# AOTA-Advanced Over The Air

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

**Information Technology**

By

Udit Pilania (151470)

Under the supervision of

Mr. Vikash Kumar

To


Department of Computer Science & Engineering and Information Technology

# TABLE OF CONTENTS


| SNO. | Page No. |
|---|---|
| Student Declaration | iii |
| Certificate | iv |
| Acknowledgement | v |
| Summary | vi |
| List of Tables and Figures | vii |
| **1. Introduction** | 1-3 |
| 1.1 General Introduction | |
| 1.2 Relevant current/open problems | |
| 1.3 Problem Statement | |
| **2. Literature Survey** | 4-18 |
| **3. Analysis, Design and Modeling** | 19-26 |
| 3.1 Overall description of the project | |
| 3.2 Functional requirements | |
| 3.3 Non Functional requirements | |
| 3.4 Logical database requirements | |
| 3.5 Design Diagrams | |
| 3.5.1Use Case diagrams | |
| 3.5.2 Class diagrams / Control Flow Diagrams | |
| 3.5.3 Sequence Diagram/Activity diagrams | |

**4. Implementation details and issues** 27-35

4.1 Implementation details

4.2 Implementation Issues

4.4 Risk Analysis and Mitigation

**5. Testing** 36-37

5.1 Testing Plan

5.2 Component decomposition and type of testing required

5.3 List all test cases in prescribed format

**6. Findings and Conclusion** 38

6.1 Findings

6.2 Conclusion

**7. Appendix** 39

A. References in IEEE style

## DECLARATION

I hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Noida

Name: Udit Pilania

Enrolment No: 151470

# CERTIFICATE

This is to certify that the project entitled, “AOTA" submitted by "Udit Pilania" in partial fulfillment of the requirements for the award of " Bachelor of Technology" in "Information Technology" at the "Jaypee University of Information Technology" is an authentic work carried out by him under my supervision and guidance.

Mr.Vikash Kumar

(Technical Lead, Thales)

## ACKNOWLEDGEMENT

Perseverance, inspiration and motivation have played a great role in the success of any venture. It would be incomplete to submit this report without acknowledging the people behind this endeavor and without this report without acknowledging the people behind this endeavor and without whose support I wouldn't have able to achieve this. It gives us immense pleasure to express our gratitude to everyone who shared with me their precious time and effort during the project. A mammoth job like this calls for both intellectual nourishment and encouragement. I would like to thank Mr. Vikash Kumar( Technical Lead, Thales) for giving me an opportunity to work on such a fabulous project.

Signatures of Students:
Udit Pilania(151470)

## SUMMARY

The Advanced Over The Air (AOTA) enables telecom operators to manage 2G, 3G and LTE operations of push and pull modes to the cards (UICC). It is a solution which allows upgrading from the 2G/3G to 4G solution and manage cards remotely on the field. The operations that can be performed on the cards are remote application management (RAM) and remote file management (RFM) through push and pull mode. The push mode sends APDU commands immediately to the card which are triggered by OTA and the pull mode is triggered by the card, where automatic request is sent from the card to the server on regular intervals for remote management of the card. Push mode refers to SMS bearer and Pull mode refers to HTTP bearer.

OTA Manager Migration Tool is a standalone Java application which is used to migrate data from earlier versions of OTA Manager (versions 4.x and 5.x) database to OTA Manager V6.5 database. This tool is based on a generic architecture which supports data migration for different pairs of source and target data systems. It is basically provided so that the data can be transferred from OTA to AOTA for all customers.

RUM is the processing module of AOTA which carries out a lot of major functionalities of AOTA including the generation of APDU commands for UICC management. This is the principle objective of RUM and because it manages a lot of functions for remotely managing the cards, it is the central processing engine of AOTA. It is also used for other functions like in activating a card, updating file or application in card and send a force polling SMS.

# LIST OF TABLES AND FIGURES


| Fig/Table no. | Contents | Page no. |
|---|---|---|
| Fig 2.1 | UICC Provisioning | 4 |
| Fig 2.2 | Life Cycle of Subscription | 5 |
| Fig 2.3 | Mass Management | 5 |
| Fig 2.4 | Apdu analyze | 14 |
| Fig 2.5 | SRM2SRM Migration Tool Architecture | 16 |
| Fig 2.6 | Execution of a Migration Plan | 17 |
| Fig 3.3.1 | Installation and Integrity requirement | 22 |
| Fig 3.5.1 | Subscription management use case diagram | 24 |
| Fig 3.5.2 | RUM Service Activation use case diagram | 25 |
| Fig 3.5.3 | Provision OTA diagram | 25 |
| Fig 3.5.4 | Activation diagram | 26 |
| Fig 3.5.5 | Life cycle diagram | 26 |
| Fig 4.1.1 | Architecture Diagram | 27 |
| Fig 4.1.2 | Push mode diagram | 28 |
| Fig 4.1.2 | Pull mode diagram | 29 |
| Fig 4.1.4 | RUM APDU functionality | 31 |
| Table 4.3.1 | Risk Analysis and Mitigation plan | 35 |
| Table 5.1.1 | CMM Traffic using MySQL | 37 |
| Table 5.1.2 | MMGT traffic using MySQL | 38 |
| Table 5.2.1 | Type of tests | 39 |

Table 5.3.1 Test cases 39-40

# INTRODUCTION

## General Introduction

A smart card contains many applications and files which are needed to be managed by the mobile network operators. Advanced OTA (Over the Air) is a solution to manage UICC remotely and in a short term other Secure Elements such as Embedded Secure Element or Secure Memory Card. UICC (Universal Integrated Circuit Card) is a telecom dedicated secure element (previously called smart card). This secure element is capable of executing security critical functions and contains authentication secrets for connecting to GSM/LTE networks. AOTA is first a solution based on RoH (Remote management over HTTP) based on Pull mode. The operations that can be performed on the cards are remote application management (RAM) and remote file management (RFM) through push and pull mode. The push mode sends APDU commands immediately to the card which are triggered by OTA and the pull mode is triggered by the card, where automatic request is sent from the card to the server on regular intervals for remote management of the card. Earlier, there was only OTA and now with the development of AOTA, all the telecom customers also need to migrate all of their data from OTA to AOTA which brings in the use of migration tool. The migration tool provides the application which enables the transfer of all the data of customer from OTA to AOTA. This data is huge and takes a lot of time to migrate, thus the performance and accuracy both are handled in this tool efficiently. AOTA has many components but the central component is RUM. It manages a lot of functionalities for card management such as provisioning of the card, activation and deactivation, active retry and network detection. RUM can also generate APDU commands for executing all the functions which may also include sending any update through different services. It also communicates with almost all of the components of AOTA. The MIOS services are added to provide the changing of IMSI. These services have the capability to change the IMSI when required. The Multi IMSI Applet is a UICC-based application that manages a set of two to ten IMSIs for the subscriber. There are three services added while the other were previously available.

## Relevant current/open problems

Remote card management is a necessity for all mobile network operators. The smart card contains many applications and files that need to be activated or updated at certain times, as needed. For that, the solution provided was OTA (Over The Air) which had the push mode. This means that all orders could be sent to the card via SMS support, which also facilitates the immediate sending of orders and the execution of the corresponding task. But, there may be times when the card is not present in the network and would not receive the SMS sent by the server. This SMS would expire before the card again comes back in the network. In order to provide a better solution, AOTA (Advanced Over The Air) was provided which also had the pull mode. This pull mode would be using the http bearer and it allows the card to request the server for commands and remotely manage it. This way, it would be insured that the card was in network when command was sent from the server. Now the telecom operators had all their huge amount of data in OTA which needs to be migrated to AOTA, and thus the migration tool provides this data transformation. The RUM being the main component of AOTA, communicates with many components and has many services which does the task of updating and other functions. These functions are improved by adding the new services to provide a better solution.

## Problem Statement

The problem of remotely managing cards in order to send remote application management and remote file management commands is solved by the AOTA solution. It makes it possible to go from the 2G / 3G solution to the 4G solution and to manage the remote cards in the field. It has push mode as well as pull mode. The solution provided by the OTA had only the push mode which had the disadvantage of the expiry of the SMS if the card was off-network. This is solved using the AOTA Pull mode. The migration tool is required to migrate all data from the OTA Manager database (versions 4.x and 5.x) to the OTA Manager V6.5 database. It includes the application that provides migration and scripts to help it. This is a need for the clients because the data is very large and, in order to use the AOTA, the migration to its database is necessary. Among many components of AOTA, RUM is the central processing component and helps in the remote management of the card through many functions. It is necessary to help communicate with many components and provide the services to update files or applications. RUM also generates APDU commands that are sent to the card for any service.

## Literature Survey (Including Products & Technologies)

UICC and non-UICC preloading The UICC element must be provisioned before performing any operation. This pre-provisioning consists of providing information about map profile attachments, 03.48 security keys, and IMSI and ICCID information about 2G / 3G cards. For 4G cards, pre-provisioning is the same as for 2G / 3G cards, with the exception of TLS-PSK security information (the UICC card is on the operator's premises but has not been sold , modified or used). Subsequently, an operator (backend) will assign a MSISDN to the existingsubscription to activate it, platform side.

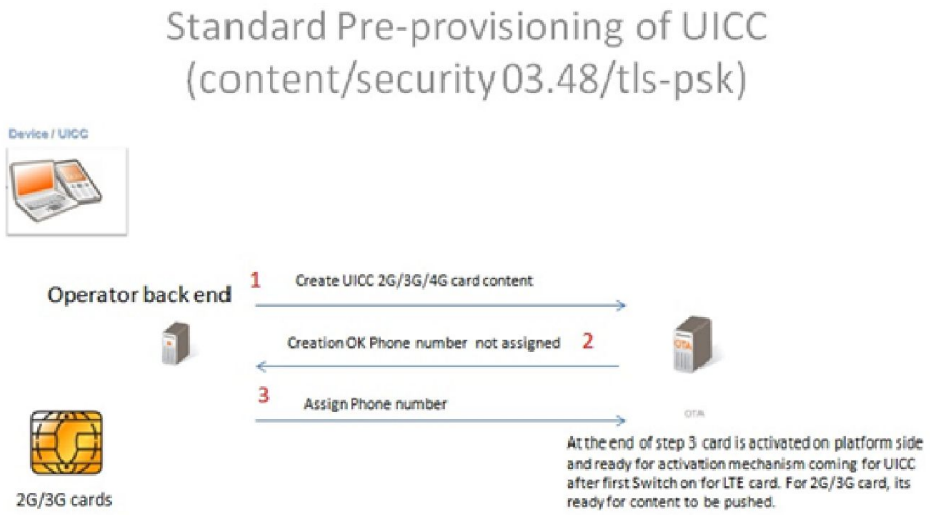



Fig 2.1 UICC Provisioning

Subscription Life Cycle (Device change)

A subscription can have various states during its life cycle. Card can be lost/damaged. In such cases, a UICC replacement is required and run pre-provisioning and activation at switch on.

Another Use case can happen if the mobile equipment is damaged. In that case, end user will probably change it. Advanced OTA Solution V6.5.2 needs to be informed of the ME change to reflect it in the subscription. This use case has a prerequisite which is to have DDE applet installed and activated in the UICC whatever the protocol used will be (sms or http)

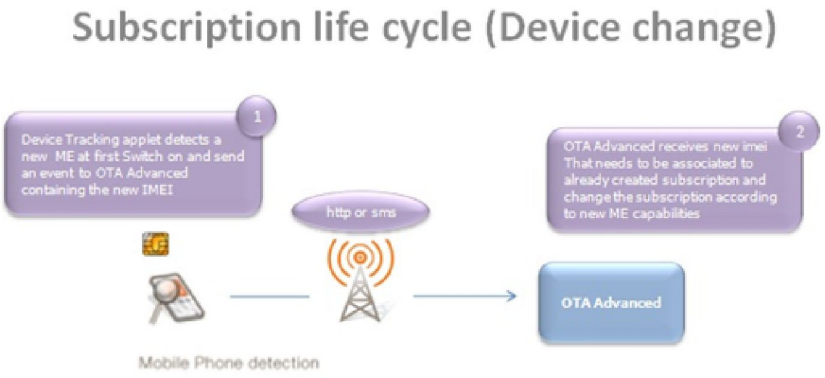



Fig 2.2 Life Cycle of Subscription

Mass Management in Polling Mode

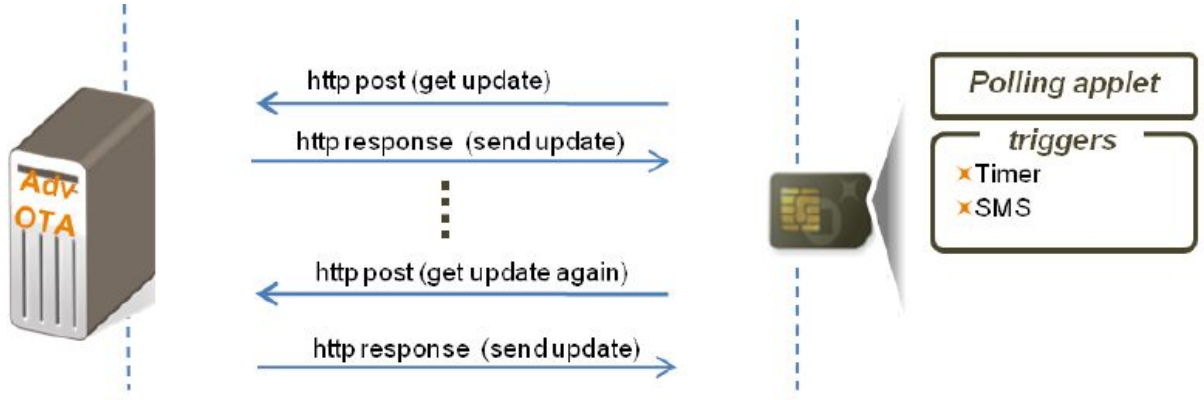



Fig 2.3 Mass Management

1. The UICC, according to a schedule timer or an SMS push, connects and gets an update from the AOTA server by sending an HTTP post.
2. The AOTA server responds by sending APDU commands according to the request.

3. A session can have several exchanges from several products, so the card sends another request and the server responds.
4. In mass management, multiple campaigns can be handled in parallel. An MMGT campaign is capable of sending all types of RAM and RFM updates, filter different targets by Card profile, Handset profile, or subscriber profile, and do various monitoring activities with features such as progress rates, estimated end dates, and graphs of http polling requests received.

Single Campaign Management in Push Mode (via SMS)
In Advanced OTA Solution V6.5.2, administrator builds a scenario based on RAM/RFM services and loads a target MSISDN file in the system. After defining the campaign duration, planning and validity period, the campaign met the required criteria, then the campaign starts. Subsequently, all SMS MT messages are sent to the end user for the update.

1. Define the scenario and the target file for the campaign.
2. Create the campaign and tune the related parameters such as campaign duration and validity period.
3. Perform an update of the pending RFM and RAM operations for the cards belonging to the segment.
4. Device receives the update and will notifies the ESME that MT is correctly received.
5. Monitor the percentage of subscribers in which the campaign has been applied.

Last Polling Date Campaign Management in Push Mode: For users who wants to select all the UICC LTE cards that are not polling after a defined date.
1 From Operation workspace, Pull Campaign widget, Manage Target Groups page, create a group.
2 Create the scenario in Campaign management and define the period of time in which the campaign is valid.
3 Receive the update.
4 Monitor the percentage of subscribers in which the template has been applied.

Device Detection: Device Detection Management is used to monitor the presence of the device and tracks the changes in the smartcard parameters. For example, smart card parameters such as
IMEISV/IMEI/MEID, Terminal Profile and additional smart card information.

What is HTTP Protocol?

The HyperText Transfer Protocol (HTTP) is a communication protocol used between a client and a server. In the OTA world, the UICC card is a client, and the Advanced OTA is a server. It is always the client that sends requests to a server and always the server that responds.
HTTP request is composed of a header where the command type and the address of the server are defined, and the data in the message body.

A status code of the request is returned to the client in the message body.
When the client sends a request to the server:

- The request is defined in a message composed of a header and a body.
- The header starts with a method (command) which defines a requested action such as the access to a resource
- The targeted server is identified by a URI (or URL) available in the header.

Depending on the command used, Parameters/data may be contained in the body.
When the targeted server returns a response with:
The header contains the status code (for example, Server reached, or not).
The body contains a response.
There are seven types of HTTP requests (or methods) available:

- GET: Requests the specified resource (without a message body).
- PUT: Uploads (add or replace) a resource (with a message body) on the server.
- POST: Submits data (with a message body) to be processed to the identified resource.
- DELETE: Removes the specified resource (with a message body) on the server.
- HEAD: Requests meta-information on a resource (without a message body).

- OPTION: Returns the HTTP methods supported by specific URI (for debugging purposes).
- TRACE: Echoes back the received requests (for debugging purposes).

Note: The last three methods are typically used by developers.

There are five groups of HTTP response status codes that can be returned by AOTA server: Informational message, Command executed successfully, Redirection, Client error (such as the “404 error! Page requested doesn't exist”), Server error. Card retry can only be performed on the status errors.

Why Does AOTA Use HTTP?

- The HTTP protocol is chosen to manage the cards on LTE networks mainly because:
- The Internet is widely available.
- Cost reduction from reusing existing tools and knowledge.
- HTTP is easy to be deployed to existing operator networks.
- HTTP has field-proven protocols and architecture (Internet protocols) to achieve performance, reliability, and scalability.

Main Principles:

The following lists the conditions for the successful operation of OTA over HTTP:

- The UICC card is an HTTP client.
- The AOTA server is an HTTP server.
- The protocol with a remote server is based on HTTP POST.
- Connection is at the initiative of the UICC card.
- The general dialog mechanism is based on the following: The UICC card requests the OTA server for the next commands (next URI) to be executed.
- HTTP POST messages are sent by the Admin Agent to the AOTA server to request for commands to be executed by the Admin Agent. The same HTTP POST request is also used to return the results of the command executed.
- The Content-Type field specifies the type of message to indicate the RFM/RAM commands or responses that are transported.
- The body contains the data associated to the type of message. The content of the body is dependent on the Content-Type of the communication. The APDU and POR are include in the body of the message.

- The Next-URI is provided by AOTA to Agent in HTTP POST response if additional commands have to be executed.

Next URI Mechanism: The Next-URI mechanism allows the use of POST method only. The following shows the process flow of how the connection is established between the server and the card. As the card is the client, it is responsible to open the connection. It can be triggered by an internal event from an application such as polling or by an external event such as an SMS.

**1** The card opens the connection.

**2** Once the connection has been established, an HTTP request is sent to the server with the address of the server and the name of the application to target the corresponding use case.

**3** The server responds with a status code and a result in the body including the APDU commands.

**4** The APDU commands are executed by the card and the result is sent to the server. The result (for example, the POR) is in the body.

**5** If some commands still have to be executed, a new response is sent.

**6** The server indicates that there are no more commands to execute on the card when the status code is 204 and there is no Next URI.

OTA over HTTPS: In order to secure communication between the AOTA server and the UICC card, the cryptographic protocol is used—Transport Layer Security Pre-Shared Key(TLS/PSK).

Note: Only the message body is encrypted.

- Encryption and decryption techniques are used for secure transfer.
- The security keys are generated corresponding to each card using algorithms.
- It is custom for the card.

This means that the secret key exchanged between the card and the server is unique for each connection.

Retry Mechanism

The Admin Agent is responsible for the connection to the AOTA server and for the accomplishment of the session.

If a communication error occurs during the processing of the administration flow, the Admin Agent will try to reconnect according to a card issuer specific retry policy:

The Admin Agent makes several attempts for resuming the administration session. A waiting period between two attempts and the maximum number of attempt is specified by the retry policy. If the communication is re-established, the Admin Agent tries to resume the HTTP dialog by navigating the last URI of this administration session and setting a Resume HTTP header tag.

If the maximum number of attempts has been reached, the administration session request will be abandoned.

If the TLS session failed to establish the connection for security/authorization reason, the administration session will be immediately discarded.

Mass Management Pull Mode: AOTA is the industry's first platform to enable the pull mode for campaign updates whereby a fleet of UICCs are updated at once to reflect subscription changes. For example, to refresh roaming preferences.

In other words, the UICC in the mobile device also initiates the connection for an update automatically, based on a preplanned schedule. This delivers much more efficient update campaigns, another important benefit to operators.

The pull approach optimizes the success rate by reaching all active subscribers virtually. It reduces the burden on OTA server and eliminates the need to create a large campaign team.

It concentrates server resources on connected devices, unlike the push mode which often attempts to initiate the update to mobile devices that are unconnected which results in the waste of server resources and causing many useless retries.

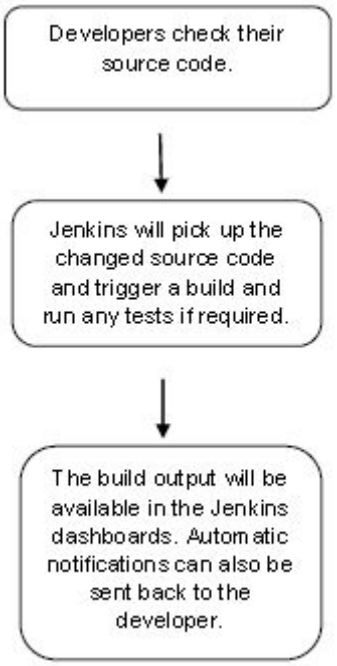
In addition, AOTA relies on a connected mode that significantly reduces the number of problems caused when a mobile device loses its network connection in the middle of an update, and recovers seamlessly when it does happen. The features in this solution are designed for optimum efficiency for a better MNO experience across all networks.

Campaign Management Push Mode: AOTA relies on its proven campaign managers to perform push campaigns on cards that are not using pull mode for some reasons.
AOTA facilities the combination of push and pull approaches in order for mass management of dynamic updates on cards.

Unified Campaign Management: In addition of the two campaign engines described in the previous sections, Advanced OTA Solution V6.5.2 Mass Manager allows the combining of push and pull modes in a single campaign to manage a heterogeneous set of cards using multiple bearers (SMS and HTTP) taking advantage of pull mode as often as possible.
This conciliation introduces the mechanism of automatic bearer selection simplifying operations in terms of monitoring and reporting.
WebLogic is a server software application that runs on a middle tier, between back-end databases and related applications and browser-based thin clients. WebLogic is a leading e-commerce online transaction processing (OLTP) platform, developed to connect users in a distributed computing environment and to facilitate the integration of mainframe applications with distributed corporate data and applications.
WebLogic server is based on Java 2 Platform, Enterprise Edition (J2EE), the standard platform used to create Java-based multi-tier enterprise applications. J2EE platform technologies were developed through the efforts of BEA Systems and other vendors in collaboration with the main developer, Sun Microsystems. Because J2EE applications are standardized modules, WebLogic can automate many system-level tasks that would otherwise have demanded programming time.

The main features of WebLogic server include connectors that make it possible for any legacy application on any client to interoperate with server applications, Enterprise JavaBean (EJB) components, resource pooling, and connection sharing that make applications very scalable. An administration console with a user interface makes management tasks more efficient and features such as Secure Sockets Layer (SSL) support for the encryption of data transmissions, as well as authentication and authorization mechanisms, make applications and transactions secure.

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.




Continuous integration is a development practice that requires developers to embed code in a shared repository at regular intervals. This concept was intended to eliminate the problem of subsequent detection of problems in the generation lifecycle. Continuous integration requires developers to frequently use versions. The common practice is that every time code validation takes place, a construct must be triggered. Continuous deployment is a software engineering approach in which teams produce software in short cycles, ensuring reliable software release at all times. Its goal is to create, test and publish software with a higher speed and frequency. This approach helps reduce the costs, time, and risks of making changes by allowing incremental updates to production applications. A simple and repeatable deployment process is important for continuous delivery.

SCI – Is the framework for AOTA. AOTA groups operational SCI pages and CCI related in the “Operation” workspace.

This workspace contains several pages which exist in the default SCI framework:

Advanced Campaigns is used for HTTP campaigns which allow you to operate a

Classic Campaigns is used for SMS campaigns which allow you to operate an SMS campaign.

Template Management is used for RUM Template Management which allows you to create template for pull campaign usage, and also to send single HTTP request from there.

Unitary Push Management is used for SMS Push requests which allows you to send single SMS request to card.

Provisioning Management is used for cards and security related provisioning which allows you to create cards in batches via SSO/CCI.

Subscriber Repository is used for data reference (such as MNO, model and brand) management and subscriber viewing which allows you to view devices belong to the particular subscriber and operations made.

UICC-SE Repository is used for card instance viewing, card profile viewing and creation. It includes both the CRM and CCI card management parts as well.

Applet Repository is used for applet management which allows you to view and create applets.

OTA Service Repository is used for 2G/SG services definition and management which allows you to create/remove 2G/3G services.

Record History Service is used for HTTP operation History recording in which you can find the detailed information (such as start/end time, ending status and service executed) of HTTP operation. Device Tracking is used for Device Detection Manager which records the DM event such as changing mobile of subscriptions.

RUM Template Manager is an SCI-based application used by administrators to easily manage the RUM templates. Depending on the access rights, the user can create, delete, or update templates. A template contains a set of predetermined parameters to be applied remotely to the SIM card files by using the selected RUM service.

The template defined by the Template Manager is used in MMGT scenario.

Services: To create a template, from the Operation workspace, select Template Management page, followed by RUM-Template Manager. The Template Manager Widget is displayed.

The purpose of RUM Services are to produce APDUs for Sim Cards

One RUM Service is responsible for managing one entity (file or application) in the card.

- A RUM Service performs two main tasks.

- Produce APDUS requests for cards
- Process response APDUs received from cards

In code level, RUM service is a Java class that client can execute

- Customer can provide parameters for the service
- Generated APDU changes depending on the provided parameters
- Response from the card is also processed by the same service
- Service may perform modifications to the card file image on the AOTA database so it is in line with the file on the card.

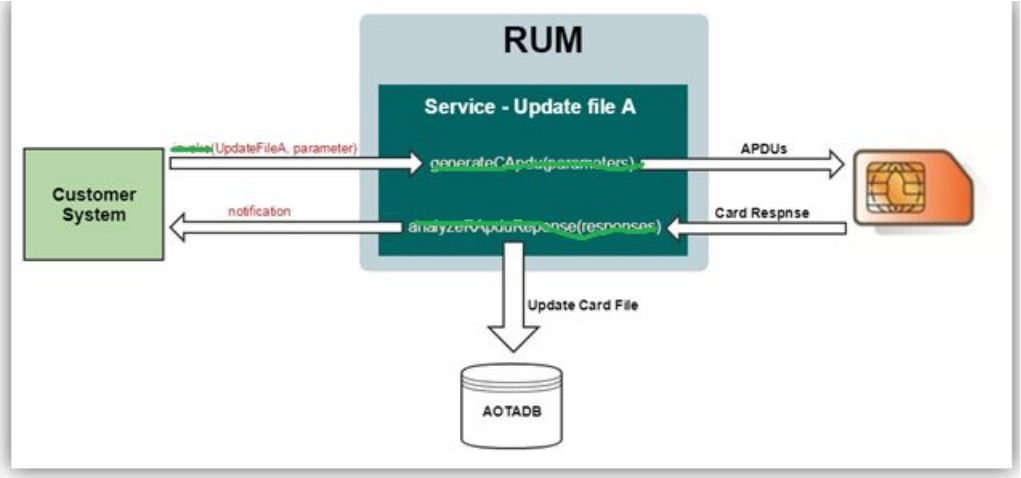



Fig 2.4 Apdu analyze

RFM & RAM Management: A service manages File or Applet on the card by sending APDU.

RFM = Remote File Management: Management of files under SIM, USIM, ISIM and CSIM applications. Based on ETSI, 3GPP standards

RAM = Remote Applet Management. Management of Load File and Application/Security Domain Based on Global Platform standard.

A service can also manages card content on server side (image of content of the card).All services can be invoked SOAP & J2EE APIs Several kinds of services

Unitary: service parameters shall be repeated at each invocation

Template: service parameters are stored and the service invocation just refers to the template defining them.

Template is used to apply same updates for a huge list of targets)

- Or both
- Several implementations of services
- Basic (simple update) RAM, RFM without capacity to support communication lost.
- High-level (scenario) A high-level service splits execution into basic services and adapts the execution depending on the SE response.
- High-level services are able to manage REENTRANCE (communication lost).

RUM supports a lots of services for RFM

- For all interpreters SIM, USIM, ISIM, CSIM
- Some 'SIM' services,can be targeted through USIM
- One update service manages only 1 file (true for 90% of services).
- RAM services can manage Load File and Application:
- Some generic services (createApplication, LockApplication…).
- Some services for applet configuration.
- Some services for applet life cycle (enable, disable).
- Some specific services for dedicated applet (Polling, DDE, …)

SRM2SRM Migration Tool is a standalone Java application which is used to migrate data from earlier versions of Subscriber Repository Manager database to
Subscriber Repository Manager database. This tool is based on a generic architecture which supports data migration for different pairs of source and target data systems.
The following are the two major components of SRM2SRM Migration Tool:

- Migration Framework

This component provides the core functionalities of the migration process. This is independent from the data system being migrated.

- Migration Plug-in:This is the pluggable component which is specific to each source and target of the system being migrated. It sits on top of the Migration Framework and handles the reading, transformation, and writing tasks for the migration process.

The following figure shows the high-level architecture of SRM2SRM Migration Tool.

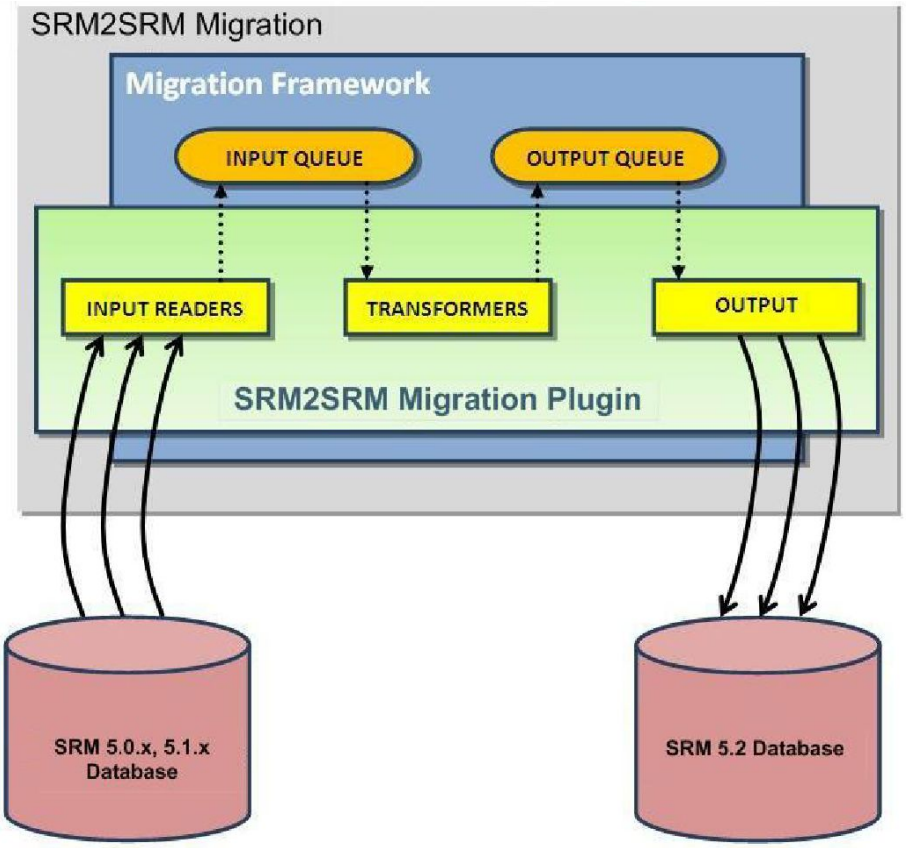



Fig 2.5 SRM2SRM Migration Tool Architecture

SRM2SRM Migration Tool executes the migration process as defined in a migration plan. A migration plan is composed of orders and steps.

An order is a collection of migration process definitions (known as steps) and it is independent from the other orders. SRM2SRM Migration Tool executes all the orders defined in the migration plan in parallel, hence, orders must be mutually exclusive and there should be no conflicts.

A step is a sequence of migration processes which comprise an order. A step is dedicated to migrate one data object model (which may include all its references) from the source system to a new data object model on the target system. Steps within the order are executed sequentially one after another.

The migration plan for earlier Subscriber Repository Manager versions to Subscriber Repository Manager has only one order for each migration phase of which is mapped to the schema that is to be migrated. The following table lists the orders along with the corresponding schemas in the source and target databases

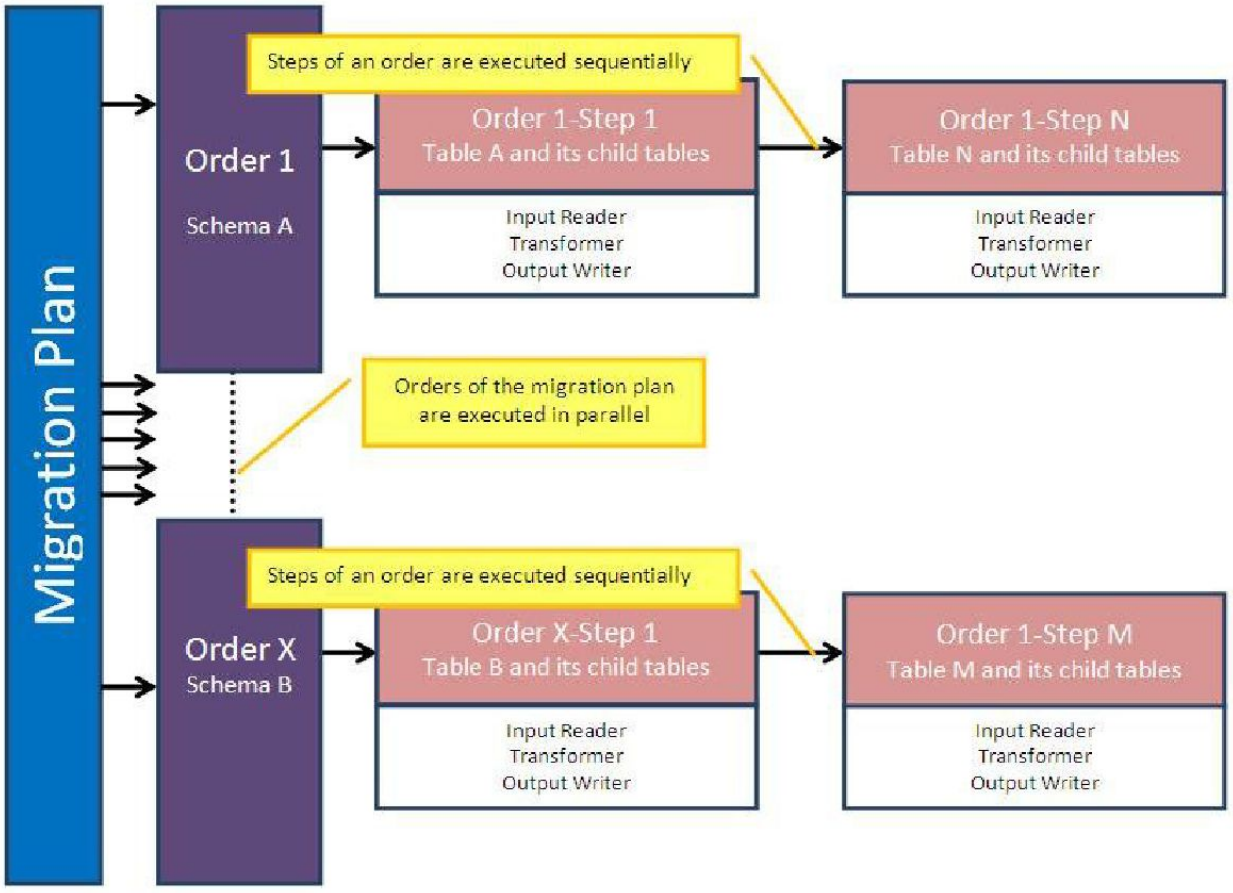



Fig 2.6 Execution of a Migration Plan

Delta data refers to the parallel changes of the data records in the source solution during migration. In the migration context, these are referred as “dirty” data because the migrated data in the target system is not the latest. For each parent table (corresponding to a step in the migration plan), the tool creates a database trigger and a Delta table. The trigger records any updates or deletions of the parent table records in the Delta table. During the execution of a

step, once all rows in the parent table has been processed (that is, no rows with GUMI_STATUS
= UNPROCESSED), the tool processes the Delta table for that particular step.
To process each delta record, the following tasks are performed:
1 Search the GUMI table with record having the status to be MIGRATED and find them in the target database based on its unique constraint value.
2 Perform update on all the record's attribute in the target table.
3 Reset the migration status to UNPROCESSED.
4 Read the parent table to migrate any remaining UNPROCESSED records. Any new insertions to the parent table will also be picked up.
5 Delta processing continues until there are no more UNPROCESSED delta records.

The Multi IMSI Applet is a UICC-based application that manages a set of two to ten IMSIs for the subscriber.
This application can be installed using a specific file system or internal buffers to store its configuration parameters.
Once the application is deployed, it is necessary to provide the possibility to fully administrate the application.

The Home IMSI is used in the subscriber’s home country and during roaming to any country where the operator has a bilateral agreement in place. The International IMSI is used when abroad for connection to a roaming hub. The first two IMSIs are mandatory, in addition, the operator can also configure up to nine International IMSIs that will be used when the subscriber travels to a specific country, provided the operator has a partnership with the local operator in that country.

# ANALYSIS, DESIGN AND MODELING

## Overall description of the Project

Advanced OTA Solution is part of a fully-integrated software solution providing you with a robust, reliable and scalable infrastructure to manage your (U)SIM installed base.

**AOTA Solution** allows you to:

- Deliver new applications directly to your subscribers.
- Easily adapt your service portal for diverse subscriber types.
- Control your subscribers' mobile environment.

AOTA is Gemalto's latest generation of over-the-air platform which enables mobile network operators to manage 2G, 3G and LTE benefits of push and pull modes to the cards. Pull mode is an important innovation introduced by AOTA in which the UICC initiates the dialogue.

The main characteristics of the AOTA platform are:

- Reliability and excellent performance with a proven technology of more than 15 years.
- Capability to offer the most appropriate bearer (SMS, CATTP and HTTP) for each use case.

Each card is dependent on its environment (type of card, device capabilities and subscription characteristics).

The AOTA platform is also involved in the instant activation of subscriptions at the first switch-on using HTTP channel.

The architecture contains the various machines and components installed on each machine either in a WebLogic or Veritas Cluster (shown in grey dotted background and hatched background respectively) that are required in the AOTA Solution. The following figure shows the global architecture of the solution in a typical deployment and its dependencies to one another.

OTA Manager Migration Tool executes the migration process as defined in a migration plan.

The AOTA and OTA vary so much in the architecture that we need a separate tool to automate the entire migration process, this is where the **OTA Manager Migration Tool** comes into play. OTAM tool is a standalone Java application which is used to migrate data from earlier versions of OTA Manager (versions 4.x and 5.x) database to OTA Manager V6.3 database. This tool is based on a generic architecture which supports data migration for different pairs of source and target data systems.

The following are the two major components of OTA Manager Migration Tool:

- Migration Framework: This component provides the core functionalities of the migration process. This is independent from the data system being migrated.
- Migration Plug-in: This is the pluggable component which is specific to each source and target of the system being migrated. It sits on top of the Migration Framework and handles the reading, transformation, and writing tasks for the migration process.

A migration plan is composed of orders and steps.

- An order is a collection of migration process definitions (known as steps) and it is independent from the other orders. OTA Manager Migration Tool executes all the orders defined in the migration plan in parallel, hence, orders must be mutually exclusive and there should be no conflicts.

- A step is a sequence of migration processes which comprise an order. A step is dedicated to migrate one data object model (which may include all its references) from the source system to a new data object model on the target system. Steps within the order are executed sequentially one after another.

Two important terms with data migration are the Gumi table and Delta Data. The Gumi tables are the status tables on the source and destination, which has the ending status of the migration of each parent table in the migration plan and Delta data refers to the parallel changes of the data records in the source solution during migration. In the migration context, these are referred as "dirty" data because the migrated data in the target system is not the latest. For each parent table which referenced by the GUMI table (corresponding to a step in the migration plan), the tool creates a database trigger and a Delta table.

The processing module of AOTA is **RUM** and it is third component that I got to work on, in Gemalto. Also there are several other services which are provided by RUM.

- Subscription provisioning in AOTA
- Activate card for the first time
- Handle Device Change of a Card (SIM card switching mobile phone)
- Update file / application in a card
- Send a force polling SMS (request an immediate polling request from the card)

## Functional requirements

Functional requirements for AOTA (since AOTA V6.1).

- Common database repository for all cards (2G/3G/4G) (since AOTA V6.1).
- Introduction of the new Unified Mass Manager (since AOTA V6.2).
- Introduction of new SMS features—Subscription Activation, Push Profile, and Active Retry (since AOTA V6.3).
- Integration of Roaming Module with Advanced OTA Solution V6.5.2.1

The following are the new RUM API introduced:

- Get Subscription Information With Interpreted Content.
- Get Detailed History of Services.
- Get Detailed Processing Services.

## Non Functional requirements

1. Software/System Architecture: The application software architecture shall be realized in a way that enables user to do their maintenance jobs, trouble shooting etc. in an efficient way.
2. Virtual Environment: Moving the virtual machine on the fly to a different hardware. Moving the used disk space of a virtual machine on the fly to another data store. Automatic distribution of virtual machines on different hosts for load optimization. The following demands must not be used by applications.
   - Special hardware requirements and interfaces (e.g. X25 and so on…)
   - Hardware Dongles
   - External storage media in continuous access (e.g. USB-Stick, CD-ROM, Discs,…..)
   - Access to physical interfaces
   - Direct access to PCI Bus
   - Disk Mirroring
   - Firm connection to special blades
3. Installation And Integration Requirement:

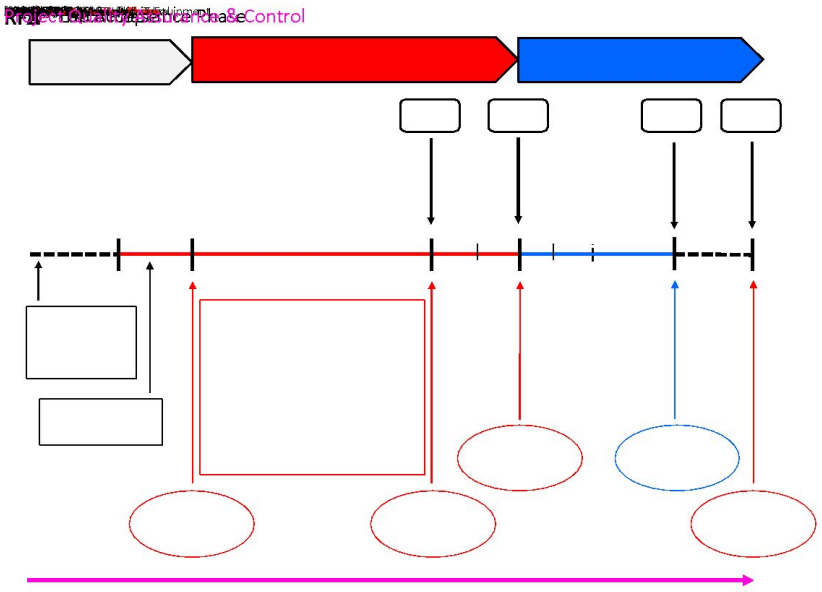


Fig 3.3.1 Installation and Integrity requirement

4. Service Monitoring:
    - Performance Measurement Interval
    - PM Data Alarming
    - Specific Counter for KPI Measuring
    - Explanation of PM Data & Counters
    - Service Affecting Alarms
    - Alarm Auto Clearing

5. Operation Requirements:
    - Access
    - Full Access for system administrators
    - Remote access
    - Access via encrypted protocols only
    - File transfer
    - Graphical user interfaces and command line interfaces.
    - Creation and administration of users

6. Usability:
    - Operational System must be linux or unix
    - Standard OS behaviour
    - Availability of Tools

8. Configuration:

The operating company might have the need to change certain parameters.

## Logical database requirements

- MySQL Version 6.3
- Oracle Version 11g

The database for all the schemas should be added in the workbench according to different dbs and SID for different machines. The data is automatically added to these schemas on running the automated installation of db script. The connection should be proper and all of the JPA code should be able to insert and retrieve from these tables. The data could be compatible according to the kind of db installed in the machine, i.e with MySQL or Oracle.

## Design Diagrams

**Use Case Diagram**

Subscription Management:

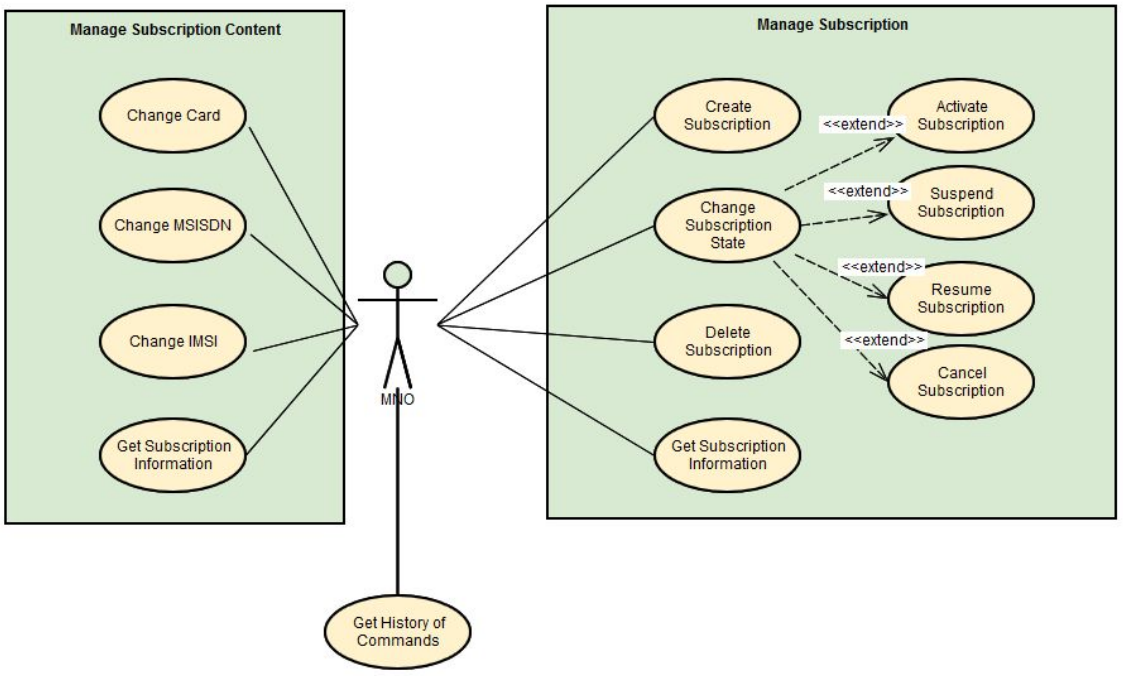



Fig 3.5.1 Subscription management use case diagram

RUM Service Activation:

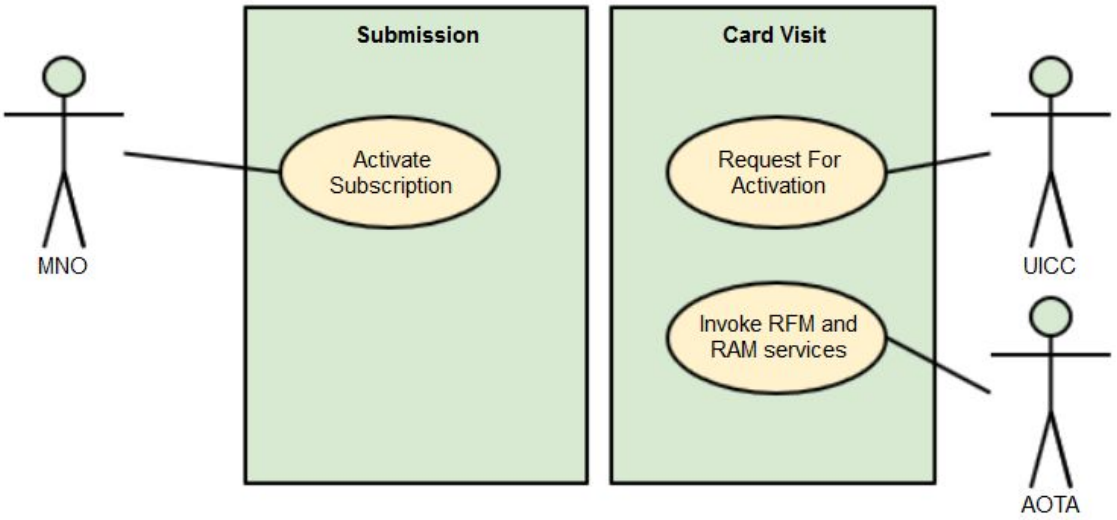



Fig 3.5.2 RUM Service Activation use case diagram

Control Flow Diagram:

First Step: Provision OTA

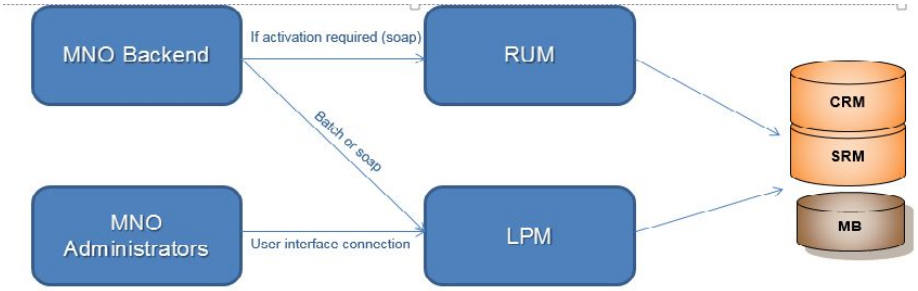



Fig 3.5.3 Provision OTA diagram

Second Step: Activation

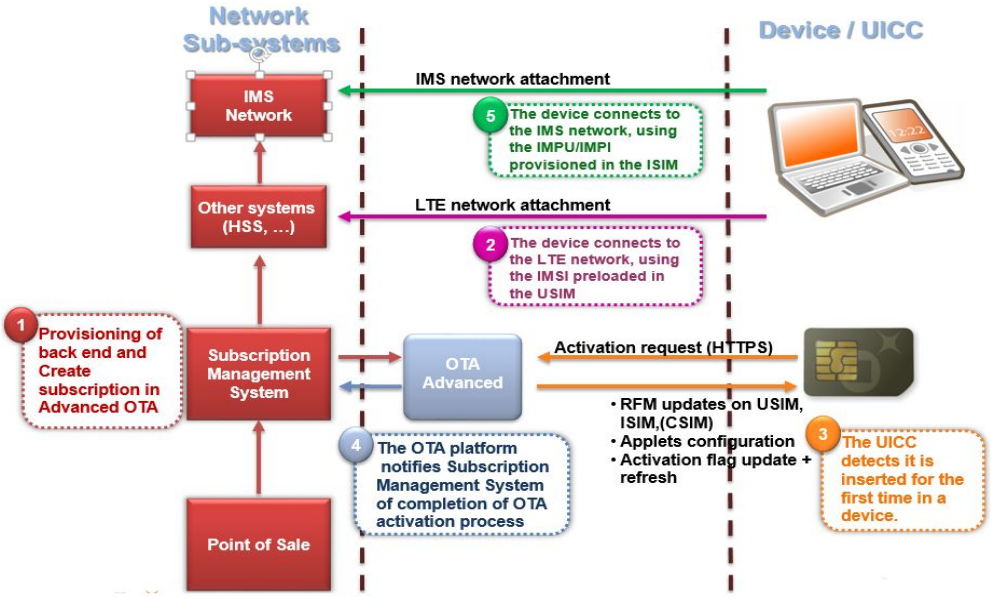


Fig 3.5.4 Activation diagram

Life-Cycle of RUM Services:

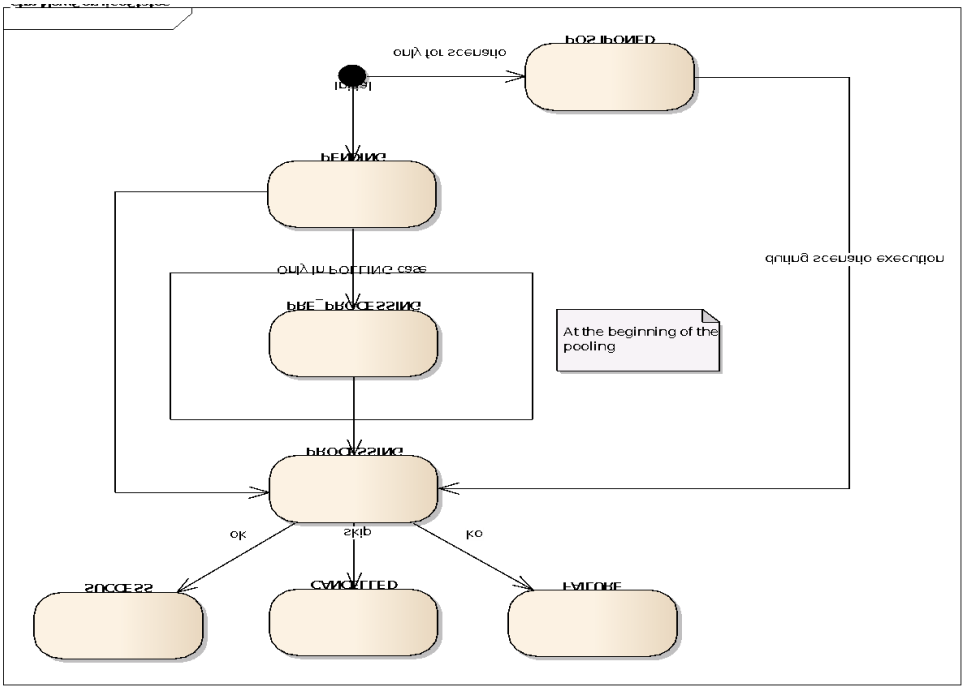



Fig 3.5.5 Life cycle diagram

# IMPLEMENTATION DETAILS AND ISSUES

## Implementation details

The Advanced OTA (AOTA) enables mobile network operators to manage 2G, 3G and LTE operations of push and pull modes to the cards (UICC). The operations that can be performed on the cards are remote application management (RAM) and remote file management (RFM). Push mode refers to SMS bearer and Pull mode refers to HTTP bearer.

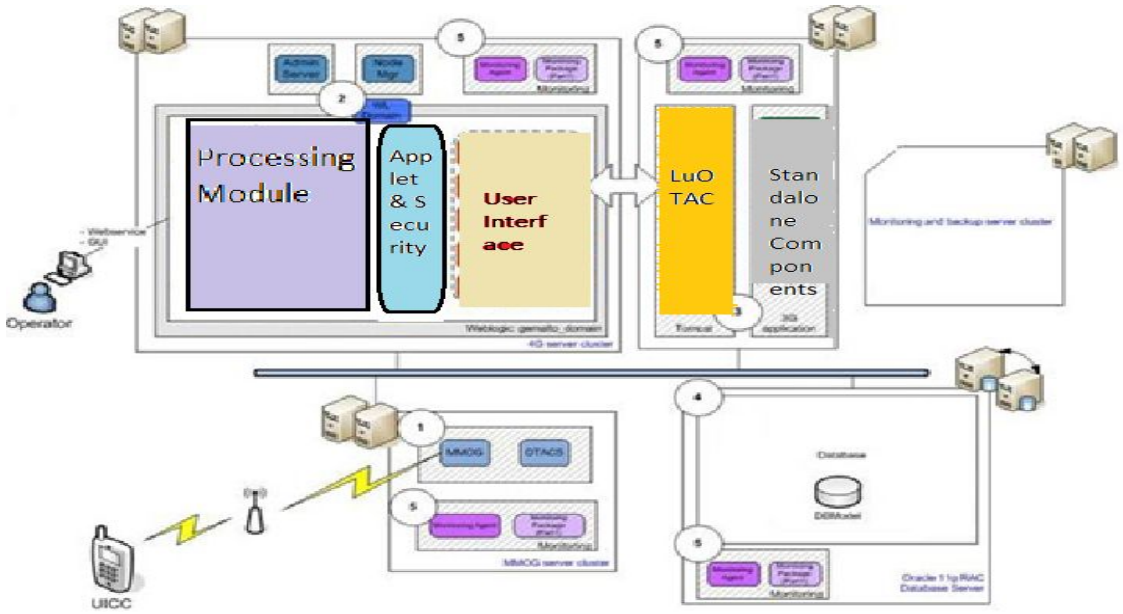



Fig 4.1.1 Architecture Diagram

For pull mode, the UICC initiate HTTP(s) request to AOTA to check if there are any operations to be executed on the UICC. If there are operations to be executed on the UICC, APDU commands would be returned as part of the HTTP(s) response. The HTTP(s) request can be for activation, polling use cases.

For Polling:

- The client submits an invocation to AOTA by invoking an AOTA API to register the services to be executed on the UICC. AOTA may wait for the next HTTP polling request from UICC or send a force polling SMS to UICC to trigger an immediate polling request.
- UICC performs a HTTP polling request to AOTA. Since services are registered, APDU commands would be returned as part of HTTP response to UICC.
  UICC performs the execution of the APDU commands and send another HTTP request to AOTA with the APDU responses of the executed commands.

- If there are more commands to be executed, AOTA would returned the set of commands to be executed as part of the HTTP response to UICC.

  This process continues till there are no more commands to be executed on UICC.
- The client would be notified with the results of the invocation.


Fig 4.1.2 Push mode diagram

For push mode, AOTA initiate the execution by sending SMS messages containing the APDU commands to be executed on the UICC. Classic OTAM is used to perform the operations if the bearer is SMS.

- The client submits an invocation to AOTA by invoking an AOTA API.
- The invocation immediately triggers the execution of the service to generate the command packets (CP) containing a list of APDU commands.

  The CPs are split into SMS messages accordingly if required, and then they are sent to UICC.
- The APDU responses of the executed commands are returned as a PoR to AOTA.
- The client would be notified with the results of the invocation.

Fig 4.1.3 Pull mode diagram

UICC needs to be provisioned first before performing any operations. This pre-provisioning consists of providing card profile attachment information. For 4G cards, pre provisioning is the same as 2G/3G cards except for security information.

A subscription can have various states during its life cycle. Card can be lost/damaged. In such cases, a UICC replacement is required and run pre-provisioning and activation at switch on.
Another Use case can happen if the mobile equipment is damaged. In that case, end user will probably change it. Advanced OTA Solution V6.5.2 needs to be informed of the ME change to reflect it in the subscription. This use case has a prerequisite which is to have DDE applet installed and activated in the UICC whatever the protocol used will be (sms or http).

In mass management, multiple campaigns can be handled in parallel. An MMGT campaign is capable of sending all types of RAM and RFM updates, filter different targets by Card profile, Handset profile, or subscriber profile, and do various monitoring activities with features such as progress rates, estimated end dates, and graphs of http polling requests received.

In Advanced OTA Solution V6.5.2, administrator builds a scenario based on RAM/RFM services and loads a target MSISDN file in the system. After defining the campaign duration, planning and validity period, the campaign met the required criteria, then the campaign starts. Subsequently, all SMS MT messages are sent to the end user for the update.

Device Detection Management is used to monitor the presence of the device and tracks the changes in the smartcard parameters.
OTA Manager Migration Tool is a standalone Java application which is used to migrate data from earlier versions of OTA Manager (versions 4.x and 5.x) database to OTA Manager V6.5 database. This tool is based on a generic architecture which supports data migration for different pairs of source and target data systems.
The following are the two major components of OTA Manager Migration Tool:

- Migration Framework

This component provides the core functionalities of the migration process. This is independent from the data system being migrated.

- Migration Plug-in

This is the pluggable component which is specific to each source and target of the system being migrated. It sits on top of the Migration Framework and handles the reading, transformation, and writing tasks for the migration process.

The Migration tool has reader, transformer and writer as a high level architecture. The reader reads the data from the database and passes it on to the transformer. The transformer then transforms the data, with or without the use of file system. Then the writer would write the transformed data to the target database.

OTA Manager Migration Tool executes the migration process as defined in a migration plan.

A migration plan is composed of orders and steps.

- An order is a collection of migration process definitions (known as steps) and it is independent from the other orders. OTA Manager Migration Tool executes all the orders defined in the migration plan in parallel, hence, orders must be mutually exclusive and there should be no conflicts.
- A step is a sequence of migration processes which comprise an order. A step is dedicated to migrate one data object model (which may include all its references) from the source system to a new data object model on the target system. Steps within the order are executed sequentially one after another.

Delta data refers to the parallel changes of the data records in the source solution during migration. In the migration context, these are referred as “dirty” data because the migrated data in the target system is not the latest. For each parent table, the tool creates a database trigger and a Delta table. The trigger records any updates or deletions of the parent table records in the Delta table. During the execution of a step, once all rows in the GUMI table has been processed, the tool processes the Delta table for that particular step.

To process each delta record, the following tasks are preformed:

**1** Delete the dirty record from the target database.

**2** Reset the migration status to UNPROCESSED.

**3** Read the GUMI table to migrate any remaining UNPROCESSED records. Any new insertions to the parent table will also be picked up.

**4** Delta processing continues until there are no more UNPROCESSED delta records.

RUM is the processing modules of AOTA and carries out lots of major functionalities of AOTA.

The goal of AOTA is to perform file or application updates on the card. This update commands are sent in the form of APDUs.

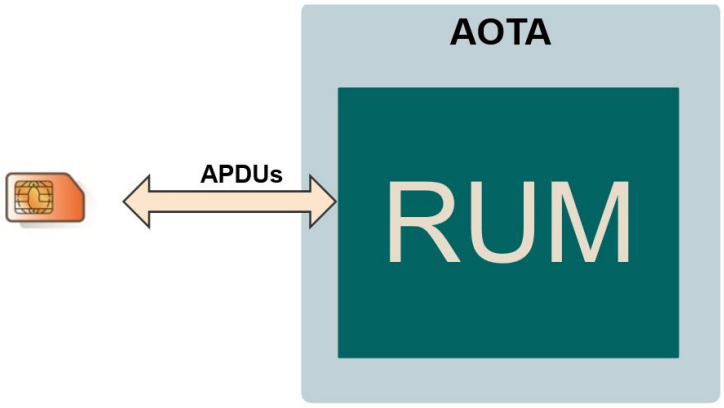



Fig 4.1.4 RUM APDU functionality

There are 4 main step in a successful service execution:

1. Create a Subscription in AOTA
2. Activate it on both AOTA side and on the real card
3. Provision some services to be executed in AOTA
4. Card Visit - Polling workflow (HTTP exchanges)

Service is basically an entity which generates APDUs for SIM Cards. Services can be simply categorized into two groups.

1. RFM services (Remote File Management)
2. RAM Services (Remote Application Management)

Basic Service is also referred as a simple updates, a service which performs update on RAM, RFM without capacity to support communication loss.

High-level Service is also referred as scenario.

A high-level service splits execution into several Basic Services and adapts the execution depending on the SE response.

High-level services are able to manage REENTRANCE (communication loss).

Templates are preconfigured services which can be created from the GUI (new SCI). Generally, C-APDU generation is done at creation time or first usage.

Only Service Templates can be targeted by the Mass Manager.

RUM Service Templates are linked to CRM content templates which can be Card File Template or Application configuration Template

Templates provides import/export methods which could be CSV, binary. There are three major step that shall be done to create a new RUM service.

1. Creation of the service / template service
2. Add the service configuration to the database
3. Creation of the RUM widget to allow creation of the template for the service

To create the new service as a template service, some extra work shall be done. The service implementation class (stateless EJB) needs to have an EJB remote interface which extends the interface. All required methods defined in the interface needs to be implemented. In RUM Template Management in SCI, it is necessary to allow the user to create the template containing the service data for a new service. This feature is not available for all the services and shall depend on the customer requirement.

The MIOS services were added to provide new functionality. MIOS stands for Multi Imsi OTA Services. These services have the capability to change the IMSI when required. The Multi IMSI Applet is a UICC-based application that manages a set of two to ten IMSIs for the subscriber.

The Home IMSI is used in the subscriber's home country and during roaming to any country where the operator has a bilateral agreement in place. The International IMSI is used when abroad for connection to a roaming hub. The first two IMSIs are mandatory, in addition, the operator can also configure up to nine International IMSIs that will be used when the subscriber travels to a specific country, provided the operator has a partnership with the local operator in that country.

This application can be installed using a specific file system or internal buffers to store its configuration parameters.

Once the application is deployed, it is necessary to provide the possibility to fully administrate the application.

The Multi IMSI OTA Services is offering deployment choice between 2 set of OTA services dedicated to manage Multi IMSI Applet:

- MIOS Multi-Bearer services (SMS & HTTP)
- MIOS SMS services based on legacy AOTA (SMS only)

**Multi Imsi Forced Switch Imsi service**

This is a RAM Multi service (template and unitary), done as "simple service" (which means only a simple generator), with support to HTTP and SMS, available from SOAP and Java API and it can be used to trigger the applet to perform the Automatic IMSI Switching – Triggered on Status Event process or Manual IMSI Selection depending on the type of Forced Switch Command sent. There are three types of Forced Switch command:

- Forced Switch Command – Automatic
- Forced Switch Command – Manual
- Forced Switch Command – Retain

As a template, this service shall support an import/export mechanism with csv format.
To do that, it is necessary to create the translator associated to this service.

**Multi Imsi Set Default Imsi service**

This is a RAM Multi service (template and unitary), done as "simple service" (which means only a simple generator), with support to HTTP and SMS, available from SOAP and Java API and it can be used to trigger the appl7et to set the default IMSI.

As a template, this service shall support an import/export mechanism with csv format.
To do that, it is necessary to create the translator associated to this service.

**Multi Imsi Get applet's configuration parameters service**

This is a RAM Unitary service only, done as "simple service" (which means only a simple generator), with support to HTTP and SMS, available from SOAP and Java API. This service does not have a UI and is a unitary service. The template cannot be created as it does not support the template part through the UI.

## Implementation issues

The issues faced during the implementation were related to the migration tool and the RUM services that were added:

- Logging was to be inserted
- PL/SQL script to be created for correcting an issue of duplicate entries.
- Some scripts to be made compatible with new OpenDJ 3.
- Combined duplicate scripts and made required changes to script files.
- Made the files for online and offline mode compatible to different versions of jars.
- The apdu to be sent was to be corrected.
- The test cases which were supposed to check the restriction, needed to have try catch to succeed and show the expected.

- While creating a new service, xsd were not correctly updated.

## Risk Analysis and Mitigation Plan

| Id | Description | Risk Area | Probability | Impact | RE(P*I) | Risk Selected for Mitigation | Mitigation Plan |
|---|---|---|---|---|---|---|---|
| 1 | An API is not compatible with lower python versions | Version | 0.1 | (0.9+0.4+0.2)/3= 0.5 | 0.1*0.5 = 0.05 | Y | Use the alternative API provided. |
| 2 | A script may hang during the deletion process if it is done after migration of a certain part. | Data base | 0.2 | (0.9+0.7+0.2)/3 = 0.6 | 0.2*0.6 = 0.12 | Y | Manually delete from db or use the other part provided instead of the API. |
| 3 | Does not handle the updated and deleted devices if it was already migrated before. | Algorithm | 0.3 | (0.8+0.5+0.2)/3=0.5 | 0.3*0.5= 0.15 | Y | Quick fix |
| 4. | The new services will support new version of applet instance and not the first one. | Algorithm | 0.1 | (0.9+0.4+0.2)/3= 0.5 | 0.1*0.5 = 0.05 | N | Just support for new version. |

Table 4.3.1 Risk Analysis and Mitigation plan

# TESTING

## Type of testing required

Functional testing: Unit testing, JATS regression testing and end-to-end testing. Each service will have unit testing which is one of the earliest testing efforts performed on the code and the earlier defects are detected, the easier they are to fix and automated JATS regression testing to ensure that the packaging of the project, its execution and its impact on all other services is void .There should be no new regressions. The final component is the end-to-end testing where we use card issued by Gemalto and run the JATS to see, how they do in actual cards. Apart from the functional testing, the performance of the tools and the services also need to be tested.

| Type Of Test | Will Test be Performed? |
|---|---|
| **Requirements Testing** | **Yes** |
| **Unit** | **Yes** |
| **Performance** | **Yes** |
| **Stress** | **No** |
| **Compliance** | **Yes** |
| **Security** | **Yes** |
| **Load** | **Yes** |

Table 5.2.1 Type of tests

## Test cases in prescribed Format

For Meta Data Services in RUM , we test the cases when the SMS address is of odd length, if the user doesn't have a password, for SMS transport , with AID specific, max length of user name, password and network access name. In case of MIOS services,

| S.No. | List Of Various Components | Type of Testing Required | Technique for writing test Cases |
|---|---|---|---|
| 1. | RUM | • Requirement<br>• Unit<br>• Integration<br>• End-to-end<br>• System<br>• Performance Load<br>• SONAR* | Black Box Testing: expected values of APDUS in XML, the computed value is compared, to check if they are the same. The SCI is the presentation layer and needs to be validated. The business logic isolation and logic layer implementation needs to be checked. |
| 2. | AOTA Migration Tool | • Requirement<br>• Unit<br>• Integration<br>• System<br>• Performance Load | Validation testing: to check the flow of the code |

Table 5.3.1 Test cases

Using the Jenkins CD job, we run the regression testing on all the JATS. The expected error are compared with the actual outputs.

* **Sonar** is a web based code quality analysis **tool** for Maven based Java projects. It covers a wide area of code quality check points which include: Architecture & Design, Complexity, Duplications, Coding Rules, Potential Bugs, Unit Test etc.

# Findings and Conclusion

## Findings

AOTA provides an important solution to its customers by providing UICC processing, including all applications and files. The project has both push and pull modes, which eliminates the card's disadvantage of not receiving SMS when not in the server and also supports the immediate action that might be needed to switch to push mode. The migration tool is particularly important because it allows the transformation of the database from the previous version to the most recent one. Given the amount of data, performance is kept high to avoid production problems. The migration tool also handles delta data, which is the data that can be changed during the migration process itself. To do this, it deletes this type of data from the target database, resets the migration status, reads all unprocessed records, and migrates them again. This happens until there are no delta data left. RUM is the main component of AOTA and communicates with almost all other components. Its most important task is to generate an APDU via which an order can be sent to the card. It also has other features that it can provide through different services.

## Conclusion

The project itself is a great solution and has many components. The main feature is to manage the smart card remotely. The previous version of OTA only supported the extraction mode. Now the AOTA supports both modes, extraction and thrust, which helps to manage things effectively. It uses SMS support and HTTP support for these purposes. The migration tool would help turn data from an old database into a new one. Different schemas are defined and several scripts are used to perform the task. It covers all aspects such as changing data during migration and also gives importance to performance. The most important component, RUM, has many tasks and provides many features such as updating or deleting a file or applet. He has many services through which he can send orders. The APDU generation is the main part that contains exactly what kind of commands should be sent and where they should be sent and what needs to be done.

## References


[1] http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html

[2] https://en.wikipedia.org/wiki/Oracle_WebLogic_Server

[3] https://jenkins.io/doc/

[4] https://www.tutorialspoint.com/jenkins/jenkins_overview.htm

[5] https://en.wikipedia.org/wiki/Jenkins_(software)

[6] https://en.wikipedia.org/wiki/Spring_Framework

[7] https://www.tutorialspoint.com/maven/maven_overview.htm

[8] Internal Documents of Gemalto