Service Provider Device Configuration
Version 12.0
18 July 2025

Security Classification: Non-Confidential

Copyright Notice

Disclaimer

Compliance Notice

The information contain herein is in full compliance with the GSMA Antitrust Compliance Policy.

This Permanent Reference Document is classified by GSMA as an Industry Specification, as such it has been developed and is maintained by GSMA in accordance with the provisions set out GSMA AA.35 - Procedures for Industry Specifications.

Table of Contents

1 Introduction

1.1 Overview

This document describes an Over The Air (OTA) mechanism that allows a Service Provider to provision mobile and non-mobile devices with the necessary configurations to use their services. It provides an alternative to the Open Mobile Alliance's (OMA) Device Management (DM) approach. For transport, the mechanism mainly relies on the Hyper-Text Transfer Protocol (HTTP).

This configuration can be initiated both from the device and from the network. It allows configuration both over Service Provider controlled access networks (e.g., cellular) and non-Service Provider controlled networks (e.g. a 3rd party provided WLAN [Wireless Local Area Network]). It also allows for the provision of messages from the Service Provider to the user potentially requiring acceptance before the provided configuration can be used.

1.2 Scope

This document covers both the device and network aspects of the configuration. It only describes the generic parts of the configuration. Service specific aspects need to be described in documents relating to that service (for example PRD [Permanent Reference Document] RCC.07 for RCS [Rich Communication Services] based services). It only covers the UNI (User-Network Interface) aspects and does not deal with the internal network and device aspects of the provisioning.

1.3 Abbreviations and Definitions

Term	Description
AC	Application Characteristic
AKA	Authentication and Key Agreement
APN	Access Point Name
AuC	Authentication Centre
BSF	Bootstrapping Server Function
B-TID	Bootstrapping Transaction Identifier
CA	Certification Authority
DDF	Device Description Framework
DER	Distinguished Encoding Rules
DM	Device Management
DNS	Domain Name System
EAP	Extensible Authentication Protocol
EAP-AKA	Extensible Authentication Protocol for 3rd Generation Authentication and Key Agreement
FQDN	Fully Qualified Domain Name
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GID1	Group Identifier 1

Term	Description
GID2	Group Identifier 2
HPLMN	Home Public Land Mobile Network
HTTP	Hyper-Text Transfer Protocol
HTTPS	Hyper-Text Transfer Protocol Secure
IMEI	International Mobile Station Equipment Identity
IMPI	Internet Protocol Multimedia Subsystem Private Identity
IMS	Internet Protocol Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
MCC	Mobile Country Code
MNC	Mobile Network Code
MO	Management Object
MSISDN	Mobile Subscriber Integrated Services Digital Network Number
NAI	Network Access Identifier
OMA	Open Mobile Alliance
OMA-CP	Open Mobile Alliance Client Provisioning
OMA-DM	Open Mobile Alliance Device Management
OMNA	Open Mobile Naming Authority, registry available at: http://www.openmobilealliance.org
OTA	Over The Air
OTP	One Time Password
PC	Personal Computer
PRD	Permanent Reference Document
PS	Packet Switched
Push Notification	A notification to the RCS client, when not connected to the RCS Service Provider, that a new incoming communication event (e.g. new Chat Message, File Transfer, Audio Message, Message or File Transfer Status / State notifications) is available for the user. The Push Notification is sent by the RCS Service Provider to wake up the RCS client for establishing a connection to the RCS Service Provider for the new incoming communication event.
Push Notification Mechanism	A mechanism that enables the RCS Service Provider to enable non-persistent connections between the RCS Service Provider and the RCS client (in contrast to the mechanism that uses persistent connections) for communication event delivery. The mechanism is not visible to the user.
RADIUS	Remote Authentication Dial In User Service
RCS	Rich Communication Services
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SMS	Short Message Service

Term	Description
Temporary Token	Temporary Token that the client obtains from a SIM authentication endpoint for the purpose of configuration provisioning
UCS2	2-byte Universal Character Set
UDH	User Data Header
UI	User Interface
UNI	User-Network Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User Experience
VAPID	Voluntary Application Server Identification (for Push Notification Mechanism)
Wi-Fi	Trademark of Industry Consortium "Wi-Fi Alliance" used as synonym for WLAN (Wireless Local Area Network)
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

1.4 References

Ref	Doc Number	Title
1.	[3GPP TS 22.011]	3GPP TS 22.011, 3rd Generation Partnership Project; Service accessibility http://www.3gpp.org
2.	[3GPP TS 23.003]	3GPP TS 23.003 Release 10, 3rd Generation Partnership Project; Numbering, addressing and identification http://www.3gpp.org
3.	[3GPP TS 23.040]	3GPP TS 23.040 Release 10, 3rd Generation Partnership Project; Technical realization of the Short Message Service (SMS) http://www.3gpp.org
4.	[3GPP TS 24.109]	3GPP TS 24.109 Release 10, 3rd Generation Partnership Project; Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details http://www.3gpp.org
5.	[3GPP TS 24.368]	3GPP TS 24.368, 3rd Generation Partnership Project; Non-Access Stratum (NAS) configuration Management Object (MO) http://www.3gpp.org
6.	[3GPP TS 33.220]	3GPP TS 33.220 Release 10, 3rd Generation Partnership Project; Generic Authentication Architecture (GAA);Generic Bootstrapping Architecture (GBA) http://www.3gpp.org
7.	[GSMA PRD-IR.67]	GSMA PRD IR.67 - “DNS/ENUM Guidelines for Service Providers & GRX/IPX Providers” http://www.gsma.com/
8.	[GSMA PRD-RCC.15]	GSMA PRD RCC.15 IMS Device Configuration and Supporting Services

Ref	Doc Number	Title
		http://www.gsma.com
9.	[GSMA PRD-PDATA.01]	GSMA PRD PDATA.01 OpenID Connect Mobile Connect Profile http://www.gsma.com/
10.	[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt
11.	[RFC2616]	Hypertext Transfer Protocol -- HTTP/1.1 IETF RFC http://tools.ietf.org/html/rfc2616
12.	[RFC3986]	Uniform Resource Identifier (URI): Generic Syntax IETF RFC http://tools.ietf.org/html/rfc3986
13.	[RFC4169]	Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) Version-2 IETF RFC http://tools.ietf.org/html/rfc4169
14.	[RFC4187]	Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) IETF RFC http://tools.ietf.org/html/rfc4187
15.	[RFC4282]	The Network Access Identifier IETF RFC http://tools.ietf.org/html/rfc4282
16.	[RFC5280]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile IETF RFC https://datatracker.ietf.org/doc/html/rfc5280
17.	[RFC6265]	HTTP State Management Mechanism IETF RFC http://tools.ietf.org/html/rfc6265
18.	[RFC7578]	Returning Values from Forms: multipart/form-data IETF RFC http://tools.ietf.org/html/rfc7578
19.	[RFC8292]	Voluntary Application Server Identification (VAPID) for Web Push http://tools.ietf.org/html/rfc8292
20.	[OMA CP Cont]	Provisioning Content, Approved Version 1.1 – 28 Jul 2009 OMA-WAP-TS-ProvCont-V1_1-20090728-A http://www.openmobilealliance.org
21.	[OMA DM DDF]	Device Description Framework, Approved Version 1.2 – 24 May 2016 OMA-SUP-DTD_DM_DDF-V1_3-20160524-A http://www.openmobilealliance.org
22.	[HTML-4.0]	HTML 4.01 Specification https://www.w3.org/
23.	[OpenID Connect]	OpenID Connect Core; OpenID Foundation http://openid.net/connect/

1.5 Conventions

"The key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional" in this document are to be interpreted as described in [RFC2119]."

2 HTTP Configuration

2.1 Overview

This mechanism is based on HTTP(S) (Hyper-Text Transfer Protocol Secure) requests sent by a device to a Service Provider's Configuration Server in order to receive the configuration data.

The HTTP(S) configuration requests may be triggered in two different ways:

- Client-triggered HTTP(S) configuration if a Service Provider supporting this mechanism is detected by the client (e.g. SIM-based or by customization).
- Network-triggered HTTP(S) configuration if a Service Provider is not detected by the client. It is used to protect against negative charging impacts in networks that do not support this type of configuration.

Client behaviour is as follows:

- If client-triggered configuration applies: when a device boots up (or when the Subscriber Identity Module [SIM] is swapped without rebooting the device [hot swap]) and no valid configuration is available for the used identity, the device sends an initial HTTP request toward the Service Provider's Configuration Server to verify the current configuration settings' version.
    - If a non-embedded mobile client or a Personal Computer (PC) client without a SIM has no valid configuration for the used identity, this check should be performed each time the client is started.
- After receiving a Short Message Service (SMS) trigger as described in section 3, there is an HTTP request sent to the Service Provider's Configuration Server to verify the current configuration settings' version.
    - If the version available on the client does not match the version on the Configuration Server, the Configuration Server will include in its response to the client's HTTP request a configuration document in Extensible Markup Language (XML) format containing all configuration settings.

NOTE: The configuration document is covered in detail in section 4 and is based on the OMA Client Provisioning (OMA-CP) syntax (see [OMA CP Cont]).

- In situations where it is necessary to force a reconfiguration of a device (e.g. SIM card swap), the device resets the version value of its on-hand configuration settings to 0. The configuration provided by the Configuration Server shall therefore always provide a version value greater than 0.
- In scenarios where the Service Provider desires that for all functionality on a device/client that is subject to configuration, the device returns to its default state, the HTTP response provided by the Configuration Server will carry an XML configuration response that carries no configuration parameters and sets the version of the configuration settings to 0, -1 or -2. That default state will be service dependent and may be simply to disable the service. That will be defined in the service specific documents for each service supporting this mechanism.

2.2 Discovery of Configuration Server Addresses

This section defines the procedures for the client to discover addresses of Configuration Servers. The Service Provider shall be able to assign additional Configuration Servers in addition to the default Configuration Server. The client shall follow the procedures of the Service Provider Device Configuration independently for each assigned Configuration Server. The client shall store the configuration data and Configuration Server status information (e.g. configuration document version, user authentication data) in relation to each assigned Configuration Server.

2.2.1 Default Configuration Server

The client shall discover the address of the default Configuration Server using the E.212 network identification data (i.e. Mobile Country Code and Mobile Network Code) of the Service Provider serving the user.

The client shall compose the default Configuration Server's Fully Qualified Domain Name (FQDN) using one of:

- the Service Provider’s MCC (Mobile Country Code) and MNC (Mobile Network Code) as follows:

  config.rcs.mnc<MNC>.mcc<MCC>.pub.3gppnetwork.org

  NOTE: The Service Provider can use this FQDN as the default and re-direct to the service provider specific FQDN by using DNS redirection or additional Configuration Server specified in Section 2.3.2. The redirection procedure is implementation specific.

- a Service Provider specific FQDN which includes the same format as above but with the domain being specific to the Service Provider as follows:

  config.rcs.mnc<MNC>.mcc<MCC>.<SP domain>

  NOTE: This FQDN could be provided by the Service Provider to the client vendor

whereby <MNC> and <MCC> shall be replaced by the respective values of the home network in decimal format and with a 2-digit MNC padded out to 3 digits by inserting a 0 at the beginning (as defined in [GSMA PRD-IR.67]).

If the client uses the SIM for user identification either directly or indirectly, then the client shall derive the Service Provider's MCC and the MNC from the International Mobile Subscriber Identity (IMSI) of the SIM and compose the Configuration Server FQDN as defined above.

If a client is connected to the network of a Service Provider that does not support the Service Provider Device Configuration, then the Configuration Server FQDN will not be resolved in the Domain Name System (DNS) (NXDOMAIN). The client shall handle this case as defined for the case of no configuration data available for all services for which the client supports configuration via the Service Provider Device Configuration, as defined in the corresponding service documentation. In this case Network-triggered HTTP(S) configuration as described in section 2.1 applies.

2.2.2 Additional Configuration Server

The Service Provider shall be able to assign additional Configuration Servers in the configuration XML document via the SERVER characteristic in the ACCESS-CONTROL characteristic defined in section 4.2.

The client shall parse the ACCESS-CONTROL characteristic for additional Configuration Server information only in configuration XML documents received from the default Configuration Server. The client shall ignore an ACCESS-CONTROL characteristic received from other than the default Configuration Server.

If the client receives a configuration XML document from the default Configuration Server containing an ACCESS-CONTROL characteristic, then it shall inspect the SERVER characteristics and update the locally stored additional Configuration Server data as follows.

The client shall compare the locally stored data with the data received in the ACCESS-CONTROL characteristic on a per additional Configuration Server basis. An additional Configuration Server is uniquely identified by its FQDN provided in the fqdn parameter of a SERVER characteristic. The values of the FQDN from local storage (i.e. received in the previous response for client configuration from the default Configuration Server) and the fqdn value in SERVER characteristic match if there is a full string match, case insensitive.

The client shall first identify the additional Configuration Servers stored locally for which no SERVER characteristic is contained in the ACCESS-CONTROL characteristic. The client shall remove the configuration data and Configuration Server status information (e.g. configuration document version, user authentication data) from the local store for Configuration Servers being no longer present. The client shall apply for the removed services the procedures for removal of service configuration data as defined in the corresponding service documentation.

Then the client shall identify the additional Configuration Servers in the SERVER characteristic of the ACCESS-CONTROL characteristic which are not stored locally yet. The client shall store the Configuration Server data provided for those additional Configuration Servers locally. Then it shall use the value received in the fqdn parameter of the SERVER characteristic to invoke an initial client configuration request per additional Configuration Server in accordance with the procedures defined in this document. If multiple additional Configuration Servers are added, then the client shall check whether for one of the Configuration Servers the "id-provider" parameter is present in the SERVER characteristic. In this case, the client shall invoke the client configuration request to this Configuration Server first.

If a SERVER characteristic of the ACCESS-CONTROL characteristic contains an fqdn parameter value that is stored locally, then the client shall handle the additional data associated with the additional Configuration Server as defined in section 2.3.2.

If the FQDN associated with an additional Configuration Server does not resolve in DNS (NXDOMAIN), then the client shall handle the services identified by the app-id values associated with this Configuration Server as defined for the case of no configuration data available for the corresponding services. The client shall retry resolution of such a FQDN at the time of reboot or when the client starts next time.

If the resolution of the FQDN of an additional Configuration Server does not resolve in DNS for requests other than the initial service configuration request, then the client shall trigger a configuration request with the default Configuration Server to allow it to update the additional server configuration data.

2.3 Discovery of Supported Services

One or more Configuration Servers can be assigned by the Service Provider to manage client configuration data. The Service Provider shall be able provide permissions for Configuration Servers to manage client configuration via the ACCESS-CONTROL characteristic defined in section 4.2, based on the procedures defined in this section.

The client shall parse the ACCESS-CONTROL characteristic for additional Configuration Server information only in configuration XML documents received from the default Configuration Server. An ACCESS-CONTROL characteristic received from other than the default Configuration Server shall be ignored by the client.

The default Configuration Server provides a list of app-id values for authorised services

- for the default Configuration Server itself in the DEFAULT characteristic of the ACCESS-CONTROL characteristic. If the DEFAULT characteristic does not contain authorised app-id values, then the default Configuration Server does not provide any service configuration data itself but manages only the additional Configuration Server data, i.e. it acts as a redirect Configuration Server.
- for additional Configuration Servers via their SERVER characteristic of the ACCESS-CONTROL characteristic.

The client shall store and apply configuration data for a service only if the corresponding app-id value is authorised for this Configuration Server via the ACCESS-CONTROL characteristic. Parts of a configuration XML document received from a Configuration Server that is not authorised via the app-id value in the ACCESS-CONTROL characteristic for this Configuration Server shall be ignored by the client.

The client configuration data received from individual default or additional Configuration Server shall be stored and applied independent from client configuration data from other Configuration Servers.

If the ACCESS-CONTROL characteristic is absent in a configuration XML document for client configuration from a default Configuration Server, then the client shall apply the service configuration data with no further authorisation.

2.3.1 Default Configuration Server

If a client receives a configuration response for client configuration with an ACCESS-CONTROL characteristic present from the default Configuration Server

- without having received an ACCESS-CONTROL characteristic in the previous configuration response for client configuration (i.e. no service authorisation data is stored on the client)
- and the previous configuration response did contain client configuration data (i.e. there is already client configuration data stored on the client),

then the client shall remove the parts of the stored client configuration not matching with an app-id value contained in the DEFAULT characteristic of the ACCESS-CONTROL characteristic. The client shall apply for these services the procedures for removal of service configuration data as defined in the corresponding service documentation. For the parts of the stored client configuration matching with an app-id value contained in the DEFAULT characteristic of the ACCESS-CONTROL characteristic the client shall update the stored client configuration data with the data received from the configuration XML document from default Configuration Server in accordance with the definitions in this document and the corresponding service documentation. For example, the version handling of configuration XML document as defined in section 2.4.1 applies based on the previous version stored on the client.

If a client receives a configuration response for client configuration with an ACCESS-CONTROL characteristic absent from the default Configuration Server with having received an ACCESS-CONTROL characteristic in the previous configuration response for client configuration (i.e. service authorisation data is stored on the client), then the client shall update the stored client configuration data with the data received from the configuration XML document from default Configuration Server in accordance with the definitions in this document and the corresponding service documentation. For example, the version handling of configuration XML document as defined in section 2.4.1 applies based on the previous version stored on the client. Subsequently the client shall accept client configuration data from the default Configuration Server with no further authorisation.

If a client receives a configuration response for client configuration with an ACCESS-CONTROL characteristic present from the default Configuration Server with having received an ACCESS-CONTROL characteristic in the previous configuration response for client configuration (i.e. service authorisation data is stored on the client), then the client shall compare the list of app-id values stored locally on the client with the list of app-id values received in the DEFAULT characteristic. The client shall identify the app-id values from the local store that are not present in the DEFAULT characteristic. The client shall remove the client configuration data for these services from the local store and apply for these services the procedures for removal of service configuration data as defined in the corresponding service documentation.

2.3.2 Additional Configuration Server

If a client receives a configuration response for client configuration with an ACCESS-CONTROL characteristic containing a SERVER characteristic from the default Configuration Server with a fqdn parameter value which is stored already locally, then the client shall compare the list of app-id values stored locally on the client with the list of app-id values received in the SERVER characteristic.

The client shall first identify the app-id values from the local store that are not present in the SERVER characteristic. The client shall remove the client configuration data for these services from the local store and apply for these services the procedures for removal of service configuration data as defined in the corresponding service documentation.

Then the client shall identify app-id values in a SERVER characteristic which are not stored locally yet. The client shall store the values locally and trigger a client configuration request

to the corresponding Configuration Server identified by the fqdn value in accordance with the procedures defined in this document.

2.4 Client Configuration

The section defines the general principles of the Service Provider Device Configuration mechanism that are applicable for all access networks.

The Service Provider Device Configuration is based on a configuration request sent by the client to retrieve the configuration data. From a User Experience (UX) perspective, the user is not aware of the auto-configuration process unless the procedures for authentication and authorisation require actions by the end user.

To send an initial client configuration request to the Configuration Server, the client shall use the FQDN discovered in result of the procedures in section 2.2 and initiate a secured HTTP connection to it. If the certificate received from the Configuration Server has expired or is otherwise invalid or the connection is proposed to use a TLS version lower than version 1.2, the client shall not continue with the establishment of the connection and shall handle the configuration request as described in section 2.4.3 for the case where a 500 Internal Server error or any other HTTP error was received. Otherwise (i.e. when the certificate is valid), the client shall form a Request URI using the FQDN, without a path element and with a query string with request parameters encoded using the *application/x-www-form-urlencoded* format and send a HTTP GET Request.

NOTE: Given that for security reasons this client handling does not ensure a graceful fallback to a connection without TLS if the certificate is invalid, Service Providers should ensure that certificates are replaced before expiry.

If additional Configuration Servers are configured for the client, then the client shall initiate a new connection and negotiate the secure tunnel for each Configuration Server before initiating the configuration procedure. It is not reasonable for the client to assume that the default server and the additional server(s) share the same security certificate(s), since they may be provided by different authorities.

If additional Configuration Servers are configured for the client and simultaneous triggers for configuration requests to multiple Configuration Servers apply, then the client shall first invoke the request

- if to be contacted, to the default Configuration Server, otherwise
- if to be contacted, to the additional Configuration Server with the "id-provider" parameter set in the SERVER characteristic, otherwise
- to a randomly selected Configuration Server from the ones to be contacted.

If additional Configuration Servers are configured for the client and the client is triggered for a client configuration request to a given Configuration Server and there is a client configuration request in progress with another Configuration Server, then the client shall wait until the processing of the other client configuration request is finished. A client configuration request is considered finished if a final response is received from the Configuration Server without subsequent user interaction. If the Configuration Server invokes for a configuration request additional authentication or authorization procedures (e.g. via SMS_port zero policy, see section 2.6.2) or user messages (see section 2.4.2), then the processing of the

authentication or authorization, including a potential user interaction, is considered to be part of the processing of the configuration request.

The generic configuration request shall contain parameters as defined in Table 1.

Parameter	Description	Mandatory	Format
vers	Integer value indicating the version of configuration XML document currently stored on the client. The value shall be taken from the version parameter of the VERS characteristic in the previously received configuration XML document. If the client has no configuration XML document stored (e.g. non-existent, reset due to SIM change) then the value of the parameter shall be set to 0.	Y	Int (0 or a positive integer)
provisioning_version	String that identifies the version of the service provider device configuration supported by the client. It shall be set to "*5.0*" (without the quotes) for clients following this specification.	Y	String (4 max), Case-Sensitive
terminal_vendor	String that identifies the terminal OEM.	Y	String (4 max), Case-Sensitive
terminal_model	String that identifies the terminal model.	Y	String (10 max), Case-Sensitive
terminal_sw_version	String that identifies the terminal software version.	Y	String (20 max), Case-Sensitive
config_client_vendor	String that identifies the vendor providing the client process initiating the configuration process	N It shall be included if different from terminal_vendor	String (4 max), Case-Sensitive
config_client_version	String that identifies the version of the client process initiating the configuration request. client_version_value = Platform "-" VersionMajor "." VersionMinor Platform = Alphanumeric (9 max) VersionMajor = Number (2 char max) VersionMinor = Number (2 char max) Example: client_version=CFGAndrd-1.0	N It shall be included if not identified unambiguously by terminal_sw_version	String (20 max), Case-Sensitive
IMEI	If available, the subscriber's International Mobile Station Equipment Identity (IMEI) shall be sent as a parameter. Those Service	N	String (15 digits)

Parameter	Description	Mandatory	Format
	Providers that support a comprehensive device database can ignore the terminal_X parameters and use the IMEI instead, if it was available to the implementation.	if the OS platform allows it, it shall be included	
friendly_device_name	If provided by the user, a user friendly identification for the device may be passed along that can be used by the network when presenting the user with an overview of their devices. NOTE: this parameter needs to be included only if required for one of the services to be configured. In which case its mandatory character will be documented in the relevant service specific documents.	N, only to be provided if provided by the user	String (30 max before escaping), Case-Sensitive
app	String identifying one service supported by the client for configuration by means of its APPID for Application Characteristics (AC) or Management Object Identifier as assigned by the Open Mobile Naming Authority (OMNA). If the client supports multiple services, one "app" name/value pair per supported service shall be provided in the request. Example: If the client supports the services identified by the following APPID values: ap2204, urn:foo:mo:bar:1.0 then the "app" parameter is presented as follows in the request: app=ap2204&app=urn%3Afoo%3Amo%3Abar%3A1.0	Y	String multi-valued parameter
GID1	If available, the value of the Group Identifier 1 (GID1) SIM file content which are used for Service Provider personalisation.	N. if the SIM includes it, it shall be included	String representation of GID1 file contents - variable length
GID2	If available, the value of the Group Identifier 2 (GID2) SIM file content which are used for Service Provider personalisation.	N. if the SIM includes it, it shall be included	String representation of GID2 file contents - variable length

Table 1: Configuration request parameters

If the default Configuration Server has enabled additional Configuration Servers then configuration requests and responses shall be managed by the client on a per Configuration

Server basis. The parameters of the configuration request defined in Table 1 shall have the same values for all Configuration Servers, unless stated otherwise.

NOTE: For requirements regarding presence and values of service specific request parameters for services managed via the Serviced Provider Device Configuration, refer to the corresponding service documentation.

Please note that in case of Service Provider-specific clients, the terminal vendor, model and version parameters format and values should be agreed with the associated Service Provider prior to any device or client commercialization or update.

2.4.1 Configuration Server response

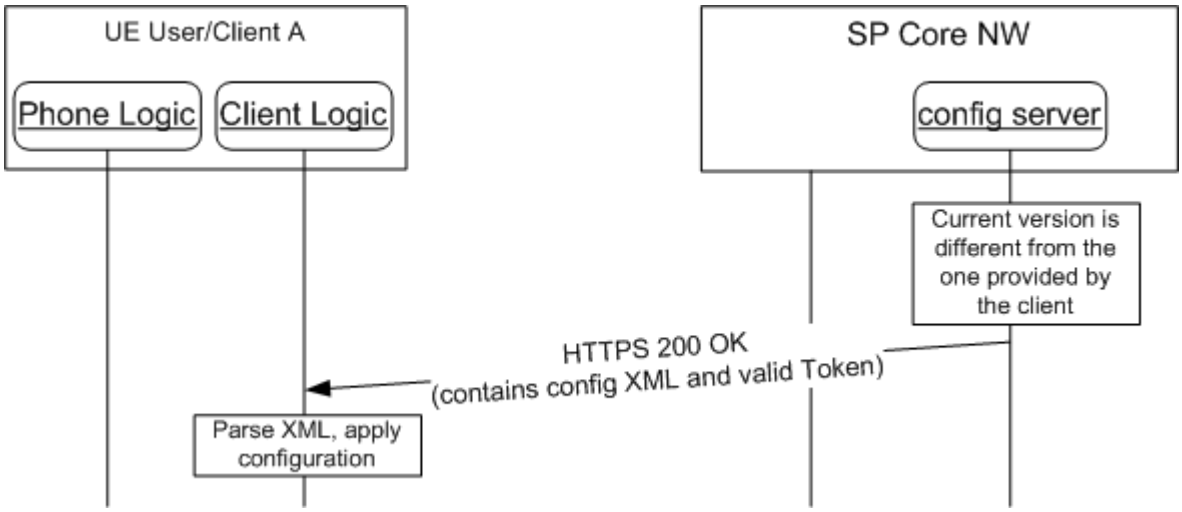

Figure 1: HTTP configuration: Server response

On reception of the HTTPS GET request the Configuration Server shall

1. first validate the client and terminal parameters. As part of this verification, the Configuration Server may challenge the client to provide proof of its authenticity and to confirm that it is running on a trust-worthy platform. If proof of client authenticity is required, then the Configuration Server shall invoke the procedure via the response defined in section 2.4.3. If the client specified support of client certificate upload by providing "client_certificate_upload" parameter (defined in section 2.12) in the request and the Configuration Server does not have this certificate yet, the Configuration Server, upon receiving this parameter, shall (re-)initiate the client certificate upload via the response defined in section 2.4.3.
2. authenticate the user. The selection of authentication method can either result from the client initiated access or client type specific configuration request or may be invoked by the Configuration Server via a Configuration Server response.

If the client is allowed to use the service and the user is authenticated, then the Configuration Server shall check if the value of the version parameter provided by the client matches the latest version of the configuration available on the Configuration Server.

If services need to be configured for the client and if the version does not match (i.e. a new configuration setting is required) or an operational procedure of the service provider requires resending of the existing configuration settings then the Configuration Server shall send a

HTTP 200 OK response with the content-type set to the value defined in section 4.1 and containing a configuration XML document in the format defined in section 4 containing the following characteristics and parameters:

- A VERS characteristic as defined in section 4.2 and as illustrated in Table 2 with the following parameters:
    - The version parameter shall be set to a positive integer value assigned to the configuration XML document in this response. If the default Configuration Server has enabled additional Configuration Servers then the version parameter value shall be stored per the Configuration Server.
    - The validity parameter as defined in section 4.2.
- Optionally, a TOKEN characteristic as defined in section 4.2 and as illustrated in Table 2.
- Optionally, an ACCESS-CONTROL characteristic as defined in section 4.2 and illustrated in Table 2. When receiving it, the client shall apply the procedures defined in section 2.2 and 2.3.
- Optionally, an USER characteristic as defined in section 4.2 and illustrated in Table 2.
- The full configuration XML document identified by the version value in the VERS characteristic. On reception, the client shall parse and apply the configuration document in accordance with the definitions of the configuration data of the applicable services.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
        <characteristic type="VERS">
                <parm name="version" value="X"/>
                <parm name="validity" value="Y"/>
        </characteristic>
        <characteristic type="TOKEN">
                <parm name="token" value="A"/>
        </characteristic>
        <characteristic type="ACCESS-CONTROL">
                <characteristic type="DEFAULT">
                        <parm name="app-id" value="ap0815"/>
                        <parm name="app-id" value="ap4711"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider1.com"/>
                        <parm name="app-id" value="ap1234"/>
                        <parm name="app-id" value="ap5678"/>
                        <parm name="id-provider" value="1"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider12.com"/>
                        <parm name="app-id" value="ap9876"/>
                        <parm name="app-id" value="ap4321"/>
                </characteristic>
        </characteristic>
        <characteristic type="USER">
                <parm name="msisdn" value="491711234567"/>
        </characteristic>

        -- full configuration XML document

</wap-provisioningdoc>
```

Table 2: HTTP configuration XML: new configuration document

If services need to be configured for the client and if the version matches (i.e. no new configuration settings required), then the Configuration Server shall send a HTTP 200 OK response with the content-type set to the value defined in section 4.1 and a configuration XML document in the format defined in section 4 containing the following characteristics and parameters:

- A VERS characteristic as defined in section 4.2 and as illustrated in Table 3 with the following parameters:
    - The version parameter shall be set to the same value as provided by the client in the HTTPS GET request
    - The validity parameter as defined in section 4.2.
- Optionally, a TOKEN characteristic as defined in section 4.2 and as illustrated in Table 3.
- Optionally, an ACCESS-CONTROL characteristic as defined in section 4.2 and illustrated in Table 3. When receiving it, the client shall apply the procedures defined in section 2.2 and 2.3.

- Optionally, an USER characteristic as defined in section 4.2 and illustrated in Table 3.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
        <characteristic type="VERS">
                <parm name="version" value="X"/>
                <parm name="validity" value="Y"/>
        </characteristic>
        <characteristic type="TOKEN">
                <parm name="token" value="Z"/>
        </characteristic>
        <characteristic type="ACCESS-CONTROL">
                <characteristic type="DEFAULT">
                        <parm name="app-id" value="ap0815"/>
                        <parm name="app-id" value="ap4711"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider1.com"/>
                        <parm name="app-id" value="ap1234"/>
                        <parm name="app-id" value="ap5678"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider12.com"/>
                        <parm name="app-id" value="ap9876"/>
                        <parm name="app-id" value="ap4321"/>
                </characteristic>
        </characteristic>
        <characteristic type="USER">
                <parm name="msisdn" value="491711234567"/>
        </characteristic>
</wap-provisioningdoc>
```

Table 3: HTTP configuration XML: no configuration changes required

If the Service Provider chooses to revert temporarily the configured functionality on the device/client to its default behaviour with the configuration query performed at boot/start-up enabled, then the Configuration Server shall send a HTTP 200 OK response with the content-type set to the value defined in section 4.1 and an XML document in the format defined in section 4 containing

- the VERS characteristic with version and validity, both set to 0 as illustrated in Table 4
- Optionally, an ACCESS-CONTROL characteristic as defined in section 4.2 and illustrated in Table 4. When receiving it, the client shall apply the procedures defined in section 2.2 and 2.3.
- Optionally, an USER characteristic as defined in section 4.2 and illustrated in Table 4.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1”>
        <characteristic type="VERS">
                <parm name=”version” value=”0”/>
                <parm name=”validity” value=”0”/>
        </characteristic>
        <characteristic type="ACCESS-CONTROL">
                <characteristic type="DEFAULT">
                        <parm name=”app-id” value=”ap0815”/>
                        <parm name=”app-id” value=”ap4711”/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name=”fqdn” value=”provider1.com”/>
                        <parm name=”app-id” value=”ap1234”/>
                        <parm name=”app-id” value=”ap5678”/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name=”fqdn” value=”provider12.com”/>
                        <parm name=”app-id” value=”ap9876”/>
                        <parm name=”app-id” value=”ap4321”/>
                        <parm name="id-provider" value="1"/>
                </characteristic>
        </characteristic>
        <characteristic type="USER">
                <parm name=”msisdn” value=”491711234567”/>
        </characteristic>
</wap-provisioningdoc>
```

Table 4: HTTP configuration XML: reset client

The Service Provider may choose to revert permanently the configured functionality on a device/client to default behaviour with the configuration query performed at boot/start-up disabled. In this case the Configuration Server shall send a HTTP 200 OK response with the content-type set to the value defined in section 4.1 and an XML document in the format defined in section 4 containing:

- the VERS characteristic with version and the validity, both set to -1 as illustrated in Table 5. If the default Configuration Server has enabled additional Configuration Servers then the version value shall be stored per the Configuration Server.
- Optionally, an ACCESS-CONTROL characteristic as defined in section 4.2 and illustrated in Table 5. When receiving it, the client shall apply the procedures defined in section 2.2 and 2.3.
- Optionally, an USER characteristic as defined in section 4.2 and illustrated in Table 5.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
        <characteristic type="VERS">
                <parm name="version" value="-1"/>
                <parm name="validity" value="-1"/>
        </characteristic>
        <characteristic type="ACCESS-CONTROL">
                <characteristic type=" DEFAULT">
                        <parm name="app-id" value="ap0815"/>
                        <parm name="app-id" value="ap4711"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider1.com"/>
                        <parm name="app-id" value="ap1234"/>
                        <parm name="app-id" value="ap5678"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn"
                        value="provider12.com"/>
                        <parm name="app-id" value="ap9876"/>
                        <parm name="app-id" value="ap4321"/>
                </characteristic>
        </characteristic>
        <characteristic type="USER">
                <parm name="msisdn" value="491711234567"/>
        </characteristic>
</wap-provisioningdoc>
```

Table 5: HTTP configuration XML: reset client and stop configuration query

If the Service Provider chooses to revert the functionality on a device/client to default behaviour (including the disabling configuration query performed at start-up) until there is a user action triggering a new query, then the Configuration Server shall send a HTTP 200 OK response with the content-type set to the value defined in section 4.1 and an XML document in the format defined in section 4 containing

- the VERS characteristic with version and the validity, both set to -2 as illustrated in Table 6. If the default Configuration Server has enabled additional Configuration Servers then the version parameter value shall be stored per the Configuration Server.
- Optionally, an ACCESS-CONTROL characteristic as defined in section 4.2 and illustrated in Table 6. When receiving it, the client shall apply the procedures defined in section 2.2 and 2.3.
- Optionally, an USER characteristic as defined in section 4.2 and illustrated in Table 6.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
        <characteristic type="VERS">
                <parm name="version" value="-2"/>
                <parm name="validity" value="-2"/>
        </characteristic>
        <characteristic type="ACCESS-CONTROL">
                <characteristic type="DEFAULT">
                        <parm name="app-id" value="ap0815"/>
                        <parm name="app-id" value="ap4711"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn" value="provider1.com"/>
                        <parm name="app-id" value="ap1234"/>
                        <parm name="app-id" value="ap5678"/>
                </characteristic>
                <characteristic type="SERVER">
                        <parm name="fqdn"
                        value="provider12.com"/>
                        <parm name="app-id" value="ap9876"/>
                        <parm name="app-id" value="ap4321"/>
                </characteristic>
        </characteristic>
        <characteristic type="USER">
                <parm name="msisdn" value="491711234567"/>
        </characteristic>
</wap-provisioningdoc>
```

Table 6: HTTP configuration XML: reset client until user input and stop configuration query

2.4.2 User Messages

Optionally (that is the tag may not be present), the XML configuration document may be used to convey a user message associated with the result of the Configuration Server response. The additional XML section is displayed in Table 7:

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1”>
        …
        <characteristic type="MSG">
                <parm name="title" value="Example"/>
                <parm name="message" value="Hello world"/>
                <parm name="Accept_btn" value="1"/>
                <parm name="Reject_btn" value="0"/>
        </characteristic>
        …
</wap-provisioningdoc>
```

Table 7: HTTP configuration: User notification/message sample

The meaning of the different parameters is described as follows:

- **Title**: The window title where the user message is displayed.
- **Message**: The message that is displayed to the user. Please note the message may contain references to HTTP addresses (websites) that need to be highlighted and converted into links by the device/client.

- **Accept_btn**: This indicates whether an “Accept” button is shown with the message on the device UI. The action associated with the Accept button on the device/client is to clear the message box.
  A value of 1 indicates that an “Accept” button has to be displayed.
  A value of 0 indicates that no “Accept” button has to be displayed.
- **Reject_btn**: This indicates whether the “Decline” button is shown with the message on the device UI. The action associated with the Reject button on the device/client side is to revert the configured services to their defined default behaviour.
  A value of 1 indicates that a “Decline” button has to be displayed.
  A value of 0 indicates that no “Decline” button has to be displayed.
  This parameter is optional, when not provided a default value of 0 shall be assumed.

NOTE: if a Reject_btn is not to be displayed (i.e. the corresponding parameter is set to 0 or is not included), the configuration shall be enabled regardless of whether the user actually presses the “Accept” button.

The *MSG* characteristic (i.e. the user message) is optional and will only be present for the following types of Configuration Server responses:

1. The response containing the full configuration settings.
2. The response disabling configuration on the device (version and validity are set to 0 or a negative value).

The device should display the message and the relevant/configured buttons in the following Configuration Server response scenarios:

- After receiving the full configuration settings, only if:
  - Working configuration was previously unavailable, including an unavailable working configuration following a SIM change; or
  - Following a terminal reset
- After receiving the disabling configuration response.

The device/client shall send language/locale settings to the Configuration Server to set the language/locale of the user message. The client should therefore include the HTTP *Accept-Language* header in all the requests and set the value of this header consistent with the device locale.

Figure 2: Configuration Server notification example

If the Service Provider has enabled additional Configuration Servers, then the client processing of user messages shall be applied for the services the Configuration Server has been authorised for. If user authorization is requested by a Configuration Server via "Accept" and/or "Decline" button, then the result of user action shall be applied only to the part of the client configuration for which the Configuration Server has been authorised by the Service Provider.

2.4.3 Response handling

In result of the processing of a configuration request, the client may receive additional responses not issuing a client configuration. In addition to results indicating a failure of the processing of the request for configuration data, there are additional responses requesting application of additional authentication mechanisms or the client authenticity verification. The corresponding client behaviour is defined in the sections describing the authentication mechanism or the client authenticity verification.

During the processing of additional authentication mechanisms or the client authenticity verification, the client may receive additional error responses related their application. These error responses are defined in the sections describing the additional mechanism.

The following responses are defined for the generic configuration request.

- If the Configuration Server returns a 200 OK response with a content-type set to the value as defined in section 2.11 for the client authenticity request, then the client shall invoke the procedure for client authenticity verification as defined in section 2.11. If the client specified support of client certificate upload by providing "client_certificate_upload" parameter defined in section 2.12 in the request, the client shall submit the client certificate together with the response from the client authenticity verification in the subsequent HTTP POST request.
- If the Configuration Server returns a 302 Found response, then the client shall invoke the procedures for authentication via OpenID Connect as defined in section 2.8.

- If the Configuration Server returns a 401 Unauthorized response with a WWW-Authenticate header containing a realm value with 3GPP-bootstrapping indication as defined in [3GPP TS 24.109], then the client shall invoke the digest authentication with the Configuration Server as defined in section 2.7. If no bootstrapped security association exists, the bootstrapping procedure is invoked first.
- If the Configuration Server returns a 403 Forbidden response, then the client shall remove the stored client configuration data from the local store. If the Service Provider has not enabled additional Configuration Servers, then the client shall apply the procedures defined for the case of no configuration data available for all services for which the client supports configuration via the Service Provider Device Configuration, as defined in the corresponding service documentation. Otherwise, the client shall apply these procedures for the services authorised by the Service Provider for the Configuration Server from which the error has been received. The client shall retry the configuration request at device boot or client start-up with the value of the "version" and "validity" parameters of the VERS characteristic both set to "0". The client shall perform in maximum 5 consecutive unsuccessful configuration requests. If the max number of retries is reached, the client shall stop retries.
- If the Configuration Server returns a 406 Not Acceptable response, then the client may inspect the value of the "Supported-Versions" HTTP header defined in section 5 for configuration request versions supported by the Configuration Server. Based on the Configuration Server's indication the client may retry the configuration request after it has adopted the request format and the behaviour to another version of the protocol. If the client does not apply for configuration using another version of the protocol, then the client shall remove the stored client configuration data from the local store. If the Service Provider has not enabled additional Configuration Servers, then the client shall apply the procedures defined for the case of no configuration data available for all services for which the client supports configuration via the Service Provider Device Configuration, as defined in the corresponding service documentation. Otherwise, the client shall apply these procedures for the services authorised by the Service Provider for the Configuration Server from which the error has been received.
- If the Configuration Server returns a 409 Conflict response, then the client shall invoke a user interaction to determine a new value for the friendly_device_name request parameter defined in Table 1. Once a new value for the configuration request parameter is available, the client shall send another configuration request containing the new value.
- If the Configuration Server returns a 503 response including a Retry-After header response, then the client shall retry the configuration request after the time specified in the “Retry-After” header.
- If the Configuration Server returns a 500 Internal Server error or any other HTTP error not listed above, then the client retry the configuration request on next reboot/the next time the client starts. If a valid configuration is available on the client, then the device/client should keep using it, even if the configuration has expired. The client shall retry the configuration request at device boot or client start-up. The client shall perform in maximum 5 consecutive unsuccessful configuration requests. If the maximum number of retries is reached, the client shall delete the locally stored configuration data. If the Service Provider has not enabled additional Configuration

Servers, then the client shall apply the procedures defined for the case of no configuration data available for all services for which the client supports configuration via the Service Provider Device Configuration, as defined in the corresponding service documentation. Otherwise, the client shall apply these procedures for the services authorised by the Service Provider for the Configuration Server from which the error has been received.

2.5 Configuration over cellular networks

2.5.1 Overview

This section describes configuration of a device carrying the SIM associated with the identity to be used for the configured services over cellular networks and other Service Provider controlled access networks that allow identifying the user's identity based on the access. . No specific handling shall be applied on such networks. A client shall always follow the procedure defined in section 2.6.

NOTE: Earlier versions of this specification included a specific procedure for authentication on cellular networks. This procedure has been deprecated now and is documented in Annex E for backwards compatibility.

2.5.2 Void

2.5.3 Void

2.6 Configuration request using SMS OTPs

This section defines a mechanism for user authentication based on an SMS exchange.

2.6.1 Non-cellular configuration

When performing the configuration, the client shall follow the SMS based configuration mechanism as detailed in this section.

If additional Configuration Servers are configured for the client and the client is triggered for a configuration request to a given Configuration Server and there is a client configuration request in progress with another Configuration Server then the client shall wait until the processing of the other client configuration request is finished. A client configuration request is considered finished if a final response is received from the Configuration Server. If the Configuration Server invokes for a configuration request additional authentication procedures (e.g. via SMS One Time Password [OTP], see section 2.6.2) or authorisation procedures (i.e. user messages, see section 2.4.2), then the processing of the authentication or authorization, including a potential user interaction, is considered to be part of the processing of the configuration request.

The non-cellular access configuration request follows the definitions of the configuration request defined in section 2.4. It is extended by additional parameters containing user identification data provided by the client. In addition, the non-cellular configuration supports a mechanism for verification of the identification data provided by the client using an OTP sent to the client or to the user via SMS.

The configuration request defined in Table 1 is extended for the use on non-cellular access by the parameters defined in Table 8. The client shall provide values for the parameters token, MSISDN and IMSI as defined below.

Depending on the client capability to access SIM data (e.g. the user's IMSI) two situations exist:

1. The client is not able to retrieve the IMSI of the SIM:

   The IMSI parameter shall always be omitted from requests. The following use cases apply for the determination of MSISDN and token.

   - If the request is not caused by a previous Configuration Server response containing a cookie and connectivity for receiving SMS is available and the client

     - has not stored a token and
     - has not stored a MSISDN,

     The client shall follow the procedure described in section 2.6.1.1. If the MO SMS Procedure described in section 2.6.1.1 would fail, the client shall prompt the user to provide a MSISDN in E.164 format.
     In this case, the value of the MSISDN parameter shall take the number entered by the user. The value of the token parameter shall be left empty as defined in Table 8.
     If connectivity for receiving SMS is not available, the client shall wait for SMS connectivity prior to initiating the configuration procedure.
   - If the request is not caused by a previous Configuration Server response containing a cookie and connectivity for receiving SMS is available and the client

     - has not stored a token and
     - has stored a MSISDN

       - derived from the msisdn parameter defined in section 4.2 or
       - if no msisdn parameter is available, derived from a previous user input, including user input from client configuration request,

     then the client may prompt the user to enter a MSISDN in E.164 format with the stored value as recommendation or may use the MSISDN without user interaction.
     In this case, the value of the MSISDN parameter shall take the value discovered by the client. The value of the token parameter shall be left empty.
     If connectivity for receiving SMS is not available, the client shall wait for SMS connectivity prior to initiating the configuration procedure.
   - If the client has not stored a token and the configuration request is caused by a previous Configuration Server response containing a cookie, the client shall not prompt the user to enter the MSISDN and set the parameters as follows:

     - the token parameter shall be left empty
     - the MSISDN parameter shall be set to the value

       - derived from the msisdn parameter defined in section 4.2 or

- if no msisdn parameter is available, derived from a previous user input, including user input from client configuration request.

It shall be omitted if none of these sources applies.

- If the client has stored a token, it shall use it to set the value of the token parameter. The MSISDN parameter shall be set to the value of the MSISDN stored with the token being either derived from the msisdn parameter defined in section 4.2 or from previous user input or shall be omitted if these sources do not apply.

2. The client is able to retrieve the IMSI of the SIM:

   The IMSI parameter shall be set in the requests to the IMSI value derived from the SIM. The following use cases apply for the determination of MSISDN and token:

   - If the request is not caused by a previous configuration response containing a cookie and connectivity for receiving SMS is available and the client

     - has not stored a token,

     then the client shall set the value of the MSISDN parameter to the MSISDN derived from the msisdn parameter defined in section 4.2 or it shall omit the parameter from the request. If the client has not stored a token, it shall leave the value of the token parameter empty as defined in Table 8.
     If connectivity for receiving SMS is not available, the client shall wait for SMS connectivity prior to initiating the configuration procedure.

   If the provisioning request returns a HTTP 200 response containing a cookie (Set-Cookie header) and a body containing an XML document that has an MO_CONFIGURATION characteristic as described in section 4.2 (an example of which can be found in Table 10) then the client shall follow the procedure described in section 2.6.1.1 starting from step 3.

   - If the provisioning request is caused by the Configuration Server response with status 403 FORBIDDEN as defined in section 2.6.3, the client shall prompt the user to enter the MSISDN. In this case, the MSISDN value shall be taken from the user input and may be the source of the MSISDN parameter values in subsequent requests.
   - If the client has stored a token, it shall use it to set the value of the token parameter as defined in Table 8.

   The client behaviour to supply the identification parameters in the request is the same regardless whether it is sent in result of a previous configuration response containing a cookie or not.

In addition to the user identification data the client shall indicate via the SMS_port parameter defined in Table 8 whether it supports the OTP handling in the background (non-zero value in SMS_port) or not.

Parameter	Description	Mandatory	Format
IMSI (International Mobile Subscriber Identity)	If available, the subscriber's IMSI shall be sent as a parameter.	N if the OS platform allows it, it shall be included	String (15 digits)
msisdn	MSISDN, in E.164 format, of the primary SIM that is used to derive the user's main identity. It shall be present if the request is sent in result of a user prompt to enter the MSISDN. In all other cases it shall be present if the client is able to discover a value from the client configuration or if the client has stored the value of a previous user input which caused a successful configuration response.	N	E.164 (+447900000001) in international format NOTE: In case that msisdn comes with a plus sign, the client shall provide the msisdn value with the plus sign encoded as per [RFC3986] section 2.1.
SMS_port	This parameter indicates support of application port addressing via the SMS User Data Header (UDH) for the validation of user identification data through an OTP. If set to 0, the client indicates to the Configuration Server that the client does not support application port addressing via SMS UDH. A standard SMS (user visible) shall be used instead. If set to a positive integer value, it indicates the port on which the client will listen to receive an SMS with an OTP sent via application port addressing in the UDH. If not set, the default port value shall be used, i.e. port 37273.	N	Int (0-65355)

Parameter	Description	Mandatory	Format
token	If the client has not stored a token (e.g. it is the first time the device requests configuration), the parameter shall contain an empty string. Otherwise, it shall contain the token value received in the token parameter of the TOKEN characteristic of the previous configuration response. If the Service Provider has enabled additional Configuration Servers then the client shall manage the token value per Configuration Server. The value of the "token" parameter sent in a request to a Configuration Server shall be derived from the previous response of this Configuration Server.	Y	String

Table 8: Additional configuration request parameters for primary devices over non-3GPP access

If the default Configuration Server has enabled additional Configuration Servers then configuration requests and responses shall be managed by the client on a per Configuration Server basis. The parameters of the configuration request defined in Table 8 shall have the same values for all Configuration Servers, unless stated otherwise.

NOTE: For requirements regarding presence and values of service specific request parameters for services managed via the Serviced Provider Device Configuration, refer to the corresponding service documentation.

On reception of the HTTPS GET request, the Configuration Server shall first validate the client and terminal parameters. As part of this verification, the Configuration Server may challenge the client to provide proof of its authenticity and to confirm that it is running on a trust-worthy platform. The procedure to do so is described in section 2.11.

NOTE: This means that procedure in section 2.11 can be invoked for already configured clients when they verify the status of their current configuration.

If accepting the client as allowed to use the service (either directly or after explicit verification as described in section 2.11) the Configuration Server shall initiate the subsequent client and Configuration Server procedures for user identification over non-PS access that are defined as follows:

1. If the token value is empty in the request and the network allows configuring of devices using this mechanism, then the Configuration Server responds with an HTTP 200 OK response that includes a new cookie (Set-Cookie header) request and optionally a configuration xml document containing a POLICY characteristic with a SMS_port zero policy parameter as defined in sections 2.6.2, if the initial configuration request did contain a SMS_port parameter with a non-zero value.

    a) Following the request, an SMS message shall be sent to the primary device, i.e. the device using the SIM associated with the MSISDN or IMSI sent in the

HTTP request. This SMS message will contain an OTP. The format of this SMS is covered in detail in section 2.6.2.

NOTE: the Configuration Server provider may implement mechanisms on the Configuration Server to protect it from suspicious or potentially malicious transactions (e.g. a client causing too many SMS messages)

b) If the client has sent in the configuration request a SMS_port parameter with value "0" or if the subsequent HTTP 200 OK response did contain a POLICY characteristic with a SMS_port zero policy parameter as defined in sections 2.6.2 then, the client shall prompt the user to enter the OTP. The prompt should request the user to enter manually the value that will be received via SMS.

c) If the user has entered the OTP after being requested as defined in step a) or if the client has received the SMS with the OTP in the format defined in step 1 in section 2.6.2 for all other cases, the client shall send a second HTTPS GET request using the following parameters in the GET request

Parameter	Description	Mandatory	Format
OTP	This is the password received on the primary device using the SIM associated with the provided MSISDN/IMSI	Y	String

Table 9: HTTP configuration for primary devices: Second and final HTTPS request GET parameters

In addition, the second HTTPS GET request shall include the cookie obtained in the previous HTTP 200 OK response at the start of this procedure so that the Configuration Server is able to correlate the initial and subsequent HTTPS requests.
If the user has been requested to enter the OTP as defined in step ii and the user aborts it, then the client may provide a mechanism to prompt the user to start the client configuration procedure from the beginning (e.g. after a timeout period) including the request to enter the MSISDN if one was requested before. This may cover scenarios where the user is not able to receive the OTP via SMS, e.g. due to missing network connection. If the user is not able to authenticate itself for a time determined by the client, the client shall remove the client configuration from the device and apply the default behaviour. It may inform the user about the consequences of the abortion.

In the success case, the subsequent procedure is identical to the one described in sections 2.4.1 and 2.4.3.

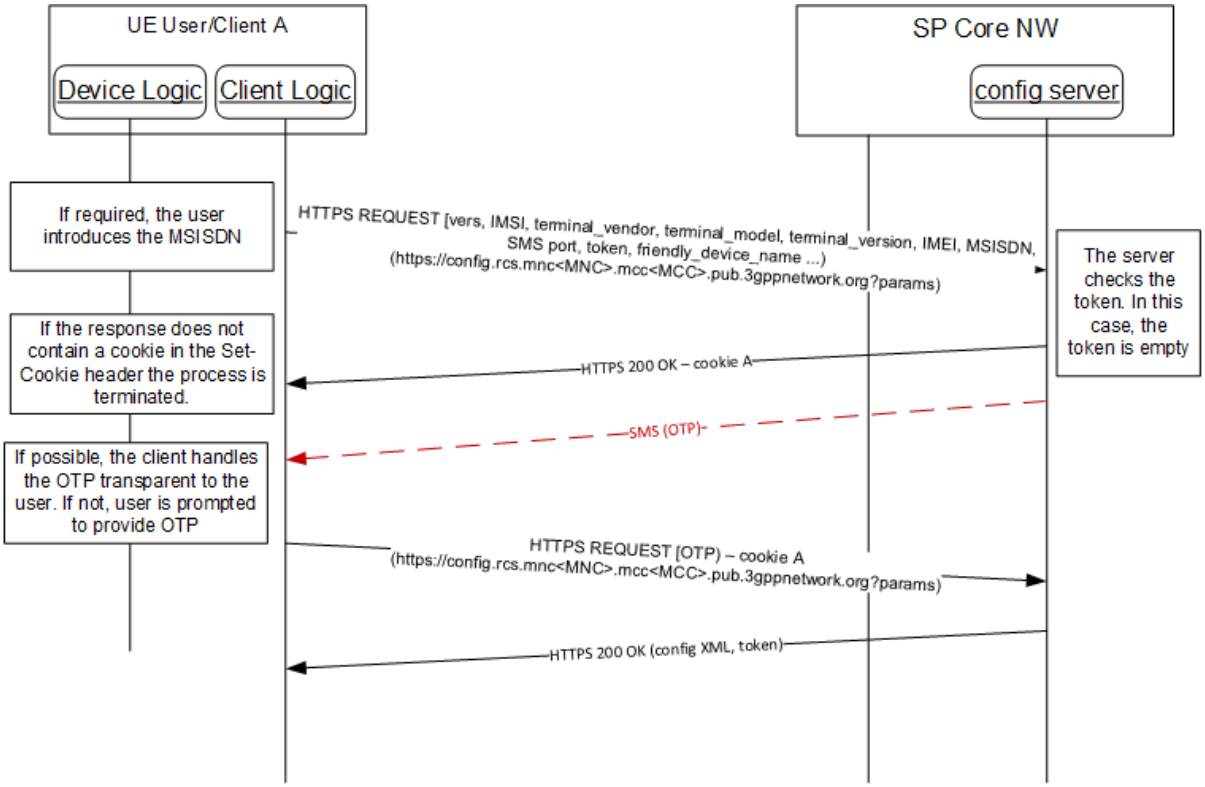

Figure 3: HTTP configuration for primary devices over non-PS access with MSISDN: empty token

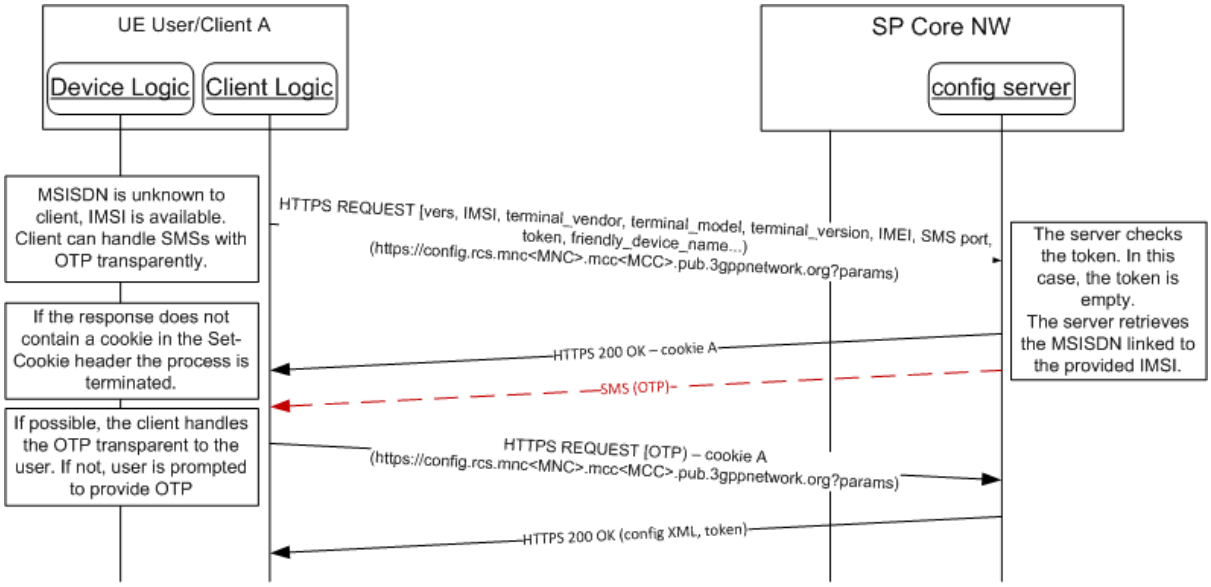

Figure 4: HTTP configuration for primary devices over non-PS access with only IMSI: empty token

If, in exception to the definitions above, the Configuration Server HTTP 200 OK response did not contain a SetCookie header, then the client shall stop processing of the request, delete locally stored configuration data and shall apply the default behaviour for services configured by the Configuration Server.

2. If a token value was sent in the request and the token is valid (i.e. the user is identified) and the token privileges the Configuration Server to issue a client configuration, then from this point the procedure is identical to the one described in sections 2.4.1 and 2.4.3. That is, the Configuration Server returns a 200 OK response with a configuration XML document.

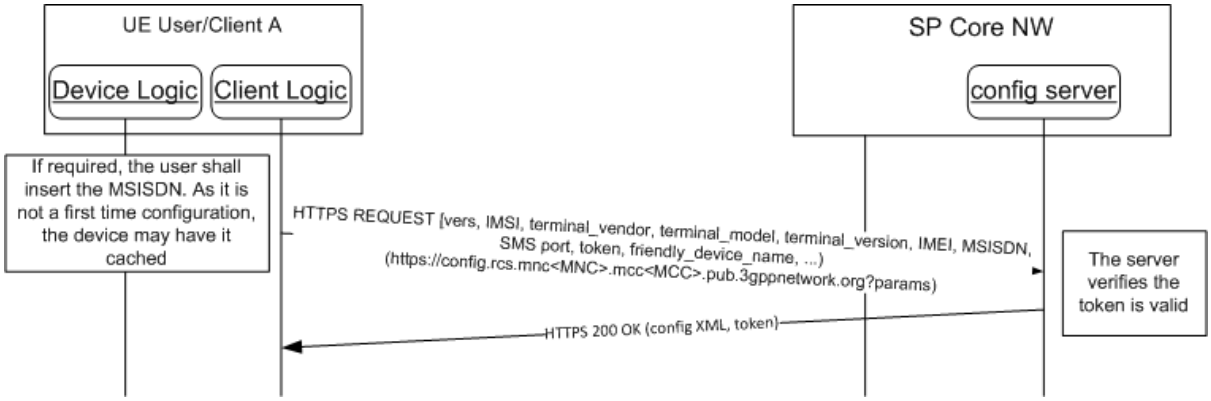

Figure 5: HTTP configuration for primary devices over non-PS access: Valid token

3. If a token value was sent in the request and the token is valid (i.e. the user is identified) but the token does not privilege the Configuration Server to issue a client configuration, then the Configuration Server responds with an HTTP 200 OK response same as defined for the case 1 in this section. In this case, the client shall delete the locally stored token and shall follow the procedure defined in case 1.

For the definition of additional use cases for the token-based user identification, refer to section 2.6.3.

2.6.1.1 Mobile Originated SMS Procedure

This mechanism applies for cases where the client might or might not have access to the IMSI or the network cannot resolve the IMSI. It is used to determine the client's MSISDN and provision a token. If invoked because the client has access to neither the IMSI nor the MSISDN as defined in section 2.6.1,

1. the client shall send the initial HTTPS request without the IMSI or msisdn parameters defined in Table 8.
2. If the response contains a cookie (Set-Cookie header), a Retry-After header and a body containing an XML document that has an MO_CONFIGURATION characteristic as described in section 4.2 (example can be found in Table 10), then the client shall continue with the procedure. Otherwise, the client shall consider the procedure as failed.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
   <characteristic type="MO_CONFIGURATION">
      <parm name="sms_message_content"
         value="Verifying your number for RCS service 1092309aazxclkjljoiear"/>
      <parm name="destination_phone_number" value="+12345678901"/>
      <parm name="sms_visibility" value=”1”/>
      <parm name=”sms_wait_interval_ms” value=”30000”/>
   </characteristic>
</wap-provisioningdoc>
```

Table 10: HTTPS configuration XML Example for Mobile Originated (MO) SMS

The meaning of the different parameters is described as follows:

- sms_message_content: The payload of the SMS to be sent automatically (contains an OTP).
- destination_phone_number: Destination phone number to send the MO SMS to.
- sms_visiblity: Indicates if the sms should be visible to the user. The values are “0” for not visible and “1” for visible.
- sms_wait_interval_ms: Time in milliseconds the client shall wait after sending the automatic SMS until the client sends the next HTTP provisioning request.

3. The client shall automatically send the SMS to the service provider specific destination MSISDN obtained from the destination_phone_number parameter in the MO_CONFIGURATION characteristic and use the value of the sms_message_content parameter in the MO_CONFIGURATION as the body of the SMS (encoded in UCS2).
4. If the sms_visiblity parameter in the MO_CONFIGURATION characteristic was set to “0”, the client shall not display the SMS to the user.
5. After sending the automatic SMS, the subsequent HTTP provisioning request shall contain all the same parameters as the original request, together with the sms_message_content parameter (see Figure 6) with the value of the {sms_message_content} encoded in web safe base64 (see Table 11) and the cookie received in the previous response (cookie header). This request shall be sent after the value of the sms_wait_interval_ms parameter is reached.

Parameter	Description	Mandatory	Format
sms_message_ content	The content of the value of the sms_message_content field of the XML configuration for MO SMS encoded in web safe base64.	Y	String

Table 11: Additional configuration request parameters for primary devices over non-3GPP access in MO SMS procedure

The Configuration Server, upon successfully matching the OTP token in the request with the OTP SMS that originated from the client with the identified MSISDN, shall returns the configuration.

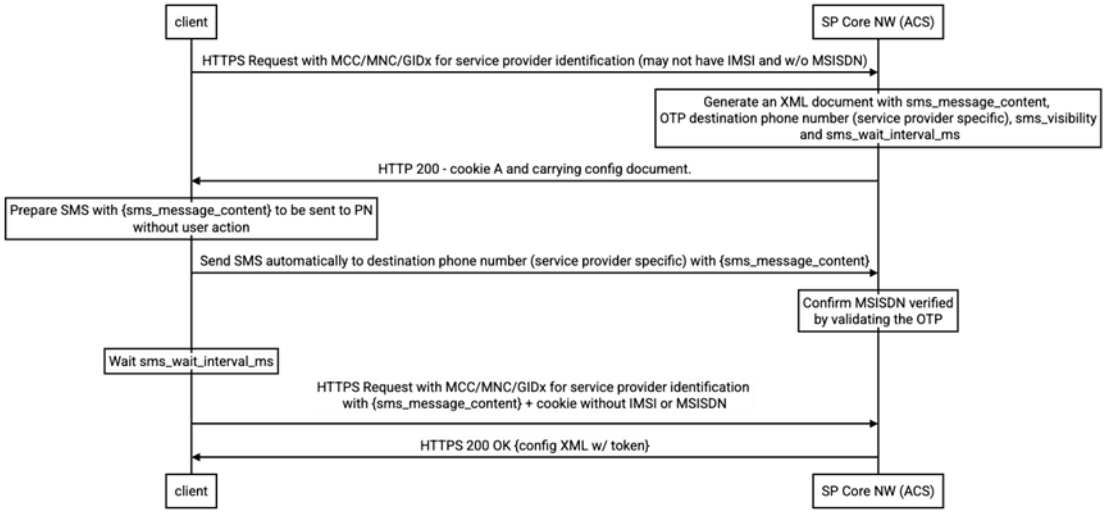

Figure 6: HTTP configuration for primary devices over non-PS access using MO SMS without IMSI or MSISDN: empty token

2.6.2 SMS format to receive the OTP value

In case of a primary device configuration, the device receiving the SMS containing the OTP password shall match the device where the client is running, the preferred approach is that the SMS is sent in a format that allows the client to intercept the OTP in a transparent manner. In order to do so, the Configuration Server shall perform the following steps:

1. If the client has indicated in the configuration request via the SMS_port parameter defined in section 2.6.1 support of the procedure to receive the OTP in the SMS message via application port addressing as defined in [3GPP TS 23.040] and the Configuration Server supports application port addressing, then the following SMS format shall be used:

   - DataCodingScheme = 00 (GSM 7 bit default alphabet) or 08 (UCS2)
   - UserDataHeaderIndicator = 1
   - UserDataHeader with

     - UDHL=06
     - IEI=05
     - IEIL=04
     - Destination port: port provided by the client in HTTP request encoded in hex. If not provided, 37273 (0x9199) shall be the default value.
     - Source Port: 0x0000 (0 in decimal)

   - Content of the message shall be the OTP encoded in UCS2

   If the device receives the SMS message in this format, it shall be routed to the client listening to the port. The client shall process the Configuration Server request transparently to the user.

2. If client has indicated in the configuration request via the SMS_port parameter defined in section 2.6.1 that it does not support the procedure to receive the OTP in the SMS message via application port addressing as defined in [3GPP TS 23.040], then the

Configuration Server shall send a standard SMS and the user shall be prompted by the client to manually provide the OTP code to the client (e.g. via a text box).

Where the Service Provider wants to send a standard SMS for the OTP code, the SMS_port parameter shall be included in the HTTPS 200 OK response sent by the Configuration Server just before the SMS that carries the OTP code (see HTTP flows presented in figures 9 and 10). The response shall carry an XML document containing the SMS_port parameter set to 0 as illustrated in Table 12:

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="POLICY">
        <parm name="SMS_port" value="0"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 12: HTTPS configuration XML: SMS_port zero policy

In this case, the OTP handling that is transparent to the user is not possible and the client prompts the user to enter the OTP that is received via SMS.

NOTE: The Service Provider should allow enough time prior to sending the SMS with the OTP code to make sure that the client has received the HTTPS 200 OK response that carries the XML with the SMS_port parameter set to zero. This response shall be sent always considering the provisioning_version parameter and thus ensuring backward compatibility. In case that the Service Provider sets a different value to the SMS_port parameter, this value shall be ignored by the client.

2.6.3 Use cases review

The error conditions and use case scenarios covered in section 2.4.3 also apply for configuration over non-PS access, but in this case, any disabling of the client shall be limited to that specific non-PS network. Further configuration attempts shall thus be done when the device connects to a cellular or another non-PS network. In addition to those errors, for the process of performing a configuration over non-PS access networks the following specific error conditions shall be taken into account and supported:

1. In the scenario where the user was prompted to provide an MSISDN, the given MSISDN may be invalid or unauthorized to retrieve the Service configuration. As a result, the initial configuration request shall be answered by the Configuration Server with an HTTP 403 FORBIDDEN error response. In this case, the client may provide the user to retry client configuration by entering the MSISDN again. If the user identification continues to fail for a number of times determined by the client, the client shall remove an existing client configuration from the device and apply the default behaviour.

   If the user aborts the identification procedure, the client shall remove an existing client configuration from the device and apply the default behaviour. The client may inform the user about the consequences of the abortion.

2. In the case where the initial configuration request did contain only the IMSI parameter for user identification (see section 2.6.1), the Configuration Server can request the

client to prompt the user for to enter a MSISDN. This is likely if the network does not support user identification based on IMSI. In that case, the Configuration Server shall answer to the initial configuration request with a HTTP 403 FORBIDDEN response. In this case, the client shall not remove an existing client configuration, request the user for input of the MSISDN and perform the procedure including the MSISDN. This is shown in the flow in Figure 7:

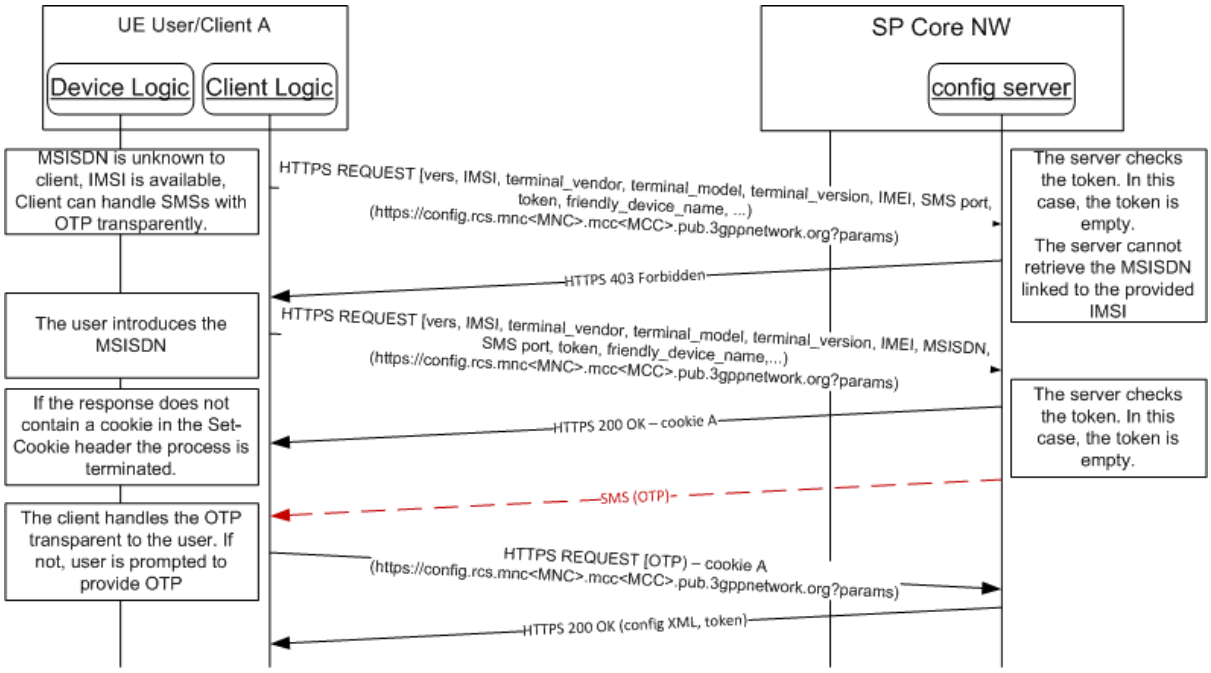

Figure 7: HTTP configuration for primary devices over non-PS access with only IMSI: Not supported by the network

3. If the client has sent a HTTP configuration as defined in step 1.c) of section 2.6.1 and The OTP password is invalid, then the Configuration Server shall reply with an HTTP 511 NETWORK AUTHENTICATION REQUIRED error response.

   If the OTP was entered by the user caused by the client supporting SMS_port value "0" only (see section 2.6.1) or by the service provider applying SMS_port zero policy (see section 2.6.2) then the client should offer the user to retry entering the OTP value. If the retry fails a number of times or if the user aborts the procedure the client shall remove an existing client configuration from the device and apply the default behaviour, see also step 1.c) in section 2.6.1.
   In all other cases, the client shall remove an existing client configuration immediately after receiving the first HTTP 511 NETWORK AUTHENTICATION REQUIRED error response and apply default behaviour.

4. If the token in the configuration request is invalid, the Configuration Server replies with an HTTP 511 NETWORK AUTHENTICATION REQUIRED error response. The client shall remove the previously stored token and re-start the client configuration as defined for cases without a stored token.

2.6.4 Security considerations

For terminals carrying the SIM associated to the user's main identity where the connection is carried out over the PS access network, the current design reduces the risk of a man-in-the-middle attack whereby a third party is able to impersonate the Configuration Server.

To secure interoperability between Service Providers and to reduce complexity on the device/client, the Configuration Server shall make use of public root certificates issued by a recognized Certification Authority (CA). That is, the root certificates are similar to those used by standard webservers that are widely recognized by browsers and web-runtime implementations both in PCs and in devices.In addition, as a Service Provider Option, the Configuration Server is able to enforce a policy for the OTP challenge on primary devices being always visible to the user, especially for the case where the client would be able to apply it transparently by use of the SMS UDH procedure. If the client receives a Configuration Response in HTTP 200 OK with a "SMS port zero" policy (see section 2.6.2) then it shall expect the reception of the OTP via user visible SMS. Thus, it shall prompt the user to enter the OTP and continue processing with the user input only.

If the SMS with the OTP is visible to the user, the user might be subject to malicious parties attempting to obtain the OTP through social engineering. SMS messages are recommended to warn against that.

2.7 HTTP(S) based client configuration mechanism with GBA Authentication

2.7.1 Overview

The General Bootstrapping Architecture (GBA) defined in [3GPP TS 33.220] provides mechanisms for AKA based user authentication using the 3GPP Authentication Centre (AuC) and the USIM or ISIM. The HTTP(s) based client configuration mechanism supports the authentication of primary devices via GBA.

NOTE: To reduce the number of SIM-based authentication solutions provided in this specification (e.g. 2.8.3.2 or 2.13), the support for GBA may be deprecated in a future version of this specification.

The Authentication Procedure consists of two parts. The basis for the user authentication between the device and network applications is a bootstrapped security association. The association provides the client with a Bootstrapping Transaction Identifier (B-TID) and key material that can be used by clients of the device to authenticate the user with network applications. In non-3GPP access, only the configuration request procedure for clients with access to SIM data defined in section 2.6.1 applies.

An application client will use the B-TID and the key material for the authentication with a specific network application. For the Service Provider Device Configuration HTTP Digest Authentication is used.

If the Service Provider's has enabled additional Configuration Servers then the set of B-TID and the key material is used to generate keys for all Configuration Servers supporting GBA authentication.

The end-to-end implementation of GBA is defined in [3GPP TS 33.220]. The protocol extension of the client configuration protocol shall be implemented according to [3GPP TS 24.109].

2.7.2 Use Case review

2.7.2.1 HTTP Digest Authentication

Precondition for the use of this procedure for authentication within the HTTP(s) based client configuration is the support of GBA as defined in [3GPP TS 33.220] on the device and in the Service Provider network or in additional Configuration Servers.

The GBA Authentication can be applied independent from the access network type (3GPP or non-3GPP). However, a client supporting GBA based HTTP digest authentication shall invoke the client configuration mechanisms in accordance with the access network type, i.e. in 3GPP access as defined in section 2.5, in non-3GPP access as defined in section 2.6.

When sending the secured configuration request the client supporting GBA shall indicate it by the addition of a GBA product token in the User-Agent header as defined in [3GPP TS 24.109].

If the Service Provider's Configuration Server or an additional Configuration Server does not support GBA based HTTP digest authentication, it returns responses as defined in sections 2.5 and 2.6 respectively. Device configuration commences as defined at the same place.

If the Service Provider's Configuration Server or an additional Configuration Server supports GBA based authentication then after verifying the client authenticity using the procedure described in section 2.11 if considered relevant, it returns an HTTP 401 Authorization Required response with a WWW-Authenticate header instructing the client to use HTTP digest Authentication with a bootstrapped security association.

If the client has no bootstrapped security association in place, it shall invoke the bootstrapping procedure defined in section 2.7.2.2 to generate it.

If the client has a bootstrapped security association in place, it shall use the stored key material and the B-TID to generate keys specific to the Configuration Server as defined in [3GPP TS 33.220]. With the key material and the B-TID, it shall generate the Authorization header to be sent in a new secured request for client configuration.

The Configuration Server will fetch the B-TID and key material from the Bootstrapping Server Function (BSF) and complete the digest authentication. If successful, a 200 OK response containing an Authentication-Info header and the configuration XML will be returned to the client. The Configuration Server may request the client to renegotiate the bootstrapped security association (e.g. due to expiry) by returning a 401 "Unauthorized" response. When the client receives the 401 "Unauthorized" response, it shall renegotiate the bootstrapped security association with the procedure defined in section 2.7.2.2.

The client shall validate the Authentication-Info header information and apply the configuration XML.

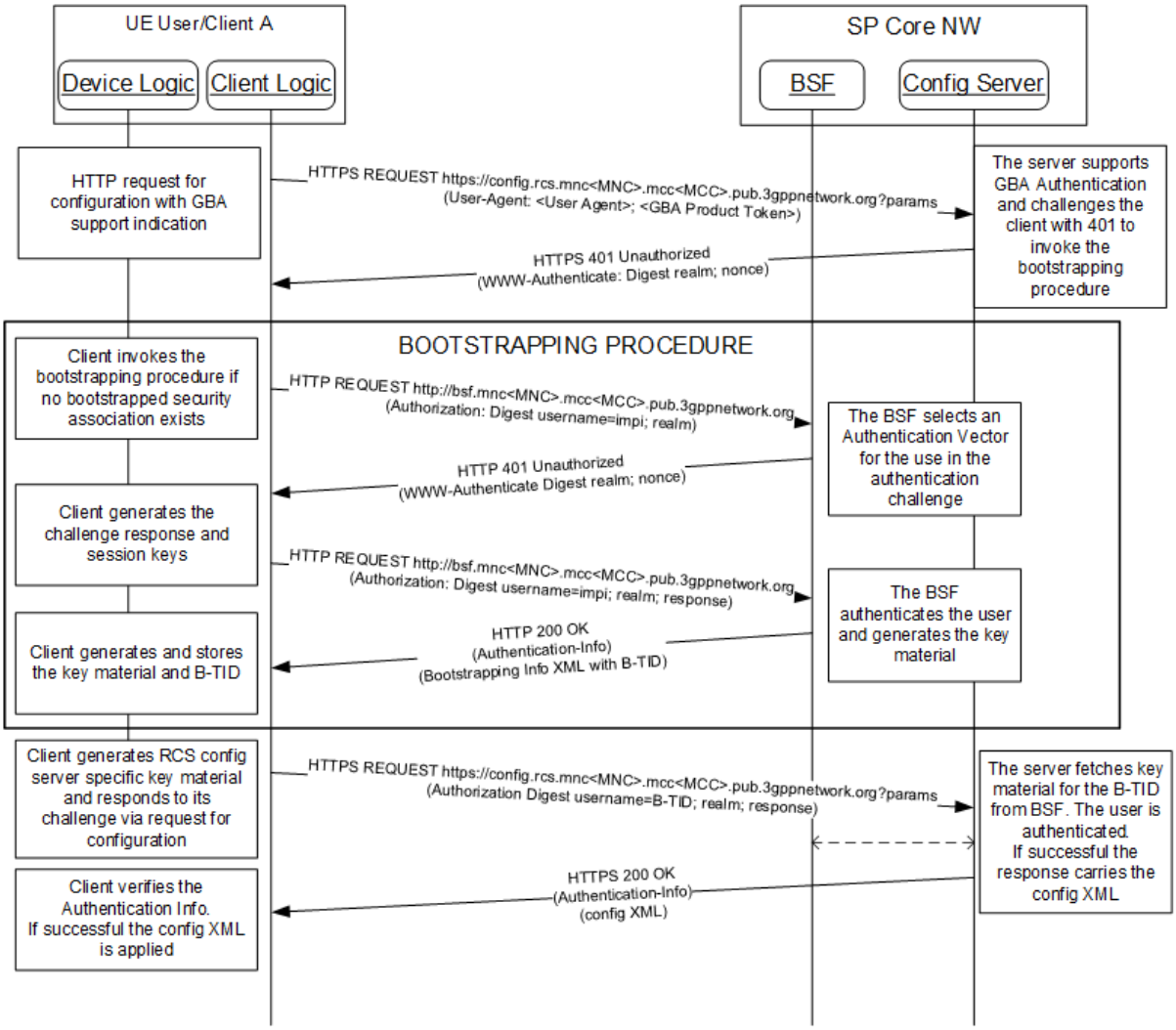

Figure 8: HTTP configuration for primary devices using GBA

2.7.2.2 Bootstrapping Procedure

The device will invoke the bootstrapping procedure with the service provider's BSF to generate a bootstrapped security association as defined in [3GPP TS 24.109]. This section provides an informative overview of the procedure.

The URI of the BSF is derived by the client from the IMSI or the user's private identity (IMPI) as defined in [3GPP TS 23.003]. The client creates an HTTP GET request with an authorization header as defined in [3GPP TS 24.109] and sends it to the BSF. The authorization header contains the user's private identity (IMPI) as username. If the device is not able to get the user's private identity (IMPI) from the SIM, it shall be constructed from the IMSI as defined in [3GPP TS 23.003].

On receipt of the request for authentication, the BSF retrieves and selects an Authentication Vector for use in the authorisation challenge [3GPP TS 33.220]. It returns a HTTP 401 Authorization Required response to the client with a WWW-Authenticate header instructing the client to authenticate itself.

The client runs the AKA algorithm [RFC4169] to calculate the challenge response which is sent back to the BSF in the Authorization header of a subsequent HTTP GET request. It also calculates the session keys.

On reception of the GET request, the BSF authenticates the user based on the challenge response received from the client. It generates the B-TID for the IMPI and stores the session keys. It informs the client about the success of the authentication in the Authentication-Info header of the 200 OK response. The response contains also the bootstrapping XML with the B-TID. The client generates the key material and stores it for subsequent authorisations.

If the default Configuration Server has enabled additional Configuration Servers then the B-TID and the key material provided by the Service Provider's BSF is applicable for all Configuration Servers supporting GBA authentication and being authorised by the default Configuration Server.

2.8 Support of OpenID Connect

2.8.1 Overview

The mechanisms defined in this section extends the Service Provider Device Configuration to support user authentication based on the OpenID Foundation OpenID Connect standard [OpenID Connect]. This allows Service Providers to evolve from custom authentication mechanisms defined in this document towards an open standard identity service as defined in [GSMA PRD-PDATA.01].

OpenID Connect provides a generic solution for user authentication independent from the type of device and the access network. It encompasses authentication, authorization and consent management features for devices using the identity of a SIM, additional devices sharing a user identity and devices using a dedicated identity.

2.8.2 OpenID Connect Authentication Flow

A Configuration Server wishing to authenticate a client through Open ID Connect shall return a HTTP 302 Response to the client, as defined in section 2.4.3, redirecting the client to the OpenID Connect authorisation endpoint.

The client shall support the procedures of the OpenID Connect Authentication Flow using the Authorization Code Flow as defined in [OpenID Connect] as follows.

If the client has sent a first client configuration request to the Configuration Server using a secured connection as defined in sections 2.4, 2.9.1 and 2.10, and if the Configuration Server returns a 302 Found response, then the client shall use the value of the HTTP Location header to connect to the Open ID Connect authorization endpoint. This request forms the OpenID Connect authentication request.

On reception of the OpenID Connect authentication request the OpenID Connect authorization endpoint performs the procedures for user authentication, authorization and consent.

If the procedures for authentication, authorization and consent succeed or fail, then the OpenID Connect authorization endpoint returns a HTTP 302 Found response. The client

shall use the value of the HTTP Location header to connect to Configuration Server. This request forms the OpenID Connect authentication response.

On reception of the OpenID Connect authentication response the Configuration Server performs the procedures for the processing of the client's configuration request. In result of the processing, the Configuration Server returns a success or failure response to the client as defined in section 2.4.

The Configuration Server and the OpenID Connect authorization endpoint shall preserve the values of the client's initial configuration request during the OpenID Connect authentication request. The path and query components of the URIs used for the OpenID Connect authentication request procedure are opaque for the client.

The client shall support for the authentication flow the procedures for HTTP state management defined in [RFC6265]. This allows the Configuration Server and the authentication endpoint to return in HTTP responses a Set-Cookie header. The client shall apply the parsing and storage procedures of the Set-Cookie header as defined [RFC6265]. It shall send the cookie header in HTTP requests to the endpoints respecting the cookie attributes in the Set-Cookie header in accordance with [RFC6265].

2.8.3 Authentication, authorisation and user consent methods for OpenID Connect

The client shall support the following methods for user authentication, authorisation and consent during processing of the OpenID authentication request.

2.8.3.1 Authentication via HTTP Digest AKA

A client supporting Authentication via AKA shall support the procedure for user authentication via HTTP Digest AKA as defined in Annex B during processing of the OpenID Connect authentication request.

2.8.3.2 Authentication via HTTP embedded EAP-AKA

A client supporting Authentication via HTTP embedded EAP-AKA shall, when sending a first client configuration request to the Configuration Server using a secured connection as defined in section 2.4, add to the configuration request the EAP_ID parameter defined in section C.2.

As per section 2.4, the Configuration Server shall first validate the client and terminal parameters and, as part of this verification, the Configuration Server may challenge the client to provide proof of its authenticity and to confirm that it is running on a trust-worthy platform as defined for the client configuration request. If proof of client authenticity is required, then the Configuration Server shall invoke the procedure defined in section 2.11 via the response defined in section 2.4.3.

Based on the request parameter, the Configuration Server detects the support of HTTP embedded EAP-AKA. To authenticate the client via HTTP embedded EAP-AKA, the Configuration Server shall invoke the OpenID Connect Authentication Code flow via 302 Found response containing a HTTP Location header as defined in section 2.8.2. The URI in the Location header shall contain the "https" scheme to enforce a secure connection to the OpenID Connect authorization endpoint. The client shall abort the processing of the request

if the URI does not contain the "https" scheme. The client shall treat this the same as if there is no Configuration Server deployed in the network.

On reception of the 302 Found response, the client shall use the value of the Location header and issue a HTTP GET to the OpenID Connect authorization endpoint.

On reception of the request, the OpenID Connect authorization endpoint invokes the authentication via HTTP embedded EAP-AKA as defined in section C.3.

After completion of the authentication via HTTP embedded EAP-AKA, the OpenID Connect authorization endpoint returns a 302 Found response. The Location header contains the authentication response for the processing in the Configuration Server. The URI in the Location header shall contain the "https" scheme to enforce a secure connection to the Configuration Server. The client shall abort the processing of the request if the URI does not contain the "https" scheme. The client shall treat this the same as if there is no Configuration Server deployed in the network.

On reception of the 302 Found response, the client shall use the value of the Location header and issue a HTTP GET to the Configuration Server.

The Configuration Server validates the authorisation information contained in the authentication response.

NOTE: One potential way for the Configuration Server to validate in real-time the access token provided by the user, is to check with the OIDC endpoint that creates the token; as shown in the optional steps in Figure 9. Depending on the relationship (i.e. service level agreement and trust level) between the Configuration Server and the OIDC endpoint, the OIDC endpoint may pass additional information (i.e. IMSI, MSISDN, Account status etc.) regarding to the UE and user to support additional security checks by the Configuration Server.

If the user is authenticated, the Configuration Server returns a configuration XML document. If the configuration document contains a token parameter in the TOKEN characteristic, then the client shall apply the processing of the token as defined in section 4.2.

The Configuration Server can re-initiate the authentication for any subsequent client configuration request by initiating the OpenID Connect authorisation code flow via the 302 Found response, based on Service Provider policy.

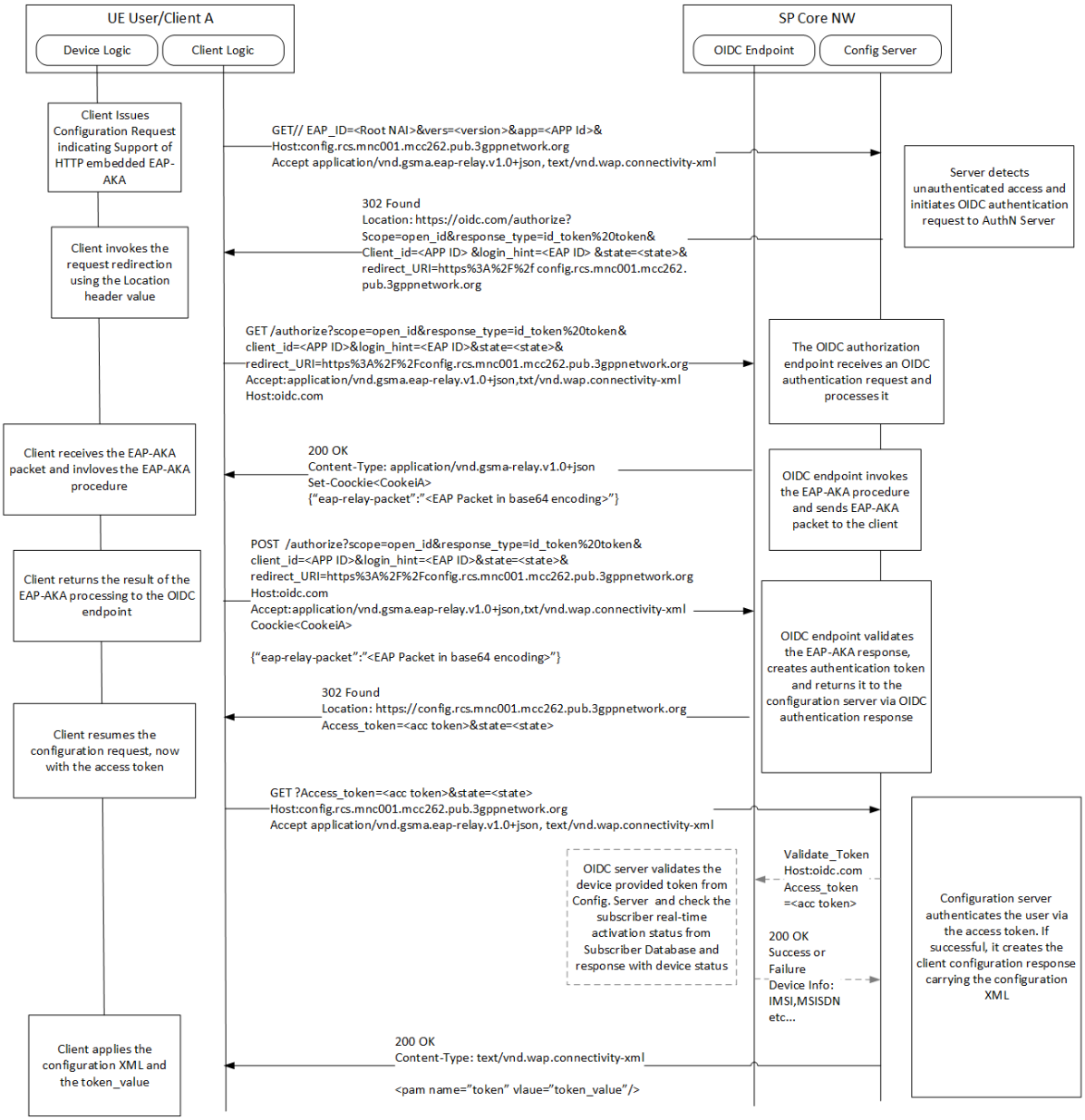

Figure 9: Authentication Flow using HTTP embedded EAP-AKA

2.8.3.3 Authentication via SMS OTP

The OpenID Connect authorization endpoint shall be able to authenticate the user during processing of the OpenID Connect authentication request via SMS OTP in the background. During the processing of the OpenID Connect authentication request, the client shall

- if it has received from the OpenID Connect authorization endpoint in result of a HTTP request a 401 Unauthorized response containing a WWW-authenticate header with the scheme set to "Basic" or "Digest" and the realm parameter set to the home network domain name of the network as defined in [3GPP TS 23.003], and
- if the client has indicated in the configuration request via the SMS_port parameter defined in section 2.3.2 support of the procedure to receive the OTP in the SMS message via application port addressing as defined in [3GPP TS 23.040]

then the client shall wait until it receives a SMS message with the format defined in section 2.6.1.

Once the SMS message is received, the client shall send the HTTP request again including an Authorization header with a value following the scheme requested by the OpenID Connect authorization endpoint. To form the content of the Authorization header the client shall use as

- the username the value of the IMSI, if available to the client, otherwise the MSISDN in international E.164 format without leading "+", and
- the password the value of the OTP received in the SMS message.

The HTTP request to the OpenID Connect authorization endpoint shall follow the procedures of the Basic or Digest HTTP Authentication.

The response of the OpenID Connect authorization endpoint to the client's HTTP request with the Basic or HTTP Digest authentication may be conveyed to the client via the HTTP 302 Found response to continue processing of the OpenID Connect authentication request or it may convey a request to the client to invoke a user interface.

2.8.3.4 Authentication, authorisation and user consent via user interface

If the OpenID Connect authorization endpoint requires during processing of the OpenID Connect authentication request actions by the user for

- authentication, e.g. user interactions to enter the MSISDN, a dedicated identity and/or a plain text password,
- authorisation and consent, e.g. an user message to confirm the client configuration

then the OpenID Connect authorisation endpoint shall request the client to initiate the user interface by returning a 200 OK response with a content body containing the web page for display. The client shall parse the web page and process the user interface flow until it receives from the OpenID Connect authorization endpoint in a HTTP response

- the request for the OpenID Connect authentication response, or
- a request for another user authentication method defined in this section.

The client shall support multiple user interface invocations during the processing of the OpenID Connect authentication request.

2.9 Configuration of additional devices sharing the same identity

This section describes the process of autoconfiguration authentication for clients on additional devices not carrying a SIM or carrying a SIM that is not used for user authentication. In this case, the additional device will share the identity assigned to a primary device with SIM.

2.9.1 First-time configuration

During first-time configuration, the device implementation/client will receive the credentials associated with the primary SIM card of the user regardless of the type of connection they are using (e.g. Wi-Fi, PS) to reach the Configuration Server.

The process is as follows:

1. As an option, the device implementation/client will offer the possibility to the user to perform manual provisioning
2. The user is prompted for the MSISDN or SIP URI of the primary device and the Service Provider associated with the primary SIM. The account created is always associated with the primary identity that the user has to input into the application. Please note that, as a pre-condition, the aforementioned identity must already be provisioned using the mechanism described in previous sections.
3. The device performs the HTTPS configuration as presented in section 2.4, however, using the GET parameters in Table 13 instead of the default ones.
4. If additional Configuration Servers are configured for the client and the client is triggered for a configuration request to a given Configuration Server and there is a client configuration request in progress with another Configuration Server then the client shall wait until the processing of the other client configuration request is finished. A client configuration request is considered finished if a final response is received from the Configuration Server. The additional authentication (i.e. SMS OTP) and the optional authorization processing via user messages (see section 2.4.2) is considered part of the processing of the configuration request.

Parameter	Description	Mandatory	Format
vers	This is either -2, -1, 0 or a positive integer. 0 indicates that the configuration must be updated (e.g., the configuration is damaged, non-existent or an update is needed following a SIM change). A positive value indicates the version of the static parameters (those which are not user dependent) so the Configuration Server can evaluate whether an update is required. -1 indicates that the device/client is providing the default behaviour for the services that would be configured and has disabled the configuration query performed at boot. This may be used by the client/device to inform the SP that the functionality was permanently disabled from the device. -2 Indicates that for the services to be configured the default behaviour needs to be provided (including the disabling of the configuration query at boot), but a configuration query might be triggered on user action. If the Service Provider has enabled additional Configuration Servers then the client shall manage the value of the "vers" parameter per Configuration Server. A "vers" value sent in a request to a Configuration Server shall be derived from the previous response of this Configuration Server.	Y	Int (-2, -1, 0 or a positive integer)
msisdn	MSISDN, in E.164 format, of the primary SIM that is used to derive the identity.	N, Mandatory if sip_uri not provided	E.164 (+447900000001) in international format NOTE: In case that msisdn comes with a plus sign, the client shall provide the msisdn value with the plus sign encoded as per [RFC3986] section 2.1.

Parameter	Description	Mandatory	Format
sip_uri	SIP URI of the primary device	N, Mandatory if msisdn is not provided	String (50 max), Case-insensitive
provisioning_version	String that identifies the version of the service provider device configuration supported by the client. It shall be set to "*5.0*" (without the quotes) for clients following this specification.	Y	String (4 max), Case-Sensitive
config_client_vendor	String that identifies the vendor providing the client process initiating the configuration process	N It shall be included if different from terminal_vendor	String (4 max), Case-Sensitive
config_client_version	String that identifies the version of the client process initiating the configuration request. client_version_value = Platform "-" VersionMajor "." VersionMinor Platform = Alphanumeric (9 max) VersionMajor = Number (2 char max) VersionMinor = Number (2 char max) Example: client_version=CFGAndrd-1.0	N It shall be included if not identified unambiguously by terminal_sw_version	String (20 max), Case-Sensitive
token	If the client has not stored a token (e.g. it is the first time the device requests configuration), the parameter shall contain an empty string. Otherwise, it shall contain the token value obtained in the last configuration response. If the Service Provider has enabled additional Configuration Servers then the client shall manage the token value per Configuration Server. The value of the "token" parameter sent in a request to a Configuration Server shall be derived from the previous response of this Configuration Server.	Y	String (24 max), Case-Sensitive
device_type	This indicates the type of device where the client is running.	Y	Possible values: - Tablet - PC - Other

Parameter	Description	Mandatory	Format
friendly_device_name	If provided by the user, a user friendly identification for the device may be passed along that can be used by the network when presenting the user with an overview of their devices. NOTE: this parameter needs to be included only if required for one of the services to be configured. In which case its mandatory character will be documented in the relevant service specific documents.	N, only to be provided if provided by the user	String (30 max before escaping), Case-Sensitive
app	String identifying one services supported by the client for configuration by means of its APPID for ACs or Management Object Identifier as assigned by the OMNA. If the client supports multiple services, one "app" name/value pair per supported service shall be provided in the request. Example: If the client supports the services identified by the following APPID values: ap2204, urn:foo:mo:bar:1.0 then the "app" parameter is presented as follows in the request: app=ap2204&app=urn%3Afoo%3Amo%3Abar%3A1.0	Y	String multi-valued parameter

Table 13: HTTP configuration for additional devices: Initial HTTPS request GET parameters

If the default Configuration Server has enabled additional Configuration Servers then configuration requests and responses shall be managed by the client on a per Configuration Server basis. The parameters of the configuration request defined in Table 13 shall have the same values for all Configuration Servers, unless stated otherwise.

NOTE: For requirements regarding presence and values of service specific request parameters for services managed via the Serviced Provider Device Configuration, refer to the corresponding service documentation.

Please note that the initial HTTP request is not required in this case since the header enrichment requirement is not applicable. Therefore, the device implementation/client will directly perform the HTTPS request as presented in Figure 10.

5. On reception of the HTTPS GET request, the Configuration Server shall first validate the client and terminal parameters. As part of this verification, the Configuration Server may challenge the client to provide proof of its authenticity and to confirm that it is running on a trust-worthy platform. The procedure to do so is described in section 2.11.

NOTE: This means that procedure in section 2.11 can be invoked for already configured clients when they verify the status of their current configuration.

If accepting the client as allowed to use the service (either directly or after explicit verification as described in section 2.11) the Configuration Server shall continue the procedure.

6. As this is a first time request, the token value is empty; the request is then identified as a first time configuration. In this case, and provided the network allows for configuring additional devices using this mechanism, the HTTP server responds with a HTTP 200 OK response carrying a new cookie (Set-Cookie header) to be used in the subsequent HTTP requests.

    a) Following the request, an SMS message shall be sent to the primary device, i.e. the phone carrying the SIM associated to the MSISDN the user introduced in step 2. This SMS message will contain an OTP. This message shall be a standard SMS (i.e. no UDH procedures required).

NOTE: When used on IP Multimedia Subsystem (IMS) networks with IMS devices, other ways may be provided to contact the primary device. Those are covered in [GSMA PRD-RCC.15]

    b) In parallel, the device performing the HTTP configuration prompts for the OTP. Therefore, the user should manually introduce the code delivered via SMS to the primary device.
    c) Once the user enters the OTP, the device performing the HTTP configuration makes a second HTTPS request using the following parameters in the GET request:

Parameter	Description	Mandatory	Format
OTP	This is the password received on the device carrying the SIM associated with the MSISDN introduced in step 2	Y	String (8 Max), Case-Sensitive

Table 14: HTTP configuration for additional devices: Second and final HTTPS request GET parameters

Please note this second HTTPS request shall carry the cookie obtained in step 6 (cookie header), therefore the Configuration Server can correlate the initial and final HTTPS requests.

    d) From this point onwards the procedure is identical to the one described in sections 2.4.1 and 2.4.3, however, with the token added as a parameter. If the request is successful, one of the possible 200 OK responses described in section 2.4.1 is provided.

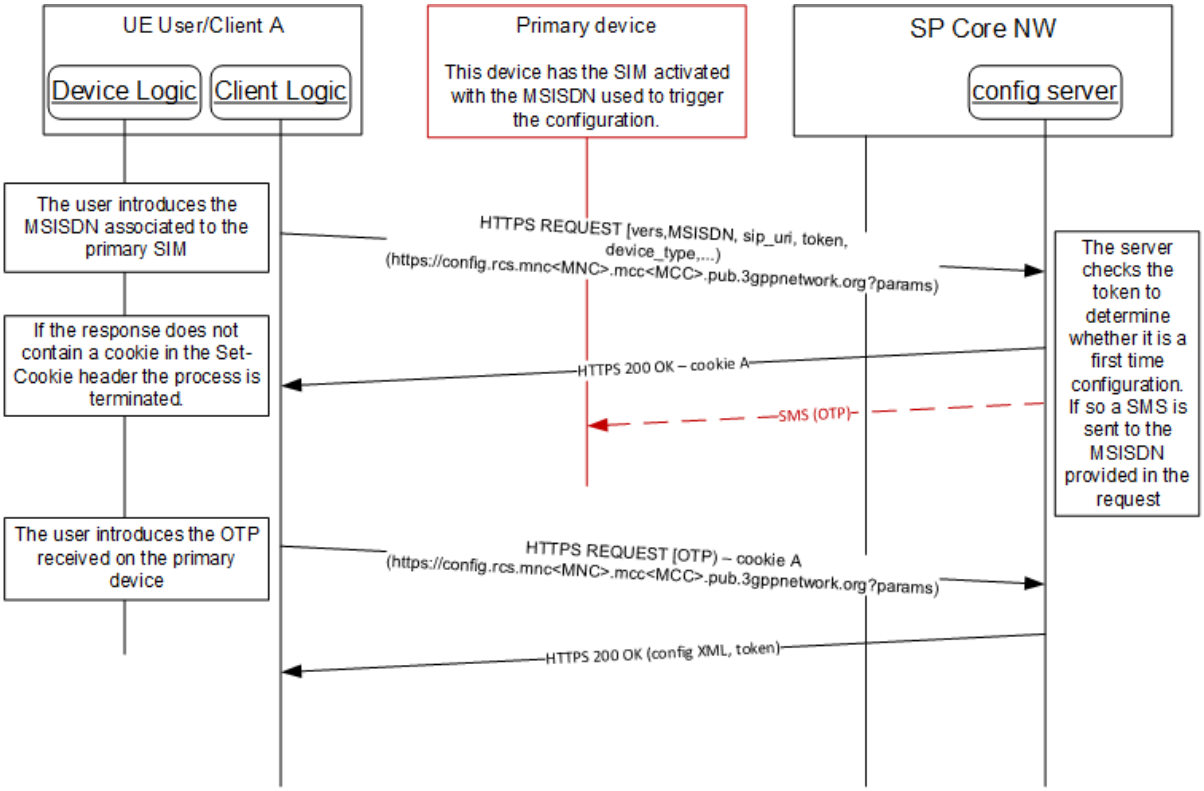

Figure 10: HTTP configuration for additional devices: First time configuration

Please note the token shall be stored with the MSISDN so it is not necessary to repeat this procedure for future requests. These values shall be removed together with the rest of the configuration when the device or client is reset.

2.9.2 Error handling

When performing a first time configuration for additional devices, there are three possible error conditions that the client has to be aware of and handle:

1. The MSISDN used is not valid or it is not authorized (including the case the primary MSISDN is not been provisioned yet to use the services being configured) to get the configuration/make use of the services. In this case, the initial request will be answered with an HTTP 403 FORBIDDEN error and the client shall inform the user of the issue and may offer to retry with a different MSISDN.
2. The OTP password introduced by the user is not valid. In this case, the Configuration Server replies again with a HTTP 511 NETWORK AUTHENTICATION REQUIRED error. It is up to the client implementation to offer the user to retry. If retrying, the client shall start the first time configuration process from the beginning.
3. The HTTP server suffers an internal error (HTTP 5XX [except 511], response coming from the Configuration Server). In this case, the user shall be informed of the circumstance and offered to retry. If retrying, the client shall start the first time configuration process from the beginning.

2.9.3 Subsequent configuration attempts and life cycle

If the client has access to the token and the MSISDN used for the first time configuration, has a value, the initial request is performed

1. An initial request like in the case of the first time configuration of additional devices is made, this time including the token parameter set to the value received on the previous successful configuration attempt
2. If successful, from this point onwards the procedure is identical to the one described in sections 2.4.1 and 2.4.3, however, with the token added as a parameter. The Configuration Server may challenge the client to provide proof of its authenticity using the procedure described in section 2.11 or continue immediately. If the procedure is successful, one of the possible 200 OK responses described in section 2.4.1 is provided.

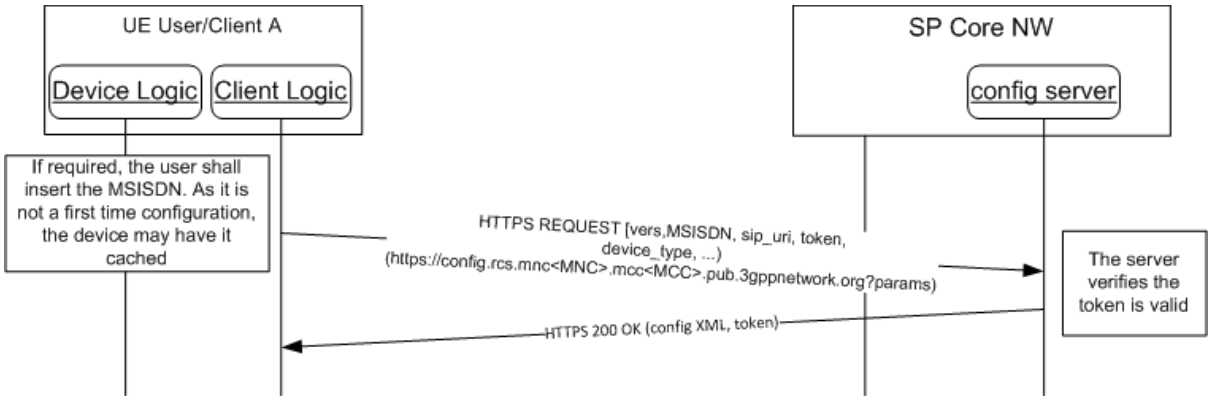

Figure 11: HTTP configuration for additional devices: Subsequent attempts

If the token and/or the MSISDN are not available (for example the device is reset), then the client shall start a first time configuration as described in section 2.9.1.

Please note the received token shall be stored with the MSISDN so it is not necessary to repeat this procedure for future requests. These values shall be removed together with the rest of the configuration when the device or client is reset.

2.9.4 Error handling

When performing a subsequent configuration for additional devices, there are three possible error conditions that the client has to be aware of and to handle:

1. The MSISDN used is not valid or it is not authorized to get the configuration/make use of the services to be configured. In this case, the initial request will be answered with an HTTP 403 FORBIDDEN error and the client shall inform the user of the issue and may offer to retry with a different MSISDN.
2. The token is no longer valid. In this case, the Configuration Server replies again with a HTTP 511 NETWORK AUTHENTICATION REQUIRED error. From this moment, the process is equivalent to the first time configuration process after the same error is received.
3. The HTTP server suffers an internal error (HTTP 5XX response coming from the Configuration Server). In this case, the user shall be informed of the circumstance and offered to retry. If retrying, the client shall start the subsequent configuration attempt procedure from the beginning.

2.9.5 Security considerations

The same security considerations described in section 2.6.4 for the standard HTTP(S) configuration mechanism also apply in this case.

2.10 Configuration of non-Cellular devices with a dedicated identity

To configure clients on devices that do not carry a SIM, but have to function with a dedicated own identity the following generic solution is provided:

1. The user obtains an OTP through means that are out of the scope of this specification (e.g. from an operator website after authentication, delivered together with the device, obtained through an operator's retail outlet, etc.). If the Service Provider has enabled additional Configuration Servers then the OTP need to be dedicated to the services managed by a given Configuration Server.
2. The user is prompted for the E.164 address or SIP URI to be used by the device and their Service Provider. The account created is always associated with the primary identity that the user has to input into the application.
3. The device performs the HTTPS configuration as presented in section 2.4, however, using the GET parameters in Table 15 instead of the default ones.
4. If additional Configuration Servers are configured for the client and the client is triggered for a configuration request to a given Configuration Server and there is a client configuration request in progress with another Configuration Server then the client shall wait until the processing of the other client configuration request is finished. A client configuration request is considered finished if a final response is received from the Configuration Server. The additional authentication (i.e. SMS OTP) and the optional authorization processing via user messages (see section 2.4.2) is considered part of the processing of the configuration request.

Parameter	Description	Mandatory	Format
vers	This is either -2, -1, 0 or a positive integer. 0 indicates that the configuration must be updated (e.g., the configuration is damaged, non-existent or an update is needed following a SIM change). A positive value indicates the version of the static parameters (those which are not user dependent) so the Configuration Server can evaluate whether an update is required. -1 indicates that the device/client is providing the default behaviour for the services that would be configured and has disabled the configuration query performed at boot. This may be used by the client/device to inform the SP that the functionality was permanently disabled from the device. -2 Indicates that for the services to be configured the default behaviour needs to be provided (including the disabling of the configuration query at boot), but a configuration query might be triggered on user action. If the Service Provider has enabled additional Configuration Servers then the client shall manage the value of the "vers" parameter per Configuration Server. A "vers" value sent in a request to a Configuration Server shall be derived from the previous response of this Configuration Server.	Y	Int (-2, -1, 0 or a positive integer)
msisdn	E.164 format of the provided identity	N, Mandatory if sip_uri is not provided	E.164 (+447900000001) in international format NOTE: In case that msisdn comes with a plus sign, the client shall provide the msisdn value with the plus sign encoded as per [RFC3986] section 2.1.

Parameter	Description	Mandatory	Format
sip_uri	SIP URI of the device	N, Mandatory if msisdn is not provided	String (50 max), Case-insensitive
provisioning_version	String that identifies the version of the service provider device configuration supported by the client. It shall be set to "*5.0*" (without the quotes) for clients following this specification.	Y	String (4 max), Case-Sensitive
config_client_vendor	String that identifies the vendor providing the client process initiating the configuration process	N It shall be included if different from terminal_vendor	String (4 max), Case-Sensitive
config_client_version	String that identifies the version of the client process initiating the configuration request. client_version_value = Platform "-" VersionMajor "." VersionMinor Platform = Alphanumeric (9 max) VersionMajor = Number (2 char max) VersionMinor = Number (2 char max) Example: client_version=CFGAndrd-1.0	N It shall be included if not identified unambiguously by terminal_sw_version	String (20 max), Case-Sensitive
token	If the client has not stored a token (e.g. it is the first time the device requests configuration), the parameter shall contain an empty string. Otherwise, it shall contain the token value obtained in the last configuration response. If the Service Provider has enabled additional Configuration Servers then the client shall manage the token value per Configuration Server. The value of the "token" parameter sent in a request to a Configuration Server shall be derived from the previous response of this Configuration Server.	Y	String (24 max), Case-Sensitive
device_type	This indicates the type of device where the client is running.	Y	Possible values: - Tablet - PC - Other

Parameter	Description	Mandatory	Format
OTP	This is the password provided to the user in step 1. Set to an empty string in case a non-empty token is provided. If the Service Provider has enabled additional Configuration Servers then it is left to the Service Provider and user's discretion to supply the OTP applicable for the requested Configuration Server.	Y	String (8 Max), Case-Sensitive
friendly_device_name	If provided by the user, a user friendly identification for the device may be passed along that can be used by the network when presenting the user with an overview of their devices. NOTE: this parameter needs to be included only if required for one of the services to be configured. In which case its mandatory character will be documented in the relevant service specific documents.	N, only to be provided if provided by the user	String (30 max before escaping), Case-Sensitive
app	String identifying one service supported by the client for configuration by means of its APPID for ACs or Management Object Identifier as assigned by the OMNA. If the client supports multiple services, one "app" name/value pair per supported service shall be provided in the request. Example: If the client supports the services identified by the following APPID values: ap2204, urn:foo:mo:bar:1.0 then the "app" parameter is presented as follows in the request: app=ap2204&app=urn%3Afoo%3Amo%3Abar%3A1.0	Y	String multi-valued parameter

Table 15: HTTP configuration for non-cellular devices: Initial HTTPS request GET parameters

If the default Configuration Server has enabled additional Configuration Servers then configuration requests and responses shall be managed by the client on a per Configuration Server basis. The parameters of the configuration request defined in 2.10 shall have the same values for all Configuration Servers, unless stated otherwise.

NOTE: For requirements regarding presence and values of service specific request parameters for services managed via the Serviced Provider Device Configuration, refer to the corresponding service documentation.

Please note that the initial HTTP request is not required in this case since the header enrichment requirement is not applicable. Therefore, the device implementation/client will directly perform the HTTPS request as presented in Figure 12.

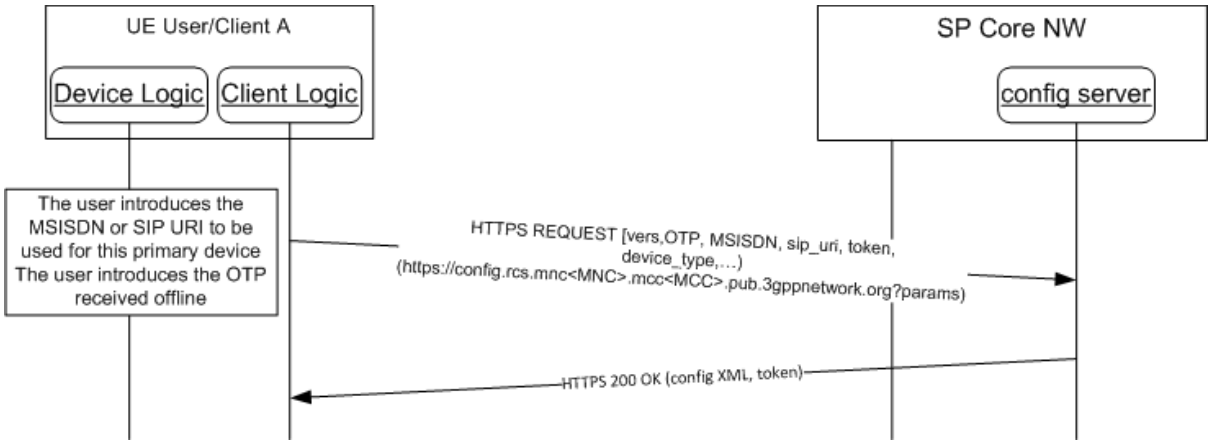

Figure 12: HTTP configuration for non-cellular devices with a dedicated identity: initial request

5. On reception of the HTTPS GET request, the Configuration Server shall first validate the client and terminal parameters. As part of this verification, the Configuration Server may challenge the client to provide proof of its authenticity and to confirm that it is running on a trust-worthy platform. The procedure to do so is described in section 2.11.

NOTE: This means that procedure in section 2.11 can be invoked for already configured clients when they verify the status of their current configuration.

If accepting the client as allowed to use the service (either directly or after explicit verification as described in section 2.11) the Configuration Server shall continue the procedure.

6. As this is a first time request, the token value is empty; the request is then identified as a first time configuration. In this case, and provided the network allows for configuring devices using this mechanism, the HTTP server responds with a HTTP 200 OK response carrying a new cookie (Set-Cookie header) to be used in the subsequent HTTP requests

From this point onwards the procedure is identical to the one described in sections 2.4.1 and 2.4.3, however, with the token added as a parameter. If the request is successful, one of the possible 200 OK responses described in section 2.4.1 is provided.

Please note the token shall be stored with the identity so it is not necessary to repeat this procedure for future requests. These values shall be removed together with the rest of the configuration when the device or client is reset.

2.10.1 Subsequent configuration attempts and life cycle

If the client has access to the token and the identity used for the first time configuration, has a value, the initial request is performed

1. An initial request like in the case of the first time configuration of the primary device is made, this time including the token parameter set to the value received on the previous successful configuration attempt
2. If successful, from this point onwards the procedure is identical to the one described in sections 2.4.1 and 2.4.3, however, with the token added as a parameter. If the request is successful, one of the possible 200 OK responses described in section 2.4.1 is provided.

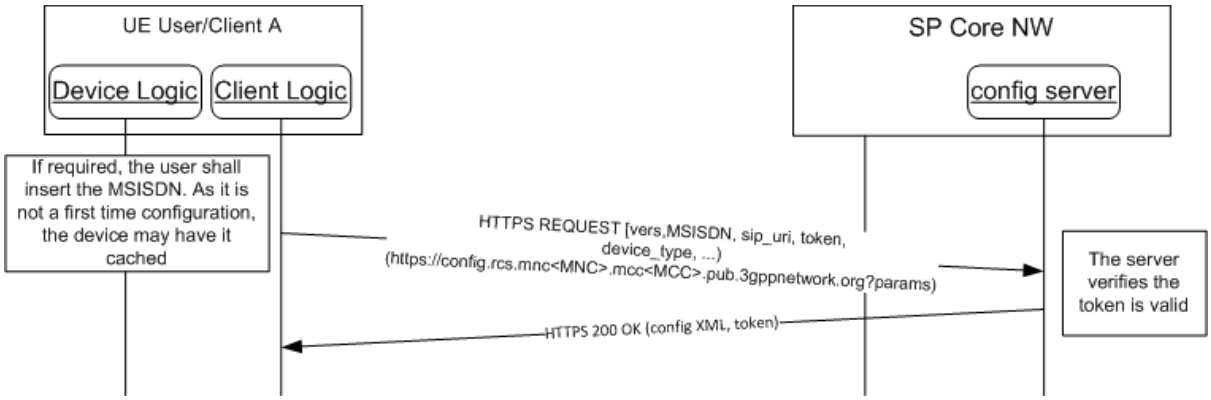

Figure 13: HTTP configuration for non-cellular devices: Subsequent attempts

If the token and/or the MSISDN are not available (for example the device is reset), then the client shall start a first time configuration as described in section 2.9.1.

Please note the received token shall be stored with the MSISDN so it is not necessary to repeat this procedure for future requests. These values shall be removed together with the rest of the configuration when the device or client is reset.

2.10.2 Error handling

In the process of performing a subsequent configuration for additional devices, there are three possible error conditions that the client has to be aware of and to handle:

1. The MSISDN used is not valid or it is not authorized to get the configuration/make use of the services to be configured. In this case, the initial request will be answered with an HTTP 403 FORBIDDEN error and the client shall inform the user of the issue and may offer to retry with a different MSISDN.
2. The token is no longer valid. In this case, the Configuration Server replies again with a HTTP 511 NETWORK AUTHENTICATION REQUIRED error. From this moment, the process is equivalent to the first time configuration process after the same error is received.
3. The HTTP server suffers an internal error (HTTP 5XX response coming from the Configuration Server). In this case, the user shall be informed of the circumstance and offered to retry. If retrying, the client shall start the subsequent configuration attempt procedure from the beginning.

2.11 Client Authenticity Verification

If the Configuration Server is aware that the client supports verifying its authenticity, the Configuration Server shall not consider the client's ability to perform the procedures required for user authentication as sufficient guarantee that the configuration will be processed and

stored securely. In this case, it shall decide to verify that the client is authentic and that the request is not originating from an application impersonating an allowed client using the procedure described in this section.

Clients shall support this procedure and include the parameters listed in Table 16 in the first HTTPS request when initiating the configuration procedure. This is in addition to the parameters described in Table 1, Table 8, Table 13 or Table 15 depending on the nature of the client and the access.

Parameter	Description	Mandatory	Format
client_authenticity_support	Describes a method supported by the client to prove its authenticity. The actual values will be well-known strings defined per platform and are considered out-of-scope of this specification. If the client supports multiple methods, one "client_authenticity_support" name/value pair per supported method shall be provided in the request.	N	String

Table 16: Initial HTTPS request GET parameters indicating support for client authenticity verification

The Configuration Server shall use the following procedure

1. On receiving the HTTPS request from the client initiating the configuration procedure (i.e. the request carrying the parameters described in Table 1, Table 8, Table 13 or Table 15 depending on the nature of the client and the access), the Configuration Server may verify whether based on those parameters access would be allowed. If that is not the case, the Configuration Server shall reject the HTTPS request with a HTTP 403 Forbidden response and terminate the procedure.

   This verification should consider whether the method(s) that the client indicated as being supported for verifying its authenticity correspond to what the Configuration Server expects based on information it has obtained on the client through other means if it has such information. Those means are considered to be out of scope for this specification.
   On deployments where the Configuration Server mandates the support of client authenticity, if a client sends a HTTPS request without the client_authenticity_support parameter the Configuration Server shall reject the HTTPS request with a HTTP 426 Upgrade Required and a reference to the missing parameter in the request.

2. If based on the provided parameters the client would be allowed access, the Configuration Server may decide to invoke one or more of the methods the client is offering to verify its authenticity if the client offered such methods as defined in Table 16:

   a) The Configuration Server shall obtain and/or generate the parameters that it needs to provide for the chosen configuration methods.

b) The Configuration Server shall then generate an XML body according to the schema in Table 17:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:gsma:params:xml:ns:cfg:clientauth:req"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns="urn:auth:params:xml:ns:cfg:auth"
                elementFormDefault="qualified"
                attributeFormDefault="unqualified">
 <xs:element name="client-authenticity-request">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="method" minOccurs="1" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="param" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:simpleContent>
          <xs:extension base="xs:string">
           <xs:attribute name="name" use="required">
            <xs:simpleType>
             <xs:restriction base="xs:string"/>
            </xs:simpleType>
           </xs:attribute>
           <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:extension>
         </xs:simpleContent>
        </xs:complexType>
       </xs:element>
       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string"/>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="id" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:integer"/>
       </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="nonce" type="xs:string" minOccurs="0" maxOccurs="1"/>
 <xs:element name="vapid" type="xs:string" minOccurs="0" maxOccurs="1"/>
</xs:schema>
```

Table 17: XML Schema for client authenticity request

In this for each of the verification methods selected a *method* element shall be instantiated for which the *name* attribute shall be set to the well-known name of the corresponding method and the *id* attribute to a generated integer value that

will be used to match the response. The parameters required for that method shall be included in *param* sub-elements of the *method* element of which the *name* attribute shall be set to the well-known name assigned to the parameter and their value as value for the *parameter* element, the generated or obtained value for that parameter shall be included.
The Configuration Server may include a nonce, which must be unique and include at least 128 bits of randomness. The nonce would be used by the client to sign and by the Configuration Server to verify the signature.
The Configuration Server may include a Voluntary Application Server Identification (VAPID) key. This VAPID key shall be used by the client for a Push Notification Mechanism (e.g. as described GSMA PRD RCC.07 in Section 2.15 for RCS Services).

c) The Configuration Server shall then generate a HTTPS 200 OK response to the outstanding request from the client and include the generated XML content as body of that response. If not already done for earlier requests in the flow e.g. the HTTP request defined in section 2.4.1, the Configuration Server shall include a Set-Cookie header as defined in [RFC2616] with as value a cookie assigned to this configuration flow. The Content-Type header of the Response shall be set to *application/vnd.gsma.cfg-clientauth+xml* and the response shall be sent to the client.

A client receiving a HTTP 200 OK response to the HTTPS request from the client initiating the configuration procedure that contains a body with Content-Type *application/vnd.gsma.cfg-clientauth+xml,* shall verify validity of the provided body against the schema defined in Table 17. If valid, for each *method* element in the body, the client shall perform the procedure to proof its authenticity for which the well-known name corresponds to the value provided in the name attribute of the *method* element. As value for the parameters required for this procedure the value of the *param* sub-element for which the *name* attribute corresponds to the well-known name of the parameter shall be used. When the procedures corresponding to all *method* elements are executed, the client shall generate a new HTTPS POST request to the Configuration Server providing the results of the procedures in a MIME *multipart/form-data* entity body as defined in [RFC7578] and including the Cookie set provided by the Configuration Server in the earlier HTTPS response. In this HTTPS POST request, the client shall include a body part for each procedure corresponding to a *method* element that was executed. This body part shall be formatted as defined in Table 18 whereby <id> is replaced with the *id* attribute of the *method* element in the XML provided by the Configuration Server for which this body part is providing the proof.

```
Content-Disposition: form-data; name="client_authenticity_result_<id>”
Content-Type: text/plain

<Authenticity Proof provided by the client>
```

Table 18: form format to be used in the HTTP POST method request to provide proof of client authenticity

A Configuration Server receiving a HTTPS POST request carrying a MIME *multipart/form-data* entity bodies formatted as defined in Table 17 shall verify whether the results of the different authenticity verification methods to which those HTTP Get parameters correspond

can be considered as a valid proof of authenticity taking into account the parameter values that it provided to the client as part of the response to the HTTPS request initiating the configuration procedure. If this validation is successful, the Configuration Server shall continue the configuration as described in the section 2.4.1, 2.6.1, 2.9.1, 2.10 or 2.13 depending on the parameters provided in the HTTPS request initiating the configuration procedure. If not considering the client or platform as valid to continue configuration the Configuration Server shall return a 200 OK response with version and validity set to -2 as described in section 2.4.1 and may include a User Message as described in section 2.4.2 to explain the reason for failure of the configuration to the user.

When successfully configured, the client may verify on a regular basis that it or the platform on which it is running have not been compromised. The means to do so are considered to be platform specific and device local and thus out of scope of this specification. Also after successful configuration, the client shall continue to handle requests from the Configuration Server to provide proof of its authenticity as described in this section. The frequency of such verification requests is up to the service provider.

Figure 14 depicts an example of the client authenticity verification flow for the case where the client authenticity proof succeeds:

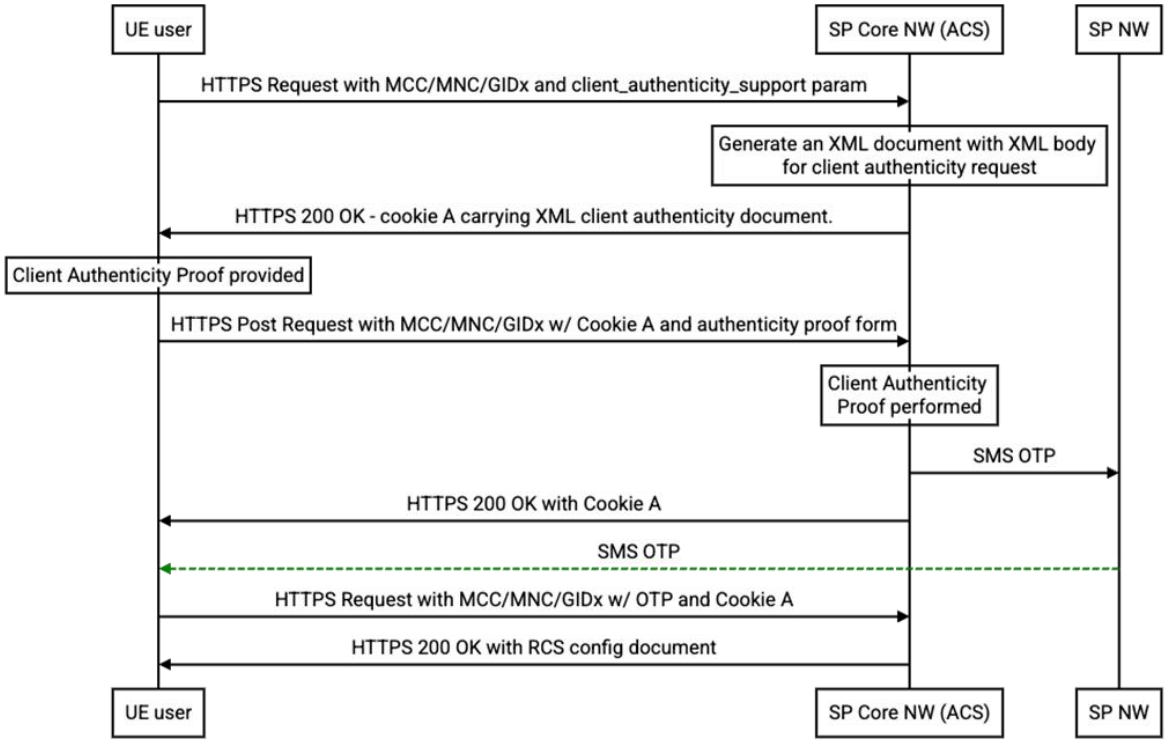

Figure 14: Example Client Authenticity Verification Flow

2.12 Client Certificate Upload

For RCS clients that SIP REGISTER in the network without SIM credentials, the Configuration Server may consider the need to support Signed SIP Digest. For that purpose, the Configuration Server may decide to direct the client to upload a certificate that shall be used to verify the SIP Digest response signature. The client certificate is provided by a trusted root.

Clients supporting this procedure shall include the parameters listed in Table 19 in the first HTTPS request when initiating the configuration procedure. This is in addition to the parameters described in Table 1, Table 8, Table 13 or Table 15 depending on the nature of the client and the access.

Parameter	Description	Mandatory	Format
client_certificate_upload	The presence of this parameter confirms the client's support of the Signed SIP Digest. The value of the parameter is the name of the form-data to be sent in the POST request.	N	String

Table 19: HTTPS request parameter to upload certificate

A client that receives a HTTP 200 OK response to the initial HTTPS request shall generate a new HTTPS POST request to the Configuration Server. This HTTP POST request shall provide the public certificate in a MIME multipart/form-data entity body as defined in [RFC7578] and include the Cookie set that the Configuration Server provided in the earlier HTTPS response. In this HTTPS POST request, the client shall include a body part with the X.509 client certificate. This body part shall be formatted as defined in Table 20, where <client_certificate_upload> is replaced with the value of the parameter passed in the earlier HTTP request.

```
Content-Disposition: form-data; name="<client_certificate_upload>”
Content-Type: text/plain

<X.509 Client certificate>
```

Table 20: form format to be used in the HTTP POST method request to upload client certificate

A Configuration Server that receives a HTTPS POST request carrying MIME multipart/form-data entity bodies formatted as defined in Table 20 shall verify the certificate validity before proceeding to store it. If this validation is successful, the Configuration Server shall continue the configuration as described in section 2.4.1, 2.6.1, 2.9.1 or 2.10, depending on the parameters provided in the HTTPS request that initiated the configuration procedure. If the Configuration Server determines that the client or platform is not valid for configuration, it shall return a 200 OK response with version and validity set to -2 as described in section 2.4.1. The Configuration Server may also include a User Message as described in section 2.4.2 to explain why the configuration failed.

NOTE: It is recommended for the client to store the private key of the provided certificate in the hardware keystore.

NOTE: This mechanism of client certificate upload can be combined with client authenticity verification (section 2.11). In this combined approach, the client would send the parameters from Table 16 and Table 18 as part of the very first request of the provisioning process. Subsequently, the client would include both the client certificate and each authenticity proof within the POST request, as described in section 2.11.

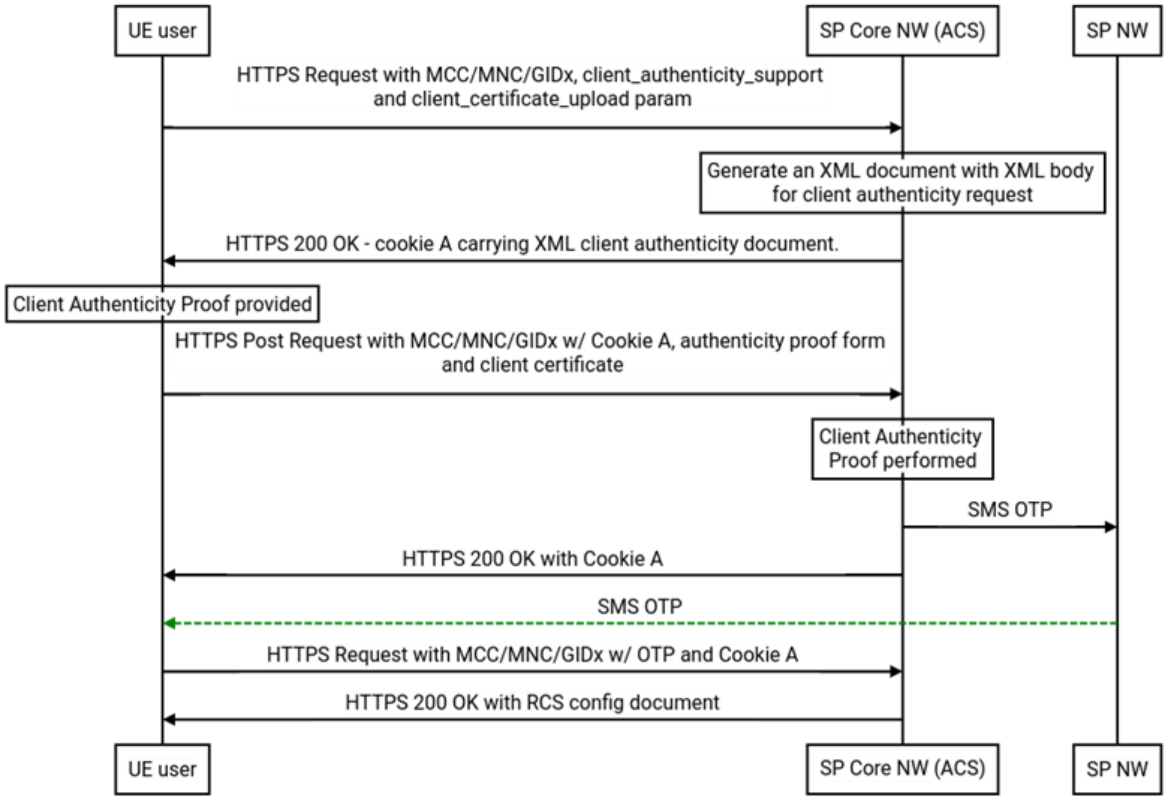

Figure 15: Example Client authenticity verification and client certificate upload flow

2.12.1 Client Certificate Details and Validation

2.12.1.1 Client Certificate Details

X.509 digital certificates shall be used for signing and verification in Signed SIP Digest. All X.509 certificates shall be signed by a trusted party, where the trusted party is recommended to be the client vendor. The Key Usage of the certificate provided shall include “digitalSignature”. The subject or subjectAltName shall contain a unique identifier for the client.

2.12.1.2 Client Certificate Validation

Client uploaded certificates shall be verified as part of a certificate chain that chains up to a trusted Root certificate. The chain may contain intermediate CA certificates.

The Configuration Server shall build the certificate chain and validate the client uploaded certificate according to the "Certification Path Validation" procedures described in [RFC5280]. In general, X.509 certificates support a liberal set of rules for determining if the issuer name of a certificate matches the subject name of another. The rules are such that two name fields may be declared to match even though a binary comparison of the two name fields does not indicate a match. [RFC5280] recommends that certificate authorities restrict the encoding of name fields so that an implementation can declare a match or mismatch using simple binary comparison. Accordingly, the Distinguished Encoding Rules (DER)-encoded tbsCertificate.issuer field of a certificate shall be an exact match to the DER-encoded tbsCertificate.subject field of its issuer certificate. An implementation may compare

an issuer name to a subject name by performing a binary comparison of the DERencoded tbsCertificate.issuer and tbsCertificate.subject fields.

NOTE: Which parties that can generate root certificates is for further study

2.13 Configuration request using Temporary Token via SIM authentication

A client capable of acquiring a Temporary Token from a SIM authentication endpoint shall proceed as follows:

1. Obtain a Temporary Token after successful SIM authentication against the SIM authentication endpoint (e.g. HTTP embedded EAP-AKA procedure in Section C.3). The Temporary Token shall allow the Configuration Server to request user information (e.g. MSISDN) deemed necessary to configure the client from the token validation endpoint.
2. In case of failure, fallback to one of procedures defined in sections 2.6, 2.7 and 2.8.
3. Otherwise, add the body part listed in Table 21 (as_temp_token) with the value obtained from step 1 in a client configuration HTTPS POST request to the Configuration Server as shown in Figure 16.

NOTE: The Temporary Token can be used for validation only once and it must have a short time-to-live.

In the HTTPS POST request, the client shall include a body part with the Temporary Token. The body part shall be formatted as defined in Table 21

```
Content-Disposition: form-data; name=”as_temp_token”
Content-Type: text/plain

<temporary token received from SIM authentication endpoint in step 1>
```

Table 21: form format to be used in the HTTP POST method request to provide temporary token

Upon receiving the request, if the Configuration Server supports Temporary Token usage, it shall validate the Temporary Token with the token validation endpoint and retrieve the user information that the token validation endpoint deems necessary to configure the client from the token validation endpoint. Upon success, the Configuration Server shall return a configuration XML document with HTTP 200 OK. Refer to Table 22 on error responses from the token validation endpoint and how the Configuration Server shall translate the error response to the client. The client shall follow section 2.4.3 for response handling.

NOTE: Additional Configuration Servers as described in section 2.2.2 can choose any method outlined in 2.6, 2.7, 2.8 or 2.13.

Response Code from Token Validation Endpoint	Scenario	Configuration Server Response Code
400, 401, 403 or 404	Failed validation of Temporary Token	403
500 Internal Server Error	Token validation endpoint runs into an internal error during procession of request	500

Table 22: Temporary Token validation error response

If the Temporary Token configuration flow is not supported by the Configuration Server, the Configuration Server shall return a HTTP 403 response. The client shall follow section 2.4.3 for response handling.

The flow is summarised in Figure 16.

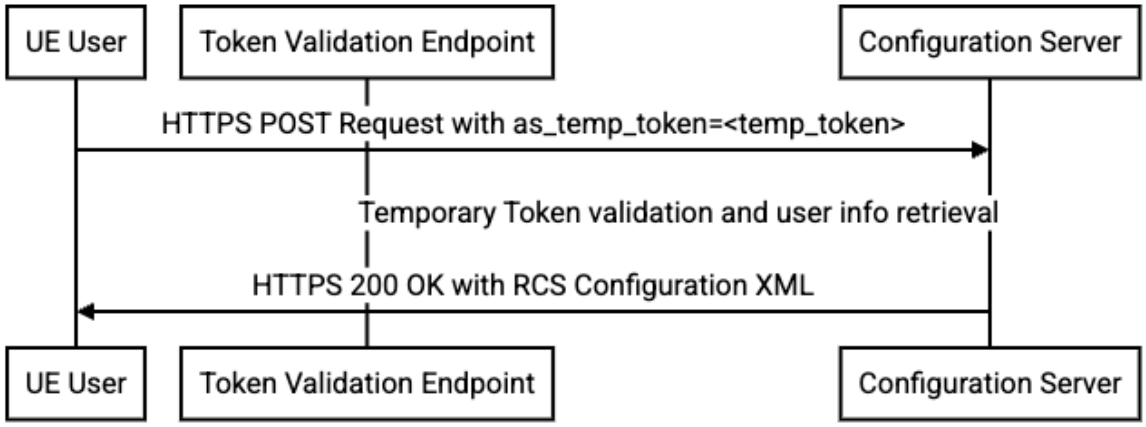

Figure 16: Configuration Flow using Temporary Token

As client and Configuration Server shall support client authenticity verification as described in section 2.11 and 2.12, the client shall perform SIM authentication and obtain the Temporary Token before sending the client authentication verification HTTPS POST request.

The client shall include the Temporary Token as as_temp_token=<temp_token> in the body of client authenticity HTTPS POST request, along with the Cookie, authenticity and client certificate payloads described in Table 18 and Table 20.

The flow is summarised in Figure 17.

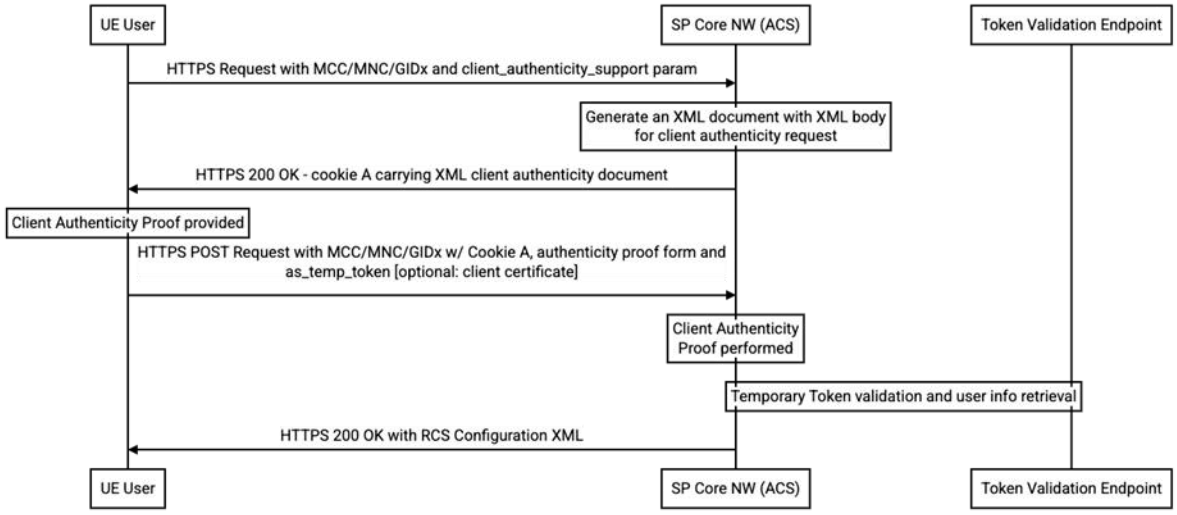

Figure 17: Example Configuration Flow using Client Authenticity and Temporary Token

A Configuration Server receiving a HTTPS POST request shall process it as per section 2.11, 2.12 and this section depending on the payload of HTTPS POST request.

2.14 Push Notification Mechanism

If the client supports a Push Notification Mechanism (e.g. as described in Section 2.15 of GSMA PRD RCC.07) and encryption of contents in the HTTP body, then the client shall include the parameters listed in Table 23 in the first HTTPS GET request when initiating the configuration procedure. This is in addition to the parameters described in Table 1, Table 8, Table 13, or Table 15 depending on the nature of the client and the access.

Parameter	Description	Mandatory	Format
client_pushkey_upload	The presence of this parameter confirms the client's support of the Push Notification Mechanism. The value of the parameter is the name of the form-data to be sent in the POST request.	N	String

Table 23: HTTPS request parameter to upload public key

The Configuration Server shall use the following procedure:

- If the Configuration Server receives the client_pushkey_upload parameter in the initial request and it supports the Push Notification Mechanism, it shall, in the XML response body defined in section 2.11, include a VAPID key as defined in [RFC8292].

A client that receives a HTTP 200 OK response to the initial HTTP request shall generate a new HTTPS POST request to the Configuration Server. This HTTPS POST request shall provide the public key in a MIME multipart/form-data entity body as defined in [RFC7578] and include the Cookie set that the Configuration Server provided in the earlier HTTPS response.

In this HTTPS POST request, the client shall include a body part with the parameter client_pushkey_upload. This body part shall be formatted as defined in Table 24, where <client_pushkey_upload> is replaced with the value of the parameter passed in the earlier HTTP request.

```
Content-Disposition: form-data; name="<client_pushkey_upload>"
Content-Type: text/plain

<Public key of client>
```

Table 24: form format to be used in the HTTP POST method request to upload client pushkey

In this HTTPS POST request, the client shall also inform the RCS Service Provider a URI where Push Notification shall be delivered to the client. The client shall include a body part with the parameter client_push_uri. The body part shall be formatted as defined in Table 25.

```
Content-Disposition: form-data; name="client_push_uri"
Content-Type: text/plain

<Push URI for client>
```

Table 25: form format to be used in the HTTP POST method request to provide client push URI

A Configuration Server that receives a HTTPS POST request carrying MIME multipart/form-data entity bodies formatted as defined in Table 24 and Table 25 and shall store the parameter client_pushkey_upload and client_push_uri for this client.

3 Network requested configuration request

There are use cases where the Configuration Server needs to enforce a first time configuration or a client reconfiguration at any given time.

The present section presents the enhancements that need to be implemented both on the network side and on the client to support a network requested reconfiguration.

NOTE: The described mechanism only covers reconfiguration requests related to primary devices. In addition, this option is only available to platforms and clients that support the application port addressing via UDH header handling as per [3GPP TS 23.040].

The Configuration Server will trigger the client for configuration via a network originated SMS.

A client shall be able to receive a request for configuration and process it accordingly in the following configuration states:

- prior to a first configuration
- if a configuration exists for a number of active services.
- if a previous client configuration reverted all services to their default behaviour.

If the Service Provider has enabled additional Configuration Servers then reconfiguration triggers are requested for an individual Configuration Server.

The SMS to trigger the client reconfiguration shall be formatted as follows:

- DataCodingScheme = 00 (GSM 7 bit default alphabet) or 08 (UCS2)
- UserDataHeaderIndicator = 1
- UserDataHeader with
    - UDHL=06
    - IEI=05
    - IEIL=04
    - Destination port: 0x9199 (37273 in decimal)
    - Source Port: 0x0000 (0 in decimal)
- User Data

  The User Data shall be encoded using the GSM 7 bit default alphabet or UCS2 as indicated in the DataCodingScheme.
  The content of the User Data is defined as follows:

```
user-data = user-id "-rcscfg" [ "," parm ]
                      ; "rcscfg" includes RCS for historic reasons
user-id = IMSI | IMPI
IMSI = *15DIGIT ; for composition of IMSI refer to [3GPP TS 23.003]
IMPI = username "@" realm ; for encoding of username and realm
                          ; refer to [RFC4282]
parm = fqdn-parm | extension
fqdn-parm = fqdn-key "=" fqdn-value
fqdn-key = "fqdn"
fqdn-value = realm ; for encoding of realm refer to [RFC4282]
extension = 1*(parm-chars)
parm-chars = %x20-2B | %x2D-7E
```

  For example: If the IMSI is
  214011001388741,
  then the value of the text in the User Data of the SMS message shall be
  214011001388741-rcscfg
  For example: If the private identity is
  214011001388741@ims.mnc001.mcc214.3gppnetwork.org,
  the value in the User Data of the SMS message shall be
  214011001388741@ims.mnc001.mcc214.3gppnetwork.org-rcscfg
  For example: If the private identity is
  214011001388741@ims.mnc001.mcc214.3gppnetwork.org, and
  the FQDN of the Configuration Server is "cfg.operator.com"
  then the value of the text in the User Data of the SMS message shall be
  214011001388741@ims.mnc001.mcc214.3gppnetwork.org-
  rcscfg,fqdn=cfg.operator.com

If the client receives a SMS message encoded as defined above prior to a first configuration and

- it contains an IMSI value and the value received in the request matches the IMSI of the SIM, and
- it does not contain a fqdn value,

then the client shall perform a HTTP configuration as per defined in section 2.6, 2.7 or 2.13 depending on client capabilities and current connectivity. The client shall discover the default Configuration Server address as defined in section 2.2.1.

If the client receives a SMS message encoded as defined above any time after the first configuration and the client is configured for IMS-based services and it contains an IMPI value and the value received in the request matches the IMS Private User Identity of the IMS client configuration (see [GSMA PRD-RCC.15]) then

- if the fqdn value does not contain a value, then the client shall perform a HTTP configuration as per defined in section 2.6, 2.7 or 2.13 depending on client capabilities and current connectivity. The client shall use the Configuration Server address discovered in accordance with the definitions in section 2.2.1,
- otherwise, if the fqdn value does contain a value and the value matches an fqdn value received from the default Configuration Server in a fqdn value of the SERVER characteristic as defined in section 4.2, then the client shall perform a HTTP configuration as defined in section 2.6, 2.7 or 2.13 depending on client capabilities and current connectivity using the fqdn value,

otherwise, the client shall ignore the SMS message.

If the client receives a SMS message encoded as defined above any time after the first configuration and the client it is not configured for IMS-based services and it contains an IMSI value and the value received in the request matches the IMSI of the SIM, then

- if the fqdn value does not contain a value, then the client shall perform a HTTP configuration as per defined in section 2.6, 2.7 or 2.13 depending on client capabilities and current connectivity. The client shall use the Configuration Server address discovered in accordance with the definitions in section 2.2.1,
- otherwise, if the fqdn value does contain a value and the value matches an fqdn value received from the default Configuration Server in a fqdn value of the SERVER characteristic as defined in section 4.2, then the client shall perform a HTTP configuration as per defined in section 2.6, 2.7 or 2.13 depending on client capabilities and current connectivity using the fqdn value,

otherwise, the client shall ignore the SMS message.

4 Configuration document

4.1 Configuration XML Document formatting

The configuration data and configuration control data will be represented in a configuration XML document conforming to the provisioning document type definitions of [OMA CP Cont].

The configuration XML document is conveyed to the client in the body part of a HTTP 200 OK response in accordance with the definitions in section 2. The content-type parameter contained in the HTTP 200 OK response shall indicate "*text/vnd.wap.connectivity-xml*".

The use of the Service Provider Device Configuration for the configuration of a service requires the definition of the service specific characteristics and parameters for representation in the provisioning document structure. This may be done through the definition of an AC for the provisioning document or the mapping from Management Objects (MO) of existing services to the provisioning document.

The mapping of MOs defined via the OMA DM DDF (Device Description Framework) as specified in [OMA DM DDF] to a provisioning document following the document type definition of [OMA CP Cont] is described in Annex A of this document.

If the default Configuration Server has enabled additional Configuration Servers then the client shall manage and store configuration XML document per Configuration Server. The client shall store the configuration data and make it available to the client services identified by the app-id value of the individual ACs. It is permitted that different configuration services provide configuration data for the same AC. It is left to the application implementation (the service identified by the app-id) to deal with multiple occurrences of the same Application Characteristic.

4.2 Characteristics of the Service Provider Device Configuration

In addition to the parameters and characteristic types provided by the mapping or definition of MOs as presented in the previous section, the following characteristic types for the client configuration control have been defined in this specification:

- Characteristic of type VERS

  The parameters of the VERS characteristic provide the version control of configuration XML documents or can be used to reset clients to default behaviour. The VERS characteristic shall be present in configuration documents controlling the client configuration as defined in section 2.4.1. The VERS characteristic shall be absent in configuration documents indicating a policy during the client provisioning procedure as defined in section 2.6.2.
  If the default Configuration Server has enabled additional Configuration Servers then the client shall manage and store the parameters of the VERS characteristic on a Configuration Server basis.
  The characteristic of type VERS can contain the following parameters:

  - version
    The value of the version parameter provides the version of the configuration document or indicates that the client shall apply default behaviour for services. The client shall store the value contained in the configuration XML document and send it as value of the vers parameter of the next configuration request to the Configuration Server. It can be used by the Configuration Server to evaluate whether an update of configuration data is required while processing of the configuration request.
    A positive integer value indicates the version of the configuration XML document.

The value is opaque for the client.
Version values other than positive integer indicate that default behaviour shall be applied for services with the following values:

- A value 0 indicates that the client shall provide the default behaviour for services. The client shall perform a client configuration request if the device is booted or the client is re-started. The value "0" is also the initial value of the version parameter if no configuration data is available (e.g. out of the box).
- A value -1 indicates that the client shall provide the default behaviour for services. The client shall not perform client configuration requests anymore.
- A value -2 indicates that the client shall provide the default behaviour for services. The client shall trigger a configuration request if a user action is detected that requires the reverted services to be active.

If the value of the version parameter indicates that the client shall apply the default behaviour for services, then

- if client configuration data is stored locally, then the client shall remove all configuration data from its local store.
- apply the default behaviour as defined in the corresponding service documentation.
- the client shall apply triggers for client configuration as defined for the received version value.
- the client shall apply the procedures for the client configuration defined for SIM swap and reset of the client defined in section 4.3.
- the client shall invoke network requested configuration requests if triggered.

If the Service Provider has enabled additional Configuration Servers, then the client shall apply the default behaviour defined for the version values for the services authorised by the Service Provider for the Configuration Server from which the configuration XML document is received. Otherwise, the client shall apply the behaviour defined for version values for all services for which the client supports client configuration via the Service Provider Device Configuration.

- validity

The value of the validity parameter indicates the validity of the configuration in seconds. The validity is counted from the time it is received by the client in the configuration XML document. The validity counter is reset using the value whenever received in a configuration XML document.
If the validity expires, then the client shall invoke a new configuration request to check for configuration changes and to refresh the validity.
If the default Configuration Server has enabled additional Configuration Servers then the validity shall be managed per Configuration Server.
If the Configuration Server has reverted services to their default behaviour using a zero or negative integer value in the version parameter, then the value of validity is not relevant for processing and shall be set to the same value as the version parameter.

- Characteristic of type TOKEN

  The parameters of the TOKEN characteristic provide the client with user identification data to be used, if available, when requesting configuration data via non-3GPP access networks.
  The TOKEN characteristic is optional in configuration XML documents conveyed via the mechanism defined in this specification. The TOKEN characteristic shall be absent in configuration documents indicating a policy during the client provisioning procedure as defined in section 2.6.2.
  If the default Configuration Server has enabled additional Configuration Servers then the client shall manage and store the parameters of the TOKEN characteristic on a Configuration Server basis.
  The characteristic of type VERS can contain the following parameters:

  - token
    The token parameter shall be present if the TOKEN characteristic is included in the configuration XML document.
    The value of the token parameter contains a value generated by the Configuration Server.
    The client shall store a received token value locally. A new received token value shall overwrite an existing token value from the previous configuration response.
    The client shall use the latest token value in configuration request as value of the request parameter "token", where applicable.
    If the default Configuration Server has enabled additional Configuration Servers then the token shall be stored per Configuration Server.
    The token value shall be removed together with the rest of the configuration when the device or client is reset or at the time of SIM swap.

- Characteristic of type ACCESS-CONTROL

  The ACCESS-CONTROL characteristic provides the client with access control data for the default Configuration Server and for additional Configuration Servers.
  The ACCESS-CONTROL characteristic is optional in configuration XML documents received in a client configuration response in result of a client configuration request to the default Configuration Server. The ACCESS-CONTROL characteristic shall be absent in all other configuration responses.
  The client shall delete any data derived from the ACCESS-CONTROL characteristic at the time of client reset (e.g. factory reset) or at SIM change.
  The characteristic of type ACCESS-CONTROL cannot contain parameters.
  The characteristic of type ACCESS-CONTROL can contain the following characteristics.

  - Characteristic of type DEFAULT

    The DEFAULT characteristic provides the client with information about the APPID for ACs or Management Object Identifier provided by the default Configuration Server. APPID for ACs and Management Object Identifier are assigned by the OMNA.

The DEFAULT characteristic shall be present if the ACCESS-CONTROL characteristic is present in the configuration XML document.
The characteristic of type DEFAULT can contain the following parameters:

- app-id

  The app-id parameter provides the value of an APPID for ACs or Management Object Identifier for which the default Configuration Server will provide client configuration data. APPID for ACs and Management Object Identifier are assigned by the OMNA.
  The app-id parameter can occur zero or multiple times, each representing one APPID value.

  Example of app-id parameter values:
  <parm name=”app-id” value=”ap2204”/>
  <parm name=”app-id” value=”urn:foo:mo:bar:1.0”/>

- Characteristic of type SERVER

The SERVER characteristic provides the client with the authorisation data for one additional Configuration Server, i.e. each additional Configuration Server is represented via a dedicated SERVER characteristic. At least one SERVER characteristic shall be present if the ACCESS-CONTROL characteristic is present in a configuration XML document.
The characteristic of type SERVER can contain the following parameters:

- fqdn

  The fqdn parameter value provides the FQDN of the additional Configuration Server.

  Example:
  <parm name=”fqdn” value=”config.provider.com”/>

- app-id

  The app-id parameter provides the value of an APPID or Management Object Identifier for which the additional Configuration Server is authorised to manage client configuration data. APPID for ACs and Management Object Identifier are assigned by the OMNA.
  The app-id parameter can occur one or multiple times, each representing one APPID value.

  Example:
  <parm name=”app-id” value=”ap2204”/>
  <parm name=”app-id” value=” urn:foo:mo:bar:1.0”/>

- id-provider

  The parameter indicates that the default Configuration Server has authorized an additional Configuration Server to provide user related identity information

in a USER characteristic. The id-provider must be only be present for one SERVER characteristic provided by the default Configuration Server.
The "id-provider" parameter has a fixed value of "1"

Example:
<parm name=”id-provider” value=”1”/>

If the client receives in a configuration XML document from the default Configuration Server with an "id-provider" parameter in a given SERVER characteristic for an additional Configuration Server while having not stored the "id-provider" parameter for this Configuration Server, then the client shall invoke a configuration request to this additional Configuration Server. In result of the Configuration Server response processing the client shall overwrite the locally stored user identity data with the data received in the USER characteristic.
If the client receives in a configuration XML document from the default Configuration Server all SERVER characteristics for additional Configuration Servers without a "id-provider" parameter while having stored an "id-provider" parameter for an additional Configuration Server, then the client shall overwrite the locally stored user identity data with the data received from the default Configuration Server.

- Characteristic of type USER

  The USER characteristic provides the client with identification data of the user.
  The USER characteristic is optional in configuration documents controlling the client configuration as defined in section 2.4.1.
  The USER characteristic is mandatory in configuration responses from the default Configuration Server if

  - no ACCESS-CONTROL characteristic is present in the configuration XML document or
  - there is an ACCESS-CONTROL characteristic with one or more SERVER characteristics present in the configuration XML document but none containing an "id-provider" parameter.

  The client shall ignore a USER characteristic if present in a configuration XML document received from the default Configuration Server if there is a SERVER characteristic present in the configuration XML document containing an "id-provider" parameter.
  The USER characteristic is mandatory in configuration responses from an additional Configuration Server if the SERVER characteristics related to this Configuration Server contains an "id-provider" parameter.
  The client shall ignore a USER characteristic if present in a configuration XML document received from an additional Configuration Server, if the locally stored Configuration Server data from the corresponding SERVER characteristic does not contain the "id-provider" parameter.
  The client shall delete any data derived from the USER characteristic at the time of client reset (e.g. factory reset) or change of SIM.

The characteristic of type USER can contain the following parameters:

- msisdn

  The parameter provides the client with the user's basic MSISDN. The MSISDN value shall be provided in international E.164 format, i.e. with no leading "+".

  Example:
  <parm name=”msisdn” value=”491711234567”/>

- Characteristic of type MSG

The parameters of the MSG characteristic provide the Configuration Server with the capability to convey a message to the device user. Usage of the parameters for MSG characteristics is defined in section 2.4.2.
The MSG characteristic is optional in configuration XML documents conveyed via the mechanism defined in this specification. The MSG characteristic shall be absent in configuration documents indicating a policy during the client provisioning procedure as defined in section 2.6.2.
If the default Configuration Server has enabled additional Configuration Servers then the client shall manage the parameters of the MSG characteristic on a Configuration Server basis.

- Characteristic of type MO_CONFIGURATION

The MO_CONFIGURATION characteristic provides the client the needed information for the client to trigger the Mobile Originated SMS procedure as described in section 2.6.1.1.The characteristic of type MO_CONFIGURATION can contain the following parameters:

- sms_message_content

  The payload of the SMS to be sent automatically (contains an OTP).
  Example:
  <parm name="sms_message_content" value="Verifying your number for RCS service 1092309aazxclkjljoiear"/>

- destination_phone_number

  Destination phone number to send the MO SMS to.
  Example:
  <parm name="destination_phone_number" value="+12345678901"/>

- sms_visiblity

  Indicates if the sms should be visible to the user. The values are “0” for not visible and “1” for visible.
  Example:
  <parm name="sms_visibility" value=”1”/>

- sms_ wait_interval_ms

Time in milliseconds the client shall wait before sending the second HTTP provisioning request after sending the automatic SMS obtained from the MO_CONFIGURATION parameters.
Example:
<parm name=”sms_wait_interval_ms” value=”30000”/>

- Characteristic of type POLICY

  The parameters of the POLICY characteristic provide the client with policies to be considered during the configuration procedure. Usage of parameters for POLICY characteristics is defined in section 2.6.2.
  The POLICY characteristic is optional in configuration XML documents conveyed via the mechanism defined in this specification. The POLICY characteristic shall be absent in configuration XML document controlling the client configuration as defined in section 2.4.1.
  If the default Configuration Server has enabled additional Configuration Servers then the client shall manage the parameters of the POLICY characteristic on a Configuration Server basis.
  The parameters of the POLICY characteristic are transient and shall not be stored by the client.

The following provides the definition of the data model of characteristics and parameters of the Service Provider Client Configuration using the notation of [OMA CP Cont].

```
characteristic : VERS ?
{
    parm: version
    parm: validity
    }

characteristic : TOKEN ?
{
    parm: token
    }

characteristic : ACCESS-CONTROL ?
{    characteristic: DEFAULT
     {
         parm: app-id *
     }
}
{
    characteristic : SERVER +
    {
        parm: fqdn
        parm: app-id +
........parm: id-provider ?
    }
}

characteristic : USER ?
{
```

```
        parm: msisdn
        }

    characteristic : MSG ?
    {
        parm: title
        parm: message
        parm: Accept_btn
        parm: Reject_btn ?
        }

    characteristic : MO_CONFIGURATION ?
    {
        parm: sms_message_content
        parm: destination_phone_number
        parm: sms_visibility
    }
    characteristic : POLICY ?
    {
        parm: SMS_port
        }
```

4.3 Configuration storage on the client

The client shall store service configuration and configuration control data received in the configuration XML document securely locally on the device.

If the client receives a full configuration XML document, then the client shall update the locally stored values of configuration parameters with the values received in the configuration XML document and apply the new values from this point in time onwards. Services controlled by the Service Provider Device Configuration may require the client to invoke service specific actions after the re-configuration of a configuration parameter. If a post re-configuration action is required for a configuration parameter, then the client shall determine at the time of processing of the full configuration XML document whether the value of a configuration parameter has been changed by comparing the old value (stored in the client local configuration) with the new value (received in the configuration XML document). If for the applied change of value a post re-configuration is required, the client shall invoke it accordingly. For definitions of post re-configuration actions for the transition of values of configuration parameters refer to the corresponding service documentation.

If any of the required parameters for a service are not configured or configured with an unexpected value, that service functionality should revert to default behaviour and not be presented as such to the user. This default behaviour needs to be defined for the individual services and might for example be to disable the service and its entry points. In this state, the full service functionality can only be restored by completing the first-time configuration procedure (see section 2).

If the client or device is reset (unpersonalised, reset to factory settings, etc.), then all configuration data stored on the client shall be removed. Once reset, the client or device shall apply for new configuration data as defined for clients with no valid configuration, starting with discovery of a Configuration Server.

If the client or device detects the removal of the SIM or that the SIM ready state has been left and it has configuration data stored locally associated with the identity of this SIM, then the client or device shall revert these services to their default behaviour or disable them.

If a SIM is entered again or the SIM enters the SIM ready state, then the client shall check the identity of the SIM.

If the identity of the SIM is not changed, then the client shall enable services associated with the SIM identity again. The stored configuration data remain applicable unless their validity is expired.

If the identity of the SIM is changed, then the client shall remove the stored configuration data, revert all services to their default behaviour and apply for new configuration data, starting with the discovery of a Configuration Server.

Clients functioning as secondary clients sharing the SIM identity used in a user's main device (see section 2.9) may offer multiple users the possibility to access the services (e.g. by requiring selecting which user to serve when started). In that case, the client shall store the service configuration data locally per user account.

4.4 Content Encoding

Content Encoding should be supported for efficient use of access network bandwidth for the transfer of the configuration XML document between the Configuration Server and the client.

In a HTTPs configuration request, the client should indicate the supported content decoding mechanisms via of the Accept-Encoding HTTP header as defined in [RFC2616].

Prior to sending of a configuration XML document in the HTTP 200 OK configuration response, the Configuration Server may apply a content encoding mechanism supported by the client. The Configuration Server shall indicate the applied content encoding mechanism in the Content-Encoding HTTP header in accordance with the definitions in [RFC2616].

It is recommended that clients and Configuration Servers support at least the encoding format "gzip" as described in [RFC2616].

5 Protocol version negotiation

5.1 Overview

This section defines a mechanism for version negotiation of the Service Provider Device Configuration. It is essential for clients and Configuration Servers to apply the same version of the protocol during the processing of Configuration Server request.

5.2 Version negotiation procedure

Clients and servers may support multiple versions of the Service Provider Device Configuration protocol. If the client initiates a client configuration request for the first time with a Configuration Server, it shall use the highest supported protocol version. It shall indicate the applied version in the value of the configuration request parameter "provisioning_version".

If the Configuration Server supports the version indicated by the client it shall process the request according to the procedures defined in this document. The Configuration Server may add to the configuration response a "Supported-Versions" HTTP header to indicate its supported protocol versions.

If the Configuration Server does not support the version indicated by the client, and if the value of the client configuration request parameter "provisioning_version" is "4.0" or higher, then it shall return to the client configuration request a 406 Not Accepted response. Otherwise, the Configuration Server shall return a 403 Forbidden response.

The Configuration Server may add to the response a "Supported-Versions" HTTP header indicating the versions of the protocol. The client may use the value of the header to adapt its configuration request processing to one of the versions supported by the Configuration Server and retry the request.

If the client receives a "Supported-Versions" HTTP header along with 200 OK or other responses from the Configuration Server, it may inspect the value and upgrade for the next configuration request to a higher version by adapting the behaviours and configuration request parameters accordingly.

5.3 Supported-Versions header

The Supported-Versions header may be used by the Configuration Server to indicate to the client the supported versions of the Service Provider Device Configuration in HTTP responses. It can be used to indicate multiple supported versions. If present in a HTTP response, it shall contain at least one version-tag. If absent in a HTTP response, the Configuration Server supports only the version indicated by the client in the configuration request.

The version-tag values are defined in the applicable versions of this document via the value of the "provisioning_version" configuration request parameter.

The Supported-Version HTTP header is defined using the notation of [RFC2616] as follows:

```
Supported-Versions = "Supported-Versions" ":" 1#version-tag
version-tag = 1*DIGIT "." DIGIT
Example:
Supported-Versions: 4.0, 5.0
```

6 Data Off

6.1 General Behaviour

If data access is unavailable for the client and a trigger for client configuration applies, then the client shall continue to apply the current configuration data for the configured services. Once the data access is available again, the client shall invoke the procedures for client configuration for the unprocessed triggers.

6.2 3GPP PS Data Off

A client embedded on a device supporting 3GPP PS Data Off as defined in [3GPP TS 22.011] shall support the 3GPP PS Data Off feature as follows.

If 3GPP PS Data Off is activated and device management over PS is not configured as a 3GPP PS Data Off Exempt Service and if a trigger for client configuration applies, then the client shall not send the client configuration request and shall apply the general behaviour for data off.

If 3GPP PS Data Off is activated and device management over PS is configured as a 3GPP PS Data Off Exempt Service, then the normal procedures for client configuration apply.

The Device Management over PS exemption for 3GPP PS Data Off is configured as a 3GPP PS Data Off Exempt Service via the following configuration parameters:

Configuration parameter	Description	Occurrence
Device Management over PS data off exemption	This parameter indicates whether device management is a cellular data off exempt services. The following values are defined: 0: device management is not defined as a cellular data off exempted services. 1: device management is defined as a cellular data off exempted services (default value). NOTE: When device management is disabled in cellular data access, then it is only possible to change the value when the device provisioning becomes enabled again, e.g. by connecting to non-cellular access.	Optional
Device Management over PS data off roaming exemption	This parameter indicates whether device management is a cellular data off exempt service when roaming. The following values are defined: 0: device management is not defined as a cellular data off exempted services when roaming. 1: device management is defined as a cellular data off exempted services when roaming (default value). NOTE: When device management is disabled in cellular data access, then it is only possible to change the value when the device provisioning becomes enabled again, e.g. by connecting to non-cellular access.	Optional

Table 26: Device Management Data Off configuration parameters

The configuration parameters defined in Table 26 are provided via the exempted service lists for 3GPP PS Data Off in the Non-Access Stratum Management Object defined in [3GPP TS 24.368].

The Non-Access Stratum Management Object is provided in the configuration XML document using the transformation of the OMA DM DDF format to the OMA-CP format defined in Annex A.

```
<characteristic type="APPLICATION">
    <parm name=”AppID” value=” urn:oma:mo:ext-3gpp-nas-config:1.0”/>
    <characteristic type="NODE">
        <characteristic type="3GPP_PS_data_off">
            <characteristic type="Exempted_service_list">
```

```
                <parm name=”Device_management_over_PS” value=”0”/>
                <parm name="Bearer_independent_protocol" value="0"/>
            </characteristic>
            <characteristic type="Exempted_service_list_roaming">
                <parm name=”Device_management_over_PS” value=”0”/>
                <parm name="Bearer_independent_protocol" value="0"/>
            </characteristic>
        </characteristic>
    </characteristic>
</characteristic>
```

Table 27: Example Non-Access Stratum Configuration Document Fragment

The client shall indicate the support of the Non-Access Stratum Management Object by inclusion of an "app" HTTP GET request configuration parameter with the value set to "urn:oma:mo:ext-3gpp-nas-config:1.0".

Annex A Mapping from OMA DM DDF format to OMA-CP format

A.1 General

This annex describes the transformation of a DM MO definition using the OMA DM Device Description Framework (DDF) to a provisioning document following the document type definition of [OMA CP Cont].

The DDF format is described in [OMA DM DDF]. This format is used in 3GPP to describe the Management Objects Tree.

A.2 Mapping from OMA DM DDF to OMA CP format

The rules to transform a DM MO definition using the format defined in [OMA DM DDF] to a provisioning document following the document type definitions of [OMA CP Cont] are described in Table 28 below.

Type	OMA DM DDF Format	Action	OMA CP Format
Root Node	<MgmtTree>	Create a Root node <characteristic type="APPLICATION">	<characteristic type="APPLICATION">
Root Interior Node	The root <Node> with <DFFormat> set to <node/> and a <DFType> set to an non empty <DDFName> containing the Management Object Identifier value as defined by the OMNA Example: <Node> <NodeName>…</NodeName> <DFProperties> … <DFFormat> <node/> </DFFormat> <DFType> <DDFName>XXX</DDFName> </DFType> </DFProperties> The root <Node> with <DFFormat> set to <node/> and a <DFType> set to an empty <DDFName>. The Management Object Identifier shall be taken from OMNA.	Create a node <parm name=" AppID " value=" XXX "/> where XXX is the OMNA registered Management Object Identifier value taken from <DDFName> or as defined in OMNA. In addition, create a node following the rules for the interior node as per definitions below.	<parm name="AppID " value="XXX"/>

Type	OMA DM DDF Format	Action	OMA CP Format
Interior Node Element #1	A <Node> with an non empty <NodeName> and a <DFFormat> set to <node/> Example: <Node> <NodeName>**XXX**</NodeName> <DFProperties> … <DFFormat> <node/> </DFFormat> </DFProperties>	Create a node <characteristic type="**XXX**"> where **XXX** is the node name.	<characteristic type="**XXX**">
Interior Node Element #2 (Runtime)	A <Node> with an empty <NodeName> and a <DFFormat> set to <node/> Example: <Node> <NodeName/> … <DFProperties> <DFFormat> <node/> </DFFormat> <Occurrence> <OneOrMore/> </Occurrence> … </DFProperties>	Create a node <characteristic type="**NODE**">	<characteristic type="**NODE**">

Type	OMA DM DDF Format	Action	OMA CP Format
Leaf Node	A <Node> with an non empty <NodeName> and a <DFFormat> different to <node/> Example: <Node> <NodeName>**XXX**</NodeName> <DFProperties> … <DFFormat> <int/> </DFFormat> <Occurrence> <One/> </Occurrence> … </DFProperties> </Node>	Create a node <parm name=" **XXX** " value=" "/> where **XXX** is the node name	<parm name="**XXX**" value=" "/>

Table 28: Mapping from OMA DM DDF format to [OMA CP Cont] provisioning document

The provisioning document fragment resulting from the transformation defined in Table 28 shall be embedded in a configuration XML document defined in section 4.

The presence of a Management Object Identifier as a value of the AppID parameter indicates that the mapping defined in this document has been applied.

A.3 Example

This section provides an example mapping of a DM MO definition to a provisioning document fragment to be embedded in a configuration XM document.

The example DM Management Object tree is depicted in Figure 18.

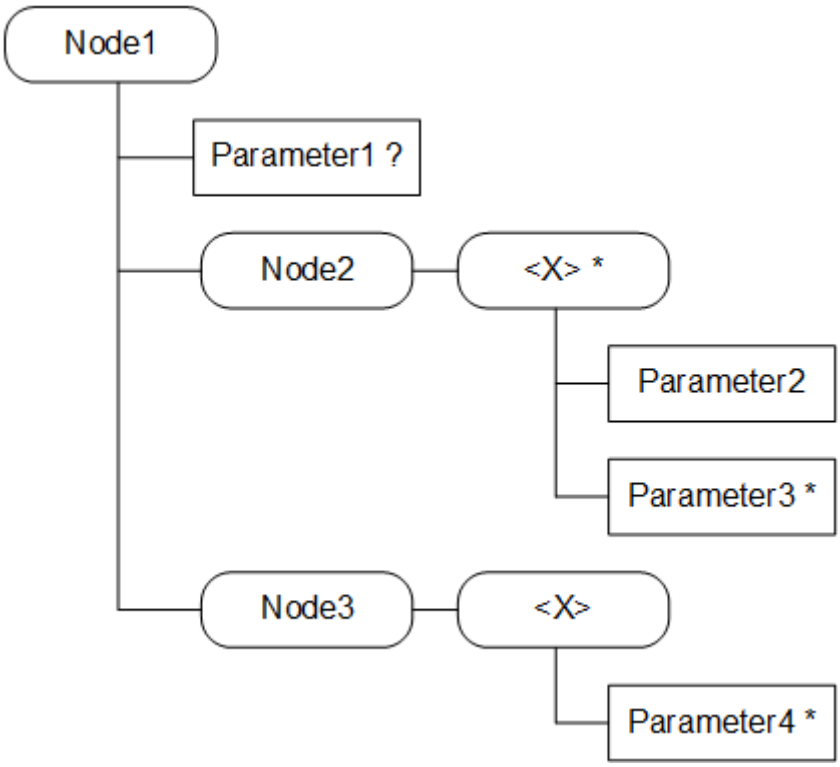

Figure 18: Example MO Tree

Table 29 depicts the DDF representing the example MO tree.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MgmtTree PUBLIC "-//OMA//DTD-DM-DDF 1.2//EN"
"http://www.openmobilealliance.org/tech/DTD/dm_ddf-v1_2.dtd">

<MgmtTree>
  <VerDTD>1.2</VerDTD>
  <Man>--The device manufacturer--</Man>
  <Mod>--The device model--</Mod>

  <Node>
    <NodeName>Node 1</NodeName>
    <DFProperties>
      <AccessType>
        <Get/>
      </AccessType>
      <Description>The Root Interior Node of the MO</Description>
      <DFFormat>
        <node/>
      </DFFormat>
      <Occurrence>
        <One/>
      </Occurrence>
      <Scope>
        <Permanent/>
      </Scope>
      <DFTitle>Example Node 1.</DFTitle>
      <DFType>
        <DDFName>urn:foo:mo:bar:1.0</DDFName>
      </DFType>
```

```
    </DFProperties>

    <Node>
      <NodeName>Parameter1</NodeName>
      <DFProperties>
        <AccessType>
          <Get/>
        </AccessType>
        <DFFormat>
          <int/>
        </DFFormat>
        <Occurrence>
          <ZeroOrOne/>
        </Occurrence>
        <Scope>
          <Permanent/>
        </Scope>
        <DFTitle>Parameter 1.</DFTitle>
        <DFType>
          <MIME>text/plain</MIME>
        </DFType>
      </DFProperties>
    </Node>

    <Node>
      <NodeName>Node2</NodeName>
      <DFProperties>
        <AccessType>
          <Get/>
        </AccessType>
        <DFFormat>
          <node/>
        </DFFormat>
        <Occurrence>
          <One/>
        </Occurrence>
        <Scope>
          <Permanent/>
        </Scope>
        <DFTitle>Node 2.</DFTitle>
        <DFType>
          <DDFName/>
        </DFType>
      </DFProperties>
      <Node>
        <NodeName/>
        <DFProperties>
          <AccessType>
            <Get/>
          </AccessType>
          <DFFormat>
            <node/>
          </DFFormat>
          <Occurrence>
            <OneOrMore/>
          </Occurrence>
          <Scope>
            <Dynamic/>
```

```
        </Scope>
        <DFTitle>The runtime Node in Node 2.</DFTitle>
        <DFType>
          <DDFName/>
        </DFType>
      </DFProperties>
      <Node>
        <NodeName>Parameter2</NodeName>
        <DFProperties>
          <AccessType>
            <Get/>
            <Replace/>
          </AccessType>
          <DFFormat>
            <int/>
          </DFFormat>
          <Occurrence>
            <One/>
          </Occurrence>
          <Scope>
            <Permanent/>
          </Scope>
          <DFTitle>Parameter 2.</DFTitle>
          <DFType>
            <MIME>text/plain</MIME>
          </DFType>
        </DFProperties>
      </Node>
      <Node>
        <NodeName>Parameter3</NodeName>
        <DFProperties>
          <AccessType>
            <Get/>
            <Replace/>
          </AccessType>
          <DFFormat>
            <int/>
          </DFFormat>
          <Occurrence>
            <ZeroOrMore/>
          </Occurrence>
          <Scope>
            <Permanent/>
          </Scope>
          <DFTitle>Parameter 3.</DFTitle>
          <DFType>
            <MIME>text/plain</MIME>
          </DFType>
        </DFProperties>
      </Node>
    </Node>
  </Node>

  <Node>
    <NodeName>Node3</NodeName>
    <DFProperties>
      <AccessType>
        <Get/>
```

```
      </AccessType>
      <DFFormat>
        <node/>
      </DFFormat>
      <Occurrence>
        <One/>
      </Occurrence>
      <Scope>
        <Permanent/>
      </Scope>
      <DFTitle>Node 3.</DFTitle>
      <DFType>
        <DDFName/>
      </DFType>
    </DFProperties>
    <Node>
      <NodeName/>
      <DFProperties>
        <AccessType>
          <Get/>
        </AccessType>
        <DFFormat>
          <node/>
        </DFFormat>
        <Occurrence>
          <One/>
        </Occurrence>
        <Scope>
          <Dynamic/>
        </Scope>
        <DFTitle>The runtime Node in Node 3.</DFTitle>
        <DFType>
          <DDFName/>
        </DFType>
      </DFProperties>
    <Node>
        <NodeName>Parameter4</NodeName>
        <DFProperties>
          <AccessType>
            <Get/>
            <Replace/>
          </AccessType>
          <DFFormat>
            <int/>
          </DFFormat>
          <Occurrence>
            <ZeroOrMore/>
          </Occurrence>
          <Scope>
            <Permanent/>
          </Scope>
          <DFTitle>Parameter 4.</DFTitle>
          <DFType>
            <MIME>text/plain</MIME>
          </DFType>
        </DFProperties>
      </Node>
    </Node>
```

```
    </Node>
  </Node>
</MgmtTree>
```

Table 29: DDF of the Example MO Tree

The assigned Management Object Identifier of the example MO tree is "urn:foo:mo:bar:1.0". The example provisioning document fragment is generated with the following instantiation data.

Parameter	Value
Parameter1	12
Node2 parameter values	
Parameter2	35
Parameter3	8
Parameter3	17
Node2 parameter values	
Parameter2	30
Parameter3	1
Node3 parameter values	
Parameter4	22
Parameter4	4
Parameter4	66

Table 30: Example Instantiation Data

The application of the mapping defined in section A.2 using the DDF shown in Table 29 with the instantiation data shown in Table 30 results in the provisioning document fragment shown in Table 31.

```
<characteristic type="APPLICATION">
    <parm name="AppID" value="urn:foo:mo:bar:1.0"/>
    <characteristic type="Node1">
        <parm name="Parameter1" value="12"/>
        <characteristic type="Node2">
            <characteristic type="NODE">
                <parm name="Parameter2" value="35"/>
                <parm name="Parameter3" value="8"/>
                <parm name="Parameter3" value="17"/>
            </characteristic>
            <characteristic type="NODE">
                <parm name="Parameter2" value="30"/>
                <parm name="Parameter3" value="1"/>
            </characteristic>
        </characteristic>
        <characteristic type="Node3">
            <characteristic type="NODE">
                <parm name="Parameter4" value="22"/>
                <parm name="Parameter4" value="4"/>
                <parm name="Parameter4" value="66"/>
            </characteristic>
```

```
        </characteristic>
    </characteristic>
</characteristic>
```

Table 31: Example provisioning document

Annex B User Authentication via HTTP Digest AKA

B.1 Overview

This section defines the procedure for clients to authenticate the user using HTTP Digest AKA as defined in [RFC4169]. Authentication and Key Agreement (AKA) is used between clients and servers for authentication and session key distribution using a challenge-response based mechanism with symmetric cryptography. AKA is performed in 3GPP between the Authentication Centre (AuC) and the USIM or ISIM.

The procedure defined in this section is applicable for the client when connected to an authentication endpoint in the network supporting HTTP Digest AKA. To support HTTP Digest AKA the client need to have access to the USIM or ISIM to request the AKA procedure. Device implementations shall ensure that only trusted clients have access to the USIM and ISIM to request the AKA procedure, e.g. based on system level certification.

The authentication endpoint in the network must have access to the 3GPP Authentication Center (AuC) to retrieve authentication vectors. The network internal procedures are out of scope of this document.

B.2 HTTP Digest AKA procedure

The procedure for HTTP Digest AKA applies as defined in [RFC4169] with the following additions.

A client supporting HTTP Digest AKA shall, if connected to an endpoint in the network via a secured HTTP connection (i.e. HTTPS), add to HTTP Requests an "Authorization" header with the following parameters:

- the "username" parameter set to the private user identity of the SIM as defined in [3GPP TS 23.003],
- the "uri" parameter set to the home network domain name of the SIM as defined in [3GPP TS 23.003],
- a "nonce" parameter with an empty value,

NOTE: the “nonce” discussed in section 2.11 is not included here. It is meant for use in service-related protocols rather than the configuration. Its use for configuration is for further study.

- a "response" parameter with an empty value.

In accordance with [RFC4169], the HTTP Digest AKA authentication is invoked by the authentication endpoint in result of the HTTP request via a 401 Unauthorized response containing a WWW-authenticate header with the Digest scheme and an "algorithm" parameter value indicating the request for AKA. The value of the "realm" parameter shall

contain the home network domain name of the network as defined in [3GPP TS 23.003]. The value of the "qop" parameter shall be set to "auth-int".

The HTTP Digest AKA procedure commences as defined in [RFC4169]. Authentication results, authentication errors and synchronisation failures shall be handled by the client and the authentication endpoint as defined in [RFC4169].

If the HTTP Digest AKA procedure is invoked by the authentication endpoint in the context of an OpenID Connect authentication request as defined in section 2.8, then the HTTP Digest AKA success or error response and the Authentication-Info header may be conveyed to the client via the HTTP 302 Found response to continue processing of the OpenID Connect authentication request.

Annex C User Authentication via HTTP Embedded EAP-AKA

C.1 Overview

This section defines the procedure for clients to authenticate the user using formats of the Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) as defined in [RFC4187] embedded in HTTP. Authentication and Key Agreement (AKA) is used between clients and servers for authentication and session key distribution using a challenge-response based mechanism with symmetric cryptography. AKA is performed in 3GPP between the Authentication Centre (AuC) and the USIM or ISIM.

The procedure defined in this section is applicable for the client when connected to an authentication endpoint in the network supporting HTTP embedded EAP AKA or an EAP relay endpoint. To support EAP-AKA the client needs to have access to the USIM or ISIM to request the AKA procedure. Device implementations shall ensure that only trusted clients have access to the USIM and ISIM to request the AKA procedure, e.g. based on system level certification.

The authentication endpoint in the network must have access to the 3GPP Authentication Center (AuC) to retrieve authentication vectors. The network internal procedures are out of scope of this document.

C.2 Transport of EAP packets over HTTP

This section defines a transport mechanism to allow clients and servers to exchange EAP packets in HTTP requests and responses. The following data elements are defined:

- EAP Identity

  This data element is sent from the client to the Configuration Server in HTTP requests to indicate support of EAP embedding in HTTP and to provide the authentication identity

- EAP packet relay

  This data element is sent from the client to the Configuration Server or from the Configuration Server to the client to relay an EAP packet via HTTP.

The EAP Identity element is defined as a URI query parameter in the URI query component as defined in [RFC3986]. The parameter is defined as follows:

Parameter	Description	Format

EAP_ID	The presence of the parameter indicates that the client supports the procedures for HTTP embedded EAP-AKA. The value of the parameter contains the authentication identity for EAP AKA. It shall take the value of the Root NAI for EAP AKA as defined in [3GPP TS 23.003].
	Example
	IMSI:
	`262011234567890`
	Root NAI for EAP AKA:
	`0262011234567890@nai.epc.mnc001.mcc262.3gppnetwork.org`
	URI query parameter:
	`EAP_ID=0262011234567890%40nai.epc.mnc001.mcc262.3gppnetwork.org`
		String

Table 32: EAP_ID Query URI parameter

The EAP packet relay element is defined as an HTTP entity body containing a JSON object. The content type of the HTTP entity body is defined as:

`application/vnd.gsma.eap-relay.v1.0+json`

The JSON schema of the EAP packet relay HTTP entity body is defined as follows:

```
{
   "$schema": "http://json-schema.org/draft-06/schema#",
   "title": "EAP packet relay",
   "description": "An object to relay EAP packets",
   "type": "object",
   "properties": {
      "eap-relay-packet": {
         "description": "Contains an EAP packet in base64 encoding",
         "type": "string"
      }
   },
}
```

Table 33: EAP packet relay schema

The property "eap-relay-packet" is used to convey an EAP packet in base64 encoding.

C.3 HTTP Embedded EAP-AKA procedure

The embedding of the EAP-AKA procedure in HTTP is based on the relay of EAP-AKA messages defined in [RFC4187] in HTTP requests responses using the EAP packet transport defined in section C.2.

EAP AKA message according to [RFC4187]	HTTP Embedding
EAP-Request/AKA-Identity	This message is not applicable for HTTP embedding. The HTTP embedding of EAP-AKA is based on the use of the permanent identity, i.e. a request for AT_PERMANENT_ID_REQ is assumed. The client shall provide its permanent identity unsolicited in the HTTP request to be authenticated.
EAP-Response/AKA-Identity	The content of the AT_IDENTITY attribute is conveyed in the EAP_ID request URI parameter defined in section C.2
EAP-Request/AKA-Challenge	This message is conveyed in the EAP payload HTTP body defined in section C.2
EAP-Response/AKA-Challenge	This message is conveyed in the EAP payload HTTP body defined in section C.2
EAP-Response/AKA-Authentication-Reject	This message is not applicable for HTTP embedding. The client shall abort the authentication procedure if the Configuration Server authentication fails.
EAP-Response/AKA-Synchronization-Failure	This message is conveyed in the EAP payload HTTP body defined in section C.2
EAP-Request/AKA-Reauthentication	This message is not applicable for HTTP embedding. Fast reauthentication is not supported for HTTP embedding.
EAP-Response/AKA-Reauthentication	This message is not applicable for HTTP embedding. Fast reauthentication is not supported for HTTP embedding.
EAP-Response/AKA-Client-Error	This message is not applicable for HTTP embedding, only the use of the permanent identity is supported.
EAP-Request/AKA-Notification	This message is not applicable for HTTP embedding.
EAP-Response/AKA-Notification	This message is not applicable for HTTP embedding.
EAP-Success	This message is not applicable for HTTP embedding. The success of the authentication is provided implicitly by the Configuration Server when providing the resource requested by the client in the initial HTTP request.
EAP-Failure	This message is not applicable for HTTP embedding. The failure of the authentication is provided implicitly by HTTP authentication failure error responses as defined for the HTTP GET processing.

Table 34: Mapping of EAP messages to HTTP

A client supporting HTTP embedded EAP-AKA, shall, if sending an HTTP GET to request a resource from an endpoint in the network via a secured HTTP connection (i.e. HTTPS), add to Request URI its permanent identity. The permanent identity shall be included in the EAP_ID request parameter defined in section C.2 by appending it to the query component of the URI using the `application/x-www-form-urlencoded` format as defined in [HTML-4.0]. If no query component exists, the client shall add one first, in accordance with the definitions in [RFC3986].

The endpoint in the network invokes the procedures for EAP-AKA in accordance with [RFC4187] if user authentication is required. To send an EAP message to the client, the network shall

- create an HTTP 200 OK response,
- create an EAP relay object by including the EAP message using the format defined in section C.2,
- add the EAP relay object as an HTTP entity body and append it to the HTTP 200 OK response,
- add a Content-Type header set to "`application/vnd.gsma.eap-relay.v1.0+json`", and
- send it to the client.

On reception of the 200 OK with a Content-Type header set to "`application/vnd.gsma.eap-relay.v1.0+json`" the client shall process the appended HTTP entity body using the format defined in section C.2 to extract the EAP message and process it in accordance with [RFC4187].

If the processing results in an EAP response message, then the client shall

- create an HTTP POST request
- create an EAP relay object by including the EAP response message using the format defined in section C.2,
- add the EAP relay object as an HTTP entity body and append it to the HTTP POST request,
- add a Content-Type header set to "`application/vnd.gsma.eap-relay.v1.0+json`", and
- send the HTTP POST request to the network using the same URI as used for the HTTP request for which the EAP relay body containing an EAP message was received.

Success and failure responses of the HTTP embedded EAP-AKA procedure are conveyed to the client in accordance with the definitions for the initial HTTP GET request, e.g. the client configuration response as defined in section 2.4.

The client shall support for the HTTP embedded EAP-AKA authentication flow the procedures for HTTP state management defined in [RFC6265]. This allows the authentication endpoint to return in HTTP responses a Set-Cookie header. The client shall apply the parsing and storage procedures of the Set-Cookie header as defined [RFC6265]. It shall send the cookie header in HTTP requests to the OpenID Connect authentication

endpoint respecting the cookie attributes in the Set-Cookie header in accordance with [RFC6265].

Annex D Reference end-to-end flow for configuration using Temporary Token (Informative)

This section provides a reference implementation of the end-to-end flow described in section 2.13. It utilizes HTTP embedded EAP-AKA for SIM authentication to generate a one-time and short-lived Temporary Token which is used by the Configuration Server to validate and complete the configuration request.

Below are the steps, also shown in Figure 19:

1. The client initiates the EAP-AKA procedure with the EAP-AKA authentication endpoint, using HTTP embedded EAP-AKA as defined in in section C.3
2. The EAP-AKA authentication endpoint invokes the procedures for EAP-AKA, as defined in section C.3.
3. The client processes the response by extracting the EAP message and processing it, as defined in section C.3.
4. The client sends a HTTP POST request with an EAP response to EAP-AKA authentication endpoint, as defined in section C.3.
5. Upon success, the EAP-AKA authentication endpoint issues a one-time use, short-lived Temporary Token to the client.
6. The client starts the configuration POST request and includes a body part with Temporary Token as as_temp_token, as specified in section 2.13.
7. The Configuration Server contacts the token validation endpoint (e.g. Entitlement Configuration Server) to validate the token.
8. The token validation endpoint validates the token and responds with user information such as MSISDN, as specified in section 2.13
9. The Configuration Server shall return a configuration XML document with a HTTP 200 OK response.

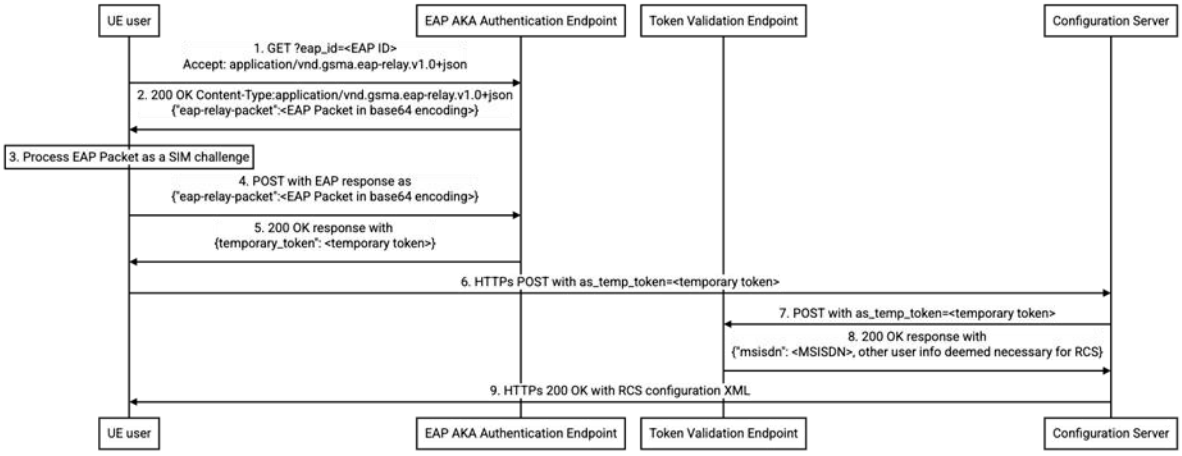

Figure 19: configuration using Temporary Token end-to-end flow

Annex E Deprecated authentication methods

The following authentication methods were specified in earlier versions of this specification but are no longer supported. They are documented here only for interaction with legacy clients.

E.1 Authentication based on cellular access

E.1.1 Overview

The Service Provider is able to implement the necessary procedures to resolve a user's Mobile Subscriber Integrated Services Digital Network Number (MSISDN) (e.g. via Remote Authentication Dial In User Service (RADIUS) requests, header enrichment) from the security association that exists for the bearer on which the client configuration request is applied.

If, depending on implementation concept, the client is not aware of the network access that it is using, then it shall follow the configuration request handling as defined in this section. This allows the Configuration Server to determine the connection status and request an alternative method of cellular access is not available.

E.1.2 Initial Request using cellular access

Network deployments may require an initial unsecured HTTP request to perform the procedures for user identification. Therefore, if connected to a cellular network the client shall always send prior to the configuration request an unsecured HTTP GET request using the FQDN discovered for a Configuration Server and for a URI with no path and query element to the Configuration Server.

As a result of the processing of the unsecured HTTP GET request, the Configuration Server may return:

- a HTTP 200 OK response with a SetCookie header. If received by the client, it shall initiate the configuration request as defined in section 2.4 with the additional parameters defined in Table 35 and respecting the value of received in the SetCookie header.

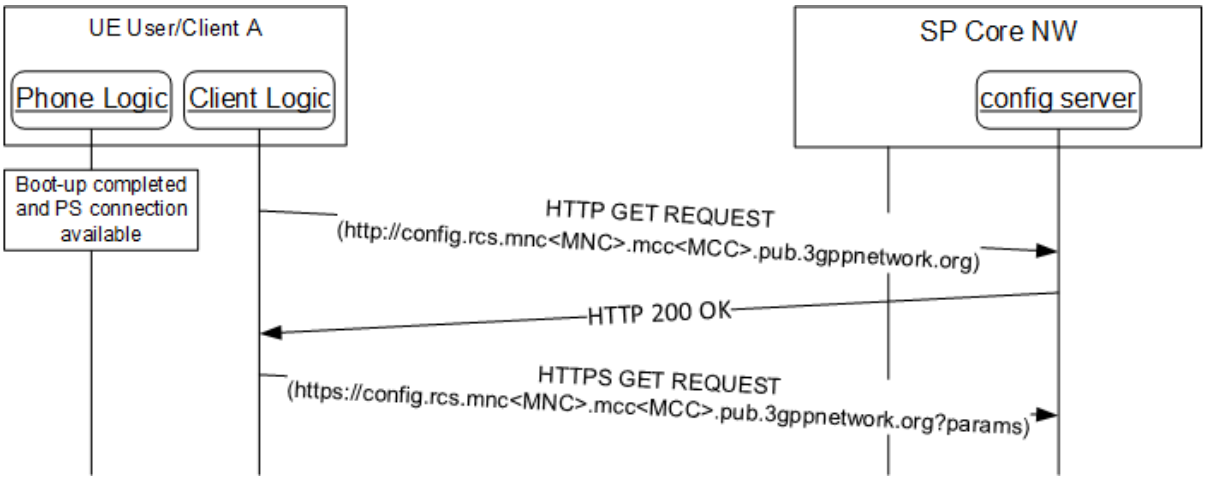

Figure 20: HTTP configuration: Initial requests

The Configuration Server should be able to correlate both HTTP and HTTPS requests from the same device via the Cooke provided by the client in the configuration request.

- a HTTP 511 Network Authentication Required response without a SetCookie header. If received, the client shall invoke procedure for client configuration as defined in section 2.6 or 2.13 for the case where the previous Configuration Server response did not include a cookie.
- a HTTP 511 Network Authentication Required response with a SetCookie header. If received, the client shall invoke procedure following the procedure defined in section 2.6 or 2.13 for the case where the previous Configuration Server response did contain a cookie (for a description of the security enhanced client configuration refer to section E.1.3).

Parameter	Description	Mandatory	Format
IMSI	If available, the subscriber's IMSI shall be sent as a parameter.	N if the OS platform allows it, it shall be included	String (15 digits)

Table 35: Additional configuration request parameters for primary devices over cellular access

E.1.3 Security considerations

For terminals carrying the SIM associated to the user's main identity the connection is carried out over the PS access network, therefore the current design reduces the risk of a man-in-the-middle attack whereby a third party is able to impersonate the Configuration Server.

To secure interoperability between Service Providers and to reduce complexity on the device/client, the Configuration Server shall make use of public root certificates issued by a recognized Certification Authority (CA). That is, the root certificates are similar to those used by standard webservers that are widely recognized by browsers and web-runtime implementations both in PCs and in devices.

To address security concerns due to mobile application system vulnerabilities (e.g. provisioning of malicious applications that appear to the Configuration Server as "trusted" applications where that authenticity isn't verified as specified in section 2.11), the security enhanced configuration mechanism could be implemented. In that case, the procedures to resolve the user's MSISDN (e.g. via RADIUS requests or header enrichment) shall be used only for acquiring the user's MSISDN and not for verifying the user's identity. Specifically, an HTTP 511 NETWORK AUTHENTICATION REQUIRED response shall be generated by the Configuration Server that contains a cookie as part of the response to the initial HTTP request (Set-Cookie header). The client shall then initiate the configuration mechanism as defined for non-3GPP access (see sections 2.6 and 2.13) without requesting the user to provide its MSISDN. The Configuration Server shall expect the client to provide that cookie in the subsequent HTTPS request (in the Cookie header).

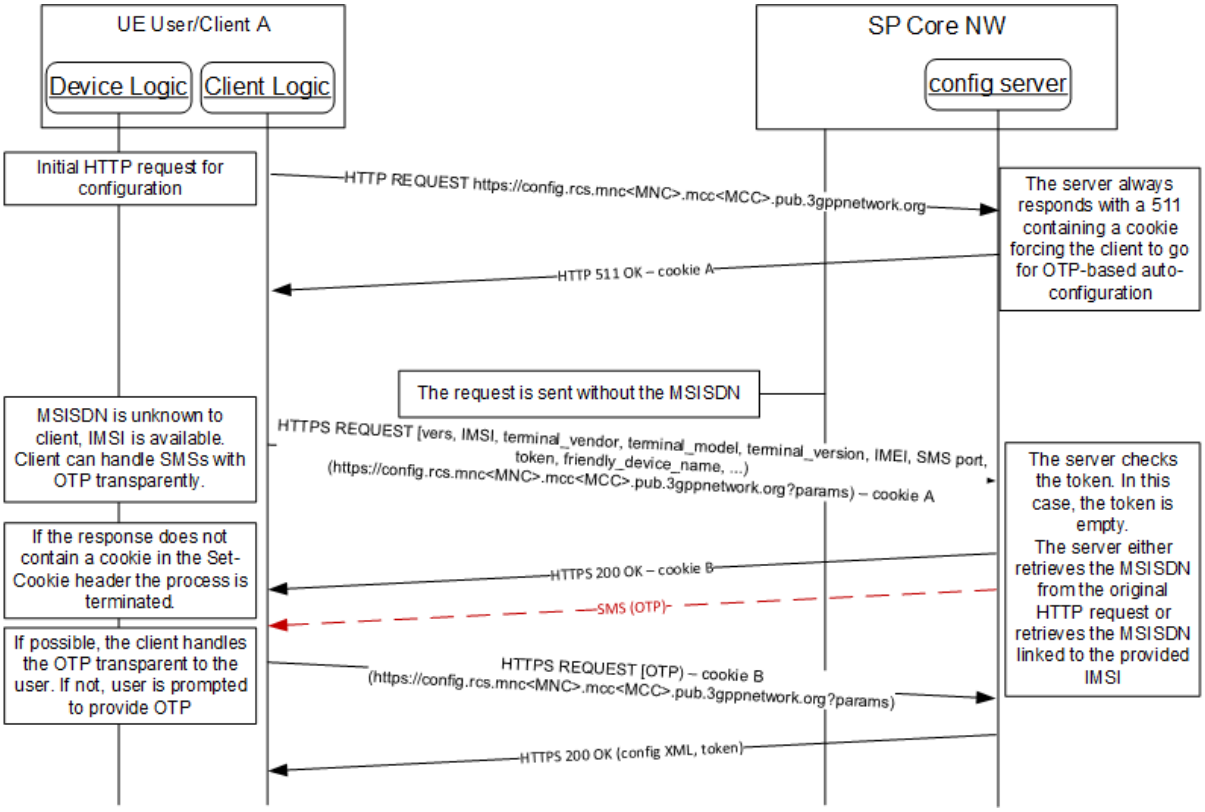

Figure 21: HTTP Configuration: Security enhanced

NOTE: The Service provider shall be able to select between the standard and security enhanced configuration mechanism. It is up to the Service Provider policy to select the most appropriate configuration mechanisms for particular configuration requests.

Annex F Document Management

F.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	02 February 2015	initial version split of from RCC.07 v5.0 to allow for more generic use	PSMC	Tom Van Pelt / GSMA
2.0	28 February 2015	Token characteristic in response over 3GPP, clarifications in response codes and description of renegotiation of bootstrapped security association	PSMC	Tom Van Pelt / GSMA
3.0	21 March 2016	Include approved CR1003	PSMC	Tom Van Pelt / GSMA
4.0	26 February 2017	Include approved CR1004	PSMC	Tom Van Pelt / GSMA
5.0	28 June 2017	Include approved CR1005	TG	Tom Van Pelt / GSMA
6.0	06 December 2018	Include approved CR1006	TG	Tom Van Pelt / GSMA
7.0	16 October 2019	Include approved CR1007	NG	Tom Van Pelt / GSMA
8.0	16 October 2020	Include approved CR1008	NG	Tom Van Pelt / GSMA
9.0	19 December 2022	Include approved CR1009: handling of invalid TLS certificates	ISAG	Tom Van Pelt / GSMA
10.0	04 June 2024	Include approved CR1010	ISAG	Tom Van Pelt / GSMA
11.0	28 February 2025	Include approved CR1011	ISAG	Tom Van Pelt / GSMA
12.0	18 July 2025	Include approved CR1012	ISAG	Tom Van Pelt / GSMA

F.2 Other Information

Type	Description
Document Owner	RCS Group
Editor / Company	Tom Van Pelt / GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.