# RESTful OTA Services

**Sarabjeet Singh**

* Research and Development, Syscom Corporation Limited

***Abstract***- Telecom Industry is based upon user satisfaction and user services. Mobile operators provide VAS (Value Added Services) to users to increase revenue and promote their services. Now a days, OTA (Over the Air) technology to update services on CARD without physically interacting CARDs. This requires implementation of different components of OTA technology to be able to perform such activities seamlessly. This paper discusses this implementation of OTA in terms of RESTful web services by removing the complex component implementation of OTA technology and creating maintainable, scalable, reusable modular services for different operations on a SIM card.

***Index Terms***- OTA, SIM, Telecom, VAS, REST

## I. Introduction

This article discusses how OTA services can be implemented using three components: JAVA, GSM Modem, SIM Card. A GSM Modem is JAVA technology allows us creation of programs and applications that allows interaction with GSM Modem.

A GSM Modem is a special hardware that accepts a SIM card and works with an operator subscription. A computer can be used to interact with a GSM modem over mobile network when in is connected via COM port of the computer. GSM Modem works with AT commands syntax. Different AT commands can be used to send SMS, make calls from GSM modem using computer system. For example, AT+CMGC command is used to Send SMS using GSM modem. AT^SMGR command can be used to read an SMS from inbox of the modem. As we can send a normal SMS using GSM modem, we can also use this modem to send OTA messages to the card. Over The Air technology is used to Manage Applications and/or data on a SIM card. This technology facilitates interaction with Card without physical interaction with the CARD. There are following required components to implement an OTA service:

Interface to send OTA requests. This interface will be used to send requests to OTA gateway for required service to be updated or replaced on the SIM card. An OTA Gateway is like an SMS gateway. It is required to understand process and convert request sent from OTA interface into a SIM understandable format and Send the request to Card. Third component required is a SMSC. SMSC or short message service center is required to manage SMS exchange between OTA gateway and SIM CARD. It facilitates a channel of interaction. Last component required for the service is Mobile Phone that contains a SIM card on which OTA request is to be sent. This whole interaction is displayed in form of a flow diagram ahead:

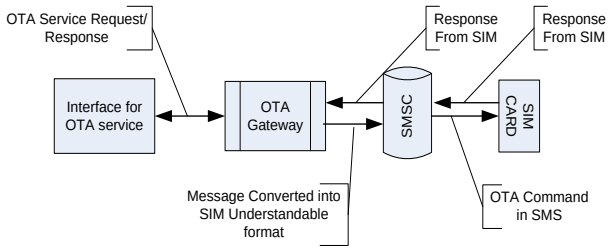



Figure 1: OTA Interaction with SIM card

**Research Elaboration** – Basic thought behind this paper is to provide an idea to replace this OTA Gateway explained in Figure 1 with a substitute that allows GSM operators to use cost effective methods to perform OTA updates on SIM Cards. GSM Modem and JAVA technology can be coupled together to create web based services that can be used to update cards for live OTA testing or performing OTA update on CARD in the fields. Here the control mechanism will be RESTful web services that will interact with a web based interface and GSM Modem for performing required actions.

Input format for the Web Service can either be an XML or JSON formatted file. JSON (JavaScript Object Notation) is easy to read and write and is easy for machines to parse and interpret. JSON is completely language independent. That gives advantage of creating requests in format of Keys and Attributes which can be parsed in JAVA or any other language like PHP easily. A simple JSON format looks like following example:

{“id”: “101”, “Name”: “JSON”, “Surname”: “Storm”}

This string contains three Keys, Id, Name and Sur name of a person. This file is easily readable, modifiable and transferable by the Web Services.

A Web interface can also be created in any language like PHP. PHP contains CURL Library that allows easy and quick creation of Web service consuming capabilities. Simple PHP code to consume a Send SMS RESTful web service takes only 7 lines of code:

```
<? php
  $url = 'http://localhost/Services/SendSMS/Hello';
    $ch = curl_init ();
    curl_setopt ($ch, CURLOPT_URL, $url);
    curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);
    $result = curl_exec ($ch);
    curl_close ($ch);
    print_r ($result);
?>
```

The Web Service is called from curl which in turn will instruct GSM Modem to Send SMS "Hello" to the required number. Destination address to send SMS can either be configured in the web service itself(For fixed cases) or can be provided from the Web interface with other fields like Validity period of the Message, SMSC to be used.

For OTA messages, Web service need to prepare SMS in format of GSM 3.48 standard. This standard specifies complete OTA SMS format. A 3.48 SMS format required following fields to be added to the SMS:

This table specified Packet elements to be added and length of the data in Octets

**Table 1: Request Packet Format**

| Element | Length | Comment |
|---|---|---|
| Command Packet Identifier (CPI) | 1 octet | Identifies that this data block is the secured Command Packet |
| Command Packet Length (CPL) | variable | This shall indicate the number of octets from and including the Command Header Identifier to the end of the Secured Data, including any padding octets required for ciphering. |
| Command Header Identifier (CHI) | 1 octet | Identifies the Command Header. |
| Command Header Length (CHL) | variable | This shall indicate the number of octets from and including the SP1 to the end of the RCICCIDS. |
| Security Parameter Indicator (SPI) | 2 octets | This describes security parameters in the Message Sent |
| Ciphering Key Identifier (Klc) | 1 octet | Key and algorithm Identifier for ciphering. |
| Key Identifier (KID) | 1 octet | Key and algorithm Identifier for RCICCIDS. |
| Toolkit Application Reference (TAR) | 3 octets | Coding is application dependent. |
| Counter (CNTR) | 5 octets | Replay detection and Sequence Integrity counter. |
| Padding counter (PCNTR) | 1 octet | This indicates the number of padding octets used for ciphering at the end of the secured data. |
| Redundancy Check (RC),Cryptographic Checksum (CC) or used, | variable | Length depends on the algorithm. A typical value is 8 octets if and for a DS could be 48 or more octets; the minimum Digital Signature (DS) should be 4 octets. |
| Secured Data | variable | Contains the Secured Application Message and possibly padding octets used for ciphering. |

Response SMS or Packet from the CARD should also be in format of 3.48 packets and it should be according to following Structure.

Structure of response packet is almost similar to request packet. It contains elements that were sent in request like padding counter, Redundancy check values, Counter and TAR etc.

**Table 2: Response Packet Format**

| Element | Length | Comment |
|---|---|---|
| Response Packet Identifier (RPI) | 1 octet | Identifies a Response Packet |
| Response Packet Length (RPL) | variable | Indicates the number of octets from and including RHI to the end of Additional Response data, including any padding octets required for ciphering. |
| Response Header Identifier (RHI) | 1 octet | Identifies the Response Header. |
| Response Header Length (RHL) | variable | Indicates the number of octets from and including TAR to the end of the RCICCIDS. |
| Toolkit Application Reference (TAR) | 3 octets | This shall be a copy of the contents of the TAR in the Command Packet. |
| Counter (CNTR) | 5 octets | This shall be a copy of the contents of the CNTR in the Command Packet. |
| Padding counter (PCNTR) | 1 octet | This indicates the number of padding octets used for ciphering at the end of the Additional Response Data. |
| Response Status Code Octet | 1 octet | Codling defined in Table 3. |
| Redundancy Check (RC), Cryptographic Checksum (CC) or Digital Signature (DS) | variable | Length depending on the algorithm indicated in the Command Header in the incoming message. A typical value is 4 to 8 octet or zero if no RCICCIDS is requested. |
| Additional Response Data | variable | Optional Application Specific Response Data, including possible padding octets used for ciphering. |

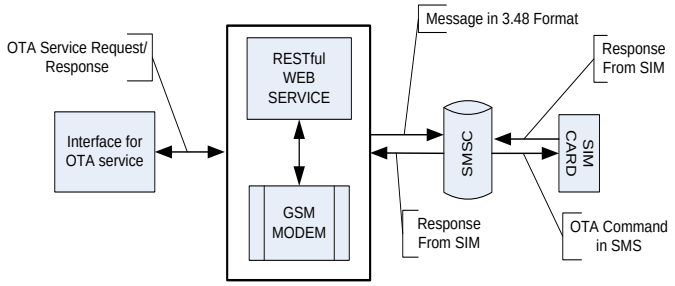
**Table 3: Response Status Code Octet**

| Status Code (hexadecimal) | Meaning |
|---|---|
| 0 | PoR OK |
| 1 | RC/CC/DS failed |
| 2 | CNTR low |
| 3 | CNTR high |
| 4 | CNTR Blocked |
| 5 | Ciphering error |
| 6 | Un-identified security error. |
| 7 | Insufficient memory to process incoming message. |
| 8 | This status code "more time" should be used if the Receiving Entity Application needs more time. |
| 9 | TAR Unknown |
| 0A-FF | Reserved for future use |

## II. Implementing OTA Gateway

Now that we know about the interaction, facilities offered by technology and requirement of OTA packet we can look into how to implement OTA Gateway which is the prime description point of this paper. In order to replace OTA Gateway we need to simulate request response pair to and from the web interface to the OTA Gateway. This will be achieved with help of PHP service consumers sending request in format of JSON to web

service. Response from the web service will also be sent in JSON format. Response will then be parsed by PHP service handler and corresponding results will be displayed to the user. Web service on the other side will implement 3.48 standards to create required OTA packet. Along with that J2EE API project called RXTX will be used for communication with GSM modem over serial port. This is the most important part of the implementation because everything depends upon the interaction of the Web Service with GSM Modem. Basic architecture of this implementation will look as following:




RESTful web service will perform task of getting input from Interface, parsing the input for required request, formation of message and instructing GSM Modem to deliver the message to required destination.

## III. Future Scope

**Use of Cloud computing for operating server from anywhere-** Web Services can be stored on cloud server and modem can be placed in a centralized location. This will allow access of services and modem from anywhere in the world either on a computer or a handheld device.

## IV. CONCLUSION

From above discussion we know now that with help of JAVA RESTful web service implementation and GSM Modem, an OTA Gateway can easily be created and used. This implementation is beneficial for telecom companies who work on SIM card application development which required OTA testing. With help of this implementation, in house OTA testing of application can easily be carried out at a very low cost and effort.

## References

[1] 3GPP "Over the Air Technology" S3-030534
[2] GSM 03.48: "Security mechanism for the SIM application toolkit"

## Authors

**First Author** – Sarabjeet Singh, Masters of Computer Applications, sarabjeet.singh2610@gmail.com