Mobile ID
Microsoft ADFS

Solution guide

Version: 1.3

Swisscom (Switzerland) Ltd

Contents

1 Introduction

Mobile ID (MID) provides strong authentication based cryptographic materials stored and protected in the SIM card in user's mobile phone. Active Directory Federation Services (ADFS) is an extensible, Internet-scalable, and secure identity access product of Microsoft that can operate across both Windows and non-Windows environments. This document provides information on how to integrate Mobile ID into ADFS. The solution allows any ADFS-aware applications to use Mobile ID to authenticate users.

This document focuses on installation, configuration, and troubleshooting of the Mobile ID Authentication Provider for ADFS, which is an essential part in the MID-ADFS integration. The Mobile ID authentication provider for ADFS appears as an additional authentication method in the ADFS Management tool and user Sign-In Web Interface. The configuration for individual ADFS-aware applications (SharePoint, Office 365, Microsoft Web Application Proxy, etc.) is out of scope of this document.

1.1 Target readership, requirements of the reader

The purpose of the Integration Guide is to provide an overview for system administrators, IT Professionals and support technicians who are responsible for designing, implementing and maintaining Active Directory Federation Services (ADFS). This manual assumes that you are familiar with the Swisscom Mobile ID service [1] and ADFS [2].

1.2 Terms and abbreviations

Abbreviation	Definition
AD	Active Directory
ADAL	Active Directory Authentication Library https://azure.microsoft.com/en-us/documentation/articles/active-directory-authentication-libraries/
ADFS	Active Directory Federation Services. https://technet.microsoft.com/en-us/windowsserver/dd448613.aspx
AP	Application Provider
CA	Certificate Authority
CRL	Certificate Revocation Lists
MID	Mobile ID. http://www.swisscom.com/mid
MFA	Multi-Factor Authentication. Authentication methods that require more than one independent "factors". A factor can be a knowledge factor ("what you know", e.g. password), possession factor ("what you have", e.g. Mobile ID SIM card), inherence factor ("what you are", e.g. fingerprint, retina pattern, voice).
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)

1.3 Referenced documents

[1] Mobile ID - Client reference guide v2.x.pdf
https://www.swisscom.ch/en/business/mobile-id/technical-details/technical-documents.html
[2] Overview of Active Directory Federation Services
https://technet.microsoft.com/en-us/windowsserver/dd448613.aspx
[3] Mobile ID public repository
https://github.com/SCS-CBU-CED-IAM
[4] Mobile ID – Microsoft Solution Guide v1.x
https://www.swisscom.ch/en/business/mobile-id/technical-details/technical-documents.html

2 Overview

Customers can use Windows Active Directory or various non-Microsoft (third party) identity provider databases to store their user directories. By using the WS-Federation (WS-Fed) and WS-Trust protocols, Active Directory Federation Services (ADFS) provides claims-based single sign-on for the services in the Microsoft Office service offering. The benefits of using identity federation is to provide the users in the enterprise with a single sign-on (SSO).

Here the main topics referred and used in this document:

- SAML 2.0 enables web-based authentication and authorization scenarios including cross-domain single sign-on (SSO), which helps reduce the administrative overhead of distributing multiple authentication tokens to the user.
- The ADAL based authentication stack enables applications like Microsoft Office 2013 to engage in browser-based authentication (also known as passive authentication) where the user is directed to a web page from the identity provider to authenticate, including MFA. Refer to [4] for additional documentation and information.
- Microsoft ADFS supports multi-factor authentication (MFA), which adds additional authentication methods to the so-called primary authentication method1. Immediately after a successful primary authentication, ADFS passes the primary authenticated user's identity to the additional authentication method, which performs the authentication and hands the result back to ADFS. At this point, ADFS continues executing the authentication/authorization policy and issues the token accordingly.

Mobile ID Authentication Provider for ADFS is an additional authentication method of ADFS. It implements the client interface of Mobile ID service, communicates via SOAP/HTTPS with Mobile ID Servers, and authenticates user with Mobile ID. The Mobile ID Authentication Provider retrieves the user attributes (mobile number, etc.) needed in the authentication process from AD.

[1] In Windows Server 2012 R2, the following primary authentication methods are supported: Integrated windows authentication, form-based authentication (i.e. username, password), and X.509 client certificate.

3 Architecture und main Scenario

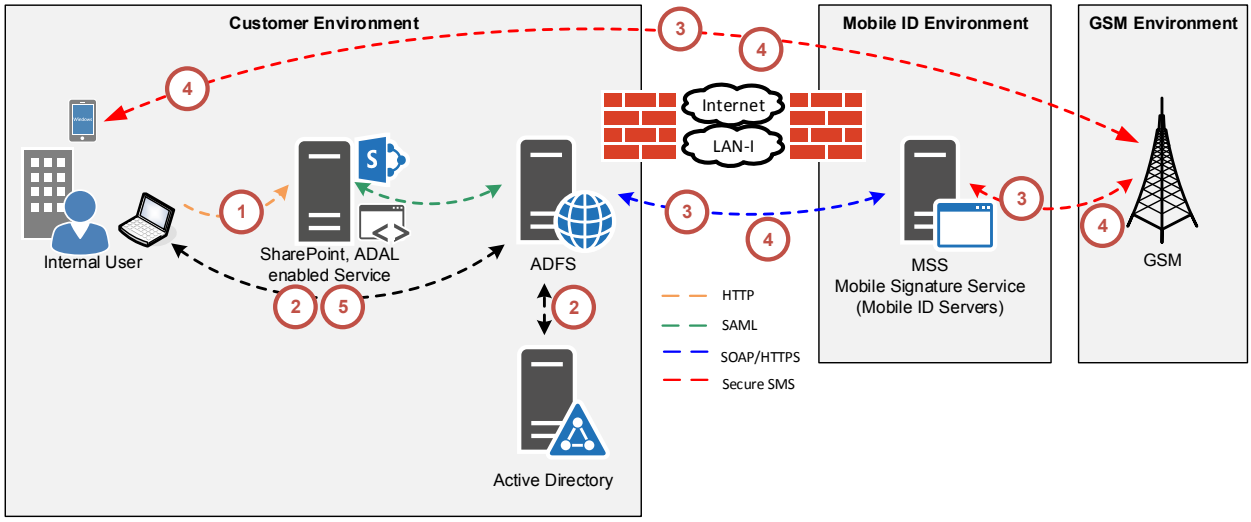

Figure 1: An example of Mobile ID and ADFS integration

In this example, the user accesses an application server (e.g. Microsoft SharePoint), which requires Multi-Factor authentication, with Mobile ID as the additional authentication method. The (simplified) sequence of data flow is as follows:

1. An unauthenticated user connects a web browser to an application server. The application server redirects the browser to a Mobile-ID enabled ADFS server(s).
2. ADFS server authenticates the user with one of the primary methods (e.g. username/password) and hand over the control to the Mobile ID Authentication Provider for ADFS.
3. The Mobile ID authentication provider sends a SOAP/HTTP request[2] in a mutually authenticated TLS/SSL connection to a Mobile ID server. The request contains the text for login prompt and the mobile number of the user. The Mobile ID server sends the specified login prompt via secure SMS to the specified mobile number.
4. The user enters his Mobile ID PIN in his mobile phone to acknowledge the login request, the mobile phone returns a signed[3] acknowledgement via secure SMS to the Mobile ID server, which (optionally) verifies the signature, embeds the signature in the SOAP response, and replies to the request in previous step.
5. The Mobile ID Authentication Provider (optionally) verifies the signature in SOAP response, and returns the authentication outcome to ADFS. ADFS builds up the claims and redirects the browser back to the application server, which then grants access to user.

[2] The MSS_SignatureReq service is invoked, see Mobile ID Client Reference Guide [1], chap. 4.2 for details.
[3] The signature is calculated with the Mobile ID private key in the SIM card, the text to be signed is specified in the SOAP request (step 3).

4 Installation and Uninstallation

4.1 System requirements and prerequisite tasks

The setup program midadfs_setup_<version>.exe of Mobile ID Authentication Provider for ADFS can be downloaded from the URL https://github.com/SCS-CBU-CED-IAM/adfs-mobileid/tree/master/binaries/ where <version> is the placeholder for version number. The current version for the time of this writing is 1.1.0.0.

4.1.1 Requirement for installation

- Windows Server 2012 R2. The role service Active Directory Federation Services (ADFS) must be installed.

4.1.2 Requirement for configuration

- A **Mobile ID Application Provider (AP) account** is available, s. Mobile ID Client Reference Guide [1], Chap.3 for more details. The account defines the following information which will be needed in the configuration:
  1. Application Provider Identifier (AP_ID)
  2. Client certificate(s) of ADFS server(s) used for the SSL/TLS communication with Mobile ID servers. It is recommended to use certificates dedicated for the Mobile ID usage.
  3. A text that will be prepended to each login prompt
- **IP connectivity**: The network infrastructure (e.g. firewall, web proxies) allows IP communication initiated from ADFS server(s) to Mobile ID server (TCP port 443) and Swisscom Certificate Authority server (http://crl.swissdigicert.ch)[5]. The following TCP connections are required:

Table 1 Required IP connectivity

Source host:port	destination host:port	protocol	description
<*adfsHost*>:<any>	mobileid.swisscom.com[6]:443	https	Mobile ID application traffic
<*adfsHost*>:<any>	crl.swissdigicert.ch:80	http	1. Used to establish the SSL connection. It is used to download the Certificate Revocation Lists (CRL) needed in validation of the Mobile ID server certificate chain. 2. Used to verify the Mobile ID user certificate chain
<*adfsHost*>:<any>	ldap.swissdigitcert.ch:389	ldap	Used to establish the SSL connection. It is used only to download the CRL when the previous CRL download via http fails.
<*adfsHost*>:<any>	ocsp.swissdigicert.ch:80	http	1. Used to establish the SSL connection. The connection is used to retrieve OCSP during the validation of the Mobile ID server certificate.

[4] This connectivity is needed to download the Certificate Revocation Lists (CRL) of the CA's in the certificate chain of the Mobile ID server. Because the CRLs are cached in ADFS servers, traffic in this connectivity can be hard to detect.
[5] This connectivity is needed to download the Certificate Revocation Lists (CRL) of the CA's in the certificate chain of the Mobile ID server. Because the CRLs are cached in ADFS servers, traffic in this connectivity can be hard to detect.
[6] If the ADFS system is connected to Mobile ID Server via the Swisscom LAN-I service, the address 195.65.233.222 should be used instead of mobileid.swisscom.com.

			2. Used to verify the Mobile ID user certificate chain
<*adfsHost*>:<any>	aia.swissdigitcert.ch:80	http	1. Used to establish the SSL connection. It is used to download the intermediate CA in the Mobile ID server certificate chain. The CA certificate is cached by the operating system. 2. Used to verify the Mobile ID user certificate chain. It is used to download the intermediate CA that signs the Mobile ID user certificate. The CA certificate is cached afterwards by the operating system.

- **TLS/SSL connectivity**: An ADFS server must establish a mutually authenticated SSL/TLS connection with a MID server before calling the MID services. The configuration requires:

  1. The SSL client certificate(s), which have been created during the creation of Mobile ID Application Provider (AP) account,
  2. Root CA certificate for MID server(s) and Mobile ID users, which can be downloaded from http://aia.swissdigicert.ch/sdcs-root2.crt or found in the folder `C:\Program Files (x86)\MobileIdAdfs\v1.x\certs`, if the Mobile ID Authentication Provider for ADFS has already been installed (Chap. 4.2),
  3. Local Administrator privilege.

  The procedure is described in Chap 4.1.3

- **User attributes in AD**: The Mobile ID Authentication Provider need to retrieve the mobile number of the user once the user has been identified with the primary authentication. The current release relies on the following LDAP attributes in Active Directory:
  - `userPrincipleName`: this is the username that the user authenticates with the primary authentication. Example: `tester1@contoso.com`
  - `mobile`:[7] a telephone number to which the Mobile ID authentication message will be sent. The number currently need to be in the format `+41`*ddddddddd* (*d*=digit where the + may be omitted). At most one telephone number can be specified. If no telephone number is specified, the user is not enabled for multi-factor authentication with Mobile ID. Example: `+41791234567`
  - `serialNumber`[8]: this attribute is only required if the serial number in user's certificate is used for verification, see [1] chap.5.3. The serial number is an alternative identifier for a user. The level of assurance is higher than the telephone number. Example: `MIDCHE0123456789`

4.1.3 Steps for setting up TLS/SSL connectivity

1. **Import the SSL client certificate file** (PFX/PKCS#12 format) into computer certificate store: Right-click your SSL Client certificate file, select `Install PFX`, the `Certificate Import Wizard` will pop up. Select `Local Machine` as `Store Location`, Click `Next` twice, then enter the passphrase of the PFX file, click `Next` and click `Finish`.
2. If the SSL client certificate is issued by a Certificate Authority trusted by your organization, you can skip this step, otherwise (e.g. self-signed certificate), you need explicitly **configure trust for the SSL client certificate**: Run `mmc.exe`, navigate to `File` > `Add/Remove Snap-in...`, select `Certificates` in left `Available`

[7] Note that it is possible to use other LDAP attribute, in that case, the configuration parameter `AdAttrMobile` (s. Chap.5.3) need to be specified.
[8] It is also possible to use other LDAP attribute. In that case, the configuration parameter AdAttrMidSerialNumber has to be set (s. Chap.4.3).

snap-ins panel, click Add >, choose Computer account, click Next, Finish, OK, the Certificates (Local Computer) snap-in is added to Management Console. In the Certificate Management Console for Local Computer; right-click Trusted People, navigate to All Task, then Import…, this opens the Certificate Import Wizard; Clicks Next, locates the PFX file in File to Import, Next, enter passphrase for the private key, clicks Next twice and Finish. Note that the service account of ADFS role service need access to the imported key/certificate.

3. **Verify the SSL client certificate** has been correctly imported and trusted: In Certificate Management Console (certmgr.msc), navigate to Personal > Certificates, double-click the certificate imported in step 1, select Certification Path, the Certificate status should displays "This certificate is OK". Do not close the console now.
4. **Configure trust to Root CA of MID servers**: In the open console, navigate to Trusted Root Certificate Authority, Right-click Certificates, select All Tasks, Import… , then Next, select the file *.crt containing the Root CA of MID servers, Next twice, Finish, confirm Yes on the Security Warning "You are about to install a certificate from a certificate authority (CA) claiming to represent: … Thumbprint (sha1): …" Click OK.
5. **Verify the SSL/TLS connectivity**: Use Internet Explorer[9] to connect to the URL https://mobileid.swisscom.com/soap/services/MSS_ProfilePort . Internet Explorer should display a Confirm Certificate dialog for picking up the client certificate and then the text

```
MSS_ProfilePort
Hi there, …
```

Notes

- The setup program for installation / uninstallation (s. Chap. 4) does not make any change in certificate store.
- It is the responsibility of ADFS administrators to monitor the expiry of SSL client certificate and renew it if necessary.

[9] Internet Explorer version 10 and 11 have been tested

4.2 Installation

Mobile ID Authentication Provider for ADFS can be installed either interactively or unattended.

In an ADFS server farm, the Mobile ID Authentication Provider for ADFS must be installed on all servers in the farm.

4.2.1 Interactive Installation

Close all running application, e.g. Windows Event Viewer before starting the installation.

Start `midadfs_setup_`*version*`.exe` and follows the wizard (see Figure 2). For the purpose of documentation, this document assumes that the default destination folder `C:\Program Files (x86)\MobileIdAdfs\v1.`*x* (*x* stands for `0` for version 1.0.y.z, `1` for 1.1.y,z, etc.) is chosen.

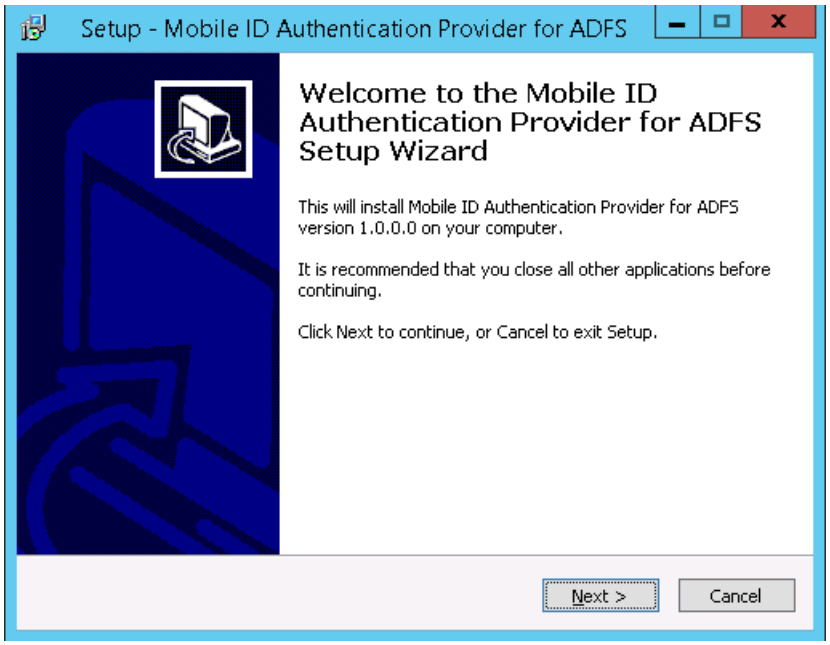

Figure 2: Setup wizard of Mobile ID Authentication Provider for ADFS

The setup program will take up to a minute to register the Mobile ID Authentication Provider for ADFS and restart the ADFS service.

4.2.2 Unattended Installation

Close all running application, e.g. Windows Event Viewer before starting the installation.

Start `midadfs_setup_`*version*`.exe` with the command line option:

`midadfs_setup_`*version*`.exe /silent`

or if the Mobile ID authentication provider needs to be installed in other folder, say `D:\midadfs\v1.`*x*,

`midadfs_setup_`*version*`.exe /silent /DIR="D:\midadfs\v1.`*x*`"`

4.2.3 Verification of Installation

On successful installation:

- A folder structure is created under `C:\Program Files (x86)\MobileIdAdfs\v1.x`.

- `Mobile ID authentication v1.x` is available as an additional authentication method in ADFS Management console (s. Figure 3) and in PowerShell (for example, `Get-AdfsAuthenticationProvider -Name MobileID11`)

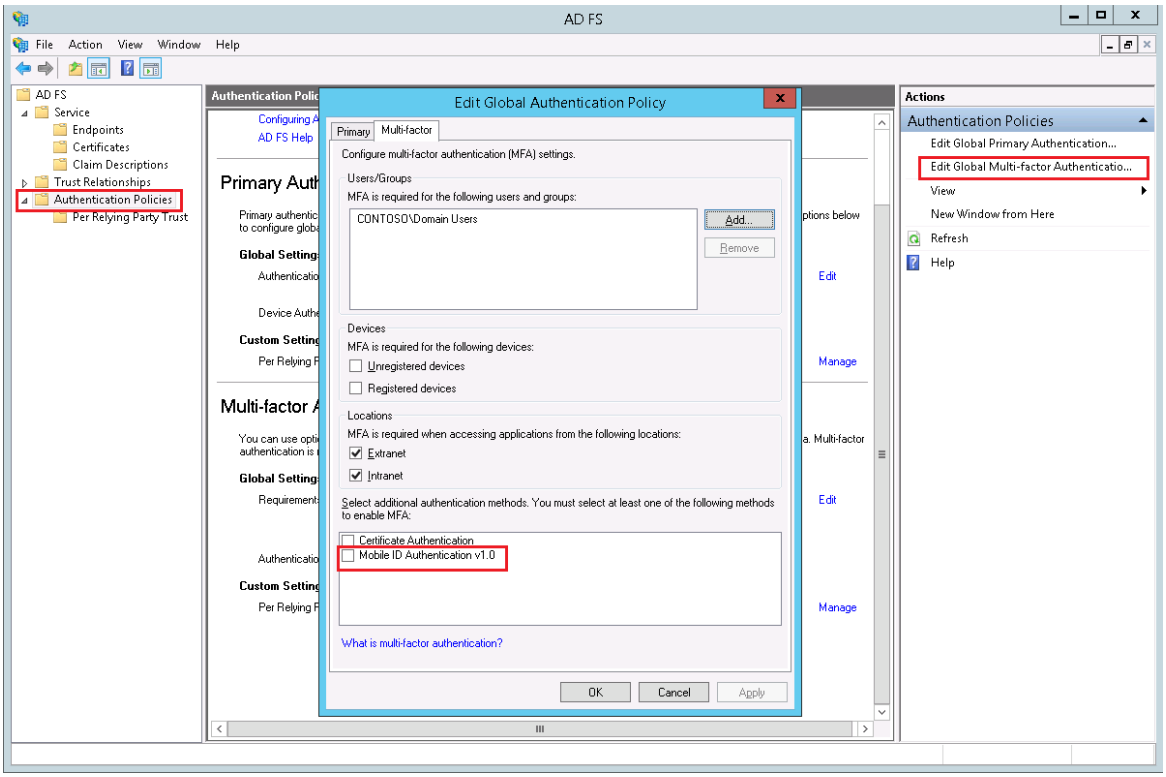

Figure 3: Mobile ID authentication provider in ADFS Management console

- There are no ADFS error messages (s. **Figure 4:**) in Windows Event Log at installation time.

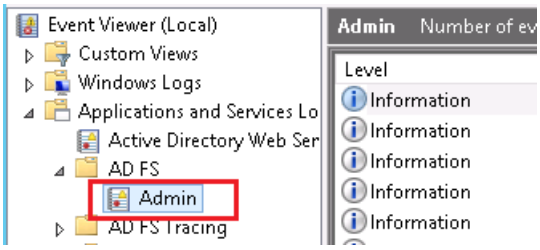

Figure 4: ADFS logging messages in Windows Event Log

Notes

- The setup program also creates log files `setup.log` and setup_trace.log in the folder `C:\Program Files (x86)\MobileIdAdfs\v1.x\inst`, which can be reviewed after installation.

4.3 Uninstallation

4.3.1 Before uninstallation

If you have adapted and edited the default **configuration file** `MobileId.Adfs.AuthnAdapter.xml` in the installation folder `C:\Program Files (x86)\MobileIdAdfs\v1.x` and want to reuse it, you should **backup** the file.

If you want to keep the **installation log files** for reference, you should backup the files `setup.log` and `setup_trace.log` in folder `C:\Program Files (x86)\MobileIdAdfs\v1.x\inst`.

The uninstallation requires a restart of ADFS service and thus may introduce a short service downtime.

4.3.2 Interactive uninstallation

In Control Panel, navigate to Uninstall or change a program, double-click the program Mobile ID Authentication Provider for ADFS version *a.b.c.d*, and follow the dialog.

The uninstallation will take a few seconds to unregister the Mobile ID authentication provider and restart the ADFS service. The actions are displayed in a PowerShell window.

4.3.3 Unattended uninstallation

Run `C:\Program Files (x86)\MobileIdAdfs\v1.x\inst\unins000.exe /silent`

4.3.4 Verification of uninstallation

After a successful uninstallation:

- Mobile ID authentication provider v1.x is no more available as an additional authentication method in ADFS Management console (s.Figure 3) and in PowerShell (Get-AdfsAuthenticationProvider –Name MobileID10)
- There are no ADFS error messages (s. Figure 4) in Windows Event Log at the time of uninstallation.

Note that the uninstaller does not delete four files (*.etwManifest.*) in the lib subfolder. Those files are needed to read the Mobile ID log entries in Windows Event Log.

4.4 Installation of multiple versions

To facilitate testing and easy rollback, it is possible to have multiple installations of Mobile ID authentication providers on the same server. One of the installations can be selected for MFA authentication in ADFS Management console.

Notes

- The installation must differ in major/minor version (i.e. the first two numbers in the version) and differ in destination folder.
- All installations share the same ADFS web scheme and same tracing files (s. Chap.6.2). They can have the same or different configuration. Their Windows EventLog definitions (s. Chap.6.1) may or may not differ.
- All installations will be loaded by ADFS service, regardless whether they have been selected as additional authentication methods.
- After the installation of version x, the version x must be configured (s. section 5). The configuration for version y (y ≠x) are not copied to version x.

4.5 Upgrade / Downgrade

A version change from x to y is done by uninstalling the version x and installing the version y. If version x is identical with version y, the current release of uninstaller still disables the Mobile ID authentication provider in ADFS. The Mobile ID authentication provider version y muss still be reconfigured (s. Chap 4) and re-selected in ADFS (s. Figure 3).

5 Configuration

In an ADFS server farm, the Mobile ID Authentication Provider for ADFS can only be configured on the primary server. The configuration is replicated[10] to or shared[11] by all slave servers automatically by ADFS.

5.1 Syntax of configuration file and basic procedure

Mobile ID Authentication Provider for ADFS is configured by a single XML configuration file, which looks like as follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<appConfig>
  <mobileIdClient attr11="value11" attr12="value12" … />
  <mobileIdAdfs   attr21="value21" attr22="value22" … />
</appConfig>
```

Notes

- The attributes of element `mobileIdClient` configures those Mobile ID behaviors, which are independent of ADFS, while the attributes of element `mobileIdAdfs` are specific to ADFS.
- The names of XML attributes are case sensitive.
- The values of XML attributes must be enclosed in quotes. Keywords (e.g. `true`) in values are case insensitive unless otherwise specified.
- Unknown attributes are ignored.
- The name and location of the configuration file can be arbitrary.

The configuration file needs to be loaded into ADFS as follows:

```
"C:\Program Files (x86)\MobileIdAdfs\v1.x\import_config.cmd" "myConfig.xml"
```

where

myConfig.xml is the (relative or absolute) path of the configuration file.

That batch command also restarts the ADFS service and its dependent services (if any) after import.

5.2 Minimal Configuration

The setup program (see Chap.4.2.1) installs a minimal configuration file `MobileId.Adfs.AuthnAdapter.xml` in `C:\Program Files (x86)\MobileIdAdfs\v1.x`, which need to be adapted.

The minimal configuration file contains all mandatory parameters and only those optional parameters with default value unsuitable in productive ADFS environment.

An example minimal configuration file looks like:

```
<?xml version="1.0" encoding="utf-8" ?>
<appConfig>
  <mobileIdClient
    AP_ID="mid://dev.swisscom.ch"
    DtbsPrefix="Test: "
    SslKeystore="LocalMachine"
```

[10] This occurs when a Windows Internal Database (WID) is used to store the configuration data.
[11] This occurs when a shared SQL database is used to store the configuration data.

```
    SslCertThumbprint="452409b86fb9541eb9dd8e3312b80a2fe2d6daac"
  />
  <mobileIdAdfs/>
</appConfig>
```

5.3 Available Configuration Parameters

Table 2 Mandatory configuration parameter in mobileIdClient element

attribute name	example value	description
`AP_ID`	`mid://dev.swisscom.ch`	Application Provider ID, as assigned by Mobile ID Service.
`SslCertThumbprint`	`452409b86fb9541eb9dd 8e3312b80a2fe2d6daac`	The SHA1 Thumbprint of client certificate used for SSL/TLS connection with Mobile ID servers (Chap.4.1.3). • The value is case insensitive. • Whitespaces in the value are ignored. • See Chap. 7.1 for how to retrieve the thumbprint of a certificate.

Table 3 Optional configuration parameter in mobileIdClient element

attribute name	default value	description
`RequestTimeOutSeconds`	`80`	Timeout value, in seconds, of Mobile ID login process.
`DtbsPrefix`	`""`	A text that will be prepended to each login prompt • The text is independent of user language. If a language-specific text is desired, it should be added to the **`LoginPrompt.`*** attributes. • White space in the text is relevant.
`PollResponseDelaySeconds`	`3`	Timespan, in seconds, between the asynchronous `RequestSignature(...)` and the first `PollSignature(...)` • Must be an integer between 1 and `RequestTimeOutSeconds`.
`PollResponseIntervalSeconds`	`1`	Timespan, in seconds, between two consecutive PollSignature(...). • Must be an integer between 1 and `RequestTimeOutSeconds`.
`ServiceUrlPrefix`	`https://mobileid. wisscom.com/soap /services/`	Prefix in the URL for SOAP requests. • The value must ends with `/`.
`SslKeystore`	`CurrentUser`	The location of SSL client certificate. Valid values are `LocalMachine` or `CurrentUser`.
`SslRootCaCertDN`	`CN=Swisscom Root CA 2, OU=Digital Certificate Services, O=Swisscom, C=ch`	Distinguished Name of the Root CA Certificate in the certificate chain of Mobile ID servers. • The parameter specifies the trust anchor in the SSL/TLS communication.
`EnableSubscriberInfo`	`false`	Whether to enable the Subscriber Info. If true, it requests MID server to include `SubscriberInfo` in response. • If the Application Provider (AP_ID) is not authorized to request Subscriber Info, the

attribute name	default value	description
		returned Subscriber Info would be empty
SeedApTransId	Some ASCII text to be used to build the unique AP_TransId in request	This text is used seed the pseudo random number generator used to build the unique AP_TransId in the request. • You can specify some unique text here to increase the randomness.
UserSerialNumberPolicy	allowAbsence, allowMismatch	Determine how the serial number in user's certificate is used in the user authentication, see Chap.5.3.1 for details.
DisableSignatureValidation	false	If the parameter is true, there is only a minimal check of signer certificate (presence of signer certificate, presence of serial number in signer certificate, time validity of signer certificate). The signature and certificates in Mobile ID server responses are not validated. If the parameter is false, the signature is validated. The verification of certificate chain depends on the parameter DisableSignatureCertValidation.
DisableSignatureCertValidation	false	Only effective if DisableSignatureValidation is false. If this parameter is true, there is only a minimal check of the signer certificate. Neither the certificates of the issuers nor certificate revocation lists are verified. If this parameter is false, the certificate chain in the Mobile ID server response is validated (including Certificate Revocation List), the key usage of the Mobile ID user certificate need to support digital signature or nonrepudiation.
SanitizePhoneNumber	false	If this parameter is "true", phone numbers read from the attribute store are transformed before use in Mobile ID calls. The transformation is specified by SanitizePhoneNumberPattern and SanitizePhoneNumberReplacement. By default, all non-digits from the input strings are removed by the transformation.
SanitizePhoneNumberPattern	\D	Only effective when SanitizePhoneNumber is true. This parameter is the regular expression[12] for matching a pattern in phone number.
SanitizePhoneNumberReplacement	(empty string)	Only effective when SanitizePhoneNumber is true. This parameter is the replace string for matched pattern defined by SanitizePhoneNumberPattern.

[12] The used regular expression engine is the .NET implementation, see https://msdn.microsoft.com/en-us/library/hs600312(v=vs.110).aspx

Table 4 Optional configuration parameter in mobileIdAdfs element

attribute name	default value	description
`AdAttrMobile`	`mobile`	Name of AD attribute which contains the Mobile Number of the user. • The name is case-insensitive and converted to lower case internally. • The AD attribute can contain at most one value.
`AdAttrMidSerialNumber`	`serialNumber`	Name of AD attribute which contains the SerialNumber (see [1], chap.5.3) in the Distinguished Name of user's certificate. • The name is case-insensitive and converted to lower case internally. • If the (multi-valued) attribute `altSecurityIdentities`[13] is chosen, only the value prefixed with `MID:<SN>` will be used for the serial number of Mobile ID. The prefix is removed in the effective serial number. • If other AD attribute is chosen, the whole value is interpreted as serial number. • The AD attribute should contain at most one serial number.
`WebClientMaxRequest`	`100`	A WebClient can be re-used to send requests. If the number of requests exceed this number, the WebClient must be re-cycled (i.e. closed and re-created)
`SessionMaxTries`	`5`	Maximum number of tries (i.e. invocation of MobileId.Iauthentication.RequestSignature) during an Mobile ID authentication session • In a *Mobile ID authentication session*, a user can retry the Mobile ID after an unsuccessful login. This is the maximum number of unsuccessful login tries in a Mobile ID authentication session.
`SessionTimeoutSeconds`	`300`	Maximum duration, in seconds, of a Mobile ID authentication session.
`SsoOnCancel`	`false`	If true, the Cancel button in Sign In pages will initiate a Single Sign Out of all sites. Otherwise, the button will initiate a Local Sign Out.
`ShowDebugMsg`	`false`	If true, display verbose messages in web browser in case of error; otherwise, less error details are leaked in browser in case of error.
`LoginNonceLength`	`5`	Length of the unique not-guessable string that should be included in the message sent to mobile device.
`LoginPrompt.en`	`Login with Mobile ID (#TransId#) ?`	The text specified here, prepended with the parameter `mobileIdClient/DtbsPrefix`, will be display in user's mobile phone. • The text can contain the (case-insensitive)
`LoginPrompt.de`	`Authentifizierung mit Mobile ID (#TransId#) ?`	

[13] The OID of the LDAP attribute is 1.2.840.113556.1.4.867. The attribute is also used for X509 certificate and Kerberos mapping by Windows, see https://msdn.microsoft.com/en-us/library/ms677943.aspx for more details.

attribute name	default value	description
`LoginPrompt.fr`	`Authentification avec Mobile ID (#TransId#) ?`	• placeholder `#TransId#`, which will be replaced by a fixed length random string at runtime. • The language (`en`, `de`, `fr`, `it`) is determined by the language preference of user's web browser.
`LoginPrompt.it`	`Autenticazione con Mobile ID (#TransId#) ?`	

There is no mandatory configuration parameter in `mobileIdAdfs` element

5.3.1 Use of Serial Number in Mobile ID verification

Mobile ID Authentication Provider for ADFS can optionally compare the serial number attribute in the user's certificate embedded in the Mobile ID response, which is never empty[14] in a valid authentication response, with the serial number attribute stored in Active Directory ("attribute store").

The configuration parameter `UserSerialNumberPolicy` determines how the verification is done. The value is either a comma-separated list of the flags or the numerical equivalent of the list. The following flags are supported (the numbers in parenthesis are their corresponding numerical value):

- `match` (0): A user is authenticated if the serial number in Mobile ID response matches the one in the attribute store. The comparison is case-sensitive.
- `warnMismatch` (1): Write a warning[15] or error message in log if the serial number in response does not match the serial number in attribute store. This flag does not affect the authentication decision.
- `allowAbsence` (2): a user is authenticated if the user has no or an empty serial number in the attribute store.
- `allowMismatch` (4): a user is authenticated if the user has a non-empty serial number in the attribute store.

The default is "`allowAbsence`, `allowMismatch`", which effectively ignores the serial number in the attribute store.

Scenarios in Authentication

Assume that the serial number in MID server response is R, the serial number in attribute store is A. R in a valid response is never empty, A may or may not be absent or empty. The result of serial number check at various `UserSerialNumberPolicy` settings is summarized in Table 5.

Table 5 Result of serial number check in various configuration

`UserSerialNumberPolicy`		result			usage scenario
text	num	A empty	A ≠ R	A = R	
`allowAbsence, allowMismatch`	6	✔	✔	✔	Serial number check is effectively disabled.
`allowAbsence`	2	✔	✘	✔	Users with serial number in attribute store are checked more stringently than users without serial numbers.
`allowMismatch`	4	✘	✔	✔	Users without serial number in attribute store cannot login with Mobile ID. However, changes in serial number are tolerated in authentication. This is intended for temporary use only.

[14] A user certificate with no or an empty serial number is considered invalid. The authentication will be rejected in this case.

[15] The warning appears as event UserSerialNumberMismatch or UserSerialNumberNotInStore in Event Log Swisscom \ MobileId \ Client.

					It is recommended to set the `warnMismatch` bit as well in this case.
`Match`	0	✘	✘	✔	Serial numbers are strictly checked.

Legend: ✔: user is successfully authenticated (subject to other checks), ✘: user is not authenticated.

5.4 Verification of Configuration

For the verification purpose, configure ADFS as follows:

1. Open the `AD FS Management Console`: start `Server Manager`, select `Tools`, `AD FS Management`.
2. In `Authentication Policies`, edit `Global Authentication Policy`, see also **Figure 3:**. For `Primary Authentication`, enable `Form Authentication` for `Extranet` and `Intranet` but do not enable `device authentication`. For Multi-factor Authentication, require MFA for both `Intranet` and `Extranet`, select '`Mobile ID Authentication v1.x`' as additional authentication method.
3. Open the URL `https://<your.adfs.server.dns>/adfs/ls/IdpInitiatedSignon.aspx` in web browser. After login with username / password, the Mobile ID login screen appears.

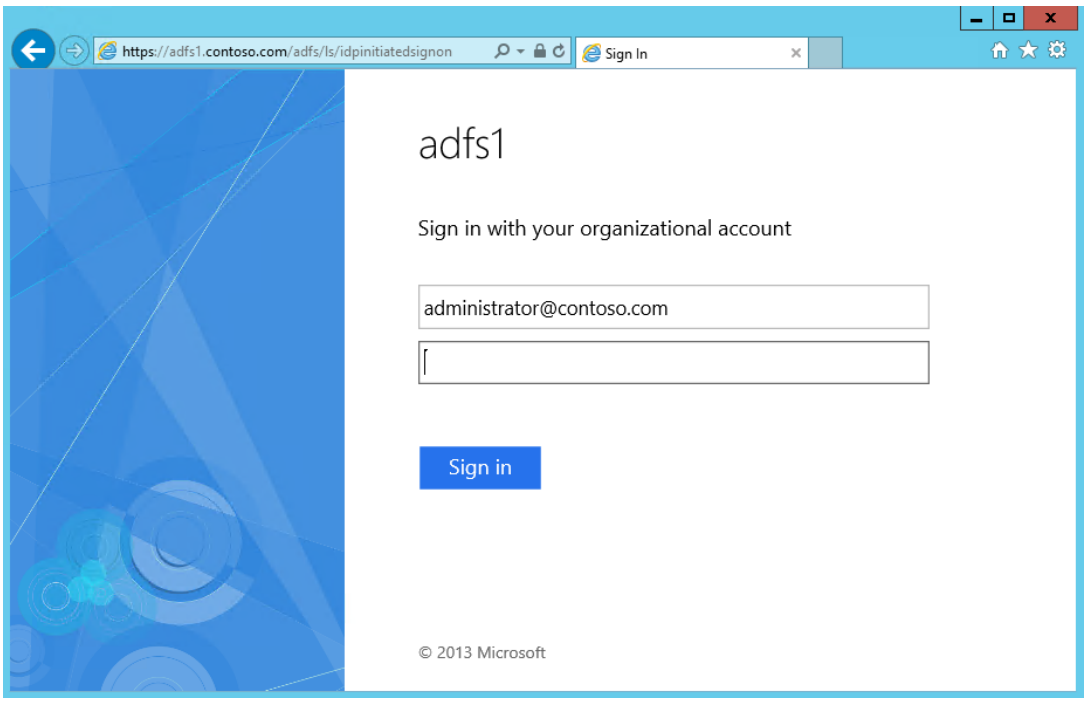

Figure 1 ADFS primary authentication

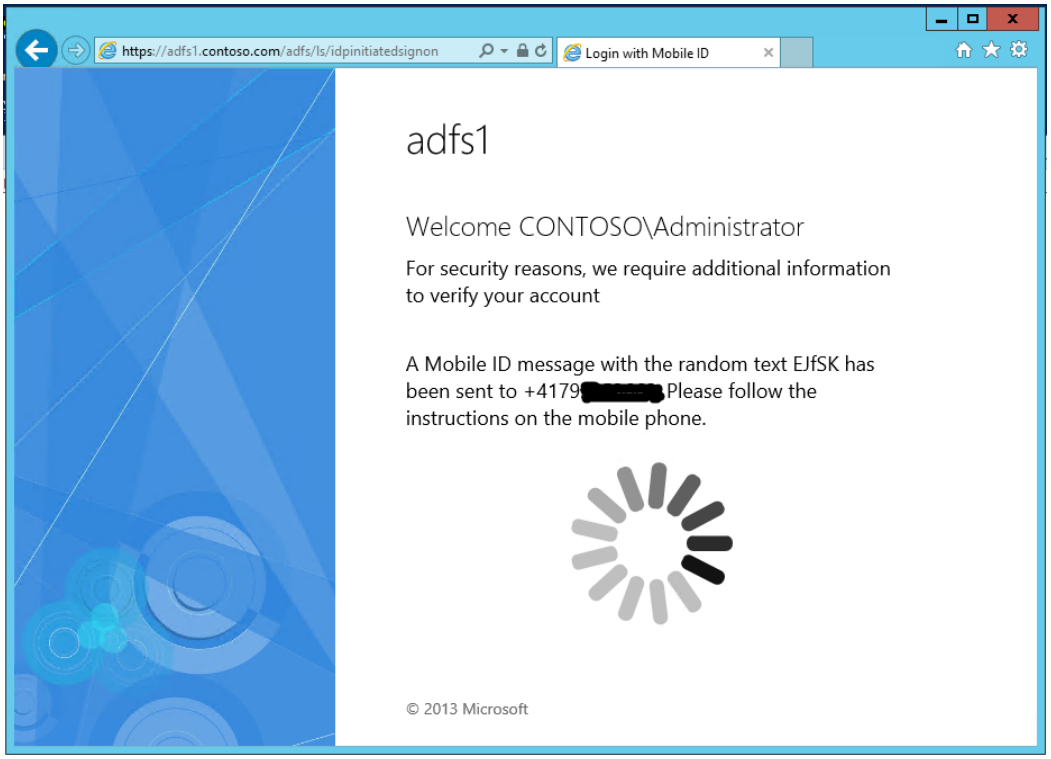

Figure 2 Mobile ID as additional authentication method

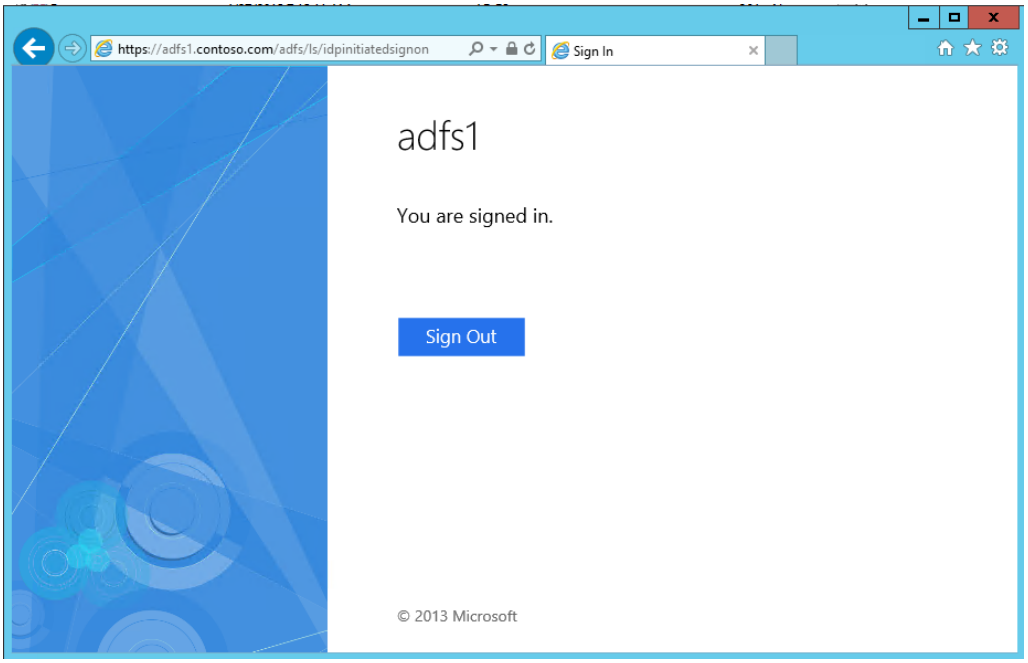

Figure 3 Authenticated with Multi-Factors

6 Operating and Trouble Shooting

6.1 EventLog

An important source of logging events is the ADFS events in Applications and Services Logs > AD FS > Admin, s. Figure 4. If the Mobile ID authentication provider cannot be started, events are frequently logged here.

If Mobile ID authentication provider can be started, it writes events in Application and Services Log > Swisscom > MobileID, see Figure 8.[16]

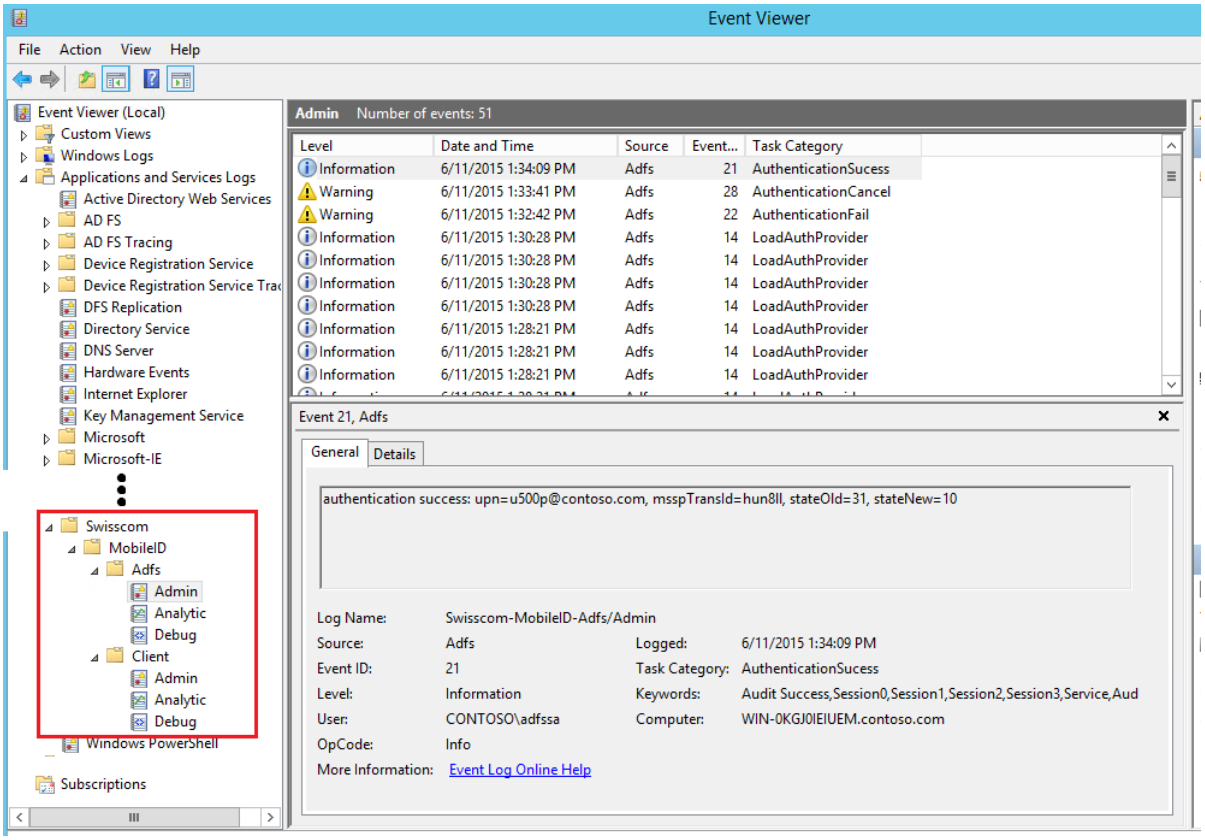

Figure 4 Logging and tracing entries of Mobile ID authentication provider, as displayed in Windows Event Viewer.

The container Client contains the lower level log entries, which are generic, and independent of ADFS, e.g. results of service call to Mobile ID servers. The container ADFS contains ADFS specific log entries, e.g. retrieval of user attributes from AD or the overall result of an authentication.

The channel[17] Admin contains log entries for typical operation and administration. The channel Analytic contains more verbose trace entries for troubleshooting. The channel Debug contains even more verbose trace entries for developers.

Logging to the Analytic and/or Debug channels are disabled by default or on operating system reboot. In order to enable or disable logging to an Analytic or Debug channel, right click the channel, select Enable Log or Disable Log respectively. There is no need to restart the ADFS service for this purpose.

[16] Mobile ID authentication provider v1.0 writes events only to Windows Log > Application.
[17] This is the terminology used in Event Tracing for Windows (ETW).

Notes

- It is also possible to consume the logs with other tools, such as wevtutil.exe (part of Windows) or perfview.exe (s. Chap 6.3 for more details).
- If Mobile ID Authentication Provider has been uninstalled and re-installed, the event viewer need to be restarted.

6.1.1 Events of Mobile ID Authentication Provider

A Windows EventLog event written by Mobile ID authentication provider looks like the one in Figure 9.

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Swisscom-MobileID-Client"
      Guid="{4D6F114F-71B1-55EB-2554-602EC88D442F}" />
    <EventID>48</EventID>
    <Version>0</Version>
    <Level>4</Level>
    <Task>3</Task>
    <Opcode>0</Opcode>
    <Keywords>0x4020f00000000021</Keywords>
    <TimeCreated SystemTime="2015-06-15T15:44:34.063755800Z" />
    <EventRecordID>215</EventRecordID>
    <Correlation ActivityID="{00000000-0000-0000-7000-0080000000D8}" />
    <Execution ProcessID="6580" ThreadID="7108" />
    <Channel>Swisscom-MobileID-Client/Admin</Channel>
    <Computer>WIN-0KGJ0IEIUEM.contoso.com</Computer>
    <Security UserID="S-1-5-21-362066223-984018922-691498886-1108" />
  </System>
  <EventData>
    <Data Name="ApTransId">X974F26F0CC207192501FC206B0BE6766</Data>
    <Data Name="MsspTransId">hv5cwr</Data>
    <Data Name="PhoneNumber">41791234567</Data>
    <Data Name="UserSerialNumber">MIDCHE3210FEDCBA</Data>
  </EventData>
</Event>
```

Figure 5 A sample signature success event in XML view. The audit relevant parameters apTransId, msspTransId, phoneNumber, and userSerialNumber are recorded.

The information in an event is described in Table 5.

Table 6 Important information items in a Windows Event

item	description
Provider	A software component, uniquely identified by Name. The following providers are included in the Mobile ID authentication provider software. • `Swisscom-MobileID-Client`: generic library for Mobile ID service calls • `Swisscom-MobileID-Adfs`: ADFS adapter
Event ID	A small integer, uniquely identify an event within a provider. Table 10 and Table 11 list all Event ID defined in the Mobile ID authentication provider.
Level	Severity or verbosity of events. It can be `Verbose` (5), `Informational` (4), `Warning` (3), `Error` (2), `and Critical` (1).
Task	Logical component or task being logged. It often encompasses several steps (OpCode). In Windows Event Viewer, it is displayed as Task Category.
OpCode	Steps within a Task. Mobile ID authentication provider uses only the following standard

item	description
	OpCode of .net framework: `Start`, `Stop`, `Info`, `Send`, and `Receive`.
Keywords	Can be used to selectively enable/disable logging of specific events, or can be used to select the events to be displayed. Mobile ID authentication provider defines the following keywords: `Audit`, `Config`, `Transport`, `Service`, `AttrStore`, `Attack`, `KeyManagement` (Client only), `Message` (Client only), `Presentation` (Adfs only).
Channel	Intended audience of the event, can be `Admin` (administrators / operators), `Analytic` (support staff), `Debug` (developer)
Data	Event-specific parameters. The frequently used parameters are described in Table 7.

Table 7 Frequently used event parameters, which appear in events with severity ≥ Informational

parameter	channels	description
AgeSeconds	Adfs	Elapsed time, in seconds, since the start of Mobile ID authentication.
ApTransId	Client	The request transaction identifier of client. It is the AP_TransID attribute in Mobile ID service requests and responses, see [1] Chap.4.2.1.
Code	Client	A numerical status code, which is either the status code of Mobile ID server (MSS Status Code or Fault Subcode Value) or is generated at client side. The implemented status code is listed in Chap. 7.4.
Content	Client, Adfs	The effective configuration.
Detail	Client, Adfs	The verbose or additional information about the server response. It is the element Detail in Fault response from MID servers, see [1] Chap.4.2.1.
HttpStatusCode	Client	numerical response code in the HTTPS response from MID servers, e.g. 500.
LdapFilter	Adfs	LDAP search filter used to retrieve the user's attributes
MsspTransId	Client, Adfs	The transaction identifier created by the MID server. It is the attribute MSSP_TransID in MID server responses, see [1] Chap.4.2.1.
PhoneNumber	Client	Phone number used to authenticate the user with Mobile ID.
Reason	Client, Adfs	The textual representation for the parameter Code. It is the element Reason in Fault response or element StatusMessage in other responses from MID servers, see [1] Chap.4.2.1.
ResponseBody	Client	The text body of HTTPS response from MID servers.
StateOld, StateNew	Adfs	(for developers only) internal state during the authentication
Upn	Adfs	User principal name of the user's Windows account
UserSerialNumber	Client	Serial Number of the user, as returned from MID servers.
xxxRequest	Client	Parameter xxx as sent in the service request.
xxxResponse	Client	Parameter xxx as seen in the service response

6.2 Trace files

Mobile ID Authentication Provider has an additional logging / tracing facility, which can be turned on for troubleshooting / debug purpose. It is controlled via the dotNet tracing configuration mechanism. The configuration file is shared with configuration file of the ADFS service, which is located in C:\Windows\ADFS\Microsoft.IdentityServer.ServiceHost.exe.config. Mobile ID Authentication Provider writes tracing messages to TraceSource MobileId.WebClient and MobileId.Adfs.AuthnAdapter. By default, logging/tracing via dotNet tracing is disabled. You can modify the configuration file to enable / adjust

tracing messages of Mobile ID Authentication Provider. The configuration is not modified by the (un)installer of Mobile ID Authentication provider.

The sample configuration segment writes all tracing messages to Windows Event Log and the files C:\midadfs\MobileIdClient.log, C:\midadfs\MobileIdAdfs.log.

```
…
<system.diagnostics>
  <switches>
    <!--  The next setting specifies the "global" logging severity threshold. In order of
decreasing verbosity,  the value can be one of "All", "Verbose", "Information", "Warning",
"Error", "Critical", "None".
    →
    <add name="MobileId.WebClient.TraceSeverity" value="All"/>
    <add name="MobileId.Adfs.TraceSeverity" value="All"/>
  </switches>

  <sources>
    …
    <source name="MobileId.WebClient" switchName="MobileId.WebClient.TraceSeverity"
switchType="System.Diagnostics.SourceSwitch">
      <listeners>
        <remove name="Default"/>
        <!-This listener writes to Windows Event Log (Log=Application,EventSource="MobileID")
→
        <add name="eventLog" type="System.Diagnostics.EventLogTraceListener"
initializeData="MobileID">
          <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information" />
        </add>
        <!-This listeners appends to a file for debugging purpose →
        <add name="logfile" type="System.Diagnostics.TextWriterTraceListener"
initializeData="C:\midadfs\MobileIdClient.log">
          <filter type="System.Diagnostics.EventTypeFilter" initializeData="All"/>
        </add>
      </listeners>
    </source>
    <source name="MobileId.Adfs.AuthnAdapter" switchName="MobileId.Adfs.TraceSeverity"
switchType="System.Diagnostics.SourceSwitch">
      <listeners>
        <remove name="Default"/>
        <!-This listener writes to Windows Event Log (Log=Application,
EventSource="MobileID.Adfs") →
        <add name="eventLog" type="System.Diagnostics.EventLogTraceListener"
initializeData="MobileID.Adfs">
          <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information" />
        </add>
        <!-This listens appends to a file for debugging purpose →
        <add name="logfile" type="System.Diagnostics.TextWriterTraceListener"
traceOutputOptions="ProcessId,ThreadId,DateTime" initializeData="C:\midadfs\MobileIdAdfs.log">
          <filter type="System.Diagnostics.EventTypeFilter" initializeData="All"/>
        </add>
      </listeners>
    </source>
  </sources>
  <trace autoflush="true" indentsize="2"></trace>
</system.diagnostics>
```

6.3 EventLog vs Trace file

The EventLog and dotNet trace file facilities are complementary. Both can be turned on simultaneously . Table 7 compares both facilities.

Table 8 Comparison of the two logging facilities

aspect	EventLog	dotNet trace file
Event content	Similar	
Event format	structured and typed	free text
performance	high	moderate
turn on/off tracing	Immediately on configuration change (no service restart is needed)	A service restart is needed.
Consumer	Event viewer, wevtutil.exe, perfview.exe, etc.	Text editor
Limitation	content length per event is limited; no guarantee to capture all `Analytic` or `Debug` events (an event is logged if any event could not be captured)	no
Recommended usage	**default**	**For debug purpose, only to capture server response which exceeds the size limit of EventLog**

7 Appendix

7.1 How to find out the SHA1 thumbprint of a certificate in Windows?

Method 1: Double-click the certificate file, SHA1 thumbprint is displayed in the Details panel, s. Figure 10.

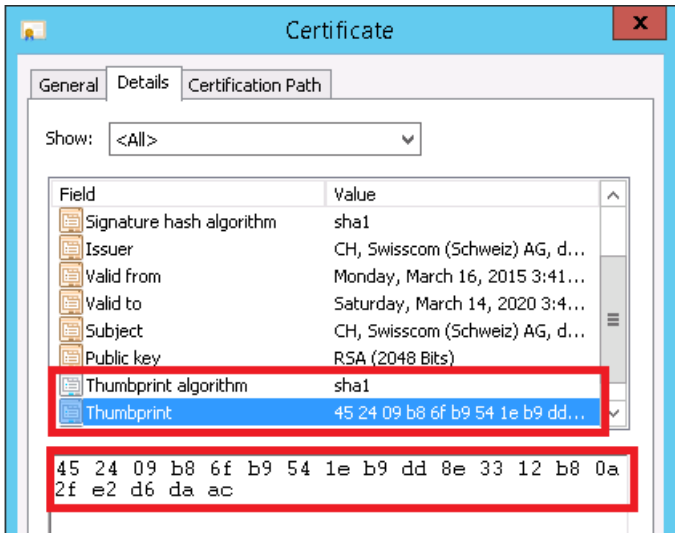

Figure 6 Display the thumbprint of a certificate

Method 2: If the location of the certificate in certificate store is known, the thumbprint can also be displayed in PowerShell with a command like: Get-ChildItem –Path cert:\\LocalMachine\My

7.2 How to find out the user serial number

A Mobile ID user can obtain his/her serial number via the Mobile ID Selfcare portal (https://www.swisscom.com/mobileid/). The link Test Mobile ID will display the serial number after the user has successfully authenticated with Mobile ID.

An administrator can also retrieve the serial number of successfully authenticated users from Windows EventLog. The relevant log entry is Swisscom-Mobile ID-Client, events with Event ID 48 (MssPollSignature).

7.3 How to collect Mobile ID logging events with perfview.exe

PerfView is a single executable file that can be used to display / select events logged with Event Tracing for Windows (ETW). It can be downloaded from http://www.microsoft.com/en-US/download/details.aspx?id=28567.

To start collecting the events written by Mobile ID authentication provider, the following additional providers need to be specified, see Figure 11.

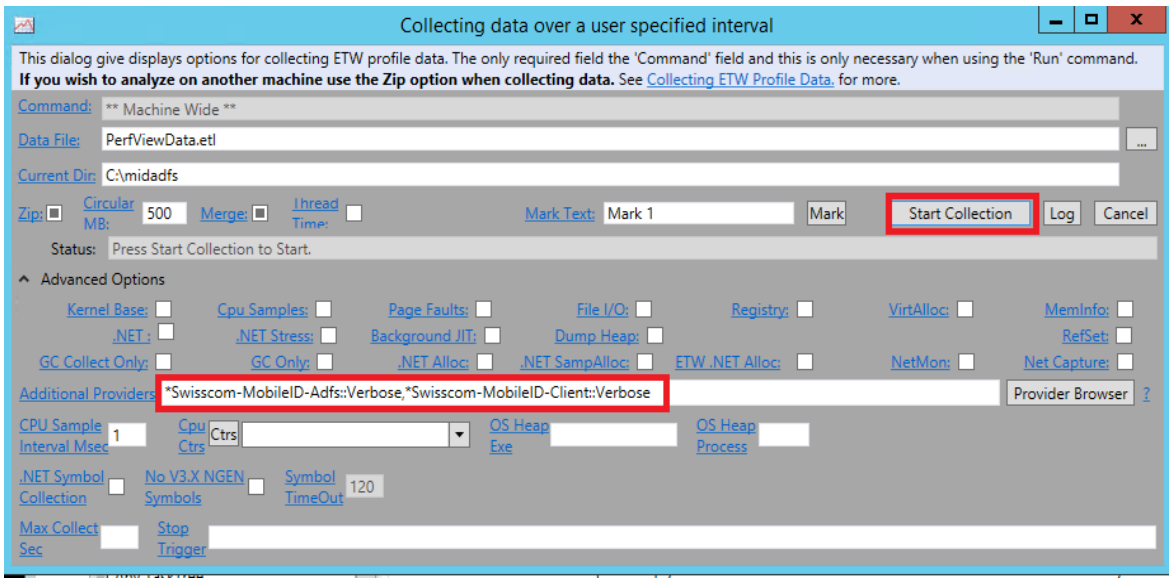

Figure 7 Collect the Mobile ID authentication provider events in PerfView.

To display the collected events, stop the collection and mark the Swisscom-MoibleID-* event types , see Figure 12.

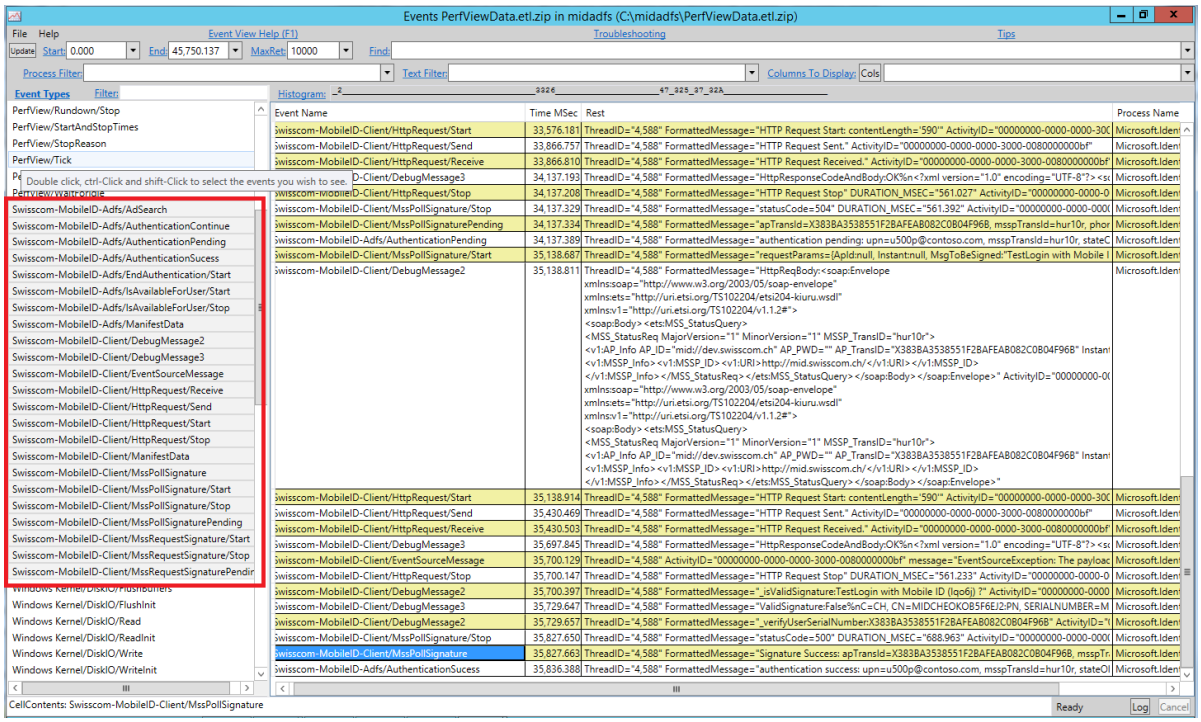

Figure 8 Display Mobile ID authentication provider events with PerfView

7.4 Catalog of service status code

The service status code has the following convention.

- Each numerical status code (e.g. 500) has an equivalent textual representation (e.g. SIGNATURE).

- A positive / uppercase value corresponds to the status code returned in MID server response, see [1], Chap.4.2.1 & 7.8.
- A negative / Pascal case value corresponds to the status code generated at the client side.
- The negative values are allocated in a similar style as HTTP status code / SMTP reply code:
  - -500 ... -599: server side error
  - -400 ... -499: client side error
  - -300 ... -399: transport / continuation / redirection
  - -200 ... -299: success
  - -100 ... -199: informational

Table 9 Status codes used by Mobile ID authentication provider

code	name	description
-508	UserCertAbsent	The server response does not contain user's certificate in the signature.
-507	UserCertExpired	The user's certificate has already expired according to the time at the client side.
-506	UserCertNotYetValid	The user's certificate is not yet valid according to the time at the client side.
-505	InvalidResponseSignature	The signature in Response is invalid or cannot be verified. This code is similar to the code INVALID_SIGNATURE, except the verification of signature is done at the client side
-504	UnknownResponse	Response has an unknown format
-503	MismatchedMsisdn	MSISDN in service response does not match the MSISDN in service request
-502	MismatchedApTransId	AP_TransID in service response does not match the AP_TransID in service request
-501	IllegalStatusCode	Mobile Service client has received a known but unexpected Fault Code or Status Code
-500	UnsupportedStatusCode	Mobile Service client has received an unsupported Fault Code or Status Code
-404	UserSerialNumberNotRegistered	User's Serial Number is not registered in the Application Provider.
-403	UserSerialNumberMismatch	User's Serial Number in MID Server Response does not match the one registered by the Application Provider.
-402	ConfigError	Client-side error in configurations
-401	InvalidInput	Input parameters of a service call are invalid or incomplete
-400	GeneralClientError	General error in client side, e.g. bug in Mobile ID client source code.
-300	CommSetupError	General error in establishing a network connection. The error can occurs in tcp, ssl, or application layer.
100	REQUEST_OK	See [1] Chap 6 & 7.8
101	WRONG_PARAM	
102	MISSING_PARAM	
103	WRONG_DATA_LENGTH	
104	UNAUTHORIZED_ACCESS	
105	UNKNOWN_CLIENT	

code	name	description
107	INAPPROPRIATE_DATA	
108	INCOMPATIBLE_INTERFACE	
109	UNSUPPORTED_PROFILE	
208	EXPIRED_TRANSACTION	
209	OTA_ERROR	
401	USER_CANCEL	
402	PIN_NR_BLOCKED	
403	CARD_BLOCKED	
404	NO_KEY_FOUND	
406	PB_SIGNATURE_PROCESS	
422	NO_CERT_FOUND	
500	SIGNATURE	
501	REVOKED_CERTIFICATE	
502	VALID_SIGNATURE	
503	INVALID_SIGNATURE	
504	OUSTANDING_TRANSACTION	
900	INTERNAL_ERROR	

7.5 Catalog of Log events of the Mobile ID authentication provider

Table 10 Log events of Swisscom-Mobile ID-Client

Id	Name	Channel	Level	Keywords	Task	Opcode	Message-Formatting	Parameters
1	DebugMessage	Debug	Verbose				{0}	Message,
2	DebugMessage2	Debug	Verbose				{0}:{1}	MessageKeyword, Message,
3	DebugMessage3	Debug	Verbose				{0}:{1} {2}	MessageKeyword, Message, Message2,
4	KeyManagementCertRetrieved	Analytic	Verbose	KeyManagement			Retrieved Cert: certname='{0}'	CertName,
5	KeyManagementCertException	Admin	Error	KeyManagement			Load Cert failed: exceptionMessage='{0}'	ExceptionMessage,
6	KeyManagementStoreError	Admin	Error	KeyManagement			Technical error while retrieving cert: storeLocation={0}, storeName={1}, findType={2}, findValue={3}, exceptionMessage={4}	StoreLocation, StoreName, X509FindType, FindValue, ExceptionMessage,
7	KeyManagementCertNotFound	Admin	Informational	KeyManagement			No valid cert found: storeLocation={0}, storeName={1}, fileType='{2}', findValue='{3}'	StoreLocation, StoreName, X509FindType, FindValue,
8	KeyManagementMultiCertFound	Admin	Informational	KeyManagement			Multiple valid certs found, the first one is used: storeLocation={0}, storeName={1}, fileType='{2}', findValue='{3}'	StoreLocation, StoreName, X509FindType, FindValue,
9	KeyManagementCertFound	Admin	Informational	Audit, KeyManagement			Found Cert: storeLocation={0}, storeName={1}, fileType='{2}', findValue='{3}'	StoreLocation, StoreName, X509FindType, FindValue,
10	HttpRequestStart	Debug	Verbose	Transport	1	Start	HTTP Request Start: contentLength='{0}'	ContentLength,
11	HttpRequestStop	Debug	Verbose	Transport	1	Stop	HTTP Request Stop	
12	HttpRequestSend	Debug	Verbose	Transport	1	Send	HTTP Request Sent.	
13	HttpRequestReceive	Debug	Verbose	Transport	1	Receive	HTTP Request Received.	
14	HttpRequestException	Admin	Error	Transport			HTTP Request Error: exceptionMessage='{0}'	ExceptionMessage,
15	HttpResponseException	Analytic	Warning	Transport			HTTP Request Error: exceptionMessage='{0}', payload='{1}'	ExceptionMessage, Payload,
19	ConfigInfo	Admin	Informational	Config			Load Config: cfg={0}	Content,
20	ServerResponseFaultCodeUnknown	Admin	Error	Message			Response ParseError: error=NoFaultCode, httpStatus='{0}', rspBody={1}	HttpStatusCode, ResponseBody,
21	ServerResponseFormatUnknown	Admin	Error	Message			Response ParseError: error=BadFaultCode, httpStatus='{0}', cursor='{1}', rspBody='{2}', exception='{3}'	HttpStatusCode, Cursor, ResponseBody, ExceptionMessage,
22	ServerResponseStatusCodeOverflow	Admin	Error	Message			Response ParseError: error=BadStatusCode, code='{0}', reason='{1}', detail='{2}'	Code, Reason, Detail,

Id	Name	Channel	Level	Keywords	Task	Opcode	Message-Formatting	Parameters
23	ServerResponseStatusCodeIllegal	Admin	Error	Message			Response ParseError: error=IllegalStatusCode, code='{0}', codeExpected='{1}', rspBody='{2}'	Code, ExpectedCodes, ResponseBody,
24	ServerResponseStatusCodeUnsupported	Admin	Error	Message			Response ParseError: error=UnsupportedStatusCode, code='{0}', reason='{1}', detail='{2}', rspBody='{3}'	Code, Reason, Detail, ResponseBody,
25	ServerResponseEmptyMssTrxId	Admin	Error	Message			Response ParseError: error='emptyMssTrxId', apTransId='{0}', rspBody='{1}'	ApTransId, ResponseBody,
26	ServerResponseMissingElement	Admin	Error	Message			Response ParseError: error='missingElement', cursor='{0}', rspBody='{1}'	Cursor, ResponseBody,
27	ServerResponseCodeMismatch	Admin	Error	Message			Response ParseError: error='illegalStatusCode', codeExpected='500', codeResponse='{0}', rspBody='{1}'	Code, ResponseBody,
28	ServerResponseEmptySignature	Admin	Error	Message			Response ParseError: error='emptySignature', rspBody='{0}'	ResponseBody,
29	ServerResponseMessageError	Admin	Error	Message			Response Error: context='{0}', error='{1}'	Context, Error,
30	ServerResponseApTrxIdMismatch	Admin	Error	Message, Attack			Hacking Attempt: error='mismatched AP_TransId', req.AP_TransId='{0}', rsp.AP_TransId='{1}', rspBody='{2}'	ApTransIdRequest, ApTransIdResponse, ResponseBody,
31	ServerResponseMsisdnMismatch	Admin	Error	Message, Attack			Hacking Attempt: error='mismatched MSISDN', req.AP_TransId='{0}', req.MSISDN='{1}', rsp.MSISDN='{2}', rspBody='{3}'	ApTransId, MsisdnRequest, MsisdnResponse, ResponseBody,
32	ServerResponseInvalidSignature	Admin	Error	Message, Attack			Hacking Attempt: error='invalidSignature', apTransId='{0}', phoneNumber='{1}', rspBody='{2}'	ApTransId, PhoneNumber, ResponseBody,
33	ServerResponseStatusTextChanged	Debug	Verbose	Message			SOAP Fault Reason ({2}) for code {0} does not match registered Reason ({1})	StatusCode, ExpectedText, SeenText,
37	UserSerialNumberMismatch	Admin	Warning	AttrStore			User Serial Numbers Mismatched: apTransId='{0}', phoneNumber='{1}', userSerialNumberInStore='{2}', userSerialNumberInResponse='{3}'	ApTransId, PhoneNumber, UserSerialNumberRequest, UserSerialNumberResponse,
38	UserSerialNumberNotInStore	Admin	Warning	AttrStore			Empty User Serial Number in Attribute Store: apTransId='{0}', phoneNumber='{1}', userSerialNumberInResponse='{2}'	ApTransId, PhoneNumber, UserSerialNumberResponse,
39	UserSerialNumberNotAccepted	Admin	Warning	AttrStore			Invalid Serial Number: apTransId='{0}', phoneNumber='{1}', userSerialNumberInStore='{2}', userSerialNumberInResponse='{3}'	ApTransId, PhoneNumber, UserSerialNumberRequest, UserSerialNumberResponse,
40	MssRequestSignatureStart	Analytic	Verbose	Service	2	Start	requestParams={0}; asynchronous={1}	RequestParams, Asynchronous,
41	MssRequestSignatureStop	Analytic	Verbose	Service	2	Stop	statusCode={0}	StatusCode,
42	MssRequestSignatureSuccess	Admin	Informational	Audit, Service	2	Info	Signature Success: apTransId='{0}', msspTransId='{1}', phoneNumber='{2}', userSerialNumber='{3}'	ApTransId, MsspTransId, PhoneNumber, UserSerialNumber,
43	MssRequestSignaturePending	Analytic	Verbose	Service			apTransId='{0}', msspTransId='{1}', phoneNumber='{2}'	ApTransId, MsspTransId, PhoneNumber,

Id	Name	Channel	Level	Keywords	Task	Opcode	Message-Formatting	Parameters
44	MssRequestSignatureWarning	Admin	Warning	Audit, Service			Signature Failure: statusCode={0}, apTransId={1}, msspTransId={2}, phoneNumber={3}, userSerialNumber={4}, detail='{5}'	StatusCode, ApTransId, MsspTransId, PhoneNumber, UserSerialNumber, Detail,
45	MssRequestSignatureError	Admin	Error	Audit, Service			Signature Error: statusCode={0}, apTransId={1}, msspTransId={2}, phoneNumber={3}, userSerialNumber={4}, detail='{5}'	StatusCode, ApTransId, MsspTransId, PhoneNumber, UserSerialNumber, Detail,
46	MssPollSignatureStart	Analytic	Verbose	Service	3	Start	requestParams={0}, msspTransId='{1}'	RequestParams, MsspTransId,
47	MssPollSignatureStop	Analytic	Verbose	Service	3	Stop	statusCode={0}	StatusCode,
48	MssPollSignatureSuccess	Admin	Informational	Audit, Service	3	Info	Signature Success: apTransId={0}, msspTransId={1}, phoneNumber={2}, userSerialNumber={3}	ApTransId, MsspTransId, PhoneNumber, UserSerialNumber,
49	MssPollSignaturePending	Analytic	Verbose	Service			apTransId={0}, msspTransId={1}, phoneNumber={2}	ApTransId, MsspTransId, PhoneNumber,
50	MssPollSignatureWarning	Admin	Warning	Audit, Service			Signature Failure: statusCode={0}, apTransId={1}, msspTransId={2}, phoneNumber={3}, userSerialNumber={4}, detail='{5}'	StatusCode, ApTransId, MsspTransId, PhoneNumber, UserSerialNumber, Detail,
51	MssPollSignatureError	Admin	Error	Audit, Service			Signature Error: statusCode={0}, apTransId={1}, msspTransId={2}, phoneNumber={3}, userSerialNumber={4}, detail='{5}'	StatusCode, ApTransId, MsspTransId, PhoneNumber, UserSerialNumber, Detail,

Legend: Column Task: 1=HttpRequest, 2=MssRequestSignature, 3=MssPollSignature

Table 11 Events of provider Swisscom-MobileID-Adfs

Id	Name	Channel	Level	Keywords	Task	Opcode	Message-Formatting	Parameters
1	WebClientCreated	Analytic	Verbose	Transport			instanceId={0}	InstanceId,
2	WebClientDestroyed	Analytic	Verbose	Transport			instanceId={0}	InstanceId,
3	AdSearch	Debug	Verbose	AttrStore			upn={0}, ldapFilter='{1}', {2}={3}, {4}={5}	Upn, LdapFilter, AttributeMobile, Mobile, AttributeUserSerialNumber, UserSerialNumber,
4	AdSearchError	Admin	Error	AttrStore			attribute store – AD search: error={0}	ExceptionMessage,
5	AttrUserNotFound	Admin	Warning	AttrStore			attribute store – user not found: upn={0}, ldapFilter='{1}'	Upn, LdapFilter,
6	AttrMobileNotFound	Admin	Warning	AttrStore			attribute store - phonenumber not found: upn={0}	Upn,
7	AttrUserSerialNumberNotFound	Admin	Informational	AttrStore			attribute store - user serial number not found: upn={0}	Upn,
8	AttrMobileMalformed	Admin	Warning	AttrStore			attribute store - phonenumber malformed (e.g. illegal length): upn={0}, phoneNumber={1}	Upn, PhoneNumber
10	IsAvailableForUserStart	Debug	Verbose	AttrStore	1	Start	upn={0}, context='{1}'	Claim, Context,
11	IsAvailableForUserStop	Analytic	Verbose	AttrStore	1	Stop	upn={1}, result={0}	Result, Upn,
12	LoadAuthProviderStart	Analytic	Verbose	Config	4	Start	instanceId={0}, version={1}	InstanceId, Version,

Id	Name	Channel	Level	Keywords	Task	Opcode	Message-Formatting	Parameters
13	LoadAuthProviderStop	Analytic	Verbose	Config	4	Stop	instanceId={0}	InstanceId,
14	ConfigInfo	Admin	Informational	Config	4	Info	load config: cfg='{0}'	Content,
15	ConfigError	Admin	Error	Config			config: error='{0}'	Message,
17	PresentationWarning	Debug	Warning	Presentation			presenstion warning: reason='{0}, message='{1}'	Reason, Message,
18	TryEndAuthenticationStop	Debug	Verbose	Service	3	Stop	TryEndAuthencation returned	
19	TryEndAuthenticationStart	Debug	Verbose	Service	3	Start	formAction='{0}', context={1}, proofData='{2}', request='{3}'	FormAction, Context, ProofData, Request,
20	AuthenticationGeneralError	Admin	Error	Service			authentication: error='{0}'	Message,
21	AuthenticationSuccess	Admin	Informational	Audit, Service			authentication success: upn={2}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId,
22	AuthenticationFail	Admin	Warning	Audit, Service			authentication failure: upn={2}, reason={4}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId, Reason,
23	AuthenticationTimeout	Admin	Warning	Audit, Service			authentication failure: upn={3}, reason=Timeout, msspTransId={4}, ageSeconds={2}, stateOld={0}, stateNew={1}	StateOld, StateNew, AgeSeconds, Upn, MsspTransId,
24	AuthenticationTechnicalError	Admin	Error	Audit, Service			authentication error: upn={2}, reason={4}, msspTransId={3}, stateOld={0}, stateNew={1}, Detail='{5}'	StateOld, StateNew, Upn, MsspTransId, Reason, Detail,
25	SessionTimeout	Admin	Error	Service			authentication session timeout: upn={3}, msspTransId={4}, ageSeconds={2}, stateOld={0}, stateNew={1}	StateOld, StateNew, AgeSeconds, Upn, MsspTransId,
26	SessionTooMuchRetries	Admin	Error	Service			too much authentication retries: upn={3}, msspTransId={4}, retries={2}, stateOld={0}, stateNew={1}	StateOld, StateNew, Retries, Upn, MsspTransId,
27	AuthenticationContinue	Analytic	Verbose	Service			upn={2}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId,
28	AuthenticationCancel	Admin	Warning	Audit, Service			authentication canceled: upn={2}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId,
29	AuthenticationPending	Analytic	Verbose	Service			authentication pending: upn={2}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId,
30	AuthenticationBadForm	Admin	Warning	Audit,Attack, Service			authentication request error: upn={2}, reason={4}, msspTransId={3}, stateOld={0}, stateNew={1}	StateOld, StateNew, Upn, MsspTransId, FormAction,

Legend: Column Task: 1=IsAvailableForUser, 2=BeginAuthentication, 3=EndAuthentication, 4=LoadAuthProvider

7.6 Change Logs

version	date	description
1.3	2016-02-25	Update the reference tables to reflect the software release v1.2.x (Table 4, Table 11). Add specification for firewall configuration (Table 1). Add clarification on the installation of multiple version (section 4.4). Add notes on the installation in a ADFS farm (section 5)
1.2	2016-01-27	Add more background information and reference to Mobile ID Microsoft Solution Guide (section 2 & section 3)
1.1	2015-06-15	Initial public release