

KeyTalk - API

Author	MR vd Sman
Creation date	14-March-2017
Last updated	24-January-2019
Document version	2.4.1
Document status	Qualified
Product	KeyTalk certificate and key management & enrolment virtual appliance
Data classification	Public

TABLE OF CONTENTS

1. INTRODUCTION	2
1.1 Purpose	2
1.2 Scope	2
2. CERTIFICATE RETRIEVAL API (RCDP V2)	3
2.1 RCDPv2 versions	3
2.2 KeyTalk config file	3
2.3 RCDPv2 overview	4
2.4 RCDPv2 communication phases	5
2.5 Messages sent in all phases	6
2.5.1 End Of communication	6
2.5.2 Error	6
2.6 Phase 1 (handshake)	8
2.6.1 Hello	8
2.6.2 Handshake	8
2.7 Phase 2 (authentication)	10
2.7.1 Request authentication requirements	10
2.7.2 Authentication	12
2.7.3 Change password	18
2.8 Phase 3 (service provision)	19
2.8.1 Check for the last messages	19
2.8.2 Generate certificate on the server	20
2.8.3 [as of v2.2.0] Query CSR requirements	22
2.8.4 [as of v2.2.0] Generate certificate from the client CSR	23
3. PUBLIC API	25
3.1 Public API versions	25
3.2 API overview	25
3.2.1 Retrieve self-service availability	25
3.2.2 Retrieve address book URLs	27
3.2.3 [as of v1.1.0] Retrieve availability of S/MIME certificate enrollment to external parties for self-service	28
4. SELF-SERVICE API	30
4.1 API versions	30
4.2 API overview	30
4.2.1 Enroll S/MIME certificates for external parties	30
5. CERTIFICATE AUTHORITY RETRIEVAL API (CA API)	34
5.1 CA API versions	34
5.2 CA API overview	34
5.2.1 Retrieve internal signing CA	34



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 (RCDP V2)

This section describes certificate retrieval API called RCDP version 2. The motivation to develop a new API over the existing legacy RCDPv1 was as follows:

- Offload handcrafted security to the standard SSL/TLS stack implemented by HTTPS protocol
- Use RESTful way of communication based on simple HTTP GET requests and JSON responses
- Simplify the API to make it easier to develop KeyTalk clients and related services

2.1 RCDPv2 versions

RCDP version	Supported KeyTalk server	Changes wrt the previous RCDP version
2.0.0	4.6.0 and up	
2.1.0	5.3.0 and up	Allow caller to request a certificate download URL in the phase 3 <code>cert</code> request instead of a certificate body.
2.2.0	5.3.1 and up	<ul style="list-style-type: none"> - Allow submitting CSR for signing - Include TPM Virtual Smart Card requirement flag as a part of auth-requirements response
2.3.0	5.3.3 and up	<ul style="list-style-type: none"> - Allow for integrated Kerberos authentication - Return "LOCKED <time>" when the user is still locked iso "DELAY <time>" - Use HTTP POST iso HTTP GET for authentication and password change requests
2.4.0	5.5.0 and up	Add flag instructing the caller to store certificate to the System (Machine) Store instead of the Personal Store

2.2 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 its 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

2.3 RCDPv2 overview

Communication in RCDPv2 is encapsulated in RESTful calls over HTTPS using standard port 443. Optional out-of-band certificate downloads are made possible over HTTP using port 8000.

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

HTTP Header	Required	Description
GET	YES	/rcdp/2.X.Y/<action>?<request-params>
Host	YES	Should contain the FQDN or IP (v4 or v6) of the KeyTalk virtual appliance.
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:

```
GET
/rcdp/2.4.0/authentication?service=DEMO_SERVICE&PASSWD=change%21&HWSIG=12345
6&USERID=DemoUser &ips=%5B%2281.175.103.107%22%5D&caller-hw-
description=Windows+7%2C+BIOS+s%2Fn+1234567890 HTTP/1.1
Host: keytalkdemo.keytalk.com
Cookie: keytalkcookie=a622bb821bec1f5315668c8f9a8e780f
```

A typical set of HTTP response headers:

```
HTTP/1.1 200 OK
Content-type: application/json
Cache-Control: no-cache
Set-Cookie: keytalkcookie=a622bb821bec1f5315668c8f9a8e780f

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

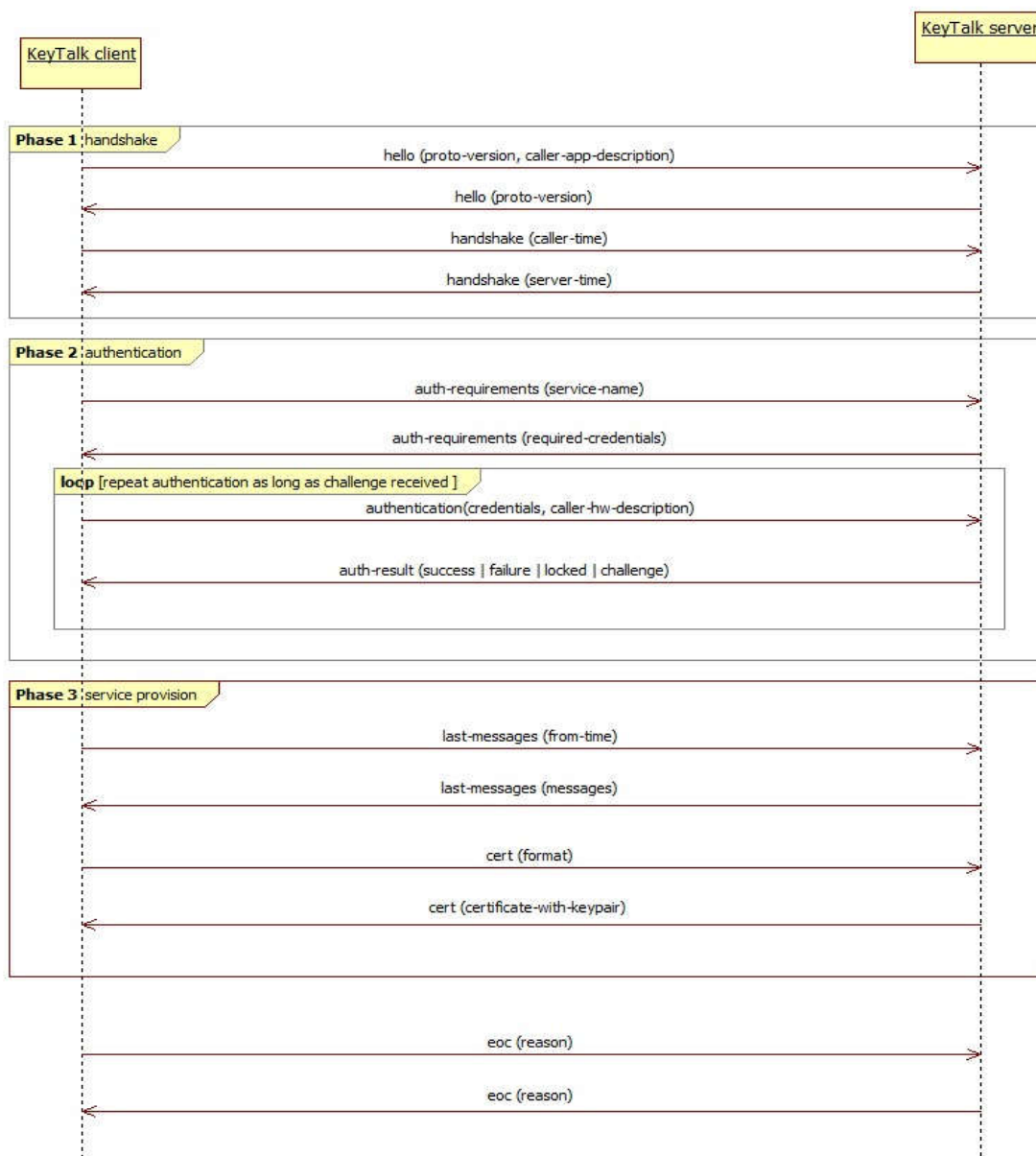
2.4 RCDPv2 communication phases

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.

2.5 Messages sent in all phases

2.5.1 End Of communication

Request

GET /rcdp/<version>/eoc

Example:

/rcdp/2.4.0/eoc
/rcdp/2.4.0/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.

2.5.2 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

GET /rcdp/<version>/error

Example:

/rcdp/2.4.0/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 match 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.

2.6 Phase 1 (handshake)

2.6.1 Hello

Agree on RCDP API version and establish session ID.

Request

GET /rcdp/<version>/hello

Example:

```
/rcdp/2.4.0/hello  
/rcdp/2.4.0/hello?caller-app-description=Demo+KeyTalk+client
```

Query parameters

parameter	type	required	description
caller-app-description	<i>string</i>	no	optional description of the caller application

RCDP API version proposed by a 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.

2.6.2 Handshake

Confirm version handshake and exchange time information.

Request

GET /rcdp/<version>/handshake

Example:

```
/rcdp/2.4.0/handshake?caller-utc=2016-04-22T10%3A44%3A35.746255Z
```

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 supports API version proposed by the server on the previous step, it proceeds with this version in HTTP GET path. Otherwise the caller ends communication.

Response

HTTP 200 - application/json

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

2.7 Phase 2 (authentication)

2.7.1 Request authentication requirements

Request authentication requirements from the server.

Request

GET /rcdp/<version>/auth-requirements

Example:

/rcdp/2.4.0/auth-requirements?service=DEMO_SERVICE

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service 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": service URIs,
  [optional] "resolve-service-uris": if service URIs need to be resolved,
  [optional] "calc-service-uris-digest": if service URIs digest needs to be calculated,
  [as of v2.2.0] [optional] "use-tpm-vsc-authentication": if TPM Virtual Smart
  Card authentication should be used,
  [as of v2.3.0] [optional] "use-kerberos-authentication": if Kerberos
  authentication should be used,
}
```

credential-types

JSON array of credential types required to authenticate against the given service. Supported credential types are: "USERID", "HWSIG", "PASSWD", "PIN" and "RESPONSE".
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-vcs-authentication

Boolean flag ("true" or "false") requesting a caller to make use of PM Virtual Smart Card to generate a certificate signing request (CSR). The CSR will be then sent 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*.

Example:

```
{
  "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"
}
```

2.7.2 Authentication

Authenticate the caller against the selected service using the supplied set of credentials. Multiple authentication rounds might be needed e.g. for RADIUS SecurID or RADIUS EAP AKA/SIM authentication.

Request

[as of v2.3.0] HTTP POST is used for authentication

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=DemoUser&HWSIG=123456&P
ASSWD=change%21&resolved=%5B%7B%22ips%22%3A+%5B%2281.175.103.107%22%5D%2C+%22
uri%22%3A+%22https%3A%2F%2Fdemo.keytalk.com%2F%22%7D%5D" \
-X POST https://test.keytalk.com/rcdp/2.4.0/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.
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. Required if resolve-service-uris was previously set in auth-requirements response. Example: <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>

			<pre> }] </pre>
digests	JSON array	if requested	<p>JSON array of objects containing service URIs accompanied with SHA-256 hexadecimal digest of the underlying file. 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": "01c7198fb614bf8746b46062aa551 dff4506dd553ad96817622c76dafa8dc354" }, { "uri": "file://%Program Files%\vpn\vpn2.exe", "digest": "01c7198fb614bf8746b46062aa551 dff4506dd553ad96817622c76dafa8dc355" }] </pre>
[as of v2.3.0] kerberos-ticket	JSON object	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>

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

[\[as of v2.3.0\]](#) "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)

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.

[\[as of v2.3.0\]](#) when LOCKED is received in `auth-status`, indicates the time in seconds the caller is *still* suspended from repeating its authentication attempt. Before v2.3.0 this was communicated as a part of DELAY `auth-status`.

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:" }],
}
```

Extra challenge is requested (RADIUS EAP-AKA UMTS challenge-response authentication)

```
{
  "status": "auth-result",
  "auth-status": "CHALLENGE",
  "challenges": [{ "name": "UMTS AUTN",
"value": "01010101010101010101010101010101"},
                  { "name": "UMTS RANDOM",
"value": "101112131415161718191a1b1c1d1e1f" }],
  "response-names": [ "RES", "IK", "CK" ]
}
```

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. Example: RADIUS SecurID 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. Example: RADIUS EAP AKA/SIM authentication.

2.7.2.1 Challenge-response authentication

Request

[as of v2.3.0] HTTP POST is used for authentication

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
"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/2.4.0/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:" } ],
}
```

2.7.3 Change password

Change user password. Password change facility has to 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

[as of v2.3.0] HTTP POST is used for changing password
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" \
-H "Expect:" \
-d "old-password=changeme&new-password=changed" \
-X POST https://test.keytalk.com/rcdp/2.4.0/change-password
```

Query parameters

parameter	type	required	description
old-password	string	yes	Current (old) user password.
new-password	string	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 has to 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 amount of seconds.

2.8 Phase 3 (service provision)

2.8.1 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

GET /rcdp/<version>/last-messages

Example:

```
/rcdp/2.4.0/last-messages  
/rcdp/2.4.0/last-messages?from-utc=2018-04-26T06%3A49%3A55.614010Z
```

Query parameters

parameter	type	required	description
from-utc	<i>UTC string in ISO 8601 including date and time</i>	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"}  
  ]  
}
```

2.8.2 Generate certificate on the server

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

Request

GET /rcdp/<version>/cert

Example:

```
/rcdp/2.4.0/cert?format=P12
/rcdp/2.4.0/cert?format=PEM&include-chain=True
/rcdp/2.4.0/cert?format=P12&out-of-band=True
```

Query parameters

parameter	type	required	default value	description
format	"P12" or "PEM"	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
	boolean	no	false	Request the entire certificate chain including subordinate and root CAs.
out-of-band	boolean	no	false	[as of v2.1.0] When set, the server will send back URL to download the certificate instead of the certificate itself.

Response

HTTP 200 - application/json

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

  "cert": certificate in the desired format 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 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.

  \[as of v2.2.0\] "store-to-system": if set, instruct the caller to store the generated
```

```
certificate to the System Store (Machine Store) instead of the Personal Store. Defaults to false.
}
```

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

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwGgAwIBAgIIWurOaAAAAABYwDQYJKoZIhvcNAQELBQAwGygxHzAdBgkq\n\nhkiG9w0B
CQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\n\nnDBNLZlXlUYWxrIElU
IFNlY3VyaXR5MRGwFgYDVQQLDA9GYWN0b3J5IERlZmFlbHx\n\nnIDAeBgNVBAMM0tleVRhbGsgRGVt
byBTaWduaW5nIENBMB4XDTE4MDUwMzA3NTUw\n\nnNFOXDTE4MDUwMzA5NTUwNfowGzAxETAPBgNVBA
MMCERlbW9vc2VyMQswCQYDVQQL\n\nnEwJOTDEWMBQGA1UECAwNTm9vcmtQmFyYmFudDESMBAGA1UE
BwwJRWluZGhvdnVu\n\nnMRQwEgYDVQKDATaW9leCBHcm9leCDEMMAoGA1UECwwDU0VTMR4wHAYJKo
ZIhvcN\n\nnAqkBFg90ZXN0dWlAc2lvdXguZXUwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEK\n\nnAoIBAQDJGKTHSL16vsgxIjXvDOTKLk2q518JaIF9Q9ews88NmpVV9cDbOPR
xwsns\n\nnSd1kNAXEYi05ScmIc5pGpIV8hyyNjtZl7tiolVO0ALkXgk7hG7wO2Rz+bAQzCdvS\n\nnoJjtz06gZPYcQVlfq+ENMt
39ibLqfuAnMLjVpn44fwfQxQFeEsd4do074E1bUXh7\n\nn7KzaoxsiDayIITYZe5Azz90147ffg3pR
Dtq\n\nn6IDYmr7xlBMOq+7QObKBU0pgwNkn\n\nn3JtgkBspxGEXok6S1qNBqJ199NjJdYjiWjHa\n\nn9vSpHSN8RF2s9xrBanLM3S+fnr6\n\nnBx34P6cBoTcc1lZ9Dpr8IYNJWkanAgMBAAGj
fTB7MAkGA1UdEw
QCMAAwHQYDVRO1\n\nnBBYwFAYIKwYBBQUHAWIGCCsGAQUFBwMBMA
sGA1UdDwQEAwID+DAqBglghkgB
hvhC\n\nnAQIEHRYbQ1VTVF9QQVNTV0RfSU5URVJOQX
fVEVTVFVJMBYGA1UdEQQPM2CC25z\n\nnLnNb3V4LmVlMA0GCSqGSIb3DQEB
CwUAA4ICAQCCKF1OTJqL3eg1JgJdbLPz
Do74\n\nnfqZbEBPnKebFe6nQ6calHJrZNG857WGdfVKfX
SorkwGHmdSN1\n\nn0XM+ySipcNOWQf\n\nnm909rxKQigk4n\n\nntvjNCiVXRa125t5pUR1ZSyul1SWQAJYc2nPj
zas15B8SwJOiet\n\nnJV80z1pgLfh2GU7hGNIWVqJLF\n\nnUO\n\nntxZ1lWlsZ64iih49owTsLt9CL06pD6KPN6\n\nnWvmzLNoK\n\nnouEeRnYgkyWXvlahGy5N2bPwlq+7+s
3BOYRo3APL4N6iVEOUfYDE78K\n\nn05g5zdhVbn717CMx1sQpXggyF5X\n\nnztQLkrUB5kLT9D7eCBnLDVdjELz12KJar\n\nnbny9eumkCg+Y9PCZN2513o1zU1DLGaH9\n\nn9KdCf6yEca3D3NvnbfCmrDvx10AN+Ht3L
n4XU2L5Rx2rqwB9tj3rZy8i6BK7\n\nnA+ARfg6Tqki5FQ9k667q2hBRPTr69bLeML5at
nyn\n\nnbeKjnYnzCRcfXDgnJIKZdfKt2PBM7lh508Hn
n6aaRZUfHBKHXjMxwuXNMdq9m\n\nnHk6+H8rbRipV\n\nn4xCzEFYvaqlpYO3lOzLlrw8AohRlUzX7UF
GmLDbpn3G2qeikD1Z\n\nnhySYtxjmjXE0DVnPLX05+MR08Eq3hC6QDys3gBZgP3n
ILvFEZliOax4fqBT3ijJ9\n\nnoxMI+OJsawZMG0u00w==\n\nn-----
END CERTIFICATE-----\n\n-----BEGIN ENCRYPTED PRIVATE KEY-----
\nMIIFDjBAGBqkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIQ9o+wzvbXQQCaggA\n\nnMBQGCCCgG
SIb3DQMHBAAipsAoCJT4gVgSCBMhTb\n\nn8ws1tw9uhH12t9mozccMJQeSAe\n\nnIDxu86RaxgbaMchJ2GnfQjFPou1Ik28eU4Pbi60Epd1G
SBAtRRTK9ZsIOcv+26vN\n\nnjrh4gFsLqa9LC\n\nnRB6T7gQFK6nSj+9332d+jCr4tKBIJvSu6hmTGTOr
aePhb8ic8B\n\nnisiShphmz91N91M311qYKMzhW\n\nnMZg043u2TBjzx1LdsFicIH\n\nnKJ8LXkYQNYM0G663y\n\nnqWpnygyjWvzIL7oL5rZh5pv7ygFTuUTy\n\nn1akDW3inuCBfn3\n\nnZy1374IHeAk4V\n\nnhGQ\n\nnc7FmpF15FTZAYICuKQqSTzUKOd+9oqlg8YrbcPbHrcMH43UTeaJ
zjklc3R5K\n\nnmQk\n\nn6a2ggjPc2z4LoFOYetoPUointBlnRetk7QEHWQdWWW5WfFGRjrbK2t0jZLLV
zXuS\n\nnZ0QYBoHeGzFYH0AeYB01DAcT8OC9PAB4r\n\nnvEFdKyXD85OdYdIp4cAbYm5IBB8bYd\n\nnnnf9JIV8iifiHy38of6FpHI3AwPZqZTTDaR+arLTj
pmpN6d9bRfMNYWUWnJsv0W0o\n\nnd1YuWU\n\nnOE0tdvVQKnU1T9FdhbjyW6nQpR8uwYLi\n\nnBIjpvCUK6ZAe\n\nn+1lik0Z2+\n\nnCxn1bu225MOay2YLS3B
izXUkMcQAO4JE5tEj9vMsEa4VHvt9zcsfpT4vZIGmG2h\n\nnU9UoY2XGhZ4jIEvtqO2ihz7V1ow+k
07eD6H1lMHws9CPZkKh03Z94FK1V\n\nnSf53U6\n\nndnR1sAmuuI5HJroXYyX6N5cLguSnwyyvOWRP
rUjQWPZrfvLzndpro6IFPils7L4\n\nn2fR1DEHwe\n\nnVV0Stf31CV6N88KRyGN+gBWrvkGKJ8EozhEz2
qToqLBUOCLQ+FVO1E\n\nnuYS30hejXc8wYKFupwsolhpJUP2B4zC4EbsmTnn7s
S55Yk+9NCetE\n\nnk0VMf\n\nnPVVN\n\nnWG0kFhq5CCmtkx8fVvq0nnnNuZS4Hy+tb1EeqMvRv
QQ62eRCR94msYG2LCVxRUiB\n\nnNrKQvBM3\n\nnRbxjQFVULr6Wjw9I8dLenjffjou47JLSMShax1DeAG5iBb0GzLZP6Wlh\n\nnOyXIYusR
ePxv40GPZsCBRQD2c6fdk52U3Bgk7asctp1L9Y1qP711bJwnuFtygt+7\n\nnz+7b38PLltxMRyMCoLD78kugFAP2St0iGGdzdUEWoIP\n\nnIZT2SmMo578CPum3RSHt\n\nnu3lCtHfzrMIqo1uTgV+HDswTrLwZt\n\nnVDcaZZUP9a6Vyfd83jQXCKFeBk2udM\n\nnHDo5TC6EvLAv9cXqGRW8VSxkJ1WdyxhIdjNS
CN+CrECX\n\nnPTbmV5MP9gydnqDSJDq\n\nnpCHXZr6dca6vAUGYn5ouQuhrTjsSRsk4M5ZhwgYt9xwCc
fNE+juVeweWEJM1GnxP\n\nnmEW3ffFSE+NNDfyOPWEA5XEGPr3xF7g9Bj51T4Yk0XV
k\n\nnED3hTx0VI8g2IZGrvt40\n\nnyh+\n\nnOxyxB9zUzsleQVDitmqnqti3nXReHwyenO0p9frC5J\n\nno4ibYKkPF91H9\n\nnUK\n\nnh8SCSLpWBil\n\nn8RBQ8kd0Pms5G\n\nnZ2TNS6dnwrXZU+solpl+Kk+T+TTjKkDp8U1xkv\n\nnWCL
AUSbs8g00289SjGjhPge0c4UWRiKLEl1j6jDx0g3yHoJU8bi6pMnJzVeg7IhLF\n\nnxK8=\n\nn-----
END ENCRYPTED PRIVATE KEY-----\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):8000/cert/?cbf498dc683c4e0499fd7e2d27640917"
}
```

2.8.3 *[as of v2.2.0]* Query CSR requirements

Client might want to generate a key pair itself and submit the CSR to KeyTalk server for signing. Before generating a key pair the client should ask the server for the initial parameters for the CSR such as key size, signing algorithm and certificate subject.

Request

GET /rcdp/<version>/csr-requirements

Example:

/rcdp/2.2.2/csr-requirements

Response

HTTP 200 - application/json

```
{
  "status": "csr-requirements",
  "key-size": key size in bits,
  "signing-algo": algorithm to use for CSR signing,
  "subject": dictionary of subject fields to use in CSR
}
```

Example:

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

2.8.4 [as of v2.2.0] 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 `csr-requirements` call described in 2.7.3.

Request

```
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-----
%0AMIIC1jCCAb4CAQAwgZAxCAJBgNVBAYTAk5MMRIwEAYDVQQHDA1FaW5kaG92ZW4x%0ADDAKBgN
VBAsMA1NFUzEUMBIGA1UECgwLU2lvdXggR3JvdXAxFjAUBgNVBAGMDU5v%0Ab3JkLUJhcmJhbnQxE
TAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKoZIhvcNAQkBFg90%0AZXN0dWlAc2lvdXguZXUwggEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAAoIBAQDQ%0AfyCCkM7cbVhpBCSx1Nf%2BFdqa9banKf9sPRW
5VwBFYP5siLdsywnNqrFYcV0w6ss%0Ath21qK9bkjZoyiKpbzvzgQw08NlbBmJfj700018HUn2xL
vp2z6J6q3Z4rAR4d8jx%0ApwcdRlPeJO5b3OtBaURKILaJTjtsUVyCXr%2B6u%2FgiuaD0DGBKsIQ
ccyAWGy%2B1zNer%0AsmUib%2FsnWHEaAPJtvg7T2amaWACKcqIOppR%2BHDJUUNSYyju9xZqCLjx
6Y2%2B2ZXHK%0AMPfcFsP%2F8GCYgZ2%2FAilWtsVzKSaRwmTVJfBsy50gW3YmwIOQYghl52NIDQu
BJeoT%0AmQFxsKXpqcWjpP3KTOS5AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAbUVCaYm%2F%0A
wl0tZaLgtCP2mIVVH%2FgHvTeVFs1436Lz%2FaKT5q1QRee81C2us1z9G7h3PG%2BM6w1N%0AUJau
wqQ2mR2c1VAidROdT52syNPR4jXeR11%2F7a%2FmsZFqaw3%2FLlwVtBJHEfOA6apU%0AJSVWi6%2
F3kUjD0FhYHAufKm2nJ10qGnwC5xpzuvYOQsUFFobLZoyGq5NNEgnSpK8X%0A9A9j5kKGBom9eQOr
Wwx%2F0UlwRqLpt6l76Gt5%2B1Mp5BtTCPK2uboHvJiPu4aJUuHh%0Afx9ZjKox73V%2BleOEmNSY
fesuQPE5AwifkE988NFixGXOHw7uQdWc9SFsYFRFZG2p%0AYb%2Bm9iFyUY8AHw%3D%3D%0A-----
END+CERTIFICATE+REQUEST-----%0A" \
-X POST https://test.keytalk.com/rcdp/2.4.0/cert
```

Request POST parameters:

parameter	type	required	default value	description
csr	<i>string</i>	yes	n/a	Base64 encoded PKCS#10 certificate signing request
include-chain	<i>boolean</i>	no	false	Request the entire certificate chain including subordinate and root CAs.
out-of-band	<i>boolean</i>	no	false	When set, the server will send back URL to download the certificate instead of the certificate itself.

Response

HTTP 200 - application/json

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwwGAgIBAgIIWurNEwAAABUwDQYJKoZIhvcNAQELBQAwwGAgHxAdBgkq\n\nhkiG9w0B
CQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\n\nnDBNLZlXlUYWxrIElUeIF
NlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmFlbHx\n\nnIDAEgNVBAMMF0tleVRhbGsgRGVt
byBTAwduaW5nIENBMB4XDTE4MDUwMzA3NDky\n\nnM1oXDTE4MDUwMzA5NDkyM1owgZAx\n\nCzAJBgNVBA
YTAk5MMRiWEAYDVQQLDA1FaW5k\n\nnaG92ZW4xZDdAKBgNVBAsMA1NFUzEUMBIGA1UECgwLU21vdXgg
R3JvdXAxFjAUBgNV\n\nnBAGMDU5vb3JkLUJhcmJhbnQxETAPBgNVBAMMCERlbW9Vc2VYMR4wHAYJKo
ZIhvcN\n\nnaQkBFg90ZXN0dWlAc21vdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\n\nnao
IBAQDGFyCCKM7cbVhpBCSx1Nf+FDqa9banKf9sPRW5VwBFYP5siLdsywnKqRf\n\nnYcV0w6ssth21
qK9bkjZoyiKpbzvzgQw08N1bBmJfj700018HUn2xLvp2z6J6q3Z4\n\nnrAR4d8jxpwcdr1PeJO5b3O
tBaURKILaJTjtsUVyCxr+6u\n\n/giuaD0DGBKsIQccyAW\n\nnGy+1zNersmUib\n\n/snWHEaAPJtvG7T2a
maWACKcqIOppR+HDJUUNSYYju9xZqCLjx6\n\nnY2+2ZXHKmpFcFsP\n\n/8GCGZ2\n\n/AilWtsVzKSaRWm
TVJfBsy50gW3YmwIOQYghl52NI\n\nnDQuBJeoTmQFxsKXpqcWjpp3KTOS5AgMBAAGj\n\nfTB7MAkGA1Ud
EwQCMAAwHQYDVR01\n\nnBBYwFAYIKwYBBQUHAWIGCCsGAQUFBwMBMAwGA1UdDwQEAwID+DAqBg1ghk
gBhvhC\n\nnaQIEHRYbQ1VTVF9QQVNTV0RfSU5URVJOQUxvFEVTVFVJMBYGA1UdEQQPMA2CC25z\n\nnLn
Npb3V4LmV1MA0GCSqGSIb3DQEBCwUAA4ICAQCCca0C1I9Dw+io7IIQmZ8UKzhq\n\nn8MWcbpthcgFH
PHdxqFYIftWYOzXCN8FVq96oHH2e09anBYopGyHW+a5oMbY8bKbP\n\nnvGD6\n\n/Cs1C8nFFqkQfRTH6
nanDSq18S\n\n/4uc3bMaIQvWzv5mEYpiTktKCSUMfv7FLN\n\nns64I\n\n/UQNg1EhHMu1UyL0NM3xU8QY
mz+k6qnkw2C3M5Y9eprUT9iZxXCm4XGJo7j\n\nnUPBIRBXUCsaPz+UdK0Syq2H1\n\n/IsRet5iPRJIU\n\n/B4FjdUJlD1R68ZAYNnyOeDQI7f\n\nnEJWUeBYC2QwdlXW3FqKdwi928wksRPy4x3Fyz9\n\n/f32chZ
QOihe378HP9PDiTZQ\n\nnFCIWSsrO+WUJToehK2ErgqWCrH0Ydw5ZuIV1vVivGzlgmDhMIQY6uPn
Yasa1kQw\n\nnspY2JyvlZA\n\n/9mhCvfupwB6L4QIA8yJwNoM3MASZgq4fVklkxm\n\n/k1pRMPB2bsGy4u
\nnFLyMoodTAYJfpzH\n\n/gCwWnrYowqW2T67HsPqBBiOonsuaA0h4k\n\n/m88i4ypcv5f48wJ\n\nnzcxaXq
RqWqzxw\n\n/efkYg5m4HdncAPU05NxxJmP17n77188MzvKc0wVbA+22vCBgCi\n\nnMaOYWhnkTuBN90A
oaYAJwelbkLlbTFMZJjsNPvvS5sAk119NihCrXS8ZWtZrFGYz\n\nngPkm+UPWboYdQbKCRg==\n\n---
--END CERTIFICATE-----\n"
```

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

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwwGAgIBAgIIWurNEwAAABUwDQYJKoZIhvcNAQELBQAwwGAgHxAdBgkq\n\nhkiG9w0B
CQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\n\nnDBNLZlXlUYWxrIElUeIF
NlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmFlbHx\n\nnIDAEgNVBAMMF0tleVRhbGsgRGVt
byBTAwduaW5nIENBMB4XDTE4MDUwMzA3NDky\n\nnM1oXDTE4MDUwMzA5NDkyM1owgZAx\n\nCzAJBgNVBA
YTAk5MMRiWEAYDVQQLDA1FaW5k\n\nnaG92ZW4xZDdAKBgNVBAsMA1NFUzEUMBIGA1UECgwLU21vdXgg
R3JvdXAxFjAUBgNV\n\nnBAGMDU5vb3JkLUJhcmJhbnQxETAPBgNVBAMMCERlbW9Vc2VYMR4wHAYJKo
ZIhvcN\n\nnaQkBFg90ZXN0dWlAc21vdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\n\nnao
IBAQDGFyCCKM7cbVhpBCSx1Nf+FDqa9banKf9sPRW5VwBFYP5siLdsywnKqRf\n\nnYcV0w6ssth21
qK9bkjZoyiKpbzvzgQw08N1bBmJfj700018HUn2xLvp2z6J6q3Z4\n\nnrAR4d8jxpwcdr1PeJO5b3O
tBaURKILaJTjtsUVyCxr+6u\n\n/giuaD0DGBKsIQccyAW\n\nnGy+1zNersmUib\n\n/snWHEaAPJtvG7T2a
maWACKcqIOppR+HDJUUNSYYju9xZqCLjx6\n\nnY2+2ZXHKmpFcFsP\n\n/8GCGZ2\n\n/AilWtsVzKSaRWm
TVJfBsy50gW3YmwIOQYghl52NI\n\nnDQuBJeoTmQFxsKXpqcWjpp3KTOS5AgMBAAGj\n\nfTB7MAkGA1Ud
EwQCMAAwHQYDVR01\n\nnBBYwFAYIKwYBBQUHAWIGCCsGAQUFBwMBMAwGA1UdDwQEAwID+DAqBg1ghk
gBhvhC\n\nnaQIEHRYbQ1VTVF9QQVNTV0RfSU5URVJOQUxvFEVTVFVJMBYGA1UdEQQPMA2CC25z\n\nnLn
Npb3V4LmV1MA0GCSqGSIb3DQEBCwUAA4ICAQCCca0C1I9Dw+io7IIQmZ8UKzhq\n\nn8MWcbpthcgFH
PHdxqFYIftWYOzXCN8FVq96oHH2e09anBYopGyHW+a5oMbY8bKbP\n\nnvGD6\n\n/Cs1C8nFFqkQfRTH6
nanDSq18S\n\n/4uc3bMaIQvWzv5mEYpiTktKCSUMfv7FLN\n\nns64I\n\n/UQNg1EhHMu1UyL0NM3xU8QY
mz+k6qnkw2C3M5Y9eprUT9iZxXCm4XGJo7j\n\nnUPBIRBXUCsaPz+UdK0Syq2H1\n\n/IsRet5iPRJIU\n\n/B4FjdUJlD1R68ZAYNnyOeDQI7f\n\nnEJWUeBYC2QwdlXW3FqKdwi928wksRPy4x3Fyz9\n\n/f32chZ
QOihe378HP9PDiTZQ\n\nnFCIWSsrO+WUJToehK2ErgqWCrH0Ydw5ZuIV1vVivGzlgmDhMIQY6uPn
Yasa1kQw\n\nnspY2JyvlZA\n\n/9mhCvfupwB6L4QIA8yJwNoM3MASZgq4fVklkxm\n\n/k1pRMPB2bsGy4u
\nnFLyMoodTAYJfpzH\n\n/gCwWnrYowqW2T67HsPqBBiOonsuaA0h4k\n\n/m88i4ypcv5f48wJ\n\nnzcxaXq
RqWqzxw\n\n/efkYg5m4HdncAPU05NxxJmP17n77188MzvKc0wVbA+22vCBgCi\n\nnMaOYWhnkTuBN90A
oaYAJwelbkLlbTFMZJjsNPvvS5sAk119NihCrXS8ZWtZrFGYz\n\nngPkm+UPWboYdQbKCRg==\n\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):8000/cert/?cbf498dc683c4e0499fd7e2d27640917"
}
```

3. PUBLIC API

There is a set of API to query various information from KeyTalk server without the need to authenticate.

3.1 Public API versions

REST API version	Supported KeyTalk server	Changes wrt the previous API version
1.0.0	5.3.2 and up	n/a
1.1.0	5.5.0 and up	Add enrolment of S/MIME certs for external parties
1.2.0	5.5.1 and up	added synchronous flag to smime-cert-enrollment-availability call

3.2 API overview

The communication goes over HTTPS and uses port 443.

3.2.1 Retrieve self-service availability

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

Request

POST /public/1.0.0/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-----
%0AMIIFDCCAvSgAwIBAgIIWlSC7gAAAYowDQYJKoZIhvcNAQELBQAwYg9w0
BCQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK%0ADBNLZX1UYWxrIElUI
FNlY3VyaXR5MRgwFgYDVQLDA9GYWN0b3J5IERlZmF1bHQx%0AIDAEBgNVBAMMF0tleVRhbGsgRGV
tbyBTAwduaW5nIENBMB4XDTE4MDEwOTA3NTMw%0AMl0XDTI4MDEwNzA4NTMwMlowgZAxETAPBgNVB
AMMCERlbW9Vc2VyMQswCQYDVQQG%0AEwJOTDEWMBQGA1UECAwNTm9vcmtQmFyYmFudDESMBAGA1U
EBwwJRWluZGhvdnVu%0AMRQwEgYDVQQKDatTaW91eCBHcm91cDEMMMAoGA1UECwwDU0VTMR4wHAYJK
oZIhvcN%0AAQkBFg90ZXN0dWlAc21vdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK%0A
AoIBAQDI98ytSED47IFmjT0PUezLlyHULxXgYGF83G27J5%2BLkfn1xcWa2cCVH%0AotFG%2BK
ZNfspd5HAXUFRgJI%2BciypdMf3DS9ZH3PIScGzEzWNG0WnNUagHjUfsYQ6r%0A1I6MBKE86F8sjR
hyF%2FrlUpogsc24ALypBdIhQ6Ham3ni4NFsYEcBk80nKK%2BpATDJ%0APqzK1IIGsd3XhOmxjVUZ
PR70FaEbHwnbOakWfeLibwJAWtttdIL7KctSMsBLg3U2o%0AogYmm5fJqfLwZZEmwNslkuzGcHB6M1
WBYQHyp966klYnItBDFEMYKvaW2ec8tZEA%0AljGAWmIrgAMR9obUWoUuGVwLoOXzAgMBAAGjcDBu
MAKGA1UdEwQCAAwEwYDVR01%0ABAwwCgYIKwYBBQUHAWIwCwYDVR0PBAQDAgP4MCCGCWCSAGG%2
BEIBAgQaFhhDVVNU%0AX0FOT19JTL1RFUK5BTF9URVNUVUkwFgYDVORBA8wDYILbnMuc21vdXguZX
UwDQYJ%0AKoZIhvcNAQELBQADggIBAAXrf8FkL2Bh2rW%2FgJQOPHADELqk25%2F8UmMNbpauneVB
%0A0aJFHcXpsSUDQ4beJo3gyM5JvZRV5nfc8Na3v4ANb01oVJHDHA8%2BGs1l1YXy3fN1v%0AU4YAF
yiJULxELrYyuA5g0rdxXjLSSvSTWGWPKIJEGLYC2xv08kh%2BnwOijVciHFCp%0AGATpCbEO4MPwj
```

QiLpJowx5W5FhcyaRvp%2FjHl0T1IsPWRuD23oG8gcg4uRAF9CKvU%0AYE8%2BV9RrYG%2B6VYeNP
2Va6DGTxVsH2%2Fi3vP7Ida08cnVOcYcHKzOBU%2Bfr60muNpuo%0AhE%2FoGzZ7uj%2F3wja9q49
OGJaNxHjadUjk5k799e%2Fv2isBaSuNmXVobOExzjasIyF%2B%0A2VWGrxSJcfhUXSv3%2B0xnm0o
UigN7uxVl%2BriPcpvlzyNLQqzNxSqrD0iW6dIAPHj%0AFuchgAYYuRXVMHbwxqfUVUWqyBg2Pr5
ESSYB6PTDXzigChLjWJRLuEZxXmi6ztn%0AfgSMOG9XRFFR7yxQKDEuGZKfbAc0gXtCDbH3T%2B
UUESjAc7bUz4OM5Zxfea%2F8F%2FE%0AoBrzb6XROZEWAPxmbQOWRXRu01ddl%2Fi77irHAsH%2FL
Gq6RGv7qsDmic%2B3WQXZtdkX%0AslTcBslGLZkgnbWevNtA8UWHL1JdfJjKmnOv5RdYxRc6kCXE0
WQilCByQxpQXR5L%0A-----END%20CERTIFICATE-----" \

-X POST <https://test.keytalk.com/public/1.0.0/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

GET /public/1.0.0/address-book-list

Example:

/public/1.0.0/address-book-list?service=DEMO_SERVICE

Query parameters

parameter	type	required	description
service	string	yes	KeyTalk service name

Response

HTTP 200 - application/json - successful invocation

```
{
  "status": "address-book-list",
  "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",
  "address-books": ""
}
```

HTTP 400 - application/json - invalid request

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

3.2.3 *[as of v1.1.0]* 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

POST /public/1.1.0/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
<i>[as of v1.2.0]</i> synchronous	boolean	no	true	<p>When set to <code>true</code>, checks whether an immediate enrolment is possible.</p> <p>When set to <code>false</code> 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-----
%0AMIIFDCCAvSgAwIBAgIIWlSC7gAAAYowdQYJKoZIhvcNAQELBQAwYg9w0
BCQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTA5MMRwwGgYDVQQK%0ADBNLZX1UYWxrIElUI
FNlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmF1bHhQx%0AIDAEBgNVBAMMF0tleVRhbGsgRGV
tbyBTAWduaW5nIENBMB4XDTE4MDEwOTA3NTMw%0AMl0oXDTI4MDEwNzA4NTMwM1owGZAxETAPBgNVB
AMMCERlbW9Vc2V5MQswCQYDVQQG%0AEWJOTDEWMBQGA1UECAwNTm9vcmtQmFyYmFudDESMBAGA1U
EBwwJRWluZGhvdnVu%0AMRQwEgYDVQQKDAAtaW9leCBHcm91cDEMMGA1UECwwDU0VTMR4wHAYJK
oZIhvcN%0AAQkBFg90ZXN0dW1Ac2lvdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK%0A
AoIBAQDI98ytSED47IFmjT0PUezLlyHULxHULxHULxHULxHULxHULxHULxHULxHULxHULxHULxHULx
ZNFpsd5HAXUFRgJI%2BciypdMf3DS9ZH3PIScEzWNG0WnNUagHjUfsYQ6r%0A1I6MBKE86F8sjR
hyF%2FrlUpogsc24ALypBdIhQ6Ham3ni4NFsYECBk80nKK%2BpATDJ%0APqzK1IIGsd3XhOmxjVUZ
PR70FaEbHwnbOakWfeLibwJAWttdIL7KctSMsBLG3U2o%0AogYmm5fJqfLwZZEmwNslkuzGcHB6M1
WBYPQHyp966k1YnItBDFEMYKvaW2ec8tZEA%0AljGAWmIrrqAMR9obUWoUuGVwLoOXzAgMBAAGjcDBu
MAKGA1UdEwQCMAAwEwYDVR01%0ABAwwCgYIKwYBBQUHAwIwCwYDVR0PBAQDAgP4MCcGCWCGSAGG%2
BEIBAgQAFhhDVVNU%0AX0FOT19JT1RFUk5BTF9URVNUVUkWFgYDVORBA8wDYILbnMuc2lvdXguZX
UwDQYJ%0AKoZIhvcNAQELBQADggIBAIXrf8FkL2Bh2rW%2FgJQOPHADELqk25%2F8UmMNbpauneVB
%0AOaJFHCXpsSUDQ4beJo3gyM5JvZRV5nfc8Na3v4aNB01oVJHDA8%2BGs11UXy3fn1v%0AU4YAF
yiJULxELrYyuA5g0rdxXjLSSvSTWGWPKIJEG1YC2xv08kh%2BnwOijVciHFCp%0AGATpCbEO4MPwj
QiLpJowx5W5Fhcyarvp%2FjH10T1IsPWRu23oG8gCg4uRAF9CKvU%0AYE8%2BV9RrYg%2B6VYeNP
2Va6DGTXVh2%2Fi3vP7Ida08cnVocYCHKZObU%2Bfr60muNpuo%0AhE%2FoGzZ7uj%2F3wja9q49
OGJaNxHjadUjk5k799e%2Fv2isBaSuNmXVobOExzjasIyF%2B%0A2VWGrxSJcfhUXSv3%2B0xnm0o
UigN7uxV1%2BriPcpvlzyNLQqzNxSgerD0iW6dIAPHj%0AFuchgAYYurXVMHbwxhfqUVUWqyBg2Pr5
ESSYB6PTDXzigChLjWJRLuEZxWxi6ztn%0AfgSMOG9XRFFR7yxQKDEuGZKfbAcdd0gXtCDbh3T%2B
```



```
UUEsjAc7bUz4OM5Zxfea%2F8F%2FE%0AoBrzb6XROZEWAPxmbQOWRXRu01ddl%2Fi77irHAsH%2FL
Gq6RGv7qsDmic%2B3WQXZtdkX%0As1TcBslGLZkgnbWevNtA8UWHL1JdfJjKmnOv5RdYxRc6kCXE0
WQilCByQxpQXR5L%0A-----END%20CERTIFICATE-----" \
-X POST https://test.keytalk.com/public/1.2.0/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)
}
```

4. SELF-SERVICE API

There is a set of 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.

4.1 API versions

REST API version	Supported KeyTalk server	Changes wrt the previous API version
1.0.0	5.5.0 and up	n/a
1.1.0	5.5.1 and up	added synchronous flag to smime-cert-enrollment call

4.2 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.

4.2.1 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/1.1.0/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 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.</p>

Example:

```
[
  { "email": "mike.brook@example.com",
```

				<pre> "mobile": "+31645610000" }, {"email": "chuck.norris@badass.com" }] </pre>
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	<p>When set to <i>false</i>, 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.</p> <p>At the moment S/MIME asynchronous orders are only supported by KeyTalk services bound to GlobalSign PersonalSign products.</p>

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/1.1.0/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 ],
  "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 ],
  "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)  
}
```

5. CERTIFICATE AUTHORITY RETRIEVAL API (CA API)

Besides strongly authenticated TLS-secured RCDP API, KeyTalk server also supports unauthenticated plain-HTTP REST API to retrieve trusted and intermediate signing certificate authorities. CA API is meant to be called by KeyTalk clients in order to roll out the initial trust CAs on the system before RCDP API comes into play. The same effect can be achieved by deploying RCCD files, though parsing RCCD is far more complex task compared to downloading a single file over HTTP.

The calls go over plain HTTP iso HTTPS because at the stage of calling CA API a KeyTalk client is not yet supposed to possess a trusted KeyTalk communication CA to establish secure TLS connection to the server. In any case the retrieved certificates contain only public information hence no secrets are leaked by using plain HTTP.

5.1 CA API versions

REST API version	Supported KeyTalk server	Changes wrt the previous API version
1.0.0	5.3.1 and up	n/a

5.2 CA API overview

The communication goes over HTTP and uses port 8000.

5.2.1 Retrieve internal signing CA

Retrieve KeyTalk Signing CA or KeyTalk Primary CA or KeyTalk root CA for a user certificate that will be eventually received via RCDP call. Each subsequent CA is an issuer of the previous one.

The received CAs are KeyTalk internal CAs (i.e. not from GlobalSign or Microsoft CA tree) corresponding to “Signing CA” “Primary CA” and “Root CA”, on the KeyTak admin web panel. A typical KeyTalk internal CA tree is 2 level deep with self-signed Primary CA and no Root CA.

Request

```
GET /ca/1.0.0/signing
GET /ca/1.0.0/primary
GET /ca/1.0.0/root
```

Response

```
HTTP 200 - application/octet-stream - PEM-encoded CA certificate is returned in
HTTP response body
HTTP 404 - CA does not exist (e.g. for Root CA)
```