
[FMK01]	Forum Mobile Kommunikation HTML Version: http://www.fmk.at/mobilkom/detail.cfm?Textid=41&Kapitelnr=5 PDF Version: http://www.fmk.at/mobilkom/dl/FMK_4-6.pdf
[HTML01]	W3C: Hypertext Markup Language (HTML) Home Page http://www.w3.org/MarkUp/
[IDEF01]	Federal Information Processing Standards Publication 184, "Integration Definition for Information Modelling (IDEF1X)", 1993 December 21 http://www.idef.com/Downloads/pdf/Idef1x.pdf
[JCARD01]	Java Card Plattform http://java.sun.com/products/javacard/
[JCP2PD]	JCP Procedures: JCP 2 Process Document http://jcp.org/en/procedures/jcp2
[J2EE01]	"Java 2 Standard Edition Overview", Sun Microsystems Inc. http://java.sun.com/j2ee/overview.html
[J2EEPr]	Davis, R.: "Introduction to Java and J2EE" http://ca.sun.com/en/events/presentations/devdays/docs/DD_Roland_final.pdf
[J2ME01]	Java 2 Micro Edition, Sun Microsystems Inc. http://java.sun.com/j2me/
[J2SE01]	"Java 2 Standard Edition Datasheet", Sun Microsystems Inc. http://java.sun.com/j2se/1.4/datasheet.1_4.html
[J2ME01DC]	J2ME Documentation Documents http://java.sun.com/j2me/docs/
[JCARD01]	JavaCard Plattform http://java.sun.com/products/javacard/

[JCARD01SP]	Java Card 2.2.1 Specification, Public Review Draft `http://java.sun.com/products/javacard/JavaCard221.html`
[JSec]	Sun Java Security, Sun Microsystems Inc. `http://java.sun.com/j2se/1.4.1/docs/guide/security/index.html`
[JSR124V03]	Danny Coward, Dave Bowen: J2EE Client Provisioning Specification Version 1.0 Proposed Final Draft `http://jcp.org/jsr/detail/124.jsp`
[JSR124RI]	J2EE Client Provisioning Specification Reference Implementation `http://developer.java.sun.com/developer/earlyAccess/j2ee_cp/index.html`
[JSR177]	Java Community Process, JSR 177; `http://www.jcp.org/en/jsr/detail?id=177`
[JSSE]	"Java Secure Socket Extension Reference Guide", Sun Microsystems Inc. `http://java.sun.com/j2se/1.4.1/docs/guide/security/jsse/JSSERefGuide.html`
[KVM]	"White Paper on KVM and the Connected, Limited Device Configuration (CLDC)", Sun Microsystems, May 19, 2000 `http://java.sun.com/products/cldc/wp/KVMwp.pdf`
[MAKE]	GNU Make `http://www.gnu.org/software/make/`
[MIDPW]	Mobile Information Device Profile (MIDP) `http://java.sun.com/products/midp/`
[PNGF01]	Portable Network Graphics, PNG Specification, Version 1.2 `http://www.libpng.org/pub/png/pngdocs.html`
[RAND01]	Class SecureRandom `http://java.sun.com/j2se/1.4.2/docs/api/java/security/SecureRandom.html`
[RFC1738]	Berner-Lee, T., Masinter, L., and M. McCahill: "Uniform Resource Locators (URL)", RFC 1738, December 1994 `http://www.ietf.org/rfc/rfc1738.txt`
[RFC2045]	Freed & Borenstein: „Multipurpose Internet Mail Extensions (MIME) Part One Format of Internet Message Bodies", November

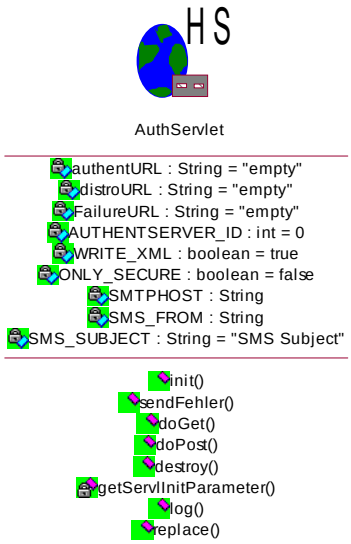
	1996 http://www.ietf.org/rfc/rfc2045.txt
[RFC2510]	Myers, M., et al.: "Internet X.509 Public Key Infrastructure Certificate Management Protocols.", June 1999 http://www.ietf.org/rfc/rfc2510.txt
[RFC2617]	Franks, et al.: "HTTP Authentication: Basic and Digest Access Authentication", June 1999 http://www.ietf.org/rfc/rfc2617.txt
[S3INST03]	Gfader Peter: „S3GIS-Projekt Download und Installationshinweise" s3gis_server_download_install.txt
[SERVSP01]	Java Servlet Specification 2.3 Final Release http://www.jcp.org/aboutJava/communityprocess/final/jsr053
[SL45i]	Siemens SL45i, J2ME-fähiges Endgerät http://www.my-siemens.com/MySiemens/CDA/Index/0,2730,HQ_en_0_product%253AMW%252FHD%252FHD%252FSL45I%252Fdesc,FF.html
[SSLSICH]	Sicherung von Diensten mittels SSL, Funktionsweise von SSL http://wwwbs.informatik.htw-dresden.de/svortrag/ai98/Poetzsch/funktion_ssl.html
[VMSpec]	The JavaTM Virtual Machine Specification http://java.sun.com/docs/books/vmspec/
[Walker01]	Simon Walker: "Basic Over the Air Server Configuration White Paper", March 3 2003 Basic_OTA_Server_Configuration.pdf
[WAP]	The Wireless Application Protocol http://www.wapforum.org
[WTK20]	Wireless Toolkit Version 2.0_01 - Production Release http://java.sun.com/products/j2mewtoolkit/download-2_0.html
[XMLW]	Extensible Markup Language (XML) http://www.w3.org/XML/
[ZIPF]	ZIP-Datei Format http://www.winzip.de/aboutzip.htm

C Abbildungsverzeichnis

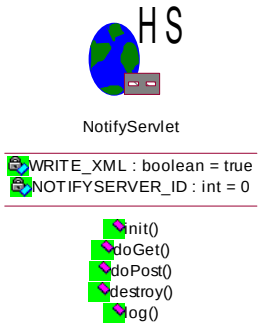
D Tabellenverzeichnis

E UML-Diagramme

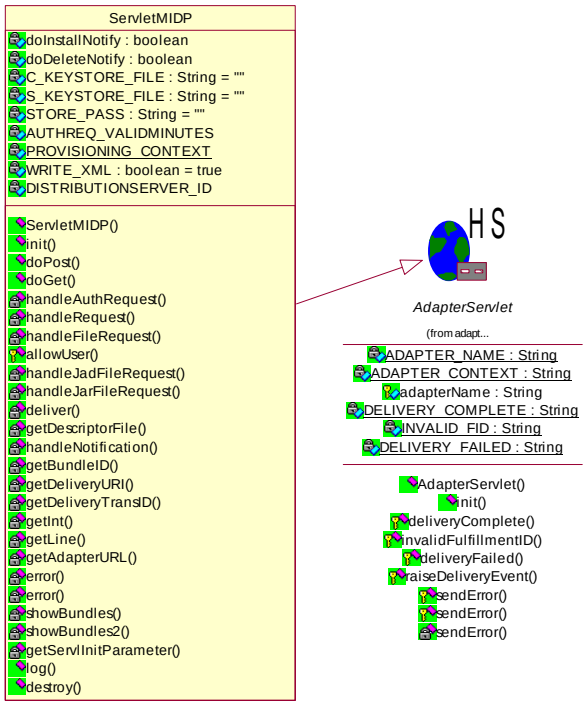
Authentication Server (Servlet erweitert HttpServlet Klasse HS)



Notification Server(Servlet erweitert HttpServlet Klasse HS)

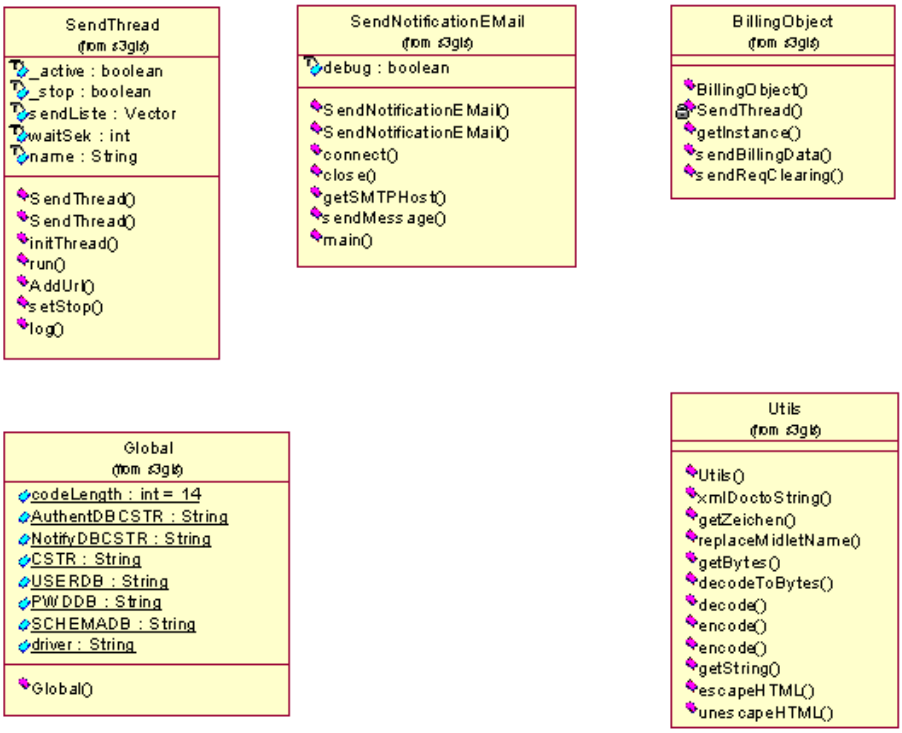


Distribution Server



S3GIS Package

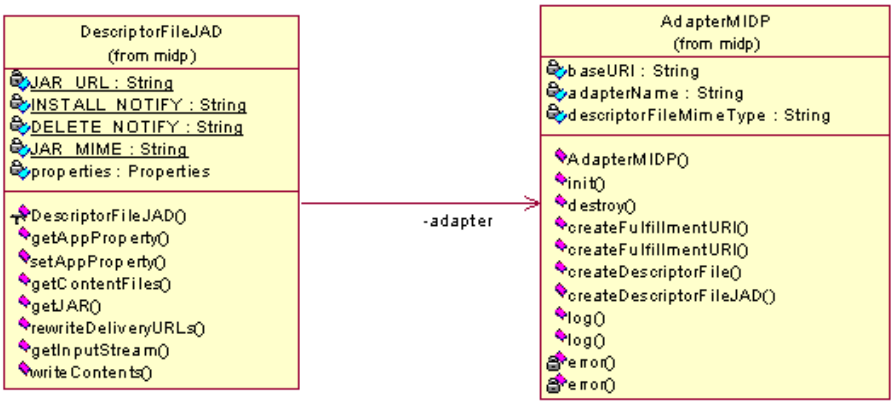
`Package: at.ac.tuwien.ict.s3gis`

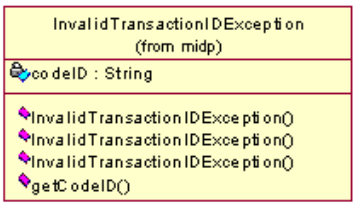


Package: at.ac.tuwien.ict.s3gis.database


TransactionTaskDB
(from database)
writeSQL : boolean
TransactionTaskDB()
createNewTrans()
getTransactionAllInfo()
getTransactionAllInfo()
getTransaction()
getIfT()
setExpired()
setExpired()
setValid()
enableTrans()
updateTransUser()
enableTrans()
disableTrans()
disableTrans()
incCountJad()
incCountJAR()
incPassFailed()
getUserPass()
getID()
getNextId()
MessageSmsCustomersObject
(from database)
XMLROOTNAME : String
customers : Vector
message : String
code : int
xmlFormat : String
sender : String
receiver : String
MessageSmsCustomersObject()
MessageSmsCustomersObject()
getDocument()
getDocumentAllowPush()
sendToURL()
sendRequest()
TransactionObject
(from database)
ID : int
enabled : int
randomD : String
fft : String
userID : String
pass : String
createTime : Timestamp
expireTime : Timestamp
expired : int
countJAD : int
countJar : int
countPassFailed : int
maxPassFailed : int
URLohneCode : String
bundleID : String
TransactionObject()

Package: at.ac.tuwien.ict.s3gis.adapters.midp

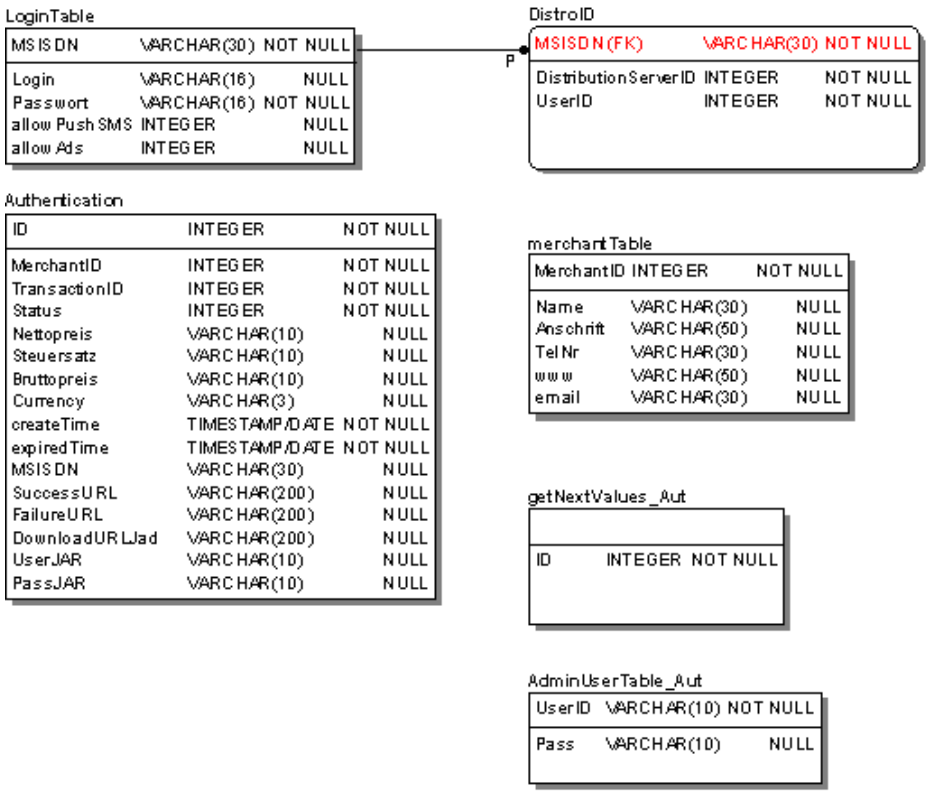




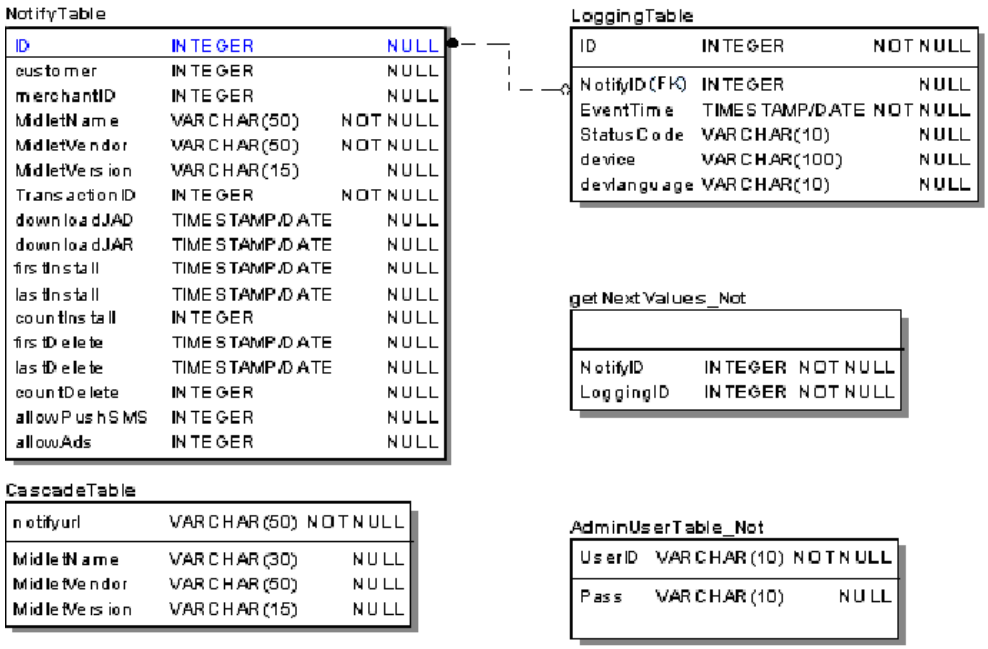
F EER-Diagramme

Alle Diagramme wurden mit Embarcadero ER/Studio [ERST01] erstellt und sind in IDEF1X Notation [IDEF01].

Authentication Server Datenbank. Details siehe [AUTHS03]



Notification Server Datenbank. Details siehe [NOTS03]



Distribution Server Datenbank. Details siehe [DISTS03]

TransactionTable

ID	INTEGER	NOT NULL
enabled	INTEGER	NOT NULL
randomID	VARCHAR(12)	NOT NULL
customer	INTEGER	NULL
fft	VARCHAR(50)	NULL
userID	VARCHAR(6)	NULL
pass	VARCHAR(6)	NULL
createTime	TIMESTAMP/DATE	NOT NULL
expireTime	TIMESTAMP/DATE	NULL
expired	INTEGER	NULL
countJad	INTEGER	NULL
countJar	INTEGER	NULL
countPassFailed	INTEGER	NULL
maxPassFailed	INTEGER	NULL
URLohneCode	VARCHAR(255)	NULL
BundleID	VARCHAR(15)	NULL

getNextValues_Dist

transID	INTEGER	NOT NULL

AdminUserTable_Dist

UserID	VARCHAR(10)	NOT NULL
Pass	VARCHAR(10)	NULL