Mobile ID RADIUS Integration Guide

Technical Documentation - Version 2.0

Table of contents

Mobile ID is a brand of Swisscom.
Swisscom (Switzerland) Ltd
Alte Tiefenaustrasse 6
CH-3050 Bern

1 Introduction

The Swisscom Mobile ID Service 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 Mobile ID 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.

The Swisscom Mobile ID authentication solution protects access to your company data and applications with a comprehensive end-to-end solution for two-factor authentication (2FA). Mobile ID can be used in multiple processes: everything from the simple addition of a second factor to an existing login, to password-free two-factor authentication, online signatures, and geolocation. It is suitable for different system landscapes and meets strict regulatory requirements.

Please visit https://mobileid.ch for further information. Do not hesitate to contact us in case of any questions.

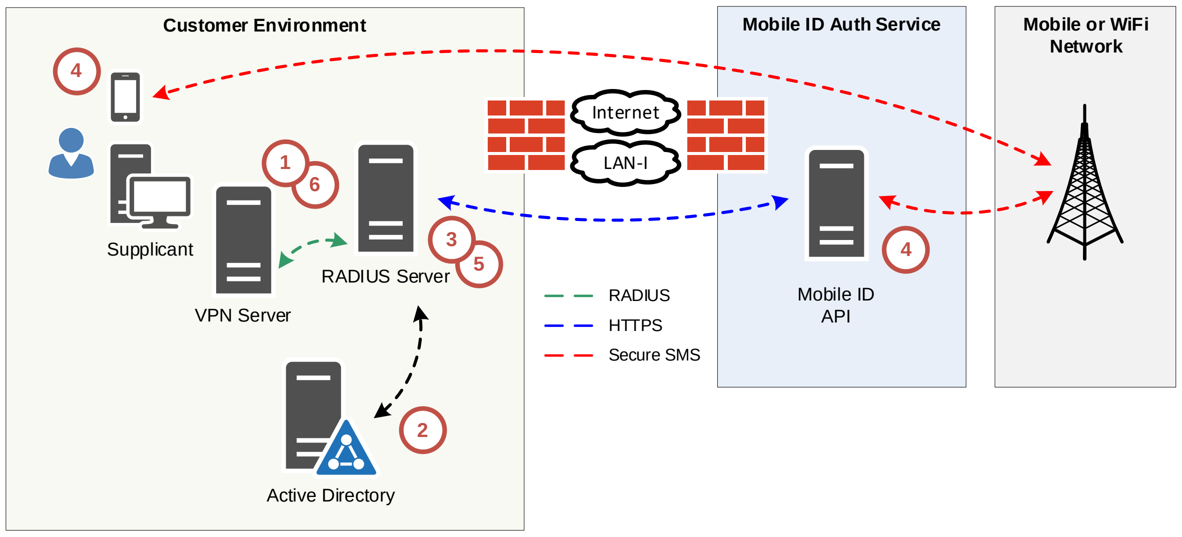
Referenced documents: Mobile ID Reference Guide, official technical documentation of the Mobile ID API.

1.1 Terms and Abbreviations

Term	Description
AP	Application Provider
AP_ID	Application Provider Identifier
DTBD	Data-To-Be-Displayed. Message that is displayed on the mobile phone (authentication context).
DTBS	Data-To-Be-Signed. Equal to DTBD. Data that will be signed with the Mobile ID signing key.
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.
LAN-I	Swisscom LAN-Interconnect Service. Also known as Enterprise WAN.
MID	Mobile ID platform providing the mobile signature service
MNO	Mobile Network Operator, also known as a wireless service provider, wireless carrier, cellular company, or mobile network carrier.
MSISDN	Number uniquely identifying a subscription in a GSM/UMTS mobile network, aka mobile phone number.
MSSP	Mobile Signature Service Provider, the Swisscom Mobile ID backend application.
OTA	Over-The-Air is a technology used to communicate with and manage a SIM card without being connected physically to the card.
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)
WSDL	The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functionality offered by a Web service.
X509	PKI and Digital Certificates
XML	Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

2 Overview

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



This shows a RADIUS enabled services like VPN (but could be basically any other RADIUS enabled service) sending their RADIUS request to a RADIUS server. This RADIUS server will invoke the Swisscom Mobile ID web service and provides the authentication result back to the clients. The RADIUS server may also be connected to an external user store, like Microsoft Active Directory (or any other user storage), where the end user details such as phone number or credentials are stored. Here's the flow:

1. The RADIUS enabled service – in this case a VPN connection request - makes an access request to a RADIUS server, which usually includes the username but not the user's mobile phone number

2. The RADIUS server verifies the user credentials against internal user store and maps the username to a valid mobile phone number

3. The Mobile ID enabled RADIUS server calls the Mobile ID API endpoint. This request includes the user's mobile phone number.

4. The Mobile ID service sends the Mobile ID authentication request to the end-user's mobile phone

5. The authentication result of the Mobile ID authentication is forwarded to the RADIUS server

6. The RADIUS server validates the response from the Mobile ID service and responds with an access accept to the RADIUS client. In this case the VPN server, which will then proceed with the VPN connection request.

3 Integration Guidelines

3.1 Mobile ID integration in RADIUS servers

The Mobile ID service[1] exposes a web services which is based on SOAP or RESTful (JSON) and it does not yet support the RADIUS protocol directly. Nevertheless, most of the RADIUS servers have extension capabilities or custom module support so that it can be adapted to integrate the Mobile ID authentication service.

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

3.1.1 RADIUS client timeout, retry and fallback handling

As the Mobile ID service requires end-user interaction, it provides no immediate response to the clients. Those clients must be able to set timeouts of at least 80 seconds. In case of retries or fallback, the RADIUS server must be capable to handle those aspects properly.

3.1.2 Mapping of user credentials to mobile numbers

If the RADIUS client provides credentials that are no valid Mobile ID service numbers (MSISDNs), the RADIUS server must provide an option to convert or map the user credentials into a valid MSISDN. Common ways to store such mappings are local files, LDAP / Active Directory and SQL databases.

3.1.3 Define the Data to be Signed (DTBS) message

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

3.1.4 Set the appropriate user language

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

3.1.5 Validate the Mobile ID authentication response

The RADIUS server must provide an option to handle and validate the Mobile ID response. Especially the signature and the unique Mobile ID credentials need to be taken in consideration. Refer to Chapter “Mobile ID Serial Number Validation” in the Mobile ID Reference Guide[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.

[1] Refer to the Mobile ID Reference Guide

3.2 Integrate Mobile ID into a FreeRADIUS server

This chapter explains the integration of the Mobile ID service into a widely adopted and deployed open-source RADIUS server. The explanations are related to the Mobile ID service call itself and it is not a complete guide for a FreeRADIUS deployment itself. It assumes basic knowledge of RADIUS and the related radius client solutions as well as FreeRADIUS server knowledge[2].

The callout to the Mobile ID service is done over a FreeRADIUS ***rlm_exec***[3] script that invokes the Mobile ID 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 Mobile ID Web Service interface

3.2.1 Install FreeRADIUS

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

3.2.2 Install and configure the Mobile ID module

Refer to the README in https://github.com/MobileID-Strong-Authentication/mobileid-enabler-freeradius

3.2.3 Patch the timeout (if needed)

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

FreeRADIUS has been updated to allow a higher ***rlm_exec*** timeout value[4]. 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/MobileID-Strong-Authentication/mobileid-enabler-freeradius#known-issues for more details about how to patch *rlm_exec* for higher timeouts.

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

3.2.4 Docker Image

Instead of installing FreeRADIUS on a dedicated server, a Docker image may be used. The Docker file on https://github.com/MobileID-Strong-Authentication/mobileid-enabler-freeradius/tree/main/docker includes a lightweight and fast FreeRADIUS 3.x server with integrated Mobile ID and LDAP/Active Directory support.

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.

[2] http://freeradius.org
[3] https://networkradius.com/doc/current/raddb/mods-available/exec.html
[4] https://github.com/FreeRADIUS/freeradius-server/pull/858

3.2.5 Advanced integration options

In the following sub-chapters, you will find advanced integration options that can be covered with the FreeRADIUS server.

3.2.5.1 Implicit and transparent user mapping to Mobile ID service

The simplest configuration is to have any RADIUS client user id processed and forwarded by the RADIUS server to the Mobile ID service. Based on the answer of the Mobile ID service the request will be either valid (access-accept) or rejected.

This assumes that:

- RADIUS server will authorize/deny users solely based on Mobile ID 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 (MIDCHE*-serialnumber) of the Mobile ID is not validated by the RADIUS server

This configuration is the default one.

3.2.5.2 Using Files: User mapping, language, password, and security

Rather than allowing each Mobile ID service user to be transparently addressed, this option shows how to configure local users[5].

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[6].

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

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

3.2.5.3 Using LDAP / Active Directory: User mapping, language, password, and security

An alternative way is to have the RADIUS server users stored and retrieved from an LDAP store[7] 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 Mobile ID 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[8].

Note: 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/MobileID-Strong-Authentication/mobileid-enabler-freeradius#updating-ldapad-with-initialcurrent-x-mss-mobileid-sn-value for more details about this.

[7] https://networkradius.com/doc/current/raddb/mods-available/ldap.html
[8] http://wiki.freeradius.org/guide/FreeRADIUS-Active-Directory-Integration-HOWTO

3.2.5.4 Using SQL database: User mapping, language, password, and security

This configuration will achieve the same results as in the previous chapter, but with an SQL database[9] 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-Pass-
word', ':=', '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.

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

mobile iD

4 Appendix

4.1 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.

4.2 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

4.3 Literature & Support

FreeRADIUS Beginner's Guide - ISBN: 1849514089 [10]

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

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

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