Mobile ID
RADIUS integration guide

Version: 1.8

Swisscom (Switzerland) Ltd

Contents

1 Introduction

The Swisscom Mobile ID (MID) provides a web service interface that can be addressed natively or over a protocol translation. This document provides information and possible solutions on how to get RADIUS enabled service integrated with the MID service.

The solution presented in this document suggests adding at the customer side a RADIUS server capable to integrate the Mobile ID as a module. This document also includes the detailed steps in order to achieve this kind of server setup.

The proposed solutions are under the sole responsibility of the customer installing and using them. There will be no support provided for this.

1.1 Target readership, requirements of the reader

This integration guide is intended for **network administrators** and **system administrators** responsible for implementing and maintaining corporate Web Service over the Internet.

This manual assumes that you are familiar with the Swisscom MID service and the related "Mobile ID - Client reference guide".

1.2 Terms and abbreviations

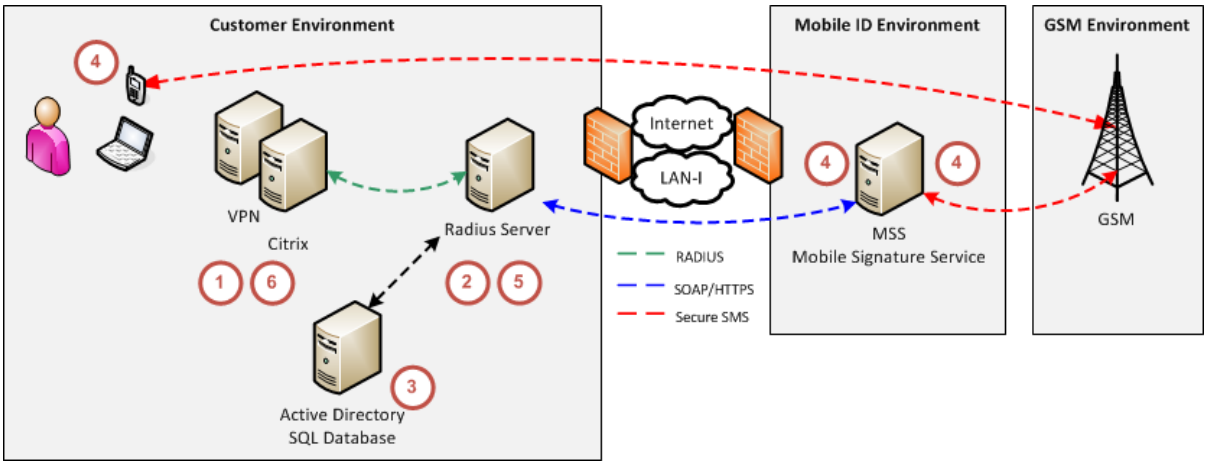
Abbreviation	Definition
AP	Application Provider
DN	Distinguished Name, often also called subject, in a X.509 certificate
Docker	https://www.docker.com is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization.
DTBD	Data to be displayed
DTBS	Data to be signed
JSON	JavaScript Object Notation is a text-based open standard designed for human readable data interchange. Although derived from the JavaScript scripting language it is language independent. The JSON format is often used for serializing and transmitting structured data over a network connection, primarily between a server and a web application, as an alternative to XML.
M-ID or MID	Mobile ID platform providing the mobile signature service
MSISDN	Number uniquely identifying a subscription in a GSM/UMTS mobile network
RESTful	Representational State Transfer is a style of software architecture for distributed systems such as the World Wide Web. It is based on the existing design of HTTP/1.0. REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses.
SOAP	Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services relying on Extensible Markup Language (XML)
WS	A Web service (WS) is a method of communication between two electronic devices over the Web (Internet). The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network". It has an interface described in a machine processable format (specifically Web Services Description Language, known by the acronym WSDL).
RADIUS	Remote Authentication Dial In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for computers to connect and use a network service.

1.3 Referenced documents

[1] Mobile ID - Client reference guide v2.8.pdf

2 Overview

Before entering into more technical details, let's have a short look at the main scenario:



This shows several RADIUS enabled services like VPN and Citrix sending their RADIUS request to a RADIUS server. This RADIUS server will invoke the Swisscom MID web service and provide the answer back to the clients. The RADIUS server may also be connected to an external user store, like Microsoft Active Directory, where the end users details like phone number or credentials are stored. Here the dataflow:

1. The RADIUS enabled service makes a request to the defined RADIUS server
2. The MID enabled RADIUS server calls the MID service
3. The RADIUS server, optionally, verifies the user credentials against internal user stores and/or maps to a valid mobile phone user
4. The MID platform ensures that the end-user signature request is allowed and forwards the signature request to the end-user's mobile phone
5. The end-user answer will be processed by the MID platform and provided to the RADIUS server
6. After verification of MID response by the RADIUS server, the RADIUS client request will be answered

3 RADIUS server capabilities for MID service use

3.1 Extensible in order to place requests to the MID web service

The MID service exposes web services based on SOAP or RESTful (JSON) and does not support the RADIUS protocol directly. Nerveless, most of the RADIUS servers have extension capabilities or flexible modules that can be adapted in order to integrate the MID service.

If a specific RADIUS server does not provide those extension capabilities, the standard RADIUS Proxy configuration should be considered as a possible option.

3.2 Proper RADIUS client timeout, retry and fallback handling

As the MID service requires end-user interaction, it provides no immediate response to the clients. Those clients must be able to set timeouts of at least 90 seconds. In case of retries or fallback, the RADIUS server must be capable to handle those aspects properly.

3.3 Translation of user credentials into mobile number

If the RADIUS client provides credentials that are not valid MID service numbers (MSIDN), the RADIUS server must provide an option to convert the user credentials into a valid MSIDN. Common ways to store such mappings are local files, LDAP / Active Directory and SQL databases.

3.4 Define the Data to be Signed (DTBS)

The RADIUS server has to define the DTBS that will be displayed on the end users mobile. This can either be a generic/global service message like “server.com: Authenticate with Mobile ID?” or a specific, user translated, message for each RADIUS client.

3.5 Set the user language

Beside the DTBS the Mobile ID signature request requires also the user language. This language is relevant for resource push from the MID service platform to the mobile user. The RADIUS server can use one global language or generate request specific communication. In this case the DTBS and user language should be consistent to avoid a language mix at the end user device.

3.6 Handle the security elements

The RADIUS server must provide an option to handle and validate the Mobile ID response. Especially the signature and the unique Mobile ID credentials needs to be taken in consideration. Refer to Chapter 5.3 “User Mapping” in [1].

Common ways to store those credentials like the unique SerialNumber of the Distinguished Name or the public key of the certificate are local files, LDAP / Active Directory and SQL databases.

4 Example: How to integrate Mobile ID into the FreeRADIUS server

This chapter presents the integration of the MID service into a widely adopted and deployed open source RADIUS server. The explanations are related to the MID service call itself and is not a complete guide/solution for FreeRADIUS deployment itself. It assumes knowledge of RADIUS and the related radius client solutions as well of the FreeRADIUS server itself[1].

The callout to the MID service is done over a FreeRADIUS rlm_exec[2] script that invokes the MID web service and is replying to the FreeRADIUS server according to its interface specifications.

Preconditions:

- Installed and running server with a Linux distribution like Ubuntu
- Functional and tested MID Web Service interface

4.1 Install FreeRADIUS

Refer to the official documentation on how to install the FreeRADIUS itself.

4.2 Install and configure the Mobile ID module

Refer to https://github.com/SCS-CBU-CED-IAM/freeradius-mobileid

4.3 Patch the timeout (if needed)

FreeRADIUS has a hard limit in regards to the timeout value for a module. This value may be too low and will not allow the end user to reply in proper time.

FreeRADIUS has been updated to allow a high enough rlm_exec timeout value[3]. This update may not be present on the binary version available on your distribution. If this is the case you can either install it from the sources or manually patch the binaries. Refer to https://github.com/SCS-CBU-CED-IAM/freeradius-mobileid#patching-rlm_exec-for-higher-timeout for more details.

Note: patching on Windows must not be done as the timeout is ignored by the rlm_exec module on this platform.

4.4 Docker Image

Instead of installing FreeRADIUS on a dedicated server, a Docker image can be used. The Docker image on https://github.com/SCS-CBU-CED-IAM/freeradius-mobileid/tree/master/docker contains a lightweight and fast FreeRADIUS 3.x server with integrated Mobile ID and LDAP/Active Directory support as described in chapter 4.5.3.

The FreeRADIUS Docker Image is publicly available in DockerHub:
https://hub.docker.com/r/swisscomtds/freeradius-mobileid

Detailed information about how to run the image and which parameters must be provided can be found in the Docker README file available under both URLs mentioned above.

[1] http://freeradius.org
[2] http://wiki.freeradius.org/modules/Rlm_exec
[3] https://github.com/FreeRADIUS/freeradius-server/pull/858

4.5 Advanced integration options

In this chapter you will find several advanced integration options that can be covered with the FreeRADIUS server.

4.5.1 Implicit and transparent user mapping to MID service

The simplest configuration is to have any RADIUS client user id processed and forwarded by the RADIUS server to the MID service. Based on the answer of the MID service the request will be either valid or rejected.

This assumes that:

- RADIUS server will authorize/deny users solely based on MID service decision
- The RADIUS client will provide a valid MSISDN number as user id
- The password, even if provided, will be ignored by the RADIUS server
- The User Language is set for all users globally by the RADIUS server
- The security element of the Mobile ID is not validated by the RADIUS server

This configuration is the default one.

4.5.2 User mapping, language, password and security element – File based

Rather than allowing each MID service user to be transparently addressed, this option shows how to configure local users[4].

This allows you to define following elements:

- Specific user id's
- The corresponding MSISDN number
- An optional user language
- An optional user password
- An optional user related security element

To achieve this setup define specific users in the `<cfg>/users` file. Here some examples of user entries:

```
# user with specific mobile number and no password validation at all
"user1"    Called-Station-Id := +41791234567

# user with specific mobile number, password and language
"user2"    Cleartext-Password := "pwduser", Called-Station-Id := +41791234567, X-MSS-
Language := de

# user with specific mobile number, password and serialnumber in the DN as security
element
"user3"    Cleartext-Password := "pwduser", Called-Station-Id := +41791234567, X-MSS-
MobileID-SN := MIDCHEGU8GSH6K83
```

The passwords can also be hashed in order to avoid cleartext-password usage. Refer to the official documentation and also to this site[5]

```
# user with specific mobile number and encrypted password
"user4"    Crypt-Password := "saV9ejDMWuP92", Called-Station-Id := +41791234567
```

[4] http://freeradius.org/radiusd/man/users.html
[5] http://www.packtpub.com/article/freeradius-authentication-storing-passwords

4.5.3 User mapping, language, password and security element – LDAP / Active Directory based

An alternative way is to have the RADIUS server users stored and retrieved from an LDAP store[6] like Active Directory or openLDAP.

Here we will focus on the enhancements that can be applied in order to support the MSISDN mapping and user language retrieval relevant for the MID service callout.

This allows you to define following elements:

- Specific user id's and MSISDN number in the LDAP / Active Directory store
- An optional user language from the LDAP / Active Directory store
- An optional LDAP / Active Directory password validation
- An optional user related security element

This configuration assumes an installed and configured LDAP store for your FreeRADIUS server. Refer to the online documentation of the LDAP server and the FreeRADIUS LDAP module.

Enable LDAP in the authentication section of your sites and define proper attribute mapping between RADIUS and the related LDAP attributes. Here some attribute mapping examples:

```
# FreeRADIUS 2.x: Adjust the <cfg>/ldap.attrmap
checkItem  Called-Station-Id       mobile
checkItem  X-MSS-Language          preferredLanguage
checkItem  X-MSS-MobileID-SN       msNPCallingStationID


# FreeRADIUS 3.x: Adjust the <cfg>/mods-available/ldap
update {
     ...
     #  Generic valuepair attribute
     Called-Station-Id       := 'mobile'
     X-MSS-Language          := 'preferredLanguage'
     X-MSS-MobileID-SN       := 'msNPCallingStationID '
```

Password validation: FreeRADIUS can validate the password in at least 2 different ways. By verifying an attribute of the LDAP store and by doing an LDAP bind with the user credentials.
Both solutions have specific limitation and requirements on the LDAP store. Especially the Active Directory user password validation implies the usage of SAMBA; refer to the documentation[7].

Often the RADIUS enabled solutions will do LDAP / Active Directory password validation before calling the RADIUS server itself. This means that FreeRADIUS does not have to revalidate it again and may simplify the configuration steps.

User attribute update: If needed, FreeRADIUS can also update the related LDAP user with proper security attributes after successful Mobile ID login. Refer to https://github.com/SCS-CBU-CED-IAM/freeradius-mobileid#updating-ldapad-with-initialcurrent-x-mss-mobileid-sn-value for more details about this.

[6] http://wiki.freeradius.org/modules/Rlm_ldap
[7] http://wiki.freeradius.org/guide/FreeRADIUS-Active-Directory-Integration-HOWTO

4.5.4 User mapping, language, password and security element – SQL database based

This configuration will achieve the same results as in 4.5.3 but with an SQL database[8] as storage rather than LDAP.

This allows you to define following elements:

- Specific user id's and MSISDN number in the SQL store
- An optional user language from the SQL store
- An optional user related security element

This configuration assumes an installed and configured SQL Store for your FreeRADIUS server. Refer to http://wiki.freeradius.org/guide/SQL-HOWTO and other online documentation.

Enable sql in the authentication section of your sites and specify users and their related attributes:

```
INSERT INTO radcheck (username, attribute, op, value) VALUES ('bob', 'Cleartext-
Password', ':=', 'passbob');
INSERT INTO radcheck (username, attribute, op, value) VALUES ('bob', 'Called-Station-
ID', ':=', '+41792080001');
INSERT INTO radcheck (username, attribute, op, value) VALUES ('bob', 'X-MSS-Language',
':=', 'fr');
INSERT INTO radcheck (username, attribute, op, value) VALUES ('bob', 'X-MSS-MobileID-
SN', ':=', 'MIDCHEGU8GSH6K80');

INSERT INTO radcheck (username, attribute, op, value) VALUES ('alice', 'Cleartext-
Password', ':=', 'passalice');
INSERT INTO radcheck (username, attribute, op, value) VALUES ('alice', 'Called-Station-
ID', ':=', '+41792080002');
```

Existing database schema: Additionally, it's also possible to use sql queries against a different database schema than the FreeRADIUS one. The SQL module supports SQL queries in xlat strings. This allows extracting the value of a single field and using it, either as a check item, a request item or a reply item. The strings will be of the following form:

```
%{sql:SELECT field FROM `table` WHERE field = %{User-Name}}
```

Examples of SQL queries for MSISDN number and user language:

```
Called-Station-Id = %{sql:SELECT mobile FROM `users` WHERE id = %{User-Name}}
X-MSS-Language = %{sql:SELECT language FROM `users` WHERE id = %{User-Name}}
X-MSS-MobileID-SN = %{sql:SELECT mobileIDSN FROM `users` WHERE id = %{User-Name}}
```

For more details about this check the http://wiki.freeradius.org/guide/SQL-HOWTO documentation.

[8] http://wiki.freeradius.org/modules/Rlm_sql

5 Appendix

5.1 RADIUS client capabilities and recommended settings

Retry / Retransmissions: disabled

If the RADIUS client does not get a reply from the destination RADIUS server within a specific time it will do a number of retries. The retry should at least cover the time for the end user to confirm the request. This can take up to 90 seconds; therefore we recommend disabling the retries.

Timeout: at least 90 seconds

If the RADIUS client does not get a reply from the destination RADIUS server within a specific time it will do a timeout after a specific number of retries. The timeout should at least cover the time for the end user to confirm the request. This can take up to 90 seconds; therefore we recommend setting a high enough client timeout.

5.2 Hardware Requirements

The recommended minimal hardware requirements are the following:

- 2 CPUs
- 4 GB RAM
- 50 GB Hard Disk

The sizing for the hard disk should be on the safe side basing on the following estimation:

- An average of 1 KB / authentication
- Approximately 10.000 users authenticating three times a day
- A traffic of 30 MB / day e.g. 3 GB / 100 days

5.3 Literature & Support

FreeRADIUS Beginner's Guide - ISBN : 1849514089 [9]

Manage your network resources with FreeRADIUS - eBook - http://it-ebooks.info/book/1957/

NetworkRADIUS FreeRADIUS Documentation http://networkradius.com/freeradius-documentation/

[9] http://www.packtpub.com/freeradius-master-authentication-authorization-accessing-your-network-resources/book?tag=rk/freeradiusbg-abr3/0911