Connectivity API Setup
M2M Service Portal

Setting up the Connectivity API using web service methods and tools

Telekom Deutschland GmbH

Version 0.89 EN
Date 22 August 2013 15:23:00
Status Draft

Table of contents

Introduction 3

Preface 3

Setup outline 3

Setup details 5
1 System Requirements 5
2 Basic information exchange 6
2.1 Information you must provide 7
2.2 Information provided by Deutsche Telekom 7
2.3 Telekom contacts 8
3 Certificate handling 9
3.1 The logic behind certificates 9
3.2 Certificate requirements 10
3.3 Requesting certificates 10
4 Setting up the clients and servers 11
4.1 Available frameworks 11
4.2 Importing WSDLs 11
4.3 Web service security settings 11
5 SSL client connection 11
6 SSL server connection 11
7 Connectivity Test & Approval 12

Annex A 13
1 Sample openssl configuration files 13

Introduction

The Connectivity API is an interface to administer and monitor M2M SIM cards.
The M2M service platform offers two web services M2MAdm and M2MMon, which may be called from the corresponding web service client implemented on your side. Your side in turn offers two web services M2MAdmCl und M2MMonCl which in turn are called by the M2M service platform. This API uses the XML-SOAP via HTTPS protocol with Web Services Security WSS and client side Token authentication in terms of an API key.

To setup the communication between you and the M2M service portal, both the SOAP clients and SOAP servers are required on your side each in two distinct environments for test & approval and production. The M2M Connectivity API allows using any compatible SOAP client/server; the choice is up to you.

Preface

This document will lead you through the following steps:

Apply for signature certificates used in communication to Telekom.
Setup a SOAP client and server.
Invoke a service of the M2M portal.
Receive an event notification from the M2M portal.

The document describes a sample communication setup to illustrate the process of connecting to the M2M portal and to allow you to check your configuration settings.

To reproduce the setup instructions of this document the following is required:

- A server connected to the internet with a DNS resolvable host name
- Two SSL server certificates for server authentication, signed by Deutsche Telekom or a public certificate authority
- Access to the M2M Service Platform 2.0 GUI
- An activated M2M SIM card and a device capable of creating a GPRS connection using the card
- For the SIM card activation example:
  - An M2M SIM card not yet registered in the cellular network
- Provided by Telekom:
  - Connectivity API WSDL files (file names M2MMonv30.wsdl, M2MMonClv30.wsdl, M2MAdmv30.wsdl, M2MAdmClv30.wsdl)
  - M2M web services endpoints as provided within this document
  - API sender identifier
  - M2M partner identifier
  - Test environment being used (TestInfrastructure 1 or 2)

Following all examples of this document will take you about three working days of your time. However, during the certificate creation and installation, you will have to wait for up to 10 working days. This step is mandatory before you can start using web services.

Setup outline

M2M web services always follow a request-response scheme. The request can be invoked on either the customer or the M2M portal side. The requesting party is said to use a SOAP client, while the responding party is said to use a SOAP server. The customer invokes a service e.g. for activating a SIM card or querying a card state (Figure 1). The M2M platform invokes a service when a certain event occurs and the customer has activated notification about this event e.g. change of network state of a SIM card (Figure 2).
A typical example where both communication directions are used is activation of a SIM card: first a customer service invocation will trigger the activation process. When eventually the activation completes, the M2M platform will invoke a service to notify the customer about this event.

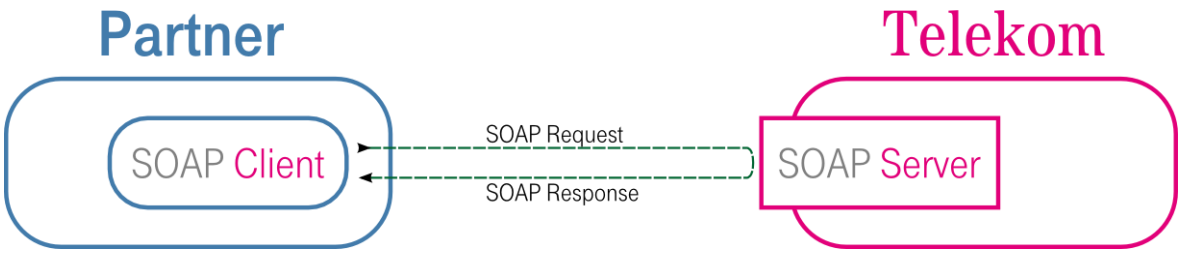

Figure 1: Connection partners in a web service call

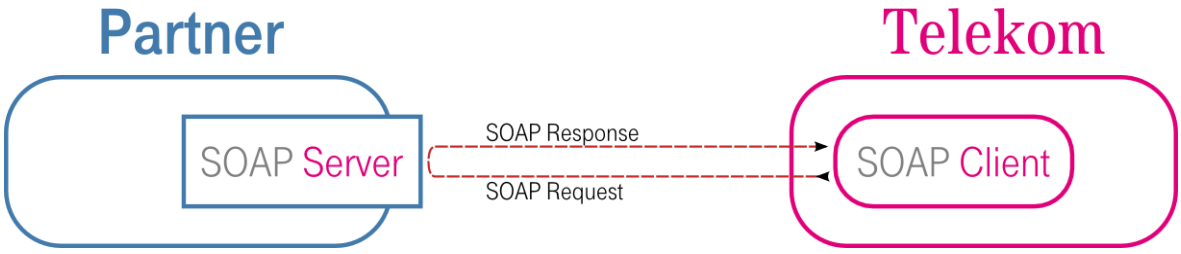

Figure 2: Connection partners in event notification

To invoke a web service you are required to execute the following steps

1. Obtain a signature certificate for secure communication
2. Create an SSL connection to the M2M service portal
3. Send request as a signed message

In step 1 a signature certificate will be created. It will be used in step 3 to sign the web service call. The signature will prove, to Telekom that the call was invoked by the authorized client. To do so, the certificate itself needs to be signed by a certificate authority (CA) of Deutsche Telekom. The SSL connection of step 2 secures the connection between client and server. However, it helps avoiding man in the middle attacks if your client checks the certificate of the M2M server.
A certificate will typically be valid for up to 3 years, so every 3 years step 1 has to be repeated to renew the certificate.

To receive a notification you are required to execute the following steps

4. Provide an SSL server certificate to authenticate your server
5. Wait for an event
6. Respond with a state code and a notification text in a signed message

In step 4 your server has to provide an SSL certificate to authenticate its identity to the M2M service portal. When the portal sends an event your server has to create a response using the signature certificate from step 1 in the same way as it would sign a request. If you omit a response or return an error, the M2M service portal might once resent the notification – depending on the notification kind.

For the certificates of steps 1 and 4 you might use any existing signing and SSL server certificates you already own, given that they are signed by a public certificate authority. If you do not own a signature certificate, Telekom offers you to create it as described in chapter 0.

Setup details

1 System Requirements

In order to connect your solution to the M2M service platform API, following requirements must be met:

- Two discrete client/server instances to adhere to the mandatory separation of test and approval (TA) and production (PE) environments. Web service end points (URLs) and private keys (and subsequently certificates) must not be shared between the two environements.
  Both instances may reside within a single physical/virtual host, however implementation in distinct hardware/software environments is strongly recommended.
- SOAP-XML HTTPS web servers exposed to the public internet. Using port 443 (HTTPS) for incoming traffic to the server is strongly recommended at least for the production environment (PE).
- At least one static IPv4 address accessible from the public internet. Using a DNS resolvable FQDN (fully qualified domain name) registered to your company instead is recommended for later address configuration amendments.
- SSL server X.509 certificates with full organization validation, issued for the respective IPv4 address or preferably FQDN by a trusted public or Deutsche Telekom internal certification authority (CA), to be sent by the host during TSL/SSL handshake. Extended Validation certificates are not required and thus optional.
- SOAP-XML HTTPS web clients capable of accessing the Deutsche Telekom B2B Gateway via HTTPS across the public internet.
- Web service security (WSS) signing X.509 certificates, to sign parts of the content of all messages (client side web service requests and server side web service responses) sent to the Deutsche Telekom B2B gateway by either client or server. SSL client certificates are not supported.

To access services provided by the optional file based/ebXML API, the following additional requirements apply:

- Dedicated HTTPS ebXML message exchange server accessible from the public internet by a static IPv4 address or preferably a DNS resolvable FQDN. The solution must conform to the ebXML standard specified by OASIS (see http://www.ebxml.org)
- Deutsche Telekom recommends using the Synchrony EndPoint Activator by Axway Inc., which is available for licensing through Deutsche Telekom. It is available for Microsoft Windows and Linux operating systems and requires 1GiB RAM (recommended 2GiB and up) and approx. 1.2GiB of HDD. For detailed system requirements and setup instruction please refer to the Axway Activator documentation.

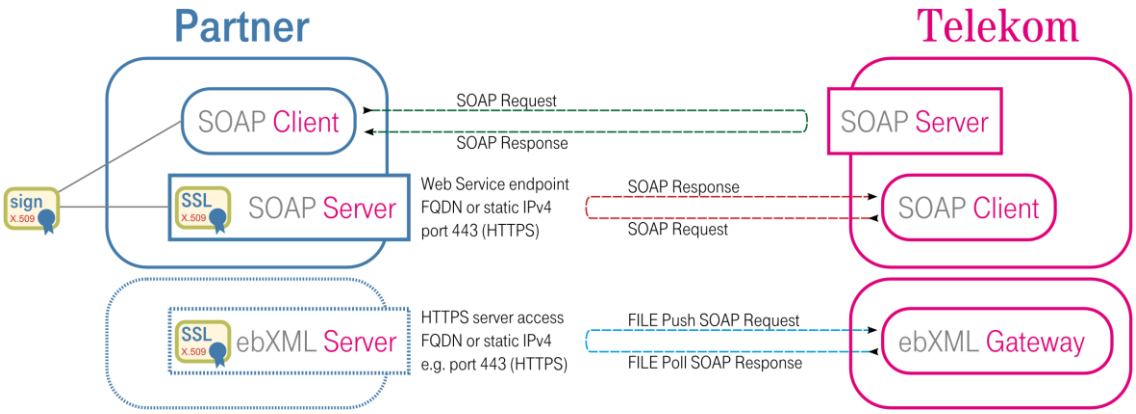

Figure 3: Partner system requirements overview

2 Basic information exchange

Telekom provides two environments: one for testing and one for production. The production environment (PE) is connected to the cellular network and accounting systems of Telekom and allows you to order true SIM cards, activate and use them, which you will be billed for. The test and approval environment (TA) allows you to test your application software without affecting the cellular network, producing bills or causing serious problems if your application still has errors. The standard process of connecting to the M2M service portal will first connect you to the test environment and once your application satisfies the requirements of the interface, you will be connected to the production environment. The only difference visible to your application will be the end points (URLs E,F/G,H) and accompanying security credentials like certificates (1,2/3,4) and API keys (J/9) associated with the respective environment which will be explained in detail later on.

For better reference to specific information and assets, the following API overview map assigns letters (Deutsche Telekom provided information) and digits (information provided by partner) to all required objects in both, the test & approval (TA) and production environment (PE).

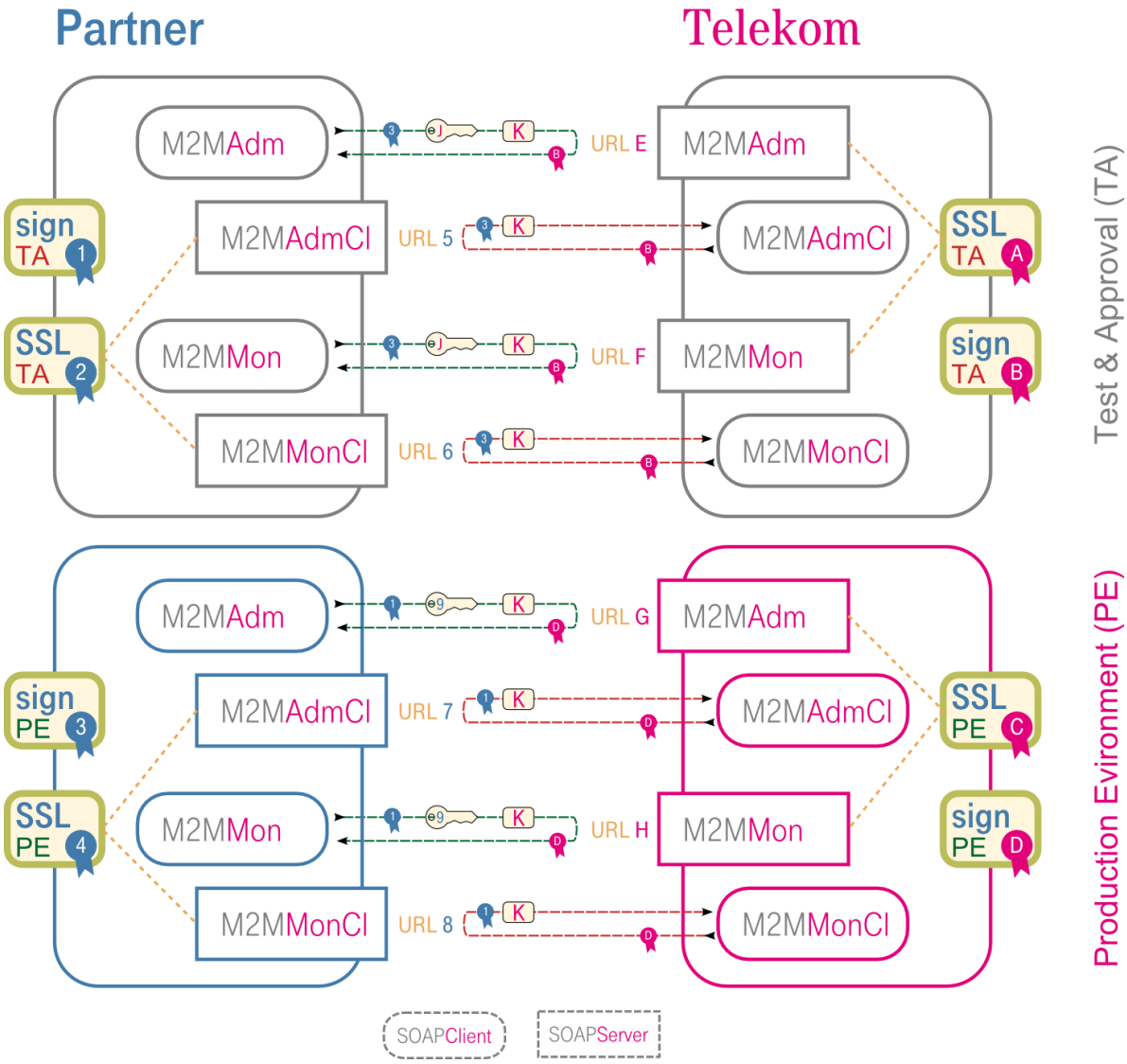

Figure 4: Connectivity web API overview

2.1 Information you must provide

You will be provided a spreadsheet table for exchanging required data between you and Deutsche Telekom. In order to set up the communication across the SOA BP B2B Gateway, the following information is required:

1 Signature certificate for the test environment TA
2 SSL server certificate for the test environment TA
3 Signature certificate for the production PE
4 SSL server certificate for the production PE
5 URL of the M2MAdmCl access point within TA
6 URL of the M2MMonCl access point within TA
7 URL of the M2MAdmCl access point within PE
8 URL of the M2MMonCl access point within PE
9 M2M Service Platform M2MApiKey within PE

Instead of purchasing appropriate certificates 1 to 4 from a public certification authority, they may as well be acquired from Deutsche Telekom in which case you will have to provide us with the corresponding certificate signing requests (CSR) instead of the certificates.

The web service end point URLs 5 to 8 must be accessible from the public internet resolving their host names via DNS. Where a DNS name cannot be provided, a static IP v4 address MAY be used instead, the use of a DNS hostname however is RECOMMENDED. While the web service end points for M2MAdmCl and M2MMonCl may be located within the same server instance, for TA and PE the end points MUST vary in at least one of hostname or network port.

For the PE, the M2M Service Platform M2MApiKey 9 which is required to be present in any request sent to the server, is a token generated within the Platform GUI and thus cannot be provided by Deutsche Telekom.

2.2 Information provided by Deutsche Telekom

In turn, Deutsche Telekom will provide you with all up to date information required to set up your web service clients and servers to connect to the M2M Service Platform in an authenticated and secure way.

A SSL server certificate of the test environment TA
B Signature certificate of the test environment TA
C SSL server certificate of the production PE
D Signature certificate of the production PE
E URL of the M2MAdm access point within TA
F URL of the M2MMon access point within TA
G URL of the M2MAdm access point within PE
H URL of the M2MMon access point within PE
J M2M Service Platform M2MApiKey within TA
K SOA-BP sender ID

All data is to be found in the spreadsheet table meant for data exchange. As the SOA-BP sender ID (K) and the M2M Service Platform M2MApiKey within TA (J) are not available unless the respective system has been provisioned with your data, this information is going to be added during the set up process as it comes available.

2.3 Telekom contacts

Your sales contact person is your main contact for ordering access to the M2M Connectivity API. You will be requested to fill in a form which queries your required connectivity type and if you would like to order your certificates via Deutsche Telekom. Once the API set up process starts, the team responsible for API setup will get in touch with you to exchange data and information you will need to develop your SOAP clients and servers. This information package will typically contain the data exchange spreadsheet table, a documentation of the available API web service requests and responses including associated WSDL files and additional type definitions and information on the web service security (WSS) requirements e.g. for time stamping and cryptographically signing messages sent from your systems.

You may find some additional contacts below to assist you in case you encounter a technical problem during setup.

Technical contacts at Telekom:

NSO DSBS – API Team
Business hours: Mo - Fr 09:00 – 16:30 CET
Phone: +49 251 78877 4774
Email: esoc-bds@telekom.de

SOA BP B2B Gateway Team
Business hours: Mo - Fr 09:00 – 16:30 CET
Phone: +49 228 181 36174
Email: security.proxies@telekom.de

3 Certificate handling

3.1 The logic behind certificates

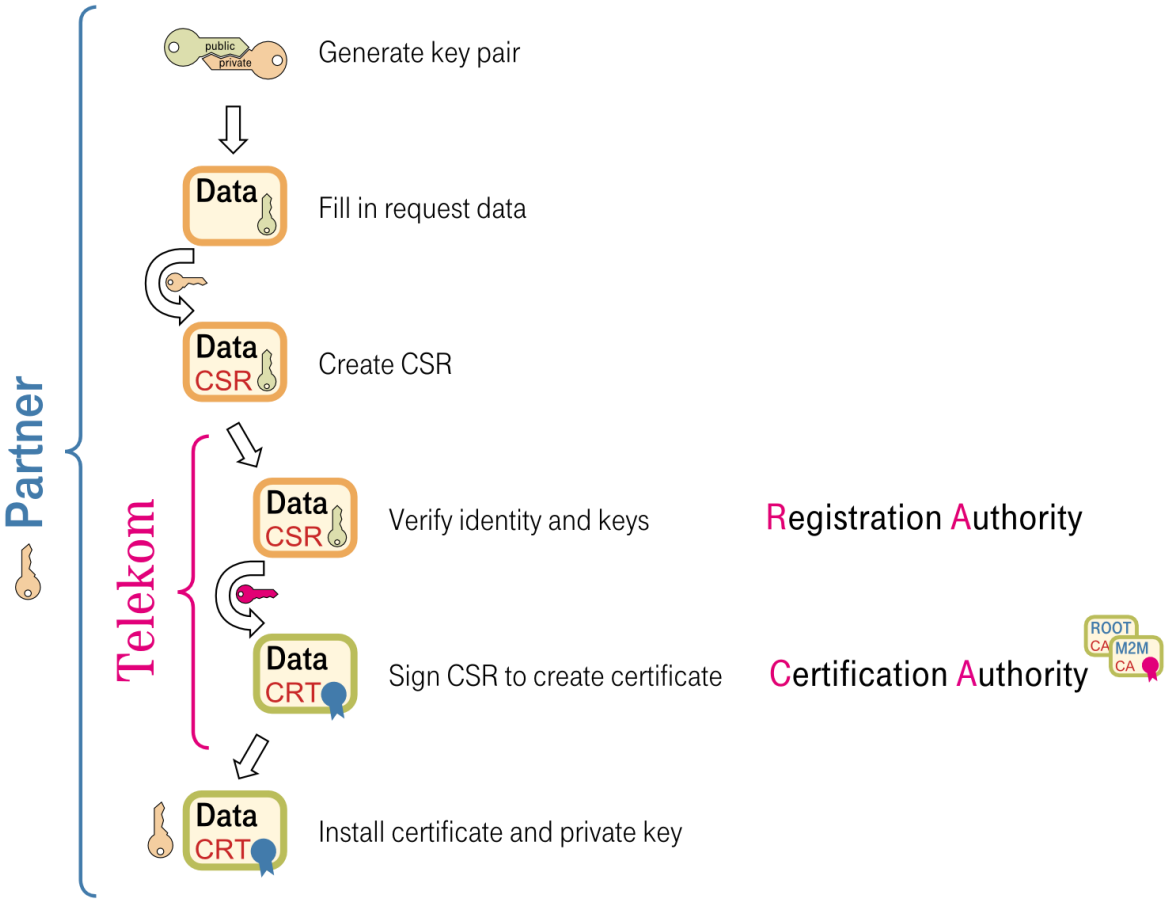

Figure 5: Steps during certificate creation

To start the certificate creation process, you create a public and private key pair. The private key represents your identity and MUST be kept secret and guarded from foreign access. The public key will become part of the certificate which will openly be exchanged in between communication partners.

Next you create a certificate signing request (CSR) containing your public key and the information you fill into the distinguished name (DN) information of the certificate. It contains your server's hostname, company address, location, country, contact person's e-mail address and some additional technical information. The CSR is created by encrypting the information and public key using your private key which is referred to as signing.

The Registration Authority (RA) will verify your identity and data in the CSR to match e.g. your company registration information and domain ownership. In addition, it will get in touch with your named technical contact via telephone to match the fingerprint of the received CSR with your local copy to assure the correct version has been transmitted and received unaltered.

The Certificate Authority (CA) will then sign your CSR and turn it into a certificate by encrypting it with its private key. This crates a chain of trust from your certificate via the intermediate CA to the root CA which allows systems to implicitly trust your certificate while it trusts the root CA and no certificate in the chain has been revoked.

Finally in order to identify your system or sign messages by this trusted certificate, you have to install both, the certificate and the associated private key you keep secret at all times, into your server/client.

3.2 Certificate requirements

Certificates used for interfacing with the B2B Gateway must meet the following specifications:

- x.509 v. 3 certificate
- Minimum encryption strength of 2048 bits RSA keys
- Issued by a Certification Authority (CA) trusted by the Deutsche Telekom B2B gateway
- Certificate subject matches partner company and address details

3.3 Requesting certificates

Creating a CSR (certificate signing request) varies across solutions and platforms. The following example will create all four certificates required in the web service API communication using the open source cryptography and SSL/TSL toolkit for reference. To adapt the configuration parameters shown here for other toolkits like the Java KeyTool, please refer to the corresponding software's manual.

- Create certificate request using openssl

Create a setup file for openssl as shown below. Adapt the keyfile, comment, country, state, location, organization, organizational unit and email address to match your organization details, where CN contains the host name of your respective endpoint and OU represents either "M2M_TA" for the test and approval or "M2M" for the production environement.
The fileds keyUsage and extendedKeyUsage must be set to reflect the type of certificate. For the SSL server certificate set *keyUsage = digitalSignature, keyEncipherment* and *extendedKeyUsage = serverAuth*. For WSS signing certificates set *keyUsage = digitalSignature* and *extendedKeyUsage = clientAuth*. A full set of example configuration files for openssl may be found in annex.

```
# OpenSSL configuration file used for generation of certificate request
#
HOME                = .
RANDFILE            = $ENV::HOME/.rnd

[ req ]
default_bits         = 2048
default_keyfile      = 1_sign_SampleOrg_TA-PrivateKey.pem
distinguished_name   = req_distinguished_name
prompt               = no
string_mask          = utf8only
req_extensions       = v3_req

[ v3_req ]
basicConstraints     = CA:FALSE
keyUsage             = digitalSignature
extendedKeyUsage     = clientAuth
nsComment            = "M2M API signature Certificate for Test and Approval (TA)"
subjectKeyIdentifier = hash

[ req_distinguished_name ]
C                    = DE                       # country
ST                   = North-Rhine-Westfalia    # state
L                    = Bonn                     # location
O                    = Sample Organization S.E. # organization
OU                   = M2M_TA                   # project name for production
CN                   = ta.sample.org            # common name: your server
emailAddress         = m2m@sample.org           # your contact person
```

Figure 6: Configuration file "1_sign_SampleOrg_TA .cnf"

Execute the command

```
openssl req -config ./1_sign_SampleOrg_TA .cnf -newkey rsa: -new -out 1_sign_SampleOrg_TA .csr
```

This will prompt for a password, create a private key, encode it with the password and write it to file *1_sign_SampleOrg_ TA-PrivateKey.pem* and create a certificate request into file *1_sign_SampleOrg_ TA.csr*

4 Setting up the clients and servers

4.1 Available frameworks

As SOAP XML is an open standard, there are various frameworks available to choose from unless you are planning to implement the standard from scratch. Well known implementations at the writing of this document are e.g. Apache Axis2, the Curl package, PHP:SOAP or Microsoft's .NET framework WCF. Each framework or class of libraries exists within its specific environment and strategy of handling security credentials, network ports, message buffers, objects and other technology and language specific structures. This document does not cover the development of your individual implementation, but will provide a basic overview of the common steps to be taken when setting up the communication between your solution and the Deutsche Telekom API.

4.2 Importing WSDLs

The web service description language (WSDL) files contain the definition of a web service interface and may typically be imported by most available toolkits and frameworks to provide a method or function stub within its specific development environment. Please refer to your chosen solution's developer guides for specific instructions.

4.3 Web service security settings

In order to access private keys for signing outbound messages, some environments use password protected solution specific or system global keystores/containers/repositories, or even hardware security devices, whereas other implementations read cryptographic credentials from flat files residing in protected areas in the file system. Please refer to your specific environment documentation on where to store the required certificates and corresponding keys in a secure way, so your server/client applications may access/read them for signing messages.

Most development environments will provide a way to add a web service security (WSS) configuration by means of extending the basic XML envelope based on a set of rules or policies added to it. In this case, two policies must be added for outgoing web service requests.

The M2M service platform access requires two security credentials in every message sent to the B2B Gateway, be it a service request from the partner client or response to a received request to the partners server.

- add a timestamp, time to live (TTL) set to 5 minutes (=300 000 milliseconds)
- add a signature and configure the appropriate private key to be used for signing

5 SSL client connection

Your SOAP client implementation must use SSL (HTTPS) to connect to the M2M service portal. The B2B Gateway server does not support SSL client certificates but will instead validate the message based on the certificate used in signing the message.
It is recommended, that you production client should verify the SSL server certificate of the B2B Gateway to prevent it from connecting to an attacker instead of the Telekom server via e.g. a forged route and send sensitive data to it.

6 SSL server connection

When the M2M portal sends an event notification, a SOAP message will arrive at the server provided by you. The message endpoint URL is to be defined in the basic data exchange spreadsheet and the server must present the appropriate SSL certificate for the respective environment during the HTTPS handshake. The B2B gateway will drop the connection if your server's identity cannot be verified. This way no sensitive data is sent to an attacker.

7 Connectivity Test & Approval

In order to be granted access to the production environment (PE), your API implementation MUST pass the integration security test within the test and approval environment (TA) before.

Even though the API should be fully implemented, this test is limited to the following scenarios:

a) synchronous request
   - M2M SIM Usage Get Retrieve the current usage volume of the specified SIM
     1. Partner *M2MMon client* initiates Session by SIMUsageGet to Telekom *M2MMon server* and immediately receives SIMUsageResponse in return.

b) asynchronous request
   - M2M SIM Activation Activate individual shipped SIM
     1. Partner *M2MAdm client* initiates session by SIMActivation to Telekom *M2MAdm server* and immediately receives a MessageResponse in return.
     2. SIM is activated in the background.
     3. Telekom *M2MAdmCl client* initiates a session by M2MAdmResponse to Partner *M2MAdmCl server* and immediately receives an M2MAdmResponseMessage in return.

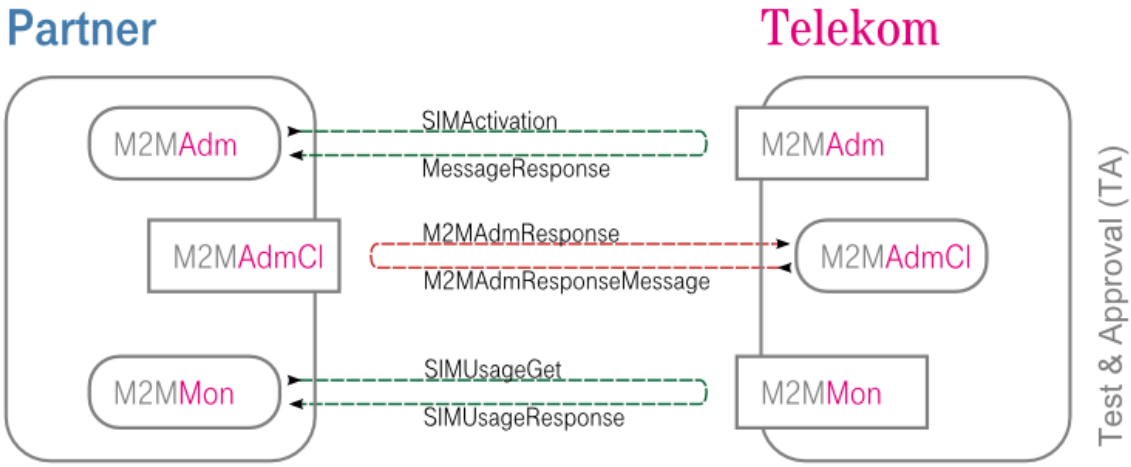

Figure 7: Test & Approval API security test cases – message flows

Once you concluded implementing your API clients and servers, Deutsche Telekom will guide you through the test scenarios. Please reserve one or two days for this common process and schedule it at least one week in advance, once you can foresee the finalization of your solution development.

Annex A

1 Sample openssl configuration files

```
# OpenSSL configuration file used for generation of certificate request
#
HOME                    = .
RANDFILE            = $ENV::HOME/.rnd

[ req ]
default_bits          = 2048
default_keyfile       = 1_sign_SampleOrg_TA-PrivateKey.pem
distinguished_name    = req_distinguished_name
prompt                = no
string_mask           = utf8only
req_extensions        = v3_req

[ v3_req ]
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature
extendedKeyUsage      = clientAuth
nsComment             = "M2M API signature Certificate for Test and Approval (TA)"
subjectKeyIdentifier = hash

[ req_distinguished_name ]
C                     = DE                                # country
ST                    = North-Rhine-Westfalia             # state
L                     = Bonn                              # location
O                     = Sample Organization S.E.          # organization
OU                    = M2M_TA                            # project name for production
CN                    = ta.sample.org                     # common name: your server
emailAddress          = m2m@sample.org                    # your contact person
```

```
# OpenSSL configuration file used for generation of certificate request
#
HOME                    = .
RANDFILE            = $ENV::HOME/.rnd

[ req ]
default_bits          = 2048
default_keyfile       = 2_ssl_SampleOrg_TA-PrivateKey.pem
distinguished_name    = req_distinguished_name
prompt                = no
string_mask           = utf8only
req_extensions        = v3_req

[ v3_req ]
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature, keyEncipherment
extendedKeyUsage      = serverAuth
nsComment             = "M2M API SSL-server Certificate for Test and Approval (TA)"
subjectKeyIdentifier = hash

[ req_distinguished_name ]
C                     = DE                                # country
ST                    = North-Rhine-Westfalia             # state
L                     = Bonn                              # location
O                     = Sample Organization S.E.          # organization
OU                    = M2M_TA                            # project name for production
CN                    = ta.sample.org                     # common name: your server
emailAddress          = m2m@sample.org                    # your contact person
```

```
# OpenSSL configuration file used for generation of certificate request
#
HOME                    = .
RANDFILE            = $ENV::HOME/.rnd

[ req ]
default_bits          = 2048
default_keyfile       = 3_sign_SampleOrg_PE-PrivateKey.pem
distinguished_name    = req_distinguished_name
prompt                = no
string_mask           = utf8only
req_extensions        = v3_req

[ v3_req ]
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature
extendedKeyUsage      = clientAuth
nsComment             = "M2M API signature Certificate for Production Environment (PE)"
subjectKeyIdentifier = hash

[ req_distinguished_name ]
C                     = DE                                # country
ST                    = North-Rhine-Westfalia             # state
L                     = Bonn                              # location
O                     = Sample Organization S.E.          # organization
OU                    = M2M                               # project name for production
CN                    = m2m.sample.org                    # common name: your server
emailAddress          = m2m@sample.org                    # your contact person
```

```
# OpenSSL configuration file used for generation of certificate request
#
HOME                    = .
RANDFILE            = $ENV::HOME/.rnd

[ req ]
default_bits          = 2048
default_keyfile       = 4_ssl_SampleOrg_PE-PrivateKey.pem
distinguished_name    = req_distinguished_name
prompt                = no
string_mask           = utf8only
req_extensions        = v3_req

[ v3_req ]
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature, keyEncipherment
extendedKeyUsage      = serverAuth
nsComment             = "M2M API SSL-server Certificate for Production Environment (PE)"
subjectKeyIdentifier = hash

[ req_distinguished_name ]
C                     = DE                                # country
ST                    = North-Rhine-Westfalia             # state
L                     = Bonn                              # location
O                     = Sample Organization S.E.          # organization
OU                    = M2M                               # project name for production
CN                    = m2m.sample.org                    # common name: your server
emailAddress          = m2m@sample.org                    # your contact person
```