

KeyTalk - Protocols

Author	MR vd Sman
Creation date	14-March-2017
Last updated	14-August-2025
Document version	2.8.4
Document status	Qualified
Product	KeyTalk certificate and key management & enrolment virtual appliance
Data classification	Public

TABLE OF CONTENTS

TABLE OF CONTENTS	2
1. INTRUCTION	5
1.1 Purpose	5
1.2 Scope	5
2. CERTIFICATE RETRIEVAL API (RCDPV2)	6
2.1 KeyTalk config file	6
2.2 RCDPV2 overview	7
2.3 RCDPV2 communication phases	8
2.4 Messages sent in all phases	9
2.4.1 End Of communication	9
2.4.2 Error	9
2.5 Phase 1 (handshake)	12
2.5.1 Hello	12
2.5.2 Handshake	12
2.6 Phase 2 (authentication)	14
2.6.1 Request authentication requirements for the template	14
2.6.2 [as of v2.8.1] Authentication requirements for the seat	18
2.6.3 Template-wide authentication	20
2.6.4 [as of v2.8.1] Seat-wide authentication	24
2.6.5 Challenge-response authentication	25
2.6.6 [as of v2.7.5] Start OTP or MFA authentication.	27
2.6.7 Change password	28
2.7 Phase 3 (service provision)	30
2.7.1 Check for the last messages.	30
2.7.2 Generate certificate on the server.	31
2.7.3 Query CSR requirements	37
2.7.4 [as of v2.7.2] Start TPM attestation.	39
2.7.5 Generate certificate from the client CSR.	40
2.7.6 [as of v2.7.4] Store certificate to the server	43
2.7.7 [as of v2.8.1] Retrieve seat shared mailbox certificates	45

2.7.8	[as of v2.7.8] Query certificate and keys scraping settings.	47
-------	--	----

3. PUBLIC API-----48

3.1	Public API versions -----	48
3.2	API overview-----	48
3.2.1	Retrieve self-service availability.	48
3.2.2	Retrieve address book URLs	50
3.2.3	Retrieve availability of S/MIME certificate enrollment to external parties for self-service	52
3.2.4	Query the certificate store to place the resulted certificate	54
3.2.5	Request a disclaimer to install to the mail client	55
3.2.6	Query certificate approver emails	56
3.2.7	Query KeyTalk server version	57
3.2.8	Query Common Name customization policy	58
3.2.9	Query which templates participate in auto-renewal of seat certificates.	59
3.2.10	Query certificate expiration margin	60
3.2.11	Download client configuration files	61
3.2.12	Check server health status	62
3.2.13	[as of v1.6.6] Retrieve message shown on KeyTalk agent automatic popup.	63
3.2.14	[as of v1.6.7] Retrieve keep-alive interval.	64
3.2.15	[as of v1.6.7] Send keep-alive	64
3.2.16	[as of v1.6.8] Check whether a seat certificate is known to have TPM key attestation.	66

4. ADMINISTRATOR API-----67

4.1	API versions -----	67
4.2	API overview-----	67
4.2.1	Enroll seat from a server-generated keypair	67
4.2.2	Enroll seat with client CSR	69
4.2.3	Revoke seat certificates	72
4.2.4	Download KeyTalk settings	74
4.2.5	Retrieve SCEP configuration.	76
4.2.6	Copy KeyTalk TEMPLATE	78
4.2.7	Import certificates	81
4.2.8	[as of v1.9.6] Create or update seat	84
4.2.9	Archive seat	86



4.2.10 List KeyTalk templates	88
4.2.11 Remove KeyTalk template.	90
4.2.12 Open slot on the seat	92
4.2.13 Create internal RA user.	94
4.2.14 Update seats	96
4.2.15 Remove seat.	98
4.2.16 [as of v1.9.4] Create template for enrolling DigiCert DV certs via ACME.	100
4.2.17 [as of v1.9.5] Create tenant.	102
4.2.18 [as of v1.9.5] Enable ACME for the template.	104
5. SELF-SERVICE API -----	106
5.1 API versions-----	106
5.2 API overview-----	106
5.2.1 Enroll S/MIME certificates for external parties	106
6. CERTIFICATE AUTHORITY RETRIEVAL API (CA API)-----	110
6.1 CA API versions-----	110
6.2 CA API overview-----	110
6.2.1 Fetch internal signing CA	110



1. INTRODUCTION

1.1 Purpose

The purpose of this document is to describe the API used by the KeyTalk system.

1.2 Scope

This document is intended for KeyTalk and its hired 3rd parties for continuous development of the KeyTalk product and related services.

More importantly this document is intended for release to the public so they may use it for their own KeyTalk related development purposes.

2. CERTIFICATE RETRIEVAL API (RCDPV2)

This section describes certificate retrieval REST API called RCDPV2.

RCDP version	Minimal KeyTalk server version	Changes
2.7.4	6.6.1 (v6) 7.2.0 (v7)	Allow store certificates to the server in the <code>service</code> phase
2.7.5	7.2.1 (v7)	Add support for multi-factor authentication and renamed “OTP” credential type to “OTP/MFA”
2.7.6	7.2.6	Allow skipping RCDP version in the request URL, which will automatically select the latest version supported by the server.
2.7.7	7.4.5	Allow retrieving certificates for all Shared Mailboxes of the given seat
2.7.8	7.4.9	Allow querying certificate and keys scraping settings
2.7.9	7.6.10	Add support for out-of-band retrieval of Shared Mailbox (SMB) certificates
2.8.0	7.7.1	Update seat authentication by acceptin UserId iso seat-name. Allow renewal of a certificate using seat authenticaiton.
2.8.1	7.7.3	Require caller to supply HWSIG prior to generating challenge for seat authentication
2.8.2	7.7.8	Add given name and surname to <code>/cert</code> request
2.8.3	7.8.0	Add extra SAN domains to <code>/cert</code> request

2.1 KeyTalk config file

In order to make use of the KeyTalk API, several details are required from the KeyTalk Real Client Communication Data file (RCCD).

This configuration file is used to feed a KeyTalk app with minimal required information to setup a proper secure connection to any KeyTalk instance.

The RCCD file is effectively a zip container and can thus easily be extracted.

As such a developer incorporating the KeyTalk API in their app, can choose to statically make use of individual files in an RCCD file, or choose to import the entire RCCD into their app or simply make use of some of the components within this RCCD file.

The content folder within the RCCD contains several files, the most important ones being:

- **RCA.der Root CA** typically only included when KeyTak’s internal private CA is generated under an already existing CA.
- **PCA.der Primary CA** with a KeyTalk self-signed private CA is usually the top of the KeyTalk internal private CA, but when RCA is included its generated under the RCA.
- **UCA.der User CA** signed under the PCA. It is the trust under which the end-point client and/or server certificates are signed and issued only in case of using the internal KeyTalk CA for issuance.
When issuing end-point client and/or server certificates under for example a connected Microsoft CA or Trusted Certificate Service Provider, ensure that



their intermediate certificates are included in your app or present on the target OS as well as these are by default not part of the current KeyTalk RCCD

- **SCA.der Server CA** signed under the PCA, it is the trust under which the KeyTalk virtual appliance certificates are generated and used.
- **user.ini** Generic configuration settings which includes the KeyTalk server URL/IP as well as the KeyTalk tenant name/SERVICE used to communicate with.
- **user.yaml** Generic configuration settings which includes the KeyTalk server URL/IP as well as the KeyTalk tenant name/SERVICE used to communicate with. Similar to user.ini, just another format.

2.2 RCDPv2 overview

Communication in RCDPv2 is encapsulated in RESTful calls over HTTPS port 443. Optional out-of-band certificate downloads are made possible via HTTP over the standard port 80 and via HTTPS of the standard port 443.

Below is a set of client HTTP headers that the client needs to send to the server.

HTTP Header	Required	Description
GET	YES	/rcdp/<version>/<action>?<request-params>
Host	YES	Should contain the FQDN or IP of KeyTalk server.
Cookie	YES except for hello	Session identifier received from KeyTalk server.

action is a request action

request-params is URL-encoded string of request parameters. Complex request parameters (arrays, dictionaries) should be JSON-encoded. All JSON objects should escape forward slashes '/' as '\\/'.

A typical set of client HTTP headers:

```
POST /rcdp/authentication HTTP/1.1

Host: keytalkdemo.keytalk.com

Cookie: keytalkcookie=a622bb821bec1f5315668c8f9a8e780f
```

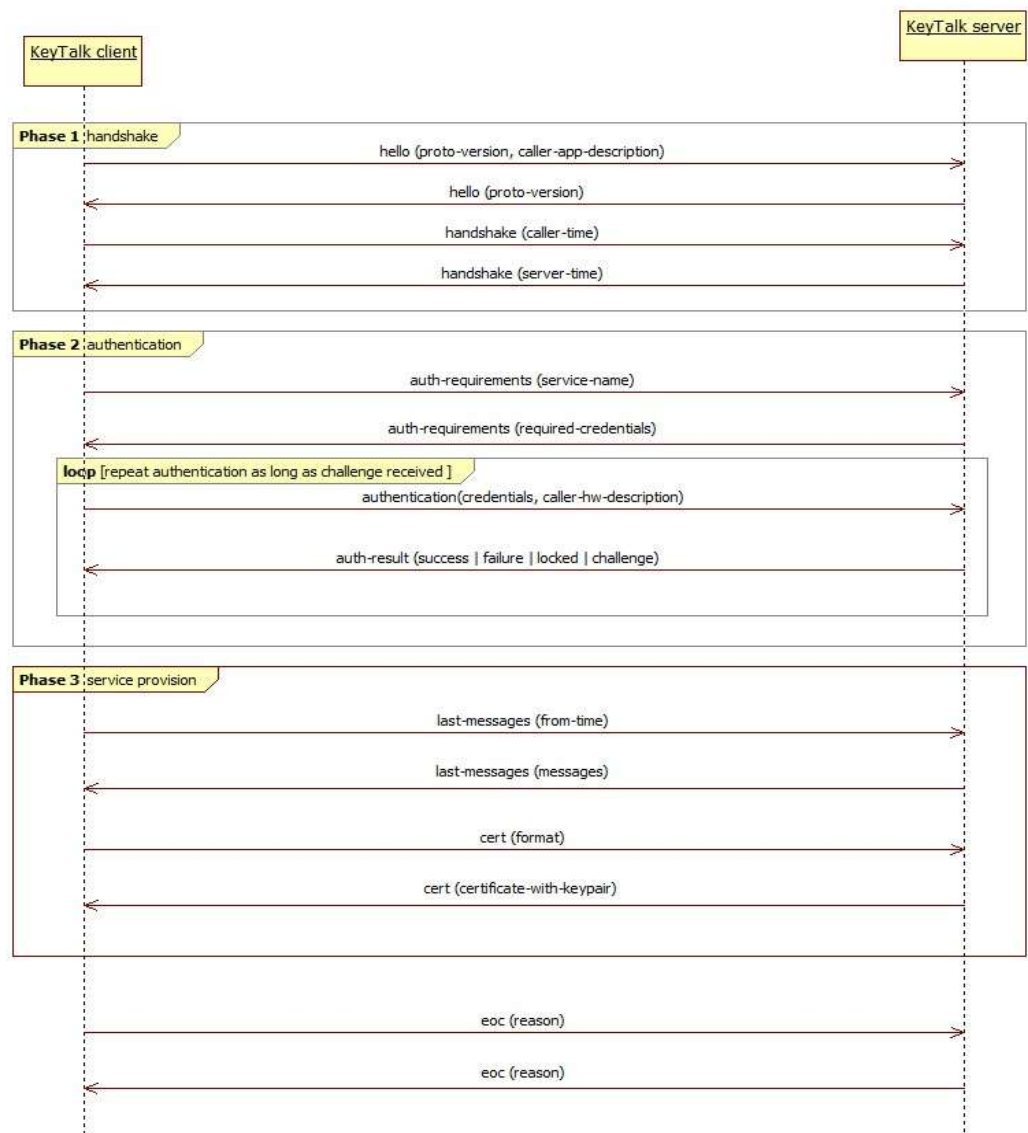
2.3 RCDPv2 communication phases

The complete RCDPv2 communication circle consists of 3 phases:

Phase1: handshake

Phase 2: authentication

Phase 3: service provision



Further we describe message semantics on each phase in detail.

2.4 Messages sent in all phases

2.4.1 End Of communication

Request

Latest protocol version: GET /rcdp/eoc

Specific protocol version: GET /rcdp/<version>/eoc

Example:

```
/rcdp/eoc
/rcdp/eoc?reason=bye%2C+server
```

Query parameters

parameter	type	required	description
reason	string	no	optional reason for ending communication

Response

HTTP 200 - application/json

```
{
  'status': 'eoc',
  [optional] 'reason': optional reason for ending communication.
}
```

End of communication can be sent at any time, initiated by any communication side.

2.4.2 Error

Errors are typically sent by the server to notify the caller on error processing its request. The client can also send errors to the server when it can't handle the server's response.

Request

Latest protocol version: GET /rcdp/error

Specific protocol version: GET /rcdp/<version>/error

Example:

```
/rcdp/error?code=1066&description=invalid+response
```

Query parameters

parameter	type	required	description
code	number	yes	numeric error code
reason	string	no	optional error description. Might be required for certain error codes. See the error code table below.

Response

HTTP 200 - application/json

```
{
  'status': 'error',
  'code': numeric error code,
  [optional] 'description': error description. Might be required for certain error codes. See
the error code table below.
}
```

Error codes

code	description	direction	remarks
1001 (ErrResolvedIpInvalid)	optional	server -> client	Sent by the server when none of IPs resolved by the client and by the server match.
1002 (ErrDigestInvalid)	optional	server -> client	Sent by the server when the client's calculated executable digest does not much the digest stored on the server.
1003 (ErrTimeOutOfSync)	difference in seconds between caller UTC and the server UTC	server -> client	Sent by the server when the client time is out of sync with the server's time.
1004 (ErrMaxLicensedUsersReached)	optional	server -> client	Sent by the server when no certificate can be supplied because the max number of licensed users has been reached.
1005 (ErrPasswordExpired)	optional	server -> client	Sent by the server when the password of the user trying to authenticate is expired and the caller is not supposed to change it.
1006 (ErrIncompatibleProtocolVersion)	optional	server -> client	Sent by the server when further communication is not possible because the client's API version does not support it (normally too old)
1007 (ErrTpmEkCertNotIssuedByEkCa)	optional	server -> client	Sent by the server to indicate that the TPM Endorsement certificate presented by the client in TPM attestation request is not issued by the TPM Endorsement CA configured on the server
1008 (ErrTpmEkCertMismatchEkPubBlob)	optional	server -> client	Sent by the server to indicate that the TPM Endorsement certificate and the public BLOB presented by the client in TPM attestation request do not correspond to each other

[as of v2.7.7] 1009 (ErrInvalidSeat)	optional	server -> client	Sent by the server to indicate that the seat supplied by the caller cannot be used for authentication typically because the seat does not exist or is archived or lacks valid certificate
--	----------	------------------	---



2.5 Phase 1 (handshake)

This is the initial phase.

2.5.1 Hello

Agree on RCDP API version and establish session ID.

Request

Latest protocol version: GET /rcdp/hello

Specific protocol version: GET /rcdp/<version>/hello

Example:

/rcdp/hello

/rcdp/hello?caller-app-description=Demo+KeyTalk+Agent

Query parameters

parameter	type	required	description
caller-app-description	string	no	optional description of the caller application

RCDP API version proposed by the caller is sent as a part HTTP GET path.

Response

HTTP 200 - application/json

```
{
  "status": "hello",
  "version": proposed API version
}
```

Session ID is returned in HTTP cookie `keytalkcookie` in Set-Cookie header.

2.5.2 Handshake

Confirm version handshake and exchange time information.

Request

Latest protocol version: GET /rcdp/handshake

Specific protocol version: GET /rcdp/<version>/handshake

Example:

/rcdp/handshake?caller-utc=2023-04-14T10%3A44%3A35Z

Query parameters

parameter	type	required	description
-----------	------	----------	-------------

caller-utc	<i>UTC string in ISO 8601 format including date and time</i>	yes	caller UTC
------------	--	-----	------------

If the caller agrees with API version proposed by the server on the previous step, it proceeds with this version. Otherwise, the caller ends communication.

Response

HTTP 200 - application/json

```
{
  "status": "handshake",
  "server-utc": server UTC in ISO 8601 format including date and time
}
```



2.6 Phase 2 (authentication)

On this phase the caller authenticates itself in order to perform further certificate-related operations against the server.

2.6.1 Request authentication requirements for the template

Request authentication requirements for the TEMPLATE (former “service”) from the server.

Request

Latest protocol version: GET /rcdp/auth-requirements

Specific protocol version: GET /rcdp/<version>/auth-requirements

Example:

/rcdp/auth-requirements?service=TEST_TEMPLATE

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk template name

Response

HTTP 200 - application/json

```
{
  "status": "auth-requirements",
  "credential-types": credential types,
  [optional] "hwsig_formula": HWSIG formula,
  [optional] "password-prompt": password-prompt,
  [optional] "service-uris": template URIs, see further,
  [optional] "resolve-service-uris": whether template URIs need to be resolved by the
  caller,
  [optional] "calc-service-uris-digest": whether template URIs digest needs to be
  calculated by the caller,
  [optional] "use-tpm-vsc-authentication": whether TPM Virtual Smart Card
  authentication should be used,
  [optional] "use-kerberos-authentication": whether Kerberos authentication should be
  used,
  [optional] "supply-computer-name": whether the machine/device name should be supplied
  to the server,
  [as of v2.7.5] [optional] "mfa-settings": whether multi-factor authentication is
  enabled for the template,
}
```

credential-types



JSON array of credential types required to authenticate against the given service. Supported credential types are: "USERID", "HWSIG", "PASSWD", "PIN", "RESPONSE" and "OTP/MFA" ("OTP" prior to [v2.7.5](#)).

Example: ["USERID", "HWSIG", "PASSWD"]

hwsig_formula

formula to calculate caller's hardware signature.

Example: "1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16". Sent when credential-types parameter contains HWSIG.

password-prompt

prompt to display to a user when a password is requested interactively e.g. "password" or "tokencode". Sent when credential-types parameter contains PASSWD.

service-uris

JSON array of RFC 3986-compliant URIs of the given service

Example:

["https://demo1.keytalk.com", "https://demo2.keytalk.com"]

or

["file://%ProgramFiles%\vpn\vpn.exe"]

resolve-service-uris

Boolean flag ("true" or "false") requesting a caller to resolve IP addresses of each supplied service-uris identifying web resources. Defaults to "false".

calc-service-uris-digest

Boolean flag ("true" or "false") requesting a caller to calculate sha-256 hexadecimal digests of each supplied service-uris identifying file resources. Defaults to "false".

use-tpm-vsc-authentication

One of ("yes", "yes-with-attestation" or "no"). This value indicates to the caller whether a TPM Virtual Smart Card should be used to generate a certificate signing request (CSR). Optionally, TPM key attestation might be used for extra security. The CSR will be then sent to KeyTalk server to create a certificate. Defaults to "false".

use-kerberos-authentication

Boolean flag ("true" or "false") requesting a caller to make use of Kerberos authentication. If Kerberos authentication happens to be not possible, the caller should fall back to regular KeyTalk authentication specified in *credential-types*.

mfa-settings

Child object containing the following multi-factor authentication settings:

```
{
  "kind": mfa-kind,
  "client-id": Azure App Client ID,
  "authority": MFA Authority,
  "scopes": mfa-scopes,
  "redirect-uri": URI to redirect to after successful authentication (agent specific),
  "post-log-out-uri": URI to redirect to after logging out (agent specific)
}
```

mfa-kind

One of the following values:



```
"mfa-popups": MFA should use Popups by Web agents  
"mfa-redirection": MFA should use Redirection by Web agents
```

mfa-scopes

JSON array containing MFA scopes, e.g. ["User.Read.All", "Mail.Read"]

Examples:

```
{
  "status": "auth-requirements",
  "credential-types": ["HWSIG", "PASSWD", "USERID"],
  "hwsig_formula": "1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16",
  "password-prompt": "Password",
  "service-uris": ["https://demo.keytalk.com"],
  "resolve-service-uris": "true"
}
```

```
{
  "status": "auth-requirements",
  "credential-types": ["HWSIG", "PASSWD", "USERID", "OTP/MFA"],
  "hwsig_formula": "1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16",
  "password-prompt": "Password",
  "service-uris": ["https://demo.keytalk.com"],
  "resolve-service-uris": "true",
  "resolve-service-uris": "true",
  "mfa-settings": {
    "kind": "mfa-popups",
    "client-id": "01234567-89ab-cdef-0123-456789abcdef",
    "authority": "https://login.microsoftonline.com/organizations",
    "scopes": ["User.Read.All", "Mail.Read"],
    "redirect-uri": "https://my.agent.com/mfa?auth",
    "post-log-out-uri": "https://my.agent.com/mfa?logout"
  }
}
```

2.6.2 *[as of v2.8.1]* Authentication requirements for the seat

Request authentication requirements for the seat from the server.

Request

Latest protocol version: GET /rcdp/auth-requirements

Specific protocol version: GET /rcdp/<version>/auth-requirements

Example:

/rcdp/auth-requirements?service=DEMO_SERVICE&USERID=Jos&computer-name=joscomp

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service (TEMPLATE) name
USERID	string	Yes	User ID of the caller
computer-name	string	Yes	Caller's machine/device name

Response

HTTP 200 - application/json

```
{
  "status": "auth-requirements",
  "hwsig_formula": HWSIG formula when configured for the TEMPLATE
  "encrypted-challenge": encrypted challenge when no HWSIG formula configured for the
  TEMPLATE,
}
```

encrypted-challenge

A random string S/MIME-encrypted with the public key of the latest HWSIG-agnostic issued certificate of the seat resolved from the supplied *USERID* and *computer-name* and further base-64 encoded for the relay. The seat should exist, should not be archived and the certificate should be valid and non-revoked, otherwise `ErrInvalidSeat` error is returned. The caller is supposed to decrypt the challenge using the private key of the seat's certificate and send it back in the subsequent /authentication request.

hwsig_formula

formula to calculate caller's hardware signature. Example:

"1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16". The caller is supposed to calculate Hardware Signature over this formula and send it back in the subsequent /auth-requirements request (see below).



Request

Latest protocol version: GET /rcdp/auth-requirements

Specific protocol version: GET /rcdp/<version>/auth-requirements

Example:

/rcdp/auth-requirements?HWSIG=CS-123456

Query parameters

parameter	type	required	description
HWSIG	string	yes	Hardware Signature of the caller's device calculated over the formula sent in <code>hwsig_formula</code> parameter in the previous <code>auth-requirements</code> server response. <code>requirements</code> response.

Response

HTTP 200 - application/json

```
{
  "status": "auth-requirements",
  "encrypted-challenge": encrypted challenge
}
```

encrypted-challenge

A random string S/MIME-encrypted with the public key of the latest issued certificate for the supplied HWSIG of the seat resolved from the supplied *USERID* and *computer-name* and further base-64 encoded for the relay. The seat should exist, should not be archived and the certificate should be valid and non-revoked, otherwise `ErrInvalidSeat` error is returned. The caller is supposed to decrypt the challenge using the private key of the seat's certificate and send it back in the subsequent `/authentication` request.

2.6.3 Template-wide authentication

Authenticate the caller against the selected service (template) using the supplied set of credentials.

Request

Latest protocol version: POST /rcdp/authentication

Specific protocol version: POST /rcdp/<version>/authentication

Content-type: application/x-www-form-urlencoded

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55a1f411396031ce91ee48d9d" \
-H "Expect:" \
-d "service=DEMO_SERVICE&caller-hw-
description=Windows+11%2C+BIOS+s%2Fn+1234567890&USERID=DemoUser&HWSIG=CS-
123456&PASSWD=change%21&resolved=%5B%7B%22ips%22%3A+%5B%2281.175.103.107%22%5
D%2C+%22uri%22%3A+%22https%3A%2F%2Fdemo.keytalk.com%2F%22%7D%5D" \
-X POST https://test.keytalk.com/rcdp/authentication
```

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service name
caller-hw-description	string	yes	Caller HW description which should be unique for the given device. For uniqueness e.g. BIOS serial number or iOS device UDID can be used. Examples: <ul style="list-style-type: none"> Windows 10, BIOS s/n 1234567890 iPAD: Jan's iPAD 234567890abcdef1234567890abcdef
USERID	string	if requested	ID of the user. Required if USERID was previously set by the server in auth-requirements response.
HWSIG	string	if requested	Hardware Signature of the caller's device calculated with the formula specified in the previous auth-requirements server response. Required if HWSIG was previously set by the server in auth-requirements response.
PASSWD	string	if requested	User password. Required if PASSWD was previously set by the server in auth-requirements response.
PIN	string	if requested	User pincode. Required if PIN was previously set by the server in auth-requirements response.
<i>[as of v2.7.5]</i> OTP/MFA	string	if requested	After receiving WAIT-FOR-OTP (see 2.6.2.2) from KeyTalk server, the caller should acquire a One Time Password (OTP) via an alternative channel (email or SMS). Required if "OTP/MFA" was previously sent by the server in auth-requirements response. Can only be used once per positive authentication. This also applies to multi-factor authentication <i>as of v2.7.5</i> .
resolved	JSON array	if requested	JSON array of objects containing service URIs accompanied with RFC 3986-compliant IPv4 or IPv6 address resolved from the URI hostname.



			<p>Required if <code>resolve-service-uris</code> was previously set in <code>auth-requirements</code> response.</p> <p>Example:</p> <pre>[{ "uri": "https://demo1.keytalk.com", "ips": ["81.175.10.107", "81.175.103.109"] }, { "uri": "https://demo2.keytalk.com", "ips": ["81.175.10.108", "[2001:db8:a0b:12f0::1]"] }]</pre>
digests	<i>JSON array</i>	if requested	<p>JSON array of objects containing service URIs accompanied with SHA-256 hexadecimal digest of the underlying file.</p> <p>Required if <code>calc-service-uris-digest</code> was previously set in <code>auth-requirements</code> response.</p> <p>Example:</p> <pre>[{ "uri": "file://%Program Files%\\vpn\\vpn.exe", "digest": "01c7198fb614bf8746b46062aa551dff4506dd553ad96817622c76dafa8dc354" }, { "uri": "file://%Program Files%\\vpn\\vpn2.exe", "digest": "01c7198fb614bf8746b46062aa551dff4506dd553ad96817622c76dafa8dc355" }]</pre>
kerberos-ticket	<i>JSON object</i>	if requested	<p>JSON object containing Kerberos Ticket Granting Ticket (TGT). Should present if <code>use-kerberos-authentication</code> was previously requested by the server in <code>auth-requirements</code>. If the caller does not supply Kerberos ticket despite requested by the server, the remaining credentials provided by the caller will be used for authentication.</p> <p>The ticket should obey the following schema:</p> <pre>{ "client-principal": client principal, "session-key": base64 encoded session key, "session-key-encoding": session key encoding, "tgt": base64 encoded ASN.1 TGT, "tgt-flags": TGT flags, "start-time": TGT validity start UTC in ISO 8601, "end-time": TGT validity end UTC in ISO 8601, "renew-till": TGT renewal due UTC in ISO 8601 }</pre>
computer-name	<i>string</i>	if requested	<p>Caller's machine/device name. Required if <code>supply-computer-name</code> was previously set by the server in <code>auth-requirements</code> response.</p>

retired-computer-name	string	conditional	<p>Only pass if all the following scenarios are true:</p> <ul style="list-style-type: none"> - supply-computer-name was previously set by the server in auth-requirements response. - The current machine/device name is different from the machine/device in the latest valid certificate. <p>Machine/device name can be found in a certificate using the following guidelines: SAN contains one or more DNS values & the certificate CN should equal a DNS value in SAN</p>
-----------------------	--------	-------------	---

Response

HTTP 200 - application/json

```
{
  "status": "auth-result",
  "auth-status": authentication-status,
  [optional] "delay": number of seconds the user is disallowed to authenticate as a result of the
(previous) failed authentication or because the user's password is expired,
  [optional] "password-validity": password validity on success,
  [optional] "challenges": requested challenges,
  [optional] "response-names": response names for the given challenges,
}
```

auth-status

authentication status. Can be one of:

"OK" - authentication successful

"DELAY" - authentication was not successful and delay parameter is set

"LOCKED" - cannot login because the user is locked on the server

"EXPIRED" - authentication not successful because the user password is expired

"CHALLENGE" - challenge is supplied by the server and challenges parameter is set

"KERBEROS-AUTH-NOK" - validation of Kerberos ticket failed (e.g. expired), a caller can try again with the remaining credentials (no user lock gets applied on failed Kerberos validations)

"WAIT-FOR-OTP" - caller's identity successfully checked, the caller should wait for the OTP to be sent (e.g. to his email represented by USERID) and re-submit authentication request with this OTP as a credential. This also applied to multi-factor authentication *as of v2.7.5* (if enabled).

delay

when DELAY is received in auth-status, indicates the time in seconds the caller is suspended from repeating its authentication attempt. Can be 0 which means a caller can try re-authenticating immediately.

When LOCKED is received in auth-status, indicates the time in seconds the caller is *still* suspended from repeating its authentication attempt.

password-validity

when authentication succeeds ("OK" received), indicates the number of seconds until the password expires or -1 if the password never expires. Password validity is supplied only when provided by an authentication backend.

challenges

when CHALLENGE is received, contains JSON array of challenges. Challenge names are meant to be displayed to a user during interactive challenge prompt. Challenge values is the value of the challenge to use for response calculation.

Example:

```
[
  {
    "name": "enter first pincode",
    "value": "981fa356"
  },
  {
    "name": "enter second pincode",
    "value": "981fa357"
  }
]
```

response-names

when CHALLENGE is received, contains JSON array of response names. When multiple responses are required by the server, response name allow identifying each response sent by the caller, thus serving as response keys. Response names can be omitted when only one response is expected by the server.

Example: ["response 1", "response 2", "response 3"]

Example:

Successful authentication:

```
{
  "status": "auth-result",
  "auth-status": "OK"
}
```

Unsuccessful authentication, the caller is suspended for 10 seconds

```
{
  "status": "auth-result",
  "auth-status": "DELAY",
  "delay": 10,
}
```

Extra challenge is requested (RADIUS SecurID authentication)

```
{
  "status": "auth-result",
  "auth-status": "CHALLENGE",
  "challenges": [{"name": "Password challenge", "value": "Enter your new PIN of 4 to 8 digits, or <Ctrl-D> to cancel the New PIN procedure:"}],
}
```

When a caller receives CHALLENGE in auth-status from the server, it should proceed as follows:

- provided the set of required credentials does not include RESPONSE, the caller should re-submit all the credentials required by the server, filling PASSWD credential with the response to the received challenge. This is called multi-phase password authentication
- provided the set of required credentials includes RESPONSE, the caller should respond with RESPONSE credential only as described below in 2.6.2.1. This is called Challenge-Response authentication.

2.6.4 *[as of v2.8.1]* Seat-wide authentication

Authenticate the caller against the seat.

Request

Latest protocol version: POST /rcdp/authentication

Specific protocol version: POST /rcdp/<version>/authentication

Content-type: application/x-www-form-urlencoded

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55alf411396031ce91ee48d9d" \
-H "Expect:" \
-d "response=12345467890&caller-hw-description=Test+7%2C+agent" \
-X POST https://test.keytalk.com/rcdp/authentication
```

Query parameters

parameter	type	required	description
RESPONSE	<i>string</i>	yes	The value of encrypted-challenge decrypted with the key of the caller's latest valid seat certificate. OpenSSL CLI counterpart for the decryption is: openssl base64 -d -A base64-ciphertext openssl cms -decrypt -signer signerkey.pem -inform DER The output is to be encoded as size#challenge
caller-hw-description	<i>string</i>	yes	Caller HW description which should be unique for the given device. For uniqueness e.g. BIOS serial number or iOS device UDID can be used. Examples: - Windows 10, BIOS s/n 1234567890 - iPad: Jan's iPad 234567890abcdef1234567890abcdef

Response

HTTP 200 - application/json

```
{
  "status": "auth-result",
  "auth-status": authentication-status,
  [optional] "delay": number of seconds the user is disallowed to authenticate as a result of the
(previous) failed authentication,
}
```

auth-status

authentication status. Can be one of:

"OK" - authentication successful

"DELAY" - authentication was not successful and delay parameter is set

2.6.5 Challenge-response authentication

Request

Latest protocol version: POST /rcdp/authentication

Specific protocol version: POST /rcdp/<version>/authentication

Content-type: application/x-www-form-urlencoded

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55alf411396031ce91ee48d9d" \
-H "Expect:" \
-d \
"responses=%7B%22CK%22%3A+%22123%22%2C+%22RES%22%3A+%22456%22%2C+%22IK%22%3A+%22789%22%7D" \
-X POST https://test.keytalk.com/rcdp/authentication
```

Query parameters

parameter	type	required	description
responses	<i>JSON object</i>	yes	JSON array of responses. Response names should be the same as returned by the server on the previous authentication request. Example: [{"name": "RES", "value": "123"}, {"name": "IK", "value": "456"}, {"name": "CK", "value": "789"}]

Response

HTTP 200 - application/json

```
{
  "status": "auth-result",
  "auth-status": authentication-status,
  [optional] "delay": authentication delay for failed authentication,
  [optional] "password-validity": password validity on success,
  [optional] "challenges": requested challenges,
  [optional] "response-names": response names for the given challenges
}
```

auth-status

authentication status. Can be one of:

"OK" - authentication successful

"DELAY" - authentication was not successful and delay parameter is set

"LOCKED" - cannot login because the user is locked on the server

"EXPIRED" - authentication not successful because the user password is expired

"CHALLENGE" - challenge is supplied by the server and challenges parameter is set

delay

when DELAY is received in auth-status, indicates the time in seconds the caller is suspended from repeating its authentication attempt. Can be 0 which means a caller can try re-authenticating immediately.

password-validity

when authentication succeeds ("OK" received), indicates the number of seconds until the password expires or -1 if the password never expires. Password validity is supplied only when provided by an authentication backend.

challenges

when CHALLENGE is received, contains JSON array of challenges. Challenge names are meant to be displayed to a user during interactive challenge prompt. Challenge values is the value of the challenge to use for response calculation.

Example:

```
[
  {
    "name": "enter first pincode",
    "value": "981fa356"
  },
  {
    "name": "enter second pincode",
    "value": "981fa357"
  }
]
```

response-names

when CHALLENGE is received, contains JSON array of response names. When multiple responses are required by the server, response name allow identifying each response sent by the caller, thus serving as response keys. Response names can be omitted when only one response is expected by the server.

Example: ["response 1", "response 2", "response 3"]

Example:

Successful authentication:

```
{
  "status": "auth-result",
  "auth-status": "OK"
}
```

Unsuccessful authentication, the caller is suspended for 10 seconds

```
{
  "status": "auth-result",
  "auth-status": "DELAY",
  "delay": 10,
}
```

Extra challenge is requested (RADIUS SecurID authentication)

```
{
  "status": "auth-result",
  "auth-status": "CHALLENGE",
}
```

```
"challenges": [{"name": "Password challenge", "value": "Enter your new PIN
of 4 to 8 digits, or <Ctrl-D> to cancel the New PIN procedure:"}],
}
```

2.6.6 *[as of v2.7.5]* Start OTP or MFA authentication.

Start OTP (One Time Password) authentication or MFA (Multi-Factor authentication) after if "OTP/MFA" was previously received by the server in `auth-requirements` response and the caller have not yet requested a One-Time Password from the server. Once successful the caller should receive an OTP for USERID supplied in the request (normally email). This OTP should later be used to authenticate against the server. The OTP should only be used once.

Request

Latest protocol version: POST `/rcdp/authentication`

Specific protocol version: POST `/rcdp/<version>/authentication`

Content-type: `application/x-www-form-urlencoded`

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55a1f411396031ce91ee48d9d" \
-H "Expect:" \
-d \
"service=DEMO_SERVICE&caller-hw-
description=Windows+7%2C+BIOS+s%2Fn+1234567890&USERID=demo%40keytalk.com&HWSI
G=123456" \
-X POST https://test.keytalk.com/rcdp/authentication
```

Query parameters

parameter	type	required	description
service	<i>string</i>	yes	KeyTalk service name
caller-hw-description	<i>string</i>	yes	Caller HW description which should be unique for the given device. For uniqueness e.g. BIOS serial number or iOS device UDID can be used. Examples: <ul style="list-style-type: none"> Windows 10, BIOS s/n 1234567890 iPAD: Jan's iPAD 234567890abcdef1234567890abcdef
USERID	<i>string</i>	if requested	ID of the user, normally a e-mail
HWSIG	<i>string</i>	if requested	Hardware Signature of the caller's device calculated with the formula specified in the previous <code>auth-requirements</code> server response. Required if HWSIG was previously set by the server in <code>auth-requirements</code> response.
computer-name	<i>string</i>	if requested	Caller's machine/device name. Required if <code>supply-computer-name</code> was previously set by the server in <code>auth-requirements</code> response.

Response

HTTP 200 - application/json

```
{
  "status": "auth-result",
  "auth-status": authentication-status,
  [optional] "delay": authentication delay for failed authentication,
}
```

auth-status

authentication status. Can be one of:

"WAIT-FOR-OTP" - the caller's identity successfully verified, and the caller should wait for the OTP to be sent (e.g. to his email represented by USERID) and re-submit the authentication request with this OTP as a credential. In case of multi-factor authentication (*as of v2.7.5*), the agent should trigger the MFA flow at this point and re-submit the authentication request with the acquired Access Token as OTP.

"DELAY" - authentication was not successful and *delay* parameter is set

"LOCKED" - cannot login because the user is locked on the server

"OK" - authentication successful. Normally should not be sent in this flow because it means that OTP is not required on the server.

delay

when *DELAY* is received in *auth-status*, indicates the time in seconds the caller is suspended from repeating its authentication attempt. Can be 0 which means a caller can try re-authenticating immediately.

Example:

Successful OTP initiation:

```
{
  "status": "auth-result",
  "auth-status": "WAIT-FOR-OTP"
}
```

Unsuccessful OTP initiation, the caller is suspended for 10 seconds

```
{
  "status": "auth-result",
  "auth-status": "DELAY",
  "delay": 10,
}
```

2.6.7 Change password

Change user password. Password change facility must be supported by the server backend such as Active Directory. A caller should normally change his password after *EXPIRED* authentication result is received from the server. A caller may also choose to change his password on successful authentication when *password-validity* parameter gives a hint that the password is about to expire.

Request

Latest protocol version: POST /rcdp/change-password

Specific protocol version: POST /rcdp/<version>/change-password

Content-type: application/x-www-form-urlencoded

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55a1f411396031ce91ee48d9d" \
-d "old-password=changeme&new-password=changed" \
-X POST https://test.keytalk.com/rcdp/change-password
```

Query parameters

parameter	type	required	description
old-password	<i>string</i>	yes	Current (old) user password.
new-password	<i>string</i>	yes	New user password.

Response

See 2.6.2 with authentication status limited to "OK", "DELAY" or "LOCKED"

"OK" means the password has been successfully changed and the user must re-authenticate with his new password.

"DELAY" means the password change did not succeed (e.g., incorrect old password or too short new password) and the caller may try again after the given number of seconds.

2.7 Phase 3 (service provision)

Once authenticated, the caller becomes eligible for (certificate-related) services from the server.

2.7.1 Check for the last messages.

Check for the last server messages. Server messages are meant for KeyTalk users e.g. to indicate planned server maintenance.

Request

Latest protocol version: GET /rcdp/last-messages

Specific protocol version: GET /rcdp/<version>/last-messages

Example:

```
/rcdp/last-messages
/rcdp/last-messages?from-utc=2023-04-26T06:49:55.614010Z
```

Query parameters

parameter	type	required	description
from-utc	UTC string in ISO 8601 including date and time	no	UTC to request the messages from. Defaults to requesting all server messages.

Response

HTTP 200 - application/json

```
{
  "status": "last-messages",
  "messages": [
    {
      "text": message text string,
      "utc": message UTC in ISO 8601 including date and time
    },
    ....
  ]
}
```

Example:

```
{
  "status": "last-messages",
  "messages": [{ "text": "This is user message number 1",
    "utc": "2017-04-06T04:15:15+0000"},
    { "text": "This is user message number 2",
    "utc": "2018-03-04T02:10:10+0000"},
    { "text": "This is user message number 3",
    "utc": "2018-05-02T00:05:05+0000"} ]
}
```



2.7.2 Generate certificate on the server.

Retrieve a server-generated certificate in the desired format along with a private key.

Request

Latest protocol version: POST /rcdp/cert

Specific protocol version: POST /rcdp/<version>/cert

Request POST parameters:

parameter	type	required	default value	description
format	"P12" "PEM" "P12v2"	yes	n/a	"PEM" to request PEM-encoded X.509 certificate and private key "P12" to request PKCS#12-encoded X.509 certificate and private key. The private key will be encrypted with "legacy" 3DES-CBC algorithm. This encryption format is considered not secure (end 2022), and new callers are strongly advised to use more secure <i>P12v2</i> format, unless it is not supported by their platform. Primarily meant for backward compatibility with mobile agents which cannot handle <i>P12v2</i> (end 2022). "P12v2" to request PKCS#12-encoded X.509 certificate and private key. The private key will be encrypted with AES256-CBC algorithm, which is superior in security compared to 3DES-CBC. New callers are advised to use this format.
out-of-band	boolean	no	false	When set, the server will send back URL to download the certificate instead of the certificate itself.
response	JSON object	if requested	n/a	Response sent by the client after the challenge status received in the previous "cert" request. Format: { "type": type of the previously received challenge, "data": Base64-encoded response }
cookie	JSON object	if requested	n/a	Cookie received in the previous response, typically of type "submitted" or "in-progress" or "challenge".
common-name	string	no		Common Name to be used in the certificate. Only honored if allowed by the server. Use <i>cn-</i>

			customization-policy Public API call to query, should return "ALLOWED".
[as of v2.8.2] given-name and surname	strings	no	Given name (first name) and surname (second name) to be used in the certificate and in CN. Only honored if allowed by the server. Use <code>cn-customization-policy Public API call to query, should return "ALLOWED-AS-GIVENNAME_SURNAME"</code> .
[as of v2.8.3] san-domains	JSON array	no	Extra SAN domains to be used in the certificate.

Response

HTTP 200 - application/json

Certificate is successfully issued by the server.

```
{
  "status": "cert",

  "cert": certificate in the desired format including CA trust chain, when available. Returned when
out-of-band is not set.
    PEM-encoded certificate has its private key encrypted with the first 30 characters of the
session ID sent by the server in keytalkcookie.
    When the certificate is delivered in PKCS#12 package, the package gets encrypted with
with the first 30 characters of the session ID sent by the server in keytalkcookie and subsequently
base64 encoded to be transported with JSON,

  "cert-url-templ": certificate download URL template returned when out-of-band is set.
    The template contains $(KEYTALK_SVR_HOST) placeholder that needs to be instantiated with
a hostname or IP address of the KeyTalk server used by the caller to make up a valid URL. The
download URL is valid for a limited amount of time (normally 5 minutes) and gets invalidated after
the first use.
    PEM-encoded certificate has its private key encrypted with the first 30 characters of the
session ID sent by the server in keytalkcookie.
    When the certificate is delivered in PKCS#12 package, the package gets encrypted with the
first 30 characters of the session ID sent by the server in keytalkcookie.

  "execute-sync": boolean flag indicating whether the caller should invoke the service URIs
synchronously (true) or asynchronously (false). Defaults to false.

  "store-to-system": if set, instruct the caller to store the generated certificate to the System
Store (Machine Store) instead of the User Store. Defaults to false.

  "apply-address-books": boolean flag indicating whether the caller should apply the address
books, typically to be used by an email client. Applying the address books should allow the user to
send encrypted emails and verify email signatures to other users registered to the address books

  "address-books": [
    {"ldap_svr_url": LDAP server URL,
```

```

        "search_base": LDAP server search base DN (e.g.
        "ou=people,dc=example,dc=com",
        "verification_ca": PEM-encoded X.509 verification CA(s) of
        the LDAPs server (optional for LDAPs URL),
        },
        ... ]

    "apply-smime-settings": boolean flag indicating whether the caller should apply S/MIME
    settings to an email client

    "set-disclaimer": boolean flag indicating whether the caller should apply a disclaimer to an
    email client signatures. The disclaimer(s) can be fetched from the Public API

    "historical-certs": array of historical certificates previously issued to this seat (user), in the
    desired format. Returned if enabled server-side provided out-of-band is not set Currently the server
    can only be configured to return S/MIME certificates.

    PEM-encoded certificates have their private keys encrypted with the first 30 characters of
    the session ID sent by the server in keytalkcookie.

    When the certificates are delivered as PKCS#12 packages, each package gets encrypted
    with the first 30 characters of the session ID sent by the server in keytalkcookie and subsequently
    base64 encoded to be transported with JSON,

    "historical-certs-url-templ": URL template to download historical certificates as a single
    GZIP package. Returned when out-of-band is set. Currently the server can only be configured to
    return S/MIME certificates.

    The template contains $(KEYTALK_SVR_HOST) placeholder that needs to be instantiated with
    a hostname or IP address of the KeyTalk server used by the caller to make up a valid URL. The
    download URL is valid for a limited amount of time (normally 5 minutes) and gets invalidated after
    the first use.

    PEM-encoded certificates have their private keys encrypted with the first 30 characters of the
    session ID sent by the server in keytalkcookie.

    When the certificates are delivered as PKCS#12 packages, each package gets encrypted with
    with the first 30 characters of the session ID sent by the server in keytalkcookie,
}

```

Example typical usage, the certificate is returned in the response body:

```

{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwGgAwIBAgIIWurOaAAAABYwDQYJKoZIhvcNAQELBQAwYg9xH2AdBgkqhkiG9w0B
CQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\nnDBNLZXL1UYWxrIEl1U
NlY3VyaXR5MRgwFgYDVQLDA9GyWN0b3J5IERlZmFlbHxQX\nnIDAeBgNVBAMMF0tleVRhbGsgRGVt
byBTAWduaW5nIENBMB4XDTE4MDUwMzA3NTUw\nnNFOXDTE4MDUwMzA5NTUwNFowZAxETAPBgNVBA
MMCERlbW9vc2VyMQswCQYDVQQG\nnEwJOTDEWMBQGA1UECAwNTm9vcmtQmFyYmFudDESMBAGA1UE
BwwJRWluZGhvdmdVu\nnMRQwEgYDVQQKDAwTaW9leCBHcm91cDEMMMAoGA1UECwwDU0VTMR4wHAYJKo
ZIhvcN\nnAQkBFg90ZXN0dWlAc2lvdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\nnAo
IBAQDJGKTHSL16vsqxIjXvDOTKlk2q518JaIF9Q9ews88NmpVV9cDbOPRxswns\nnSd1kNAXEYi05
ScmIc5pGpIV8hyyNjtZ17tiolVO0ALkXgk7hG7wO2Rz+bAQzCdvS\nnoJjtzo6gZPYcQVlfq+ENMt
39ibLqfuAnMLjVpn44fwfQxQFeEsd4do074E1bUXh7\nn7KzaoxsiDAyIITYZe5Azz90l47ffg3pR
Dtq\n/6IDYmr7x1BMOq+7QObKBU0pgwNkn\nn3JTgkBSpXGEXok6S1qNBqJ199NJjdYjIwJHa\n/9vS
pHSN8RF2s9xrBanLM3S+fnr6\nnBx34P6cBoTcc11Z9Dpr8IYNJWkanAgMBAAGjFTB7MAkGA1UdEw
QCMAAwHQYDV01\nnBBYwFAYIKwYBBQUHAWIGCCSGAQUFBwMBMAsGA1UdDwQEAwID+DAqBg1ghkgB
hvhC\nnAQIEHRYbQ1VTVF9QQVNTV0RfSU5URVJOQUxfVEVTVFVJMBYGA1UdEQQPMAC2C25z\nnLnNp
b3V4LmV1MA0GCSqGSIb3DQEBCwUAA4ICAQCYKf1OTJqL3eg1JgJdbLPzDo74\nnfqZbEBpNkeBFe6

```

```
nQ6calHJrZNG857WGdfVKfXSOrkwGHmdSN1\0XM+ySIpcNOWQf\nM9o9rxKQigk4n\tvjNCiVX
Ral25t5pURlZSyullSWQAJYc2nPjzasl5B8SwJOIet\nJV80z1pgLFh2GU7hGNIWVqJLF\U0\T
+xZ1lW1sZ64iih49owTsLt9CL06pD6KPN6\nWvmzLNoK\ouEeRnYgkyWxvlahGY5N2bPwlq+7+s
3BOYRo3APL4N6iVEOUfYDE78K\n05g5zdhVbn717CMx1sQpXggyF5X\ztQLkrUB5kLT9D7eCBnL
DVdJELz112KJar\b\ny9eumkCg+Y9PCZN2513o1zU1DLGaH9\9KdCf6yEca3D3NvnbfCmrDvx1
0AN+Ht3L\n4XU2L5Rx2rqwB9tj3rZy8i6BK7\A+ARfg6Tqki5FQ9k667q2hBRPtr69bLeML5at\
nyn\beKjnYnzCRcfXDgnJKZdfKt2PBM7lh508HNn6aaRZUfHBKHxjMxwuXNMdq9m\nHk6+H8rb
RipV\4xCzEFYvaqlpYO31OzLIrw8AohRlUzX7UFGm1Dbpn3G2qeikD1Z\nhySYTxjmjXE0DVnPL
X05+MR08Eq3hC6QDYs3gBZgP3nILvfEZliOax4fqBT3ijJ9\nnoxMI+OJsawZMG0u00w==\n-----
END CERTIFICATE-----\n-----BEGIN ENCRYPTED PRIVATE KEY-----
\nMIIFDjBABGkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIQ9o+vwzbXQQCaggA\nnMBQGCCqG
SIb3DQMHBAAipsAoCJT4gVgSCBMhTb\8ws1tw9uhH12t9mozccMJQeSAe\nIDxu86RaxgbaMcHj2
GnfQjFPoulIk28eU4Pbi6OEpdLGSBAtrRTK9ZsIOCv+26vN\nnjrh4gFsLqa9LC\RB6T7gQFK6Ns
j+9332d+jCr4tKBIJvSu6hmTGTOraePHb8ic8B\nniSHphmz91N91M311qYKMzhW\MZG043u2TBj
zx1LdsFicIH\KJ8LXkYQNYM0G663y\nnqWpnygyjWvzIL7oL5rZh5pvygFTuTY\lakDW3inu8
fn3\Zy1374IHeAk4V\hGQ\nc7FmpF15FTZAYICuKQqSTzUKOd+90qlq8YrbcPbHrcMH43UTeaJ
zjklc3R5K\mQk\n6a2ggjPc2z4LoFOYetoPUointBLnRetk7QEHWQdWWW5WfFGRrjBK2t0jZLLV
zXuS\nZ0QYBoHeGzFYH0AeYB01DacT8OC9PAB4r\veFdKyXD85OdYdIp4cAbYm5IBB8bYd\nnnf9
JIV8iifIHy38of6FpHI3AwPzqZTTDaR+arLTjpmpN6d9bRfMNYWUWnJsv0W0o\nd1YuWU\OEO
tdvVQKnU1T9FdhbjyW6nQpR8uwHYLi\BIjpvCUK6ZAe\+llik0Z2+\nCxn1bU225MOaY2YLS3B
izXUkMcQAo4JE5tEj9vMsEa4VHvt9zcsfpT4vZIGmG2h\nu9UoY2XGhZ4jIEVtqO2ihz7V1ow+k
07eD6H1HMhws9CPZkKh03Z94FK1V\Sf53U6\ndnRlsAmuuI5HJroXYX6N5cLguSnwyyvOWRPrU
UjqWPZrfvLzndpro6IFPils7L4\n2fR1DEHwe\VV0StF31CV6N88KRyGN+gBWrkvGKJ8EozhEz2
qToqLBU0CLQ+FVO1E\nuYS30hejXc8wYKFupwSolhpJU2B4zC4EbsmTnn7sS55Yk+9NCetE\k0
VMf\PVVN\nWG0kFhq5CCmtkx8fVvq0nnnNuZS4Hy+tBlEeqMvRvQQ62eRCR94msYG2LCVxRUib\
nNrKQvBM3\RbxjQFVULr6Wjw9I8dLenjffjou47JLSMShaxlDeAG5iBb0GzLZP6W1h\nOyXIYusR
ePxv40GPZsCBRqD2c6fdk52U3Bgk7asctplL9Y1qP71lbJwnuFtygt+7\nz+7b38PL1txMRYMCoL
D78kugFAP2St0iGGdzdUEWoIP\IZT2SmMo578CPum3RSht\nu31CtHfzzrMIq2o1uTGv+HDswTr
LwZ\VDcaZZUP9a6Vfyfzd83jqRXCKFeBk2udM\nHDo5TC6EvLAv9cXqGRW8VSxkjlWdyxhIdjNS
CN+CrECX\PTbmV5MP9gydnqDSJDq\npCHXZr6dca6vAUGYn5ouQuhrTjsSRsk4M5ZhwgYt9xwCc
fNE+juVeweWEJM1GnxP\nmEW3fFSE+NNdfYoPWEA5XEGPr3x7F9Bj51T4Yk0XVX\ED3htx0VI8
g2IZGrvt40\nyh+\OxyxB9zUzsleQVDitmzQnqti3nXReHwyen00p9frC5J\o4ibYKkPF91H9\
\UK\nh8SCSLpWBil\8RBQ8kD0Pms5G\Z2TNS6dnwrXZU+solpl+Kk+T+TTjKkDp8U1xkv\nWCL
AUbs8g00289SjGjhPge0c4UWRiKLElj6jDx0g3yHoJU8bi6pMnJzVeg7IhLF\nnxK8=\n-----
END ENCRYPTED PRIVATE KEY-----\n"
```

Notice again that JSON-serialization of PEM certificates requires forward slashes '/' to be escaped as '\\'

Example when the certificate download URL is returned iso a certificate itself (out-of-band request parameter was set to true):

```
{
  "status": "cert",
  "cert-url-templ":
    "http://$(KEYTALK_SVR_HOST)/cert/?cbf498dc683c4e0499fd7e2d27640917"
}
```

Response

HTTP 200 - application/json

An extra info is requested from the client before issuing a certificate.

```
{
  "status": "cert-challenge",
  "challenge-type": (string) type of the challenge. See below for the details
  "challenge-data": base64-encoded challenge contents interpreted by the client based on the
challenge type. The client is supposed to reply to this challenge by resending the cert request
augmented with the response parameter.
"cookie":
{
  "name": name,
  "value": value
}, cookie to return to the server in the subsequent cert request
```

the only supported challenge type is "select-globalsign-domainssl-approver-email" accompanied with the list of emails in challenge-data indicating emails eligible for approval the issuance of GlobalSign SSL certificates. The caller is supposed to repeat the previous cert request by adding the selected email address in the response parameter.

```
}
```

Response

HTTP 200 - application/json

The certificate request is successfully submitted ("submitted") or was previously submitted ("in-progress") and needs time or an extra action to be fulfilled. The caller is supposed to periodically check the certificate is ready using the cert request.

```
{
  "status": "submitted"|"in-progress",
  "cookie":
  {
    "name": name,
    "value": value
  }, cookie to return to the server in the subsequent cert request

  "details": (string) extra details to present to the caller
}
```

Response

HTTP 200 - application/json

The certificate request requires approval to be fulfilled. The caller is supposed to periodically check the certificate is approved using the cert request.

```
{
  "status": "pending-approval",
  "approval-ongoing": boolean indicates whether the approval for the given seat has already
been ongoing (true) or was triggered by this call (false)
}
```



2.7.3 Query CSR requirements

A client might want to generate a key pair by himself and submit the resulted CSR to KeyTalk server for signing. Before doing that the client should query the server for the initial parameters for the CSR such as key size, signing algorithm, certificate subject and a subject alternative name (SAN).

Request

Latest protocol version: GET /rcdp/csr-requirements

Specific protocol version: GET /rcdp/<version>/csr-requirements

Example:

/rcdp/csr-requirements

Response

HTTP 200 - application/json

```
{
  "status": "csr-requirements",
  "key-size": key size in bits [as of server v7.2.6 this is recommended rather than an obligatory CSR requirement, might be removed in future releases],
  "signing-algo": algorithm to use for CSR signing [as of server v7.2.6 this is recommended rather than an obligatory CSR requirement, might be removed in future releases],
  "subject": dictionary of subject fields to use in CSR, [ as of server v7.4.1 empty attribute names should be treated as "any" attribute by the caller instead of a literal value],

  "san": if set, holds an array of Subject Alternative Names to use in CSR [ as of server v7.4.1 empty SAN should be treated as "any" by the caller instead of a literal value],

  "subject"."ous": [ as of server v7.7.11 a single-value "subject"."ou" has been replaced with a multi-value "subject"."ous" ]

}
```

Example:

```
{
  "status": "csr-requirements",
  "key-size": "2048",
  "signing-algo": "sha256",
  "subject": {"cn": "TestUser",
    "c": "NL",
    "st": "Utrecht",
    "l": "Amersfoort",
    "o": "KeyTalk",
    "ous": ["Development", "Administration"],
    "e": "test@keytalk.com",
  },
}
```

```
"san": ["email:test@keytalk.com","DNS:test.keytalk.com"]
}
```

2.7.4 *[as of v2.7.2]* Start TPM attestation.

Start TPM attestation procedure from the server. On successful verification, the server will return a challenge to be used by the TPM for calculating evidence.

Request

Latest protocol version : POST /rcdp/tpm-attestation

Specific protocol version : POST /<version>/tpm-attestation

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
ek-cert	string	yes	n/a	PEM-encoded Endorsement Certificate
ek-pub-blob	string	yes	n/a	Base-64 encoded Endorsement Public Key BLOB
ak-name	string	yes	n/a	TPM Attestation Key name

Response

HTTP 200 - application/json

```
{
  "status": "tpm-attestation",
  "challenge": "Base64-encoded TPM challenge"
}
```

2.7.5 Generate certificate from the client CSR.

Retrieve a PEM-encoded certificate from the CSR supplied by the client. The CSR should be created from the parameters retrieved from the `csr-requirements` request described in 2.7.3.

Request

Latest protocol version : POST /rcdp/cert

Specific protocol version : POST /rcdp/<version>/cert

Content-type: application/x-www-form-urlencoded

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: keytalkcookie=a77c33e55a1f411396031ce91ee48d9d" \
-H "Expect:" \
-d "csr-----BEGIN+CERTIFICATE+REQUEST-----
%0AMIIIC1jCCAb4CAQAwGA1UECgwlU2lvdXggR3JvdXAxZjAUBgNVBAgMDU5v%0Ab3JkLUJhcmJhbnQx
EAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKoZIhvcNAQkBFg90%0AZXN0dWlAc2lvdXguZGUwggEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDQ%0AfyCCkM7cbVhpBCSx1Nf%2BFDqa9banKf9sPRW
5VwBFYP5siLdsywNkNqrFYcV0w6ss%0Ath21qK9bkjZoyiKpbzvzGQw08N1bBmJfj700018HUn2xL
vp2z6J6q3Z4rAR4d8jx%0ApwcdRlPeJO5b30tBaURKILaJTjtsUVyCXr%2B6u%2FgiuaD0DGBKsIQ
ccyAWGy%2BlzNer%0AsmUib%2FsnWHEaAPJtvg7T2amaWACKcqIOppR%2BHDJUUNSYyju9xZqCLjx
6Y2%2B2ZXHK%0AmpFcFsp%2F8GCGZ2%2FAi1WtsVzKSaRwMTVJfBsy50gW3YmwI0QYgh152NIDQu
BJeoT%0AmQFxsKXpqcWjpP3KTOS5AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAbUVCaYm%2F%0A
w1otZaLgtCP2mIVVH%2FgHvTeVFs1436Lz%2FaKT5q1QRee81C2us1z9G7h3PG%2BM6w1N%0AUJau
wqQ2mR2c1VAidROdT52syNPR4jXeR11%2F7a%2FmsZFqaw3%2FLlwVtBJHEfOA6apU%0AJSVWi6%2
F3kUjD0FhYHAufKm2nJ10qGnwC5xpzuVYOQsUFFobLZoyGq5NNEgnSpk8X%0A9A9j5kKGBom9eQOr
Wxw%2F0UlwRqLpt6l76Gt5%2BlMp5BtTCPK2uboHvJiPu4aJUuHh%0Afx9ZjKox73V%2BleOEmNSY
fesuQPE5AwifKE988NFixGXOHw7uQdWc9SfsYFRFZG2p%0AYb%2Bm9iFyUY8AHw%3D%3D%0A-----
END+CERTIFICATE+REQUEST-----%0A" \
-X POST https://test.keytalk.com/rcdp/cert
```

Request POST parameters:

parameter	type	required	default value	description
csr	string	yes	n/a	Base64 encoded PKCS#10 certificate signing request
out-of-band	boolean	no	false	When set, the server will send back URL to download the certificate instead of the certificate itself.
[as of v2.7.2] tpm-evidence	string	no	empty	Required when TPM attestation flow is in effect, i.e. use-tpm-vsc-authentication parameter of /auth-requirements response is set to “yes-with-attestation”. Should contain the value of TPM evidence calculated by the TPM from the challenge received on the preceding /tpm-attestation request. The evidence will be checked on the server before signing the CSR.



Response

HTTP 200 - application/json

```
{
  "status": "cert",

  "cert": PEM-encoded certificate including trust CA, if available, is returned when out-of-band is not set,

  "cert-url-templ": certificate download URL template returned when out-of-band is set. The template contains $(KEYTALK_SVR_HOST) placeholder that needs to be instantiated with a hostname or IP address of the KeyTalk server used by the caller to make up a valid URL. The download URL is valid for a limited amount of time (normally 5 minutes) and gets invalidated after the first use,

  "execute-sync": boolean flag indicating whether the caller should invoke the service URIs synchronously (true) or asynchronously (false). Defaults to false.

  "apply-address-books": boolean flag indicating whether the caller should apply the address books, typically to be used by an email client. Applying the address books should allow the user to send encrypted emails and verify email signatures to other users registered to the address books,
  "address-books": [
    {
      "ldap_svr_url": LDAP server URL,
      "search_base": LDAP server search base DN (e.g.
      "ou=people,dc=example,dc=com",
      "verification_ca": PEM-encoded X.509 verification CA(s) of the LDAPs server (optional for LDAPs URL),
    },
    ... ],
  "apply-smime-settings": boolean flag indicating whether the caller should apply smime settings to an email client.,
  "set-disclaimer": boolean flag indicating whether the caller should apply a disclaimer to an email client signatures. The disclaimer(s) can be fetched from the Public API
}
```

Example regular usage (certificate is returned in the response):

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwGgAwIBAgIIWurNEwAAABUwDQYJKoZIhvcNAQELBQAwYg9wOBF
CQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\ndBNLZX1UYWxrIE1UIF
NlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmFlbHhQx\nIDAEBgNVBAMMF0tleVRhbGsgRGVt
byBTAWduaW5nIENBMB4XDTE4MDUwMzA3NDky\nM1oXDTE4MDUwMzA5NDkyM1owGzAx
CzAJBgNVBA
YTAK5MMRIwEAYDVQQHDA1Faw5k\naG92ZW4xDDAKBgNVBASMA1NFUzEUMBGA1UECgWLU21vdXgg
R3JvdXAxXzFjAUBGNV\nnBAGMDU5vb3JkLUJhcmJhbnQxETAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKo
ZiIhvcN\naQkBFg90ZXN0dW1Ac21vdXguZmF5dGgEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\naO
IBAQDGFyCCkM7cbVhpBCSx1Nf+FDqa9banKf9sPRW5VwBFYP5siLdsywNkNqrF\nnYcV0w6ssth21
qK9bkjZoyiKpbzvzgQw08N1bBmJfj700018HUn2xLvp2z6J6q3Z4\nnrAR4d8jxpwcdr1PeJO5b3O
tBaURKILaJTjtsUVyCXr+6u\n/giuaD0DGBKsIQccyAW\nnGy+1zNersmUib\n/snWHEaAPJtvg7T2a
maWACKcqIOppR+HDJUUNSYyju9xZqCLjx6\nnY2+2ZXHKMpFcFsP\n/8GCYgz2\n/AilWtsVzKSaRwM
TVJfBsy50gW3YmwiOQYghl52NI\nndQuBJeoTmQFxsKXpqCwJpP3KTOS5AgMBAAgJfTB7MAkGAlUd
EwQCMAAwHQYDVR01\nnBBYwFAYIKwYBBQUHAWIGCCSgAQUFBwMBMAsgAlUdDwQEAwID+DAQBgIghk
gBhvhc\naQIEHRYbQ1VTVF9QQVNTVORfSU5URVJQQUxvZmVhVTVFVJMBYGA1UdEQQPMMA2CC25z\nnLn
Npb3V4LmV1MA0GCSqGSIb3DQEBCwUAA4ICAQCCca0ClI9Dw+io7IIqMZ8UKzhq\nn8MWcbpthcgFH
```

```
PHdxqFYIfTWYOzXCN8FVq96oHH2e09anBYopGyHW+a5oMbY8bKbP\nvGD6\ /Cs1C8nFFqkQfRTH6
nanDSq18S\ /4uc3bMaIQvWzv5mEYpiTKtKCSUMfv7FLN\nS64I\ /UQNglEhHMu1lUyL0NM3xU8QY
mz+k6qnkw2C3M5Y9eprUT9iZxXCm4XGJo7j\nUPBIRBXUCsaPz+UdK0Syq2H1\ /IsREt5iPRJIU\
/B4FjduJlD1R68ZAYNnyOeDQI7f\nEJWUeBYC2QwdlXW3FqKdwki928wksRpY4x3Fyz9\ /f32chZ
QOihee378HP9PDiTZQ\nFCIWSsrO+WUujToehK2ErgqwCrH0Ydw5ZuIV1vVivGzlgmDHmIQY6uPn
YasalkQw\nspY2Jyv1ZA\ /9mhCvfupwB6L4QIA8yJwNoM3MasZgq4fvk1kxm\ /k1pRMPB2bSGy4u
\nFLyMoodTAYJfpzH\ /gCwWnrYowqw2T67HsPqBBiOnsuaA0h4k\ /m88i4ypcv5f48wJ\nzcxaXq
RqWqxzw\ /efkYg5m4HdncAPU05NxxJmP17n77188MZvKc0wVbA+22vCBgCi\nMaOYWhnkTuBN90A
oaYAJwelbkLlbTFMZJjsNPvvS5sAk1l9NihCrXS8ZWtZRfGyz\nngPkm+UPWboYdQbKCRg==\n---
--END CERTIFICATE-----\n"
}
```

Notice again that JSON-serialization of PEM certificates requires forward slashes '/' to be escaped as '\\'

Example when certificate download URL is returned:

```
{
  "status": "cert",
  "cert-url-templ": "
http://$(KEYTALK_SVR_HOST) /cert/?cbf498dc683c4e0499fd7e2d27640917"
}
```


2.7.7 *[as of v2.8.1]* Retrieve seat shared mailbox certificates

Retrieves the list of shared mailboxes (SMBs) along with their certificates for all shared mailboxes that belong to the seat previously resolved from USERID and computer name submitted during authentication phase.

Latest protocol version: POST /rcdp/smbcerts

Specific protocol version: POST /rcdp/<version>/smbcerts

Request POST parameters:

parameter	type	required	default value	description
format	"P12" "PEM" "P12v2"	yes	n/a	Format of SMB certificates, same as described in section 2.7.2
<i>[as of v2.7.9]</i> out-of-band	boolean	no	false	When set, the server will send back URL to download the certificates instead of the certificate itself.

Response

HTTP 200 - application/json

SMB got successfully retrieved.

```
{
  "status": "smbcerts",
  "smbcerts": [
    {
      "upn": "shared mailbox UPN,
```

[optional, when out-of-band is false] "cert": the latest valid shared mailbox certificate in the desired format including CA trust chain, when available. PEM-encoded certificate has its private key encrypted with the first 30 characters of the session ID sent by the server in keytalkcookie. When the certificate is delivered in PKCS#12 package, the package gets encrypted with the first 30 characters of the session ID sent by the server in keytalkcookie and subsequently base64 encoded to be transported with JSON.

[as of v2.7.9] [optional, when out-of-band is true] "cert-url-templ": the latest valid shared mailbox certificate download URL template returned when out-of-band is set. The template contains \$(KEYTALK_SVR_HOST) placeholder that needs to be instantiated with a hostname or IP address of the KeyTalk server used by the caller to make up a valid URL. The download URL is valid for a limited amount of time (normally 5 minutes) and gets invalidated after the first use,

[optional] "historical-certs": array of historical certificates previously issued to this seat in the desired format. PEM-encoded certificates have their private keys encrypted with the first 30 characters of the session ID sent by the server in keytalkcookie. When the certificates are delivered as PKCS#12 packages, each package gets encrypted with the first 30 characters of the session ID sent by the server in keytalkcookie and subsequently base64 encoded to be transported with JSON

```
    },  
  
    [as of v2.7.9] [optional, when out-of-band is true] "historical-  
certs-url-templ": URL template to download historical certificates previously issued to this seat  
in the desired format as a single GZIP package. Returned when out-of-band is set. The template  
contains $(KEYTALK_SVR_HOST) placeholder that needs to be instantiated with a hostname or IP  
address of the KeyTalk server used by the caller to make up a valid URL. The download URL is valid  
for a limited amount of time (normally 5 minutes) and gets invalidated after the first use. PEM-  
encoded certificates have their private keys encrypted with the first 30 characters of the session ID sent  
by the server in keytalkcookie. When the certificates are delivered as PKCS#12 packages, each  
package gets encrypted with with the first 30 characters of the session ID sent by the server in  
keytalkcookie,  
  
    ...  
  ]  
}
```



2.7.8 *[as of v2.7.8]* Query certificate and keys scraping settings.

Query the settings for the caller (typically KeyTalk Windows Certificate Scanner agent) to scan for certificates and keys and send them to KeyTalk server using REST API call described in the sections 2.7.6.

Latest protocol version: GET /rcdp/cert-scraping-settings

Specific protocol version: GET /rcdp/<version>/cert-scraping-settings

Response

HTTP 200 - application/json

```
{
  "status": "cert-scraping-settings",
  "schedule": "never" | "one-shot" | "on-successful-authentication",
  "cert-purposes": ["smime", "client-auth", "server-auth"],
  "cert-store-types": ["personal", "system"]
}
```



3. PUBLIC API

API to query various information from KeyTalk server without the need to authenticate.

3.1 Public API versions

REST API version	Minimal KeyTalk server version	Changes wrt the previous API version
1.6.6	7.4.2	allow retrieving notification template for the message shown to end-users when KeyTalk agent automatically pops up
1.6.7	7.5.1	allow querying keep-alive interval; allow sending keep-alives
1.6.8	7.5.7	allow checking whether the given seat certificate is known to have TPM key attestation
1.6.9	7.5.15	piggy-back keep-alive internal in the response to i-am-alive request

3.2 API overview

The communication goes over HTTPS port 443 and over HTTP port 80.

3.2.1 Retrieve self-service availability.

Retrieves whether self-service is available for the given account.

Request

Latest API version: POST /public/self-service-availability

Specific API version: POST /public/<version>/self-service-availability

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
cert	string	yes	n/a	PEM-encoded X.509 user certificate previously received from KeyTalk identifying the caller

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Expect:" \
-d "cert=-----BEGIN%20CERTIFICATE-----
%0AMiIFDDCCAvSgAwIBAgIIW1SC7gAAAYowDQYJKoZIhvcNAQELBQAwYg9w0
BCQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK%0ADBNLZX1UYWxrIElUI
FNlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmF1bHhQx%0AIDAeBgNVBAMMF0tleVRhbGsgRGV
tbyBtaWduaW5nIENBMB4XDTE4MDEwOTA3NTMw%0AMloXDTI4MDEwNzA4NTMwMlowZAxETAPBgNVB
AMMCERlbW9Vc2VyMQswCQYDVQQG%0AEwJOTDEWMBQGA1UECAwNTm9vcmQtQmFyYmFudDESMBAGA1U
EBwwJRWluZGhvdnVu%0AMRQwEgYDVQQKDATaW91eCBHcm91cDEMMAoGA1UECwwDU0VtMR4wHAYJK
```



oZIhvcN%0AAQkBFg90ZXN0dWlAc2lvdXguZXUwgGEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK%0A
AoIBAQDI98ytSED47IFmjT0PUezLlyHulxDggXYgYF83G27J5%2BLkfn1xcWa2cCVH%0AotFG%2BK
ZNfspd5HAXUfRgJI%2BciypdMf3DS9ZH3PIScgEzwNXG0WnNUagHjUfsYQ6r%0A1I6MBKE86F8sjR
hyF%2Fr1Upogsc24ALypBdIhQ6Ham3ni4NFsYEcBk80nKK%2BpATDJ%0APqzK1IIGsd3XhOmxjVUz
PR7OFaEbHwnbOakWfeLibwJAWttDIL7KctSMsBLg3U2o%0AogYmm5fJqfLwZZEmwNslkuzGcHB6M1
WBYQHyp966k1YnItBDFEMYKvaW2ec8tZEA%0AljGAWmIrgAMR9obUWoUuGVwLoOXzAgMBAAGjcDBu
MAkGA1UdEwQCMAAwEwYDVR0l%0ABAwwCgYIKwYBBQUHAWIwCwYDVR0PBAQDAgP4MCCGCWCGSAGG%2
BEIBAgQaFhhDVVNU%0AX0FOT19JTlRFUk5BTF9URVNUVUkwFgYDVR0RBA8wDYILbnMuc2lvdXguZX
UwDQYJ%0AKoZIhvcNAQELBQADggIBAJXrf8FkL2Bh2rW%2FgJQOpHADELqk25%2F8UmMNbpauneVB
%0AOaJFHCXpsSUDQ4beJo3gyM5JvZRV5nfc8Na3v4aNB01oVJHDHA8%2BGsl1UXy3fN1v%0AU4YAF
yiJULxELrYyuA5g0rdxXjLSSvSTWGWPKIJEGLYC2xvO8kh%2BnwOijVciHFCp%0AGATpChEO4MPwj
QiLpJowx5W5FhcyaRvp%2FjHl0T1IsPWRuD23oG8gcg4uRAF9CKvU%0AYE8%2BV9RrYG%2B6VYeNP
2Va6DGTXVsH2%2Fi3vP7IdaO8cnVocYcHKzOBU%2Bfr60muNpuo%0AhE%2FoGzz7uj%2F3wja9q49
OGJaNxHjadUjk5k799e%2Fv2isBaSuNmXVobOEzzjasIyF%2B%0A2VWGrxSJCfhUXSv3%2B0xnm0o
UigN7uxVl%2BriPcpv1zyNLQqzNxSgerD0iW6dIAPHj%0AFuchgAYYuRXVMHbwxqfUUVUqyBg2Pr5
ESSYB6PTDXzigChLjWJRLuEZxWxmi6ztn%0AfgSMOG9XRFFR7yxQKDEuGZKfbAcdd0gXtCDbH3T%2B
UUESjAc7bUz4OM5Zxfea%2F8F%2FE%0AoBrzb6XROZEWApxmbQOWRXRu01dd1%2Fi77irHAsH%2FL
Gq6RGv7qsDmic%2B3WQXZtdkX%0AslTcBslGLZkgnbWewNtA8UWHL1JdfJjKmnOv5RdYxRc6kCXEO
WQilCBYQxpQXR5L%0A-----END%20CERTIFICATE-----" \

-X POST <https://test.keytalk.com/public/self-service-availability>

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "self-service-availability",
  "available": boolean
}
```

HTTP 400 - application/json - invalid request

```
{
  "status": "error",
  "error": error message (optional)
}
```

3.2.2 Retrieve address book URLs

Retrieves URLs of address books used by back-end LDAP/AD servers.

Request

Latest API version: GET /public/address-book-list

Specific API version: GET /public/<version>/address-book-list

Example:

/public/address-book-list?service=DEMO_SERVICE

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service (template) name

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "address-book-list",
  "apply-address-books": boolean flag indicating whether the caller should apply the address
  books, typically to be used by an email client. Applying the address books should allow the user to
  send encrypted emails and verify email signatures to other users registered to the address books,
  "address-books": [
    {
      "ldap_svr_url": LDAP server URL,
      "search_base": LDAP server search base DN (e.g.
      "ou=people,dc=example,dc=com",
      "verification_ca": PEM-encoded X.509 verification CA(s) of
      the LDAPs server (optional for LDAPs URL),
    },
    ...
  ]
}
```

Example response when no address books configured for the service

```
{
  "status": "address-book-list",
  "apply-address-books": see successful invocation,
  "address-books": ""
}
```

HTTP 400 - application/json - invalid request

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



3.2.3 Retrieve availability of S/MIME certificate enrollment to external parties for self-service

Check the availability and requirements for S/MIME certificate enrollment to external parties for the given self-service account.

Request

Latest API version: POST /public/smime-cert-enrollment-availability

Specific API version: POST /public/<version>/smime-cert-enrollment-availability

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
cert	string	yes	n/a	PEM-encoded X.509 S/MIME certificate previously received from KeyTalk identifying the caller as self-service-eligible user
synchronous	boolean	no	true	<p>When set to true, checks whether an immediate enrolment is possible.</p> <p>When set to false checks whether it is possible to place a certificate order without yielding a certificate. The order will be handed in to a configured CA, which will get responsible for communicating the certificate back to the users via e-mail.</p> <p>At the moment S/MIME asynchronous orders are only supported by KeyTalk services bound to GlobalSign PersonalSign products.</p>

Example:

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Expect:" \
-d "cert=-----BEGIN%20CERTIFICATE-----
%0AMIIFFDDCCAvSgAwIBAgIIWlSC7gAAAYowDQYJKoZIhvcNAQELBQAwYgYgXzAdBgkq%0AhkiG9w0
BCQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK%0ADBNLZXlUYWxrIE1UI
FNlY3VyaXR5MRgwFgYDVQLDA9GYWN0b3J5IERlZmF1bHhQx%0AIDAEgNVBAMMF0tleVRhbGsgRGV
tbYBTaWduaW5nIENBMB4XDTE4MDEwOTA3NTMw%0AMl0XDTI4MDEwNzA4NTMwMlowZAZETAPBgNVB
AMMCERlbW9Vc2VyMQswCQYDVQQG%0AEwJOTDEWMBQGA1UECAwNTm9vcmQtQmFyYmFudDESMBAGA1U
EBwwJRWluZGhvdnVu%0AMRQwEgYDVQQKDAwTaW9leCBHcm91cDEMAAoGA1UECwwDU0VUMR4wHAYJK
oZIhvcN%0AAQkBFg90ZXN0dWlAc2lvdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK%0A
AoIBAQDI98ytSED47IFmjT0PUezLlyHULxDbgXYgYF83G27J5%2BLkfn1xcWa2cCVH%0AotFG%2BK
Znfpsd5HAXUFRgJI%2BciypdMf3DS9ZH3PIScgEzwnXG0WnNUagHjUfsYQ6r%0AlI6MBKE86F8sjR
hyF%2Fr1Upogsc24ALypBdIhQ6Ham3ni4NFsYEcBk80nKK%2BpATDJ%0APqzK1IIGsd3XhOmxjVUZ
PR70FaEbHwnbOakWfeLibwJAWttdIL7KctSMsBLg3U2o%0AogYmm5fJqfLwZZEmwNslkuzGcHB6M1
WBYQHyp966klYnItBDFEMYKvaW2ec8tZEA%0AljGAWmIrgAMR9obUWoUuGVwLoOXzAgMBAAGjcDBu
MAkGA1UdEwQCMAAwEwYDVR01%0ABAwwCgYIKwYBBQUHAWIwCwYDVR0PBAQDAgAP4MCCGCWCGSAGG%2
BEIBAgQafhhDVVNU%0AX0FOT19JTLRFUk5BTF9URVNUVUkwFgYDVR0RBA8wDYILbnMuc2lvdXguZX
```



```
UwdQYJ%0AKoZIhvcNAQELBQADggIBAJXrf8FkL2Bh2rW%2FgJQOpHADELqk25%2F8UmMNbpauneVB
%0AOaJFHCXpsSUDQ4beJo3gyM5JvZRV5nfc8Na3v4aNb0loVJHDHA8%2BGs1lUXy3fN1v%0AU4YAF
yiJUlxErYyuA5g0rdxXjLSSvSTWGWPkIJEgLYC2xvO8kh%2BnwOijVciHFCp%0AGATpCbEO4MPwj
QiLpJowx5W5FhcyARvp%2FjHl0T1IsPWRuD23oG8gCG4uRAF9CKvU%0AYE8%2BV9RrYG%2B6VYeNP
2Va6DGTXVsH2%2Fi3vP7IdaO8cnVOcYcHKzOBU%2Bfr60muNpuo%0AhE%2FoGzZ7uj%2F3wja9q49
OGJaNxHjadUjk5k799e%2Fv2isBaSuNmXVobOEzjasIyF%2B%0A2VWGrxSJcfhUXSv3%2B0xnm0o
UigN7uxVl%2BriPcpvlzyNLQqzNxSgerD0iW6dIAPHj%0AFuchgAYYuRXVMHbwxfqUVUWqyBg2Pr5
ESSYB6PTDXzigChLjWJRLuEZxwXmi6ztn%0AfgSMOG9XRFFR7yxQKDEuGZKfbAcD0gXtCDbH3T%2B
UUESjAc7bUz4OM5Zxfea%2F8F%2FE%0AoBrzb6XROZEWApxmbQOWRXRu0lddl%2Fi77irHash%2FL
Gq6RGv7qsDmic%2B3WQXZtdkX%0As1TcBs1GLZkgnbWevNtA8UWHL1JdfJjKmnOv5RdYxRc6kCXEO
WQilCByQxpQXR5L%0A-----END%20CERTIFICATE-----" \
-X POST https://test.keytalk.com/public/smime-cert-enrollment-availability
```

Response

HTTP 200 - application/json - enrollment available for the given self-service user

```
{
  "status": "smime-cert-enrollment-availability",
  "available": true,
  "mobile-required": boolean
}
```

HTTP 200 - application/json - enrollment not available for the given self-service user

```
{
  "status": "smime-cert-enrollment-availability",
  "available": false,
  "reason": text
}
```

HTTP 400 - application/json - invalid request (e.g. user certificate is not S/MIME)

```
{
  "status": "error",
  "error": error message (optional)
}
```

3.2.4 Query the certificate store to place the resulted certificate

Query the type of the certificate store to place the certificate received using the certificate retrieval API. The result will be identical as `store-to-system` flag returned from `/cert` call of the certificate retrieval API, however received without going through the entire handshake-authentication procedure.

Request

Latest API version: GET `/public/should-cert-go-to-system-store`

Specific API version: GET `/public/<version>/should-cert-go-to-system-store`

Example:

`/public/should-cert-go-to-system-store?service=DEMO_SERVICE`

Query parameters

parameter	type	required	description
<code>service</code>	<i>string</i>	yes	KeyTalk service name

Response

HTTP 200 - `application/json` - successful invocation

```
{
  "status": "should-cert-go-to-system-store",
  "system-store": boolean flag indicating whether the certificate should be
  placed in the caller's System (Machine) store ("true") or User store
  ("false")
}
```

HTTP 400 - `application/json` - invalid request

```
{
  "status": "error",
  "error": error message (optional)
}
```



3.2.5 Request a disclaimer to install to the mail client

Request an email disclaimer based on the provided email address for the provided service. If the service is setup to supply a disclaimer the response will contain a disclaimer to be applied to the mail client.

Request

Latest API version: GET /public/set-disclaimer-for-smime-cert-email

Specific API version: GET /public/<version>/set-disclaimer-for-smime-cert-email

Example:

/public/set-disclaimer-for-smime-cert-email?service=DEMO_SERVICE&email=demo%40keytalk.com

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service name
email	string	yes	Email address for which to request a disclaimer

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "set-disclaimer-for-smime-cert-email",
  "set-disclaimer": flag determining whether a Disclaimer should be set to the mail client
  "disclaimer": {
    "name": String with the disclaimer name.
    "zip": Base64-encoded string containing the content of a mail client
signature compressed into a zip file.
    "check-interval": Interval at which to check disclaimers in minutes.
  } : optional, must be set if set-disclaimer is true. Extend all mail client signatures with the
contents contained in zip
}
```

HTTP 400 - application/json - invalid request e.g. the given service does not exist

```
{
  "status": "error",
  "error": error message (optional)
}
```

3.2.6 Query certificate approver emails

Query approver emails for the certificate. Typically used by services configured with 3rd CA, for example GlobalSign DomainSSL or GlobalSignAlphaSSL.

Request

Latest API version: GET /public/cert-approver-emails

Specific API version: GET /public/<version>/cert-approver-emails

Example:

```
/public/cert-approver-  
emails?service=DEMO_SERVICE&user=test.keytalk.com&computer-  
name=test2.keytalk.com
```

Query parameters

parameter	type	required	description
service	string	yes	Name of KeyTalk service (template)
user	string	yes	User name (typically FQDN of the server). Should be the same value as the USERID used in RCDP authentication request.
computer-name	string	yes	Name of the caller's machine/device name. Should be the same as the computer name as used in RCDP authentication request.

Response

HTTP 200 - application/json - successful invocation

```
{  
  "status": "cert-approver-emails",  
  "emails": JSON array of emails  
}
```

HTTP 400 - application/json - invalid request e.g. the given service does not exist is not configured with a CA implementing approval of cert requests via emails

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



3.2.7 Query KeyTalk server version

Query KeyTalk server version

Request

Latest API version: GET /public/version

Specific API version: GET /public/<version>/version

Query parameters

None

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "version",
  "version": KeyTalk server version string as Major.Minor.Subminor
}
```

3.2.8 Query Common Name customization policy

Query customization policy for Common Name for the given user.

Request

Latest API version: GET /public/cn-customization-policy

Specific API version: GET /public/<version>/cn-customization-policy

Example:

/public/cn-customization-policy?service=DEMO_SERVICE&user=test-user&computer-name=test.keytalk.com

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service name
user	string	yes	User name. Should be the same value as the USERID used in RCDP authentication request.
computer-name	string	yes	Name of the caller's machine/device name. Should be the same as the computer name as used in RCDP authentication request.

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "cn-customization-policy",
  "policy": "ALLOWED"|"DISALLOWED-NOT-SUPPORTED-BY-SERVICE"|"DISALLOWED-
CERT-STILL-VALID"|"ALLOWED-AS-GIVENNAME_SURNAME"
}
```

HTTP 400 - application/json - invalid request e.g. the given service and user combination does not exist

```
{
  "status": "error",
  "error": error message (optional)
}
```

3.2.9 Query which templates participate in auto-renewal of seat certificates.

Check which templates (services) seat certificate auto-renewal should be applied. When no templates are configured (empty list is returned) the agent should fall back to the old behavior by renewing certificates for a single template only.

Request

Latest API version: POST /public/templates-to-auto-renew-seat-certs

Specific API version: POST /public/<version>/templates-to-auto-renew-seat-certs

Query parameters

parameter	type	required	description
template-names	JSON array	yes	List of the template names to check

Example (for the recipients above):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d \
"template-names=%5B%22TEMPLATE1%22%2C%22TEMPLATE2%22%5D" \
-X POST https://test.keytalk.com/public/templates-to-auto-renew-seat-certs
```

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "templates-to-auto-renew-seat-certs",
  "template-names": JSON array of template names from the submitted list for which seat certificate auto-renewal is to be applied
}
```



3.2.10 Query certificate expiration margin

Query the minimal number of seconds (*margin*) before a certificate expiration when the certificate is considered as valid and does not require renewal. When the remaining time till the certificate expiry falls below this margin, the certificate is considered as expiring and ought to be renewed by KeyTalk agents.

Request

Latest API version: GET /public/cert-expiration-margin

Specific API version: GET /public/<version>/cert-expiration-margin

Example:

/public/cert-expiration-margin?service=DEMO_SERVICE&user=test-user&computer-name=test.keytalk.com

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service (TEMPLATE) name
user	string	no	User name (seat name). Should be the same value as the USERID used in RCDP authentication request. When user is not supplied or if the given user does not exist on the server, the TEMPLATE-wide setting will be used
computer-name	string	no	Name of the caller's machine/device name. Should be the same as the computer name as used in RCDP authentication request. Only needed when user is supplied.

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "cert-expiration-margin",
  "threshold-seconds": 86400
}
```

HTTP 400 - application/json - invalid request e.g. the given service and user combination does not exist

```
{
  "status": "error",
  "error": error message (optional)
}
```



3.2.11 Download client configuration files

Download client configuration files (RCCDs) to be used for customizing KeyTalk agents.

Request

Latest API version: GET /public/rccd?uid=<UID>

Specific API version: GET /public/<version>/rccd?uid=<UID>

Example:

/public/rccd?uid=1234567890ABCD

Query parameters

parameter	type	required	description
uid	string	yes	Client configuration UID. The UID can be relieved from KeyTalk template page.

Response

HTTP 200 - application/octet-stream - successful invocation

HTTP 400 - application/json - RCCD with the given UID does not exist.



3.2.12 Check server health status

Perform server health check. Initially made for Load Balancers to select right KeyTalk backend server.

Request

Latest API version: GET /public/health-check

Specific API version: GET /public/<version>/health-check

Query parameters

None

Response

HTTP 200 - application/json - server is up and running

```
{
  "status": "health-check",
  "check-result": "operational"
}
```

HTTP 521 - application/json - server is having issues preventing it from operating normally (e.g one of the daemons is not running)

```
{
  "status": "health-check",
  "check-result": "error",
  "error": error details (optional)
}
```



3.2.13 *[as of v1.6.6]* Retrieve message shown on KeyTalk agent automatic popup.

Retrieves notification template for the message shown to end-users when KeyTalk agent automatically pops up.

Request

Latest API version: GET /public/ktagent-unattended-popup-msg-template

Specific API version: GET /public/<version>/ktagent-unattended-popup-msg-template

Query parameters

None

Response

HTTP 200 - application/json

```
{
  "status": "ktagent-unattended-popup-msg-template",
  "msg-template": <message-template>
}
```

3.2.14 [\[as of v1.6.7\]](#) Retrieve keep-alive interval.

Retrieves the interval the agent is supposed to send periodic keep-alive message to the server.

Request

Latest API version: GET /public/keep-alive-interval?template-name=<template-name>

Specific API version: GET /public/<version>/keep-alive-interval?template-name=<template-name>

Response

HTTP 200 - application/json

```
{
  "status": "keep-alive-interval",
  "interval": interval value followed by a unit e.g. "3600s" or empty string if no keep-alive
interval is configured on the server for the given template
}
```

3.2.15 [\[as of v1.6.7\]](#) Send keep-alive

Send keep-alive message for the given KeyTalk user (seat). Normally sent by agents periodically according to the keep-alive interval retrieve as described in the section 3.2.14

Request

Latest API version: POST /public/i-am-alive

Specific API version: POST /public/<version>/i-am-alive

Query parameters

parameter	type	required	description
template-name	string	yes	KeyTalk service (TEMPLATE) name
user	string	yes	User name. Should be the same value as the USERID used in RCDP authentication request
computer-name	string	yes	Name of the caller's machine/device name. Should be the same as the computer name as used in RCDP authentication request.

Response

HTTP 200 - application/json

```
{
  "status": "success",
  \[as of v1.6.9\] "interval": keep-alive interval value followed by a unit e.g. "3600s" or
empty string if no keep-alive interval is configured on the server for the given template
}
```

}

3.2.16 *[as of v1.6.8]* Check whether a seat certificate is known to have TPM key attestation.

Check whether the given seat certificate is known to have TPM key attestation i.e. was previously attested by one of KeyTalk TPM endorsement CAs.

Request

Latest API version: GET /public/is-cert-tpm-attested?cert-sha1-fingerprint=<cert-sha1-fingerprint>

Specific API version: GET /public/<version>/is-cert-tpm-attested?=<cert-sha1-fingerprint>

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "status": "yes" | "no"
}
```

HTTP 400 - application/json invalid request e.g. if no seat certificate with the given name found

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{
  "status": "error",
  "error": error message (optional)
}
```



4. ADMINISTRATOR API

A set of API to be used by KeyTalk admins and managers to perform the same (subset of) tasks as using KeyTalk Web Admin Interface.

4.1 API versions

REST API version	Minimal KeyTalk server version	Changes wrt the previous API version
1.9.4	7.4.2	Allow creating templates for enrolling DigiCert DV certs via ACME
1.9.5	7.5.8	Allow creating tenants (groups of templates) and assigning templates to it. Allow specifying external customer number and external order id when copying template. Allow enabling ACME for the template
1.9.6	7.8.3	Allow creating seats

4.2 API overview

The communication goes over HTTPS and uses port 3000 secured using KeyTalk internal CA or, when, enabled server-side, using a public trusted CA such as GlobalSign or QuoVadis. All the API calls should be authenticated using credentials of the Web Admin Interface (username/password or a client certificate). Admin API requires a valid system admin/cluster admin/manager or operator privileges to execute.

4.2.1 Enroll seat from a server-generated keypair

Enroll a certificate for the given seat, creating the seat if not exist. The keypair is created server-side.

Request

Latest API version: POST /admap/api/cert-enrollment

Specific API version: POST /admap/api/<version>/cert-enrollment

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.

service	string	yes	n/a	Name of KeyTalk service (template) of the user (seat) to enroll.
deviduser	string	yes	n/a	Name of KeyTalk DevID user (seat) to enroll.
san	JSON array	no	empty	Array of Subject Alternative Names to be used in the certificate e.g. ["DNS:test.server.com", "IP:192.168.1.2"]

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/cert-enrollment
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/cert-enrollment
```

Response

HTTP 200 - application/json

```
{
  "status": "cert-enrollment",
  "cert": enrolled certificate and key in PEM format
  "created-user-auth-password": optional, authentication password of a newly created
  KeyTalk user provided the user being enrolled did not exist in an authentication back-end (only users
  of an internal db authentication backend can be automatically created this way)
}
```

HTTP 400 - application/json invalid request

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{
  "status": "error",
  "error": error message (optional)
}
```

4.2.2 Enroll seat with client CSR

An admin might want to generate a key pair by himself and submit the resulted CSR to KeyTalk server for signing. Enrolling a seat with a client-side CSR consists of 3 steps: querying the KeyTalk server for parameters to use in the CSR, generation of the CSR and submitting the CSR to the KeyTalk server for signing.

4.2.2.1 Query CSR requirements for enrolment

Prior submitting a CSR to the KeyTalk server, an admin should query the server for the initial parameters for the CSR such as key size, signing algorithm, certificate subject and a subject alternative name (SAN).

Request

Latest API version: POST /admap/api/csr-enrolment-requirements

Specific API version: POST /admap/api/<version>/csr-enrolment-requirements

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
service	string	yes	n/a	Name of KeyTalk service (template) of the seat to create a CSR for.
deviduser	string	yes	n/a	Name of KeyTalk DevID user (seat) to create a CSR for.

Response

HTTP 200 - application/json

```
{
  "status": "csr-enrolment-requirements",
  "key-size": key size in bits,
  "signing-algo": algorithm to use for CSR signing,
  "subject": dictionary of subject fields to use in CSR, [ as of server v7.4.1 empty
attribute names should be treated by the caller as "any" attribute
instead of a literal value],
```

```

    "san": if set, holds an array of Subject Alternative Names to use in CSR, [ as of server
v7.4.1 empty SAN should be treated by the caller as "any" instead of
a literal empty value],

}

```

Example:

```

{
  "status": "csr-requirements",
  "key-size": "2048",
  "signing-algo": "sha256",
  "subject": {"cn": "TestUser",
              "c": "NL",
              "st": "Utrecht",
              "l": "Amstelveen",
              "o": "KeyTalk",
              "ou": "Development",
              "e": "test@keytalk.com",
              },
  "san": ["email:test@keytalk.com", "DNS:test.keytalk.com"]
}

```

Example (for username/password authentication):

```

$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/csr-enrolment-requirements

```

Example (for client certificate authentication):

```

$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/csr-enrolment-requirements

```

4.2.2.2 Enroll seat with a client-generated CSR

Enroll a certificate for the given seat, creating the seat if not exist. The CSR is supplied by an admin and should obey the requirements retrieved with `csr-enrolment-requirements` call.

Request

Latest API version: POST /admapi/cert-enrollment-for-csr

Specific API version: POST /admapi/<version>/cert-enrollment-for-csr

Content-type: application/x-www-form-urlencoded

Request POST parameters:



parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
service	string	yes	n/a	Name of KeyTalk service (template) of the seat to enroll
deviduser	string	yes	n/a	Name of KeyTalk DevID user (seat) to enroll
csr	string	yes	n/a	Base64 encoded PKCS#10 certificate signing request.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Expect:" \
-d "auth-username=admin&auth-password=secret" \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-d "csr-----BEGIN+CERTIFICATE+REQUEST-----
%0AMiIC1jCCAb4CAQAwwZAxZAJBgNVBAYTAk5MMRIwEAYDVQQHDA1FaW5kaG92ZW4x%0ADDAKBgN
VBAsMA1NFUzEUMBIGA1UECgwLU2lvdXggR3JvdXAxXjAUBGNVBAgMDU5v%0Ab3JkLUJhcmJhbnQxE
TAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKoZIhvcNAQkBFg90%0AZXN0dW1Ac2lvdXguZXUwggEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAOIBAQDQ%0AfyCCkM7cbVhpBCSx1Nf%2BFDqa9banKf9sPRW
5VwBFYP5siLdsywnNqrFYcV0w6ss%0Ath2lqK9bkjZoyiKpbzvzgQw08NlbBmJfj700l8HUn2xL
vp2z6J6q3Z4rAR4d8jx%0ApwcdRlPeJO5b30tBaURKILaJTjtsUVyCXr%2B6u%2FgiuaD0DGBKsIQ
ccyAWGy%2BlzNer%0AsmUib%2FsnWHEaAPJtvg7T2amaWACKcqIOppR%2BHDJUUNSYyju9xZqCLjx
6Y2%2B2ZXHK%0AMpFcFSP%2F8GCYgZ2%2FAiLWtsVzKSaRWmTVJfBsy50gW3YmwI0QYghl52NIDQu
BJeoT%0AmQFxsKXpqCwJpP3KTOS5AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAbUVCaYm%2F%0A
w1otZaLgtCP2mIVVH%2FgHvTeVFs1436Lz%2FaKT5q1QRee8lC2us1z9G7h3PG%2BM6w1N%0AUJau
wQ2mR2c1VAidROdT52syNPR4jXeRl1%2F7a%2FmsZFqaw3%2FLlwVtBJHEfOA6apU%0AjSVWi6%2
F3kUjD0FhYHAufKm2nJl0qGnwC5xpzuvYOQsUFFobLZoyGq5NNEgnSpK8X%0A9A9j5kKGBom9eQOr
Wwx%2F0UlwRqLpt6l76Gt5%2BlMp5BtTCPK2uboHvJiPu4aJUuHh%0Afx9ZjKox73V%2BleOEmNSY
fesuQPE5AwifKE988NfixGXOHw7uQdWc9SFsYFRFZG2p%0AYb%2Bm9iFYUY8AHw%3D%3D%0A-----
END+CERTIFICATE+REQUEST-----%0A" \
-X POST https://test.keytalk.com:3000/admapi/cert-enrollment-for-csr
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-H "Expect:" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-d "csr-----BEGIN+CERTIFICATE+REQUEST-----
%0AMiIC1jCCAb4CAQAwwZAxZAJBgNVBAYTAk5MMRIwEAYDVQQHDA1FaW5kaG92ZW4x%0ADDAKBgN
VBAsMA1NFUzEUMBIGA1UECgwLU2lvdXggR3JvdXAxXjAUBGNVBAgMDU5v%0Ab3JkLUJhcmJhbnQxE
TAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKoZIhvcNAQkBFg90%0AZXN0dW1Ac2lvdXguZXUwggEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAOIBAQDQ%0AfyCCkM7cbVhpBCSx1Nf%2BFDqa9banKf9sPRW
5VwBFYP5siLdsywnNqrFYcV0w6ss%0Ath2lqK9bkjZoyiKpbzvzgQw08NlbBmJfj700l8HUn2xL
vp2z6J6q3Z4rAR4d8jx%0ApwcdRlPeJO5b30tBaURKILaJTjtsUVyCXr%2B6u%2FgiuaD0DGBKsIQ
ccyAWGy%2BlzNer%0AsmUib%2FsnWHEaAPJtvg7T2amaWACKcqIOppR%2BHDJUUNSYyju9xZqCLjx
```



```
6Y2%2B2ZXHK%0AMpFcFsP%2F8GCGZ2%2FAi1WtsVzKSaRWmTVJfBsy50gW3YmwI0QYgh152NIDQu
BJeoT%0AmQFxsKXpqcWjpp3KTOS5AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAbUVCaYm%2F%0A
wlotZaLgtCP2mIVVH%2FgHvTeVFs1436Lz%2FaKT5q1QRee81C2us1z9G7h3PG%2BM6w1N%0AUJau
wqQ2mR2c1VAidROdT52syNPR4jXeR11%2F7a%2FmsZFqaw3%2FLlwVtBJHEfOA6apU%0AjSVWi6%2
F3kUjD0FhYHAufKm2nJ10qGnwC5xpzuvYOQsUFFobLZoyGq5NNEgnSpK8X%0A9A9j5kKGB0m9eQOr
Wxw%2F0UlwRqLpt6l76Gt5%2BlMp5BtTCPK2uboHvJiPu4aJUuHh%0Afx9ZjKox73V%2BleOEmNSY
fesuQPE5AwifkE988NFixGXOHw7uQdWc9SFsYFRFZG2p%0AYb%2Bm9iFyUY8AHw%3D%3D%0A-----
END+CERTIFICATE+REQUEST-----%0A" \
-X POST https://test.keytalk.com:3000/admapi/cert-enrollment-for-csr
```

Response

HTTP 200 - application/json

```
{
  "status": "cert-enrollment",
  "cert": enrolled certificate and key in PEM format
  "created-user-auth-password": optional, authentication password of a newly
  created KeyTalk user provided the user being enrolled did not exist in an
  authentication back-end (only users of an internal db authentication backend
  can be automatically created this way)
}
```

HTTP 400 - application/json invalid request

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{
  "status": "error",
  "error": error message (optional)
}
```

4.2.3 Revoke seat certificates

Revoke certificates of the given seat. Requires a valid system/cluster/manager or an operator privilege to execute.

Request

Latest API version: POST /admapi/cert-revocation

Specific API version: POST /admapi/<version>/cert-revocation

Content-type: application/x-www-form-urlencoded

Request POST parameters:



parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
service	string	yes	n/a	Name of KeyTalk service of the user whose certificates are to be revoked
deviduser	string	yes	n/a	Name of KeyTalk DevID user (seat) whose certificates are to be revoked

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/cert-revocation
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "service=DEMO_SERVICE&deviduser=DemoUser" \
-X POST https://test.keytalk.com:3000/admapi/cert-revocation
```

Response

HTTP 200 - application/json

```
{
  "status": "cert-revocation",
  "num-revoked-certs": number of revoked certificates,
  "warning": [optional] warning message
}
```

HTTP 400 - application/json invalid request

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{
  "status": "error",
  "error": error message (optional)
}
```

4.2.4 Download KeyTalk settings

Download KeyTalk settings. This is the counterpart of the saving settings under the System -> Settings page of KeyTalk Web Admin interface.

Request

Latest API version: POST /admapl/settings

Specific API version: POST /admapl/<version>/settings

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
include-hsm-connection-settings	boolean	no	false	include HSM Connection Settings
include-keytalk-cert-tree	boolean	no	false	include KeyTalk Certificate Tree

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "include-shared-settings=true&include-db-connection-settings=true" \
-X POST https://test.keytalk.com:3000/admapl/settings
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "include-shared-settings=true&include-db-connection-settings=true" \
-X POST https://test.keytalk.com:3000/admapl/settings
```

Response

HTTP 200 - application/octet-stream - settings is returned in HTTP response body

HTTP 400 - application/json invalid request

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```

4.2.5 Retrieve SCEP configuration.

Retrieve Intune SCEP configuration. This is the counterpart of the Certificate and Keys -> Intune SCEP page of KeyTalk Web Admin interface.

Request

Latest API version: POST /admap/api/intune-scep-config

Specific API version: POST /admap/api/<version>/intune-scep-config

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-X POST https://test.keytalk.com:3000/admap/api/intune-scep-config
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-X POST https://test.keytalk.com:3000/admap/api/intune-scep-config
```

Response

HTTP 200 - application/json

```
{
  "status": "scep-config",
  "enabled": Boolean flag indicating whether SCEP configuration is enabled. The properties below
  are returned if "enabled" is true,
  "service-name": service name
  "recipient-cert": recipient certificate (PEM),
  "recipient-key": recipient key (PKCS#8 PEM),
  "signing-cert": signing certificate (PEM),
  "signing-key": signing key (PKCS#8 PEM),
  "issuer-certs": concatenated issuer CA chain of the signing and recipient cert above (PEM),
}
```

HTTP 400 - application/json invalid request
HTTP 401 - application/json invalid credentials
HTTP 500 - application/json - generic server-side enrolment error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.6 Copy KeyTalk TEMPLATE

Copy KeyTalk TEMPLATE along with all their properties but without seats. The caller should supply a name of the new TEMPLATE and, optional, some extra TEMPLATE properties to be altered.

The required authorization is the system admin, cluster admin or a manager. The copied template gets automatically assigned to the manager performed the call. The copied template also automatically becomes a member of each tenant assigned to the manager.

Request

Latest API version: POST /admapi/copy-template

Specific API version: POST /admapi/<version>/copy-template

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
src-template-name	string	yes	n/a	Name of the TEMPLATE to copy
new-template-name	string	yes	n/a	Name of the new TEMPLATE
digicert-central-settings	JSON	if applicable	n/a	If the source TEMPLATE is configured with DigiCert Central CA source, an admin can alter some of these settings in the new TEMPLATE. <pre>{ "product": "ssl_basic" "ssl_ev_basic" "class1_smime" "client_premium" "ssl_securesite_flex" "ssl_ev_securesite_flex" "ssl_geotrust_truebizid" "ssl_thawte_webserver" "ssl_ev_thawte_webserver", "api-key": new API-key, "account-region": "US" "Europe", "cert-validity-months": new certificate validity in months,</pre>

					"organization-id": new organization ID, if applicable, "approver-user-id": approver user ID, if applicable }
<i>[as of 1.9.5]</i>	external-customer-settings	JSON	no	n/a	External customer settings. { "customer-id": customer ID, "auth-username": username (basic HTTP authentication) for the external customer certificate order API endpoint, "auth-password": username (basic HTTP authentication) for the external customer certificate order API endpoint, "order-ids": [...] }

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret&src-template-name=TEMPL&new-template-name=TEMPL-NEW" \
-X POST https://test.keytalk.com:3000/admapi/copy-template
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "src-template-name=TEMPL&new-template-name=TEMPL-NEW" \
-X POST https://test.keytalk.com:3000/admapi/copy-template
```

Example (with custom DigiCert Central settings):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "src-template-name=TEMPL&new-template-name=TEMPL-NEW" \
-d "digicert-central-settings=%7B%22product%22%3A%22ssl_basic%22%2C%20%22api-key%22%3A%22123456%22%2C%20%22cert-validity-months%22%3A%2212%2C%20%22organization-id%22%3A%206543%7D" \
-X POST https://test.keytalk.com:3000/admapi/copy-template
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "external-customer-reserved-seats": returned only if the request included external-customer-settings parameter. Will contain a JSON dictionary mapping from the external customer order IDs to the seat name reserved by KeyTalk e.g.
  {"O20240601001": "seat-0001", "O20240601002": "seat-0002"}
}
```

HTTP 400 - application/json invalid request, e.g. a new template with the given name already exists
HTTP 401 - application/json invalid credentials
HTTP 500 - application/json - generic server-side enrolment error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.7 Import certificates

Import certificates into KeyTalk under the given TEMPLATE. Target seat names are derived from the Common Name or from SAN of the certificates. Originally made for SSL Scanner project ssls scanner.keytalk.com to send its scan results to KeyTalk.

In addition to the system admin and a cluster admin, an SSL Discovery Manager assigned to the given TEMPLATE, is also eligible to execute this API call.

Request

Latest API version: POST /admap/api/import-certs

Specific API version: POST /admapl/<version>/import-certs

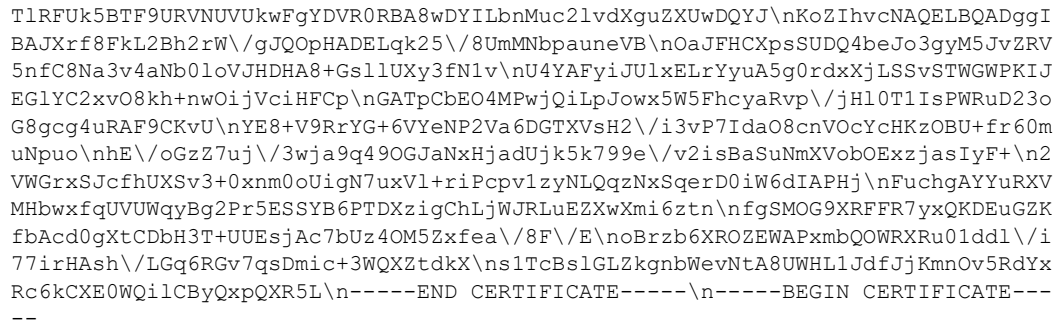
```
Content-type: application/json
```

Request POST parameters:

parameter	type	required	default value	description
custom.auth-username	<i>string</i>	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
custom.auth-password	<i>string</i>	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
custom.template-name	<i>string</i>	yes	n/a	Target KeyTalk TEMPLATE name
custom.owner-name	<i>string</i>	no	no owner	Certificates owner name. Will be used to notify on the certificate revocation and expiration.
certificates	<i>string</i>	yes	n/a	Concatenated list of PEM certificates.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/json" \
--data-raw "{
  \"certificates\": {\"-----BEGIN CERTIFICATE-----
  nMIIFDCCAvSgAwIBAgIIWlSC7gAAAYowDQYJKoZIhvcNAQELBQAwGygxHzAdBgkq\\nhkiG9w0BC
  QEWEgluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\\nDBNLZXlUYWxrIElUIFNl
  Y3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmFlbHQx\\nIDAeBgNVBAMMF0tleVRhbGsgRGVtbyB
  TaWduaW5yMiENBMB4XDTE4MDEwOTA3NTMw\\nmLOxDTI4MDEwNzA4NTMwMl9wZDZAxETAPBgNVBAMCE
  RlBw9Vc2V5MjQwYDQVQg\\nEwJ0TDEWMBQGA1UECAnTm9vcmtQcmFyYmFudDEWMBAGA1UEBwwJR
  WluZGhvdnVu\\nMRQwEgYDVQQKDAkTaW9leCBHcm91cDEMMAAoGA1UECwwDU0VTMR4wHAYJKoZIhvcN
  \\nAQkBFG90ZXN0dWlAc2lvdXguZlUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\\nAoIBAQDI9
  8ytSED47IfmjT0PUezLlyHULxDbggXyGf83G27J5+Lkfn1xcWa2cCVH\\nnotFG+KZNfpds5HaxUFrg
  JT+ciypdMf3DS9ZH3PIScgEzwNXG0WnUagHjUfsYQ6r\\n1I6MBKE86F8sjRhyF\\n/r1Upogsc24AL
  ypBdIhQ6Ham3ni4NFsYEcBk80nKK+pATDJ\\nPqzK1IIGsd3XhOmxjVUzPR7OfaEbHwnbOakWfeLib
  yJawTtdIL7KcTSMsBLg3U2o\\nogYmm5YJqfLwZEmwNslkuzGcHB6M1WBYYHyp966k1YnItBDFEMy
  kVaW2ec8tZEA\\nljGAWmIrgAMR9yobUwOUGVwLoXcAGCMBAGjcdBuMAkGA1UEwQCMAAAwEwYDVRO
  1\\nBAwwCgYIKwYBOUAAwIwCwYDVRO1PBAODAgP4MCCGAGCwGCSAGG+EIbAqOaFhhDVVNU\\nXOFOT19
```



```
\ "custom\: {
  \"auth-username\": \"admin\",
  \"auth-password\": \"secret\",
  \"template-name\": \"TEMPL\"
} } \" \
-X POST https://test.keytalk.com:3000/admapi/import-certs
```

HTTP 200 - application/json

82

HTTP 400 - application/json invalid request, e.g. a template with the given name does not exist

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side enrolment error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.8 [as of v1.9.6] Create or update seat

Create new seat or update existing seat.

Latest API version: POST /admap/api/create-seat

Specific API version: POST /admap/api/<version>/create-seat

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of KeyTalk TEMPLATE (service) the seat belongs to
seat-name	string	yes	n/a	Name of KeyTalk seat to create
cn	string	no	empty	Seat common name
san	JSON array	no	empty	Seat Subject Alternative Names. Example: ["DNS:test.server1.com", "DNS:test2.server.com"]

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "template-name=DEMO_SERVICE&seat-name=DemoUser" \
-d "san=[\"DNS:test.server1.com\", \"DNS:test2.server.com\"]" \
-X POST https://test.keytalk.com:3000/admap/api/create-seat
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name=DEMO_SERVICE&seat-name=DemoUser" \
-d "san=[\"DNS:test.server1.com\", \"DNS:test2.server.com\"]" \
-X POST https://test.keytalk.com:3000/admap/api/create-seat
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
```

```
"result": "created" | "updated"
}
```

HTTP 400 - application/json invalid request e.g. the given TEMPLATE or a seat does not exist

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side error

```
{
  "status": "error",
  "error": error message (optional)
}
```



4.2.9 Archive seat

Archive the given seat. Requires a valid system/cluster/manager privilege to execute.

Request

Latest API version: POST /admap/api/archive-seat

Specific API version: POST /admap/api/<version>/archive-seat

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the webserver	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the webserver	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of KeyTalk TEMPLATE (service) the seat belongs to
seat-name	string	yes	n/a	Name of KeyTalk seat (DevID user) to archive

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "template-name=DEMO_SERVICE&seat-name=DemoUser" \
-X POST https://test.keytalk.com:3000/admap/api/archive-seat
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name =DEMO_SERVICE&seat-name=DemoUser" \
-X POST https://test.keytalk.com:3000/admap/api/archive-seat
```

Response

HTTP 200 - application/json

```
{
  "status": "archive-seat",
  "archived": boolean indicating whether the function had effect i.e. the seat turned from non-archived to archived
}
```

```
}
```

HTTP 400 - application/json invalid request e.g. the given TEMPLATE or a seat does not exist

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.10 List KeyTalk templates

List available KeyTalk templates.

Request

Latest API version: POST /admap/api/list-templates

Specific API version: POST /admap/api/<version>/list-templates

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-X POST https://test.keytalk.com:3000/admap/api/list-templates
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-X POST https://test.keytalk.com:3000/admap/api/list-templates
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "templates": JSON array of templates
}
```



HTTP 401 - application/json invalid credentials
HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.11 Remove KeyTalk template.

Remove KeyTalk template along with all the seats and Registration Authorities bound to it.

Request

Latest API version: POST /admap/api/remove-template

Specific API version: POST /admap/api/<version>/remove-template

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of the template to remove

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret&template-name=TEMPL" \
-X POST https://test.keytalk.com:3000/admap/api/remove-template
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name=TEMPL" \
-X POST https://test.keytalk.com:3000/admap/api/remove-template
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
}
```

HTTP 400 - application/json invalid request, e.g. a template with the given name does not exist

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```

4.2.12 Open slot on the seat

Open slot on the existing seat. The call will change the status of the selected slot of the given seat from *locked* to *learn-once*. The function will have no effect when the selected slot is already *learn-once*.

Request

Latest API version: POST /admap/api/open-slot

Specific API version: POST /admap/api/<version>/open-slot

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of the TEMPLATE to remove
seat-name	string	yes	n/a	Name of the seat to open the slot for
unoccupied-only	boolean	no	False	Look up for the first unoccupied locked slot instead of simply taking the first slot.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret&template-name=TEMPL&seat-
name=seat001" \
-X POST https://test.keytalk.com:3000/admap/api/open-slot
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name=TEMPL&seat-name=seat001" \
-X POST https://test.keytalk.com:3000/admap/api/open-slot
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "extra-info": extra info (optional)
}
```

HTTP 400 - application/json invalid request, e.g. a template or s seat with the given name does not exist or the current slot state is *blocked* or *learn-always*

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side error

```
{
  "status": "error",
  "error": error message (optional)
}
```

4.2.13 Create internal RA user.

Creates a new user to the KeyTalk Internal Registration Authority database.

Request

Latest API version: POST /admap/api/create-internal-ra-user

Specific API version: POST /admap/api/<version>/create-internal-ra-user

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	<i>string</i>	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	<i>string</i>	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	<i>string</i>	yes	n/a	Name of the TEMPLATE to create a user under
user-name	<i>string</i>	yes	n/a	Name of the new user
user-password	<i>string</i>	no	empty	Password for the new user
user-password-ttl	<i>integer</i>	no	no expiry	Time-to-live in seconds for the user password
user-pincode	<i>string</i>	no	empty	Pincode for the new user
user-cert-subject	<i>JSON object</i>	no		Certificate subject overwrites for the new created user. Supported subject attributes are: "c" (country) "st" (state) "l" (city/locality) "o" (organization) "ou" (organization unit) "e" (email) Example: <pre>{ "C": "NL", "L": "Amsterdam" }</pre>
user-cert-san	<i>JSON array</i>	no		Certificate Subject Alternative Name overwrites for the new created user. Supported SAN attributes are: "DNS", "IP" and "email".

Example:

```
["ctest.server1.com",  
"DNS:test2.server.com",  
"IP:192.168.1.2"]
```

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \  
-d "auth-username=admin&auth-password=secret&template-name=TEMPL&user-  
name=user1&user-password=secret&user-password-ttl=3600&user-cert-  
subject={"C":"","NL":"","L":"","Amsteradm"}&user-cert-  
san=["DNS:test.server1.com","DNS:test2.server.com","IP:192.168.1.2"]" \  
-X POST https://test.keytalk.com:3000/admapi/create-internal-ra-user
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \  
--cert ./client-cert.pem \  
--key ./client-cert-key.pem \  
-d "template-name=TEMPL&user-name=user1&user-password=secret&user-password-  
ttl=3600&user-cert-subject={"C":"","NL":"","L":"","Amsteradm"}&user-cert-  
san=["DNS:test.server1.com","DNS:test2.server.com","IP:192.168.1.2"]" \  
-X POST https://test.keytalk.com:3000/admapi/create-internal-ra-user
```

Response

HTTP 200 - application/json

```
{  
  "status": "success",  
}
```

HTTP 400 - application/json invalid request, e.g. the template with the given name does not exist or the user with this name already exists

HTTP 401 - application/json invalid authentication credentials

HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.14 Update seats

Mass-update some seats properties for the given TEMPLATE.

Request

Latest API version: POST /admap/api/update-seats

Specific API version: POST /admap/api/<version>/update-seats

Content-type: x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Target KeyTalk TEMPLATE name
seats	JSON array	yes	n/a	<p>JSON array of seats with the properties to set.</p> <p>A seat is identified by its name given as "seat-name".</p> <p>At the moment, the only supported seat property is "mobile" standing for the mobile phone number.</p> <pre>[{ "name": "seat1", "mobile": "+312345678" }, { "name": "seat2", "mobile": "+312345679" }]</pre>



Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret&template-name=TEMPL" \
-d
"seats=[{"name":"seat1","mobile":"%2B312345678"}, {"name":"seat2",
"mobile":""}]" \
-X POST https://test.keytalk.com:3000/admapi/update-seats
```

Response

HTTP 200 - application/json

```
{
  "status": "success", received the all the supplied seats got successfully updated
}
```

HTTP 400 - application/json invalid request, e.g. a template with the given name does not exist or some of the supplied seats do not exist or the supplied seat property is not valid

HTTP 401 - application/json invalid credentials

HTTP 500 - application/json - generic server-side error

```
{
  "status": "error",
  "error": error message (optional)
}
```

4.2.15 Remove seat.

Remove seat automatically revoking seat certificate.

Request

Latest API version: POST /admap/api/remove-seat

Specific API version: POST /admap/api/<version>/remove-seat

Content-type: x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of KeyTalk TEMPLATE (service) the seat belongs to
seat-name	string	yes	n/a	Name of KeyTalk seat to remove

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "template-name=DEMO_SERVICE&seat-name=DemoUser" \
-X POST https://test.keytalk.com:3000/admap/api/remove-seat
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name =DEMO_SERVICE&seat-name=DemoUser" \
-X POST https://test.keytalk.com:3000/admap/api/remove-seat
```

Response

HTTP 200 - application/json

```
{
  "status": "remove-seat",
  "removed": True if the seat existed and got removed; False if the seat does not exist
  "warning": warning message (optional)
}
```

HTTP 400 - application/json invalid request e.g. the given TEMPLATE or a seat does not exist
HTTP 401 - application/json invalid credentials
HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```

4.2.16 *[as of v1.9.4]* Create template for enrolling DigiCert DV certs via ACME.

Allow creating templates for enrolling DigiCert DV certs via ACME.

Request

Latest API version: POST /admap/api/create-digicert-dv-acme-template

Specific API version: POST /admap/api/<version>/create-digicert-dv-acme-template

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	<i>string</i>	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	<i>string</i>	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	<i>string</i>	yes	n/a	Name of KeyTalk TEMPLATE (service)
digicert-account-region	<i>string</i>	yes	n/a	Region the DigiCert account belongs to. Can be either "Europe" or "US".
digicert-product	<i>string</i>	yes	n/a	DigiCert DV product ID such as "ssl_dv_geotrust" or "ssl_dv_rapidssl". See https://dev.digicert.com/en/certcentral-apis/services-api/glossary.html#product-identifiers for the full list.
digicert-api-key	<i>string</i>	yes	n/a	Key to call DigiCert REST API
cert-validity-months	<i>number</i>	yes	n/a	Desired certificate validity in months
mail-fetch-protocol	<i>string</i>	yes	n/a	Protocol to access the mailbox where the certificate order approval email is expected to arrive. Can be either "o365-msgraph" or "o365-imap"
azure-client-id	<i>string</i>	yes	n/a	Azure Client ID to access Office 365 mailbox where the certificate order approval email is expected to arrive



azure-client-secret	string	yes	n/a	Azure secret to access Office 365 mailbox where the certificate order approval email is expected to arrive
azure-tenant-id	string	yes	n/a	Azure Tenant ID to access Office 365 mailbox where the certificate order approval email is expected to arrive
approver-email	string	yes	n/a	Mailbox (email address) where the certificate order approval email is expected to arrive

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "template-name=DEMO_SERVICE&digicert-account-region=Europe&digicert-
product=ssl_dv_rapidssl&digicert-api-key= XXXXXXXXXXXXXXXXXXXX&cert-validity-
months=12&mail-fetch-proto=o365-msgraph&azure-client-id= abcdabcd-1234-abcd-
fc11-223344556677&azure-client-secret=secret&azure-tenant-id=01234567-ABCD-
abcd-fc11-223344556677&approver-email=approver@mydoman.com" \
-X POST https://test.keytalk.com:3000/admapi/create-digicert-dv-acme-template
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name=DEMO_SERVICE&digicert-account-region=Europe&digicert-
product=ssl_dv_rapidssl&digicert-api-key= XXXXXXXXXXXXXXXXXXXX&cert-validity-
months=12&mail-fetch-proto=o365-msgraph&azure-client-id= abcdabcd-1234-abcd-
fc11-223344556677&azure-client-secret=secret&azure-tenant-id=01234567-ABCD-
abcd-fc11-223344556677&approver-email=approver@mydoman.com" \
-X POST https://test.keytalk.com:3000/admapi/create-digicert-dv-acme-template
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "acme-directory-url": "https://test.keytalk.com/acme/directory?keytalk-
template-name=DEMO_SERVICE"
}
```

acme-directory-url – KeyTalk ACME directory URL do be used in ACME agent e.g. certbot

HTTP 400 - application/json invalid request e.g. the given TEMPLATE already exists
HTTP 401 - application/json invalid credentials
HTTP 500 - application/json – generic server-side error

```
{
  "status": "error",
  "error": error message (optional)
}
```



4.2.17 *[as of v1.9.5]* Create tenant.

Allow creating tenants (groups of templates) and assigning templates to it.

Request

Latest API version: POST /admap/api/create-tenant

Specific API version: POST /admap/api/<version>/create-tenant

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
tenant-name	string	yes	n/a	Name of KeyTalk tenant
assigned-templates	JSON array	no	n/a	KeyTalk templates to be assigned to this tenant. The templates should already exist.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "tenant-name=new-tenant" \
-d "assigned-templates=[\"template1\", \"template2\"]" \
-X POST https://test.keytalk.com:3000/admap/api/create-tenant
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "tenant-name=new-tenant" \
-d "assigned-templates=[\"template1\", \"template2\"]" \
-X POST https://test.keytalk.com:3000/admap/api/create-tenant
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
}
```

HTTP 400 - application/json invalid request e.g. the given tenant already exists
HTTP 401 - application/json invalid credentials
HTTP 500 - application/json - generic server-side error

```
{  
  "status": "error",  
  "error": error message (optional)  
}
```



4.2.18 *[as of v1.9.5]* Enable ACME for the template.

Allow requesting certificate via ACME on the template and yield ACME directory URL, which is either template-wide or tied to a seat name reserved to the previously submitted order ID.

Request

Latest API version: POST /admap/api/enable-acme

Specific API version: POST /admap/api/<version>/enable-acme

Content-type: application/x-www-form-urlencoded

Request POST parameters:

parameter	type	required	default value	description
auth-username	string	if required by the server	n/a	Caller's username. Required if the webserver is configured with username/password authentication.
auth-password	string	if required by the server	n/a	Caller's password. Required if the webserver is configured with username/password authentication.
template-name	string	yes	n/a	Name of KeyTalk template to enable requesting certs via ACME for.
external-customer-order-id	string	no	n/a	External customer order ID previously submitted using copy-template API call. Submitting the external customer order ID only affects ACME URL sent back by incorporating the name of the seat reserved for this order ID into the URL.

Example (for username/password authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
-d "auth-username=admin&auth-password=secret" \
-d "template-name=template1" \
-X POST https://test.keytalk.com:3000/admap/api/enable-acme
```

Example (for client certificate authentication):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d "template-name=template1" \
-X POST https://test.keytalk.com:3000/admap/api/enable-acme
```

Response

HTTP 200 - application/json

```
{
  "status": "success",
  "acme-directory-url": "https://test.keytalk.com/acme/directory?keytalk-
template-name=DEMO_SERVICE"
}
```

acme-directory-url – KeyTalk ACME directory URL do be used in ACME agent e.g. certbot.

HTTP 400 – application/json invalid request e.g. the given template does not exist

HTTP 401 – application/json invalid credentials

HTTP 500 – application/json – generic server-side error

```
{
  "status": "error",
  "error": error message (optional)
}
```

5. SELF-SERVICE API

There is a set of the API to be used by KeyTalk self-service user. The API requires client certificate and key as an authentication means. The Self-Service API should not be confused with the Public API which does not require authentication yet might require a client certificate (without a key) to identify a self-service user. In this case the user certificate is communicated as a part of REST API call, whereas the Self-Service API requires the certificate and the key during TLS connection establishment phase.

5.1 API versions

REST API version	Minimal KeyTalk server version	Changes wrt the previous API version
1.0.0	5.5.0	n/a
1.1.0	5.5.1	added synchronous flag to smime-cert-enrollment call
1.1.1	5.5.3	added apply-address-books flag to smime-cert-enrollment call

5.2 API overview

The communication goes over HTTPS and uses port 3000. All the Self-Service API calls should be authenticated with a certificate and key identifying the caller as KeyTalk self-service user. KeyTalk server should be configured to require certificate-based logins.

5.2.1 Enroll S/MIME certificates for external parties

Enroll or place orders for S/MIME certificates for external parties i.e. to users that are generally not registered at KeyTalk. The enrolled certificates will be communicated to the indicated email addresses. It is strongly recommended to call `smime-cert-enrollment-availability` from the [Public API](#) before enrolling a certificate to get more detailed diagnostics when the enrolment is not possible and minimize the chance of errors during enrolment.

Request

```
POST /ssapi/1.1.1/smime-cert-enrollment
Content-type: application/x-www-form-urlencoded
```

Request POST parameters:

parameter	type	required	default value	description
recipients	JSON object	yes	n/a	<p>JSON array of recipients. Each recipient contains at least an email address to send the download link of the generated S/MIME certificate to.</p> <p>A recipient might also include mobile phone number to receive the password for the S/MIME certificate and key. The mobile number is only required when it is enforced by the server</p>

configuration otherwise it is optional. If mobile phone is neither enforced by the server nor supplied by the caller the password will be included in the email.

Example:

```
[
  { "email": "mike.brook@example.com",
    "mobile": "+31645610000"
  },
  { "email": "chuck.norris@badass.com"
  }
]
```

svr-host-name	string	no	hostname extracted from the KeyTalk server's web management certificate otherwise IP address	KeyTalk server hostname to make up a download link to for the generated S/MIME certificate. This link is communicated to the S/MIME recipient hence the hostname should be routable for this person.
<i>[as of v1.1.0]</i> synchronous	boolean	no	true	When set to false, the function does not immediately yield a certificate. Instead KeyTalk will submit a request to a configured CA, which gets responsible for communicating the certificate back to the users via e-mail, normally after asking them for approval. At the moment S/MIME asynchronous orders are only supported by KeyTalk services bound to GlobalSign PersonalSign products.

Example (for the recipients above):

```
$ curl -H "Content-Type: application/x-www-form-urlencoded" \
--cert ./client-cert.pem \
--key ./client-cert-key.pem \
-d \
"recipients=%5B%7B%22email%22%3A%22mike.brook%40example.com%22%2C%22mobile%22%3A%22%2B31645610000%22%7D%2C%7B%22email%22%3A%22chuck.norris%40badass.com%22%7D%5D" \
-X POST https://test.keytalk.com:3000/ssapi/1.1.1/smime-cert-enrollment
```

Response

HTTP 200 - application/json - when synchronous enrolment requested, and it succeeded for at least one recipient.

```
{
  "status": "smime-cert-enrollment",
  "enrolled-recipients": [ emails ],
```

```

"failed-recipients": [
    {
        "email": email,
        "error": error message
    },
    ...
],
"skipped-recipients": [ emails ],

[as of v1.1.1] "apply-address-books": boolean flag indicating whether the caller
should apply the address books, typically to be used by an email client. Applying the address books
should allow the user to send encrypted emails and verify email signatures to other users registered to
the address books,

"address-books": [
    {
        "ldap-svr-url": LDAP server URL,
        "search-base": LDAP server search base DN (e.g.
        "ou=people,dc=example,dc=com",
        "verification-ca": PEM-encoded X.509 verification CA(s) of the
        LDAPs server(optional for LDAPs URL),
    },
    ...
]
}

```

When multiple recipients are supplied, the enrolment might succeed for some of them but fail along the way. If this happens the enrolment process terminates and the remaining recipients get skipped.

HTTP 200 - application/json - when asynchronous enrolment requested and at least one order has been placed successfully placed.

```

{
    "status": "smime-cert-enrollment",

    "placed-orders": [
        {
            "email": email,
            "order_id": order id
        },
        ...
    ],

    "failed-orders": [
        {
            "email": email,
            "error": error message
        },
        ...
    ],

    "skipped-orders": [ emails ],

    [as of v1.1.1] "apply-address-books": see synchronous response,

    "address-books": [
        {
            "ldap-svr-url": LDAP server URL,

```

```

        "search-base": LDAP server search base DN (e.g.
        "ou=people,dc=example,dc=com",
        "verification-ca": PEM-encoded X.509 verification CA(s) of the
        LDAPs server(optional for LDAPs URL),
        },
        ...
    ]
}

```

When multiple recipients are supplied, the order placement might succeed for some of them but fail along the way. If this happens the enrolment process terminates and the remaining recipients get skipped.

HTTP 400 - application/json - invalid request detected before enrolling any recipients (e.g. invalid mobile phone number supplied or SMTP not configured for the given KeyTalk service)

```

{
  "status": "error",
  "error": error message (optional)
}

```

HTTP 500 - application/json - generic server-side enrolment error

```

{
  "status": "error",
  "error": error message (optional)
}

```



6. CERTIFICATE AUTHORITY RETRIEVAL API (CA API)

Certificate Authority API is meant for fetching trusted and intermediate certificate authorities used by KeyTalk and is primary used by KeyTalk agents in order to roll out the initial trust CAs on the system before the RCDP API comes into play.

6.1 CA API versions

REST API version	Minimal KeyTalk server version	Changes wrt the previous API version
1.0.0	5.3.1	n/a
1.0.1	5.8.0	Allow caller requesting a desired certificate by fingerprint Allow caller specifying a desired certificate download format
1.0.2	6.2.1	Allow requesting extra Signing CAs
1.0.3	6.5.12	Allow requesting CA by fingerprint only

6.2 CA API overview

The non-SSL HTTP communication goes over the standard port 80. HTTPS communication goes over the standard port 443.

6.2.1 Fetch internal signing CA

Fetch KeyTalk internal CA certificates.

The received CAs are KeyTalk internal CAs corresponding to “Signing CA”, “Extra Signing CA”, “Communication CA”, “Primary CA” and “Root CA” on the KeyTak admin web panel.

Request

```
GET /ca/1.0.3/signing[?PEM|DER]
GET /ca/1.0.3/primary[?PEM|DER]
GET /ca/1.0.3/communication[?PEM|DER]
GET /ca/1.0.3/root[?PEM|DER]
GET /ca/1.0.3/extrasigning
GET /ca/1.0.3/signing/<cert-sha1-fingerprint>[?PEM|DER]
GET /ca/1.0.3/primary/<cert-sha1-fingerprint>[?PEM|DER]
GET /ca/1.0.3/communication/<cert-sha1-fingerprint>[?PEM|DER]
GET /ca/1.0.3/root/<cert-sha1-fingerprint>[?PEM|DER]
GET /ca/1.0.3/extrasigning/<cert-sha1-fingerprint>[?PEM|DER]
GET /ca/1.0.3/<cert-sha1-fingerprint>[?PEM|DER]
```

Response

HTTP 200 - application/octet-stream - CA certificate is returned in HTTP response body. Note that the /extrasigning request will yield a list of extra signing CAs configured on KeyTalk concatenated in a single PEM file.

HTTP 404 - CA does not exist.