

KeyTalk - Protocols

Date 03-05-2018

TABLE OF CONTENTS

	1. INTRODUCTION	2
	1.1 Purpose	2
	1.2 Scope	2
5	1.3 Definitions and abbreviations	2
	1.3.1 Definitions	2
	1.3.2 Abbreviations	2
	2. RCDP V2	3
	2.1 RCDPv2 versions	3
10	2.2 RCDPv2 overview	3
	2.3 RCDPv2 communication phases	5
	2.4 Messages sent in all phases	6
	2.4.1 End Of communication	6
	2.4.2 Error	6
15	2.5 Phase 1 (handshake)	8
	2.5.1 Hello	8
	2.5.2 Handshake	8
	2.6 Phase 2 (authentication)	10
	2.6.1 Request authentication requirements	10
20	2.6.2 Authentication	11
	2.6.3 Change password	16
	2.7 Phase 3 (service provision)	17
	2.7.1 Check for the last messages	17
	2.7.2 Generate certificate on the server	18
25	2.7.3 [as of v2.2.0] Query CSR requirements	20
	2.7.4 [as of v2.2.0] Generate certificate from the client CSR	21
30		

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to describe the protocols used by the KeyTalk system. This document is the leading source for these protocols.

1.2 Scope

This document is intended for TrustAlert and all Sioux KeyTalk team members.

1.3 Definitions and abbreviations

1.3.1 Definitions

1.3.2 Abbreviations

RDD	: RESEPT Dispatcher Daemon
RCDP	: RESEPT Client <-> RESEPT Dispatcher Daemon Protocol
RESEPT	: The historical name of KeyTalk software

2. RCDP V2

This section describes RCDP protocol version 2. The motivation to develop a new protocol over the existing RCDPv1 is as follows:

- Offload handcrafted security handshake to a standard SSL/TLS stack implemented by HTTPS protocol
- Use RESEful way of communication based on simple HTTP GET requests and JSON responses

These changes ought to significantly simplify the protocol, make it easier to test and develop clients without diving into communication security details.

2.1 RCDPv2 versions

RCDP version	Supported KeyTalk server	Changes wrt the previous RCDP version
2.0.0	5.2.0 and up	
2.1.0	5.3.0 and up	Added a possibility for the 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	Added a possibility to submit CSR for signing

2.2 RCDPv2 overview

Communication in RCDPv2 is encapsulated in RESTful calls over HTTPS using standard port 443. Optional out-of-band certificate downloads are possible over HTTP with 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	<code>/rcdp/2.X.Y/<action> ?<request-params></code>
Host	YES	Should contain the FQDN or IP (v4 or v6) of the 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:

```
GET
/rcdp/2.2.0/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
Accept-Encoding: identity
Cookie: keytalkcookie=a622bb821bec1f5315668c8f9a8e780f
```

A typical set of HTTP response headers:

```
HTTP/1.1 200 OK
```

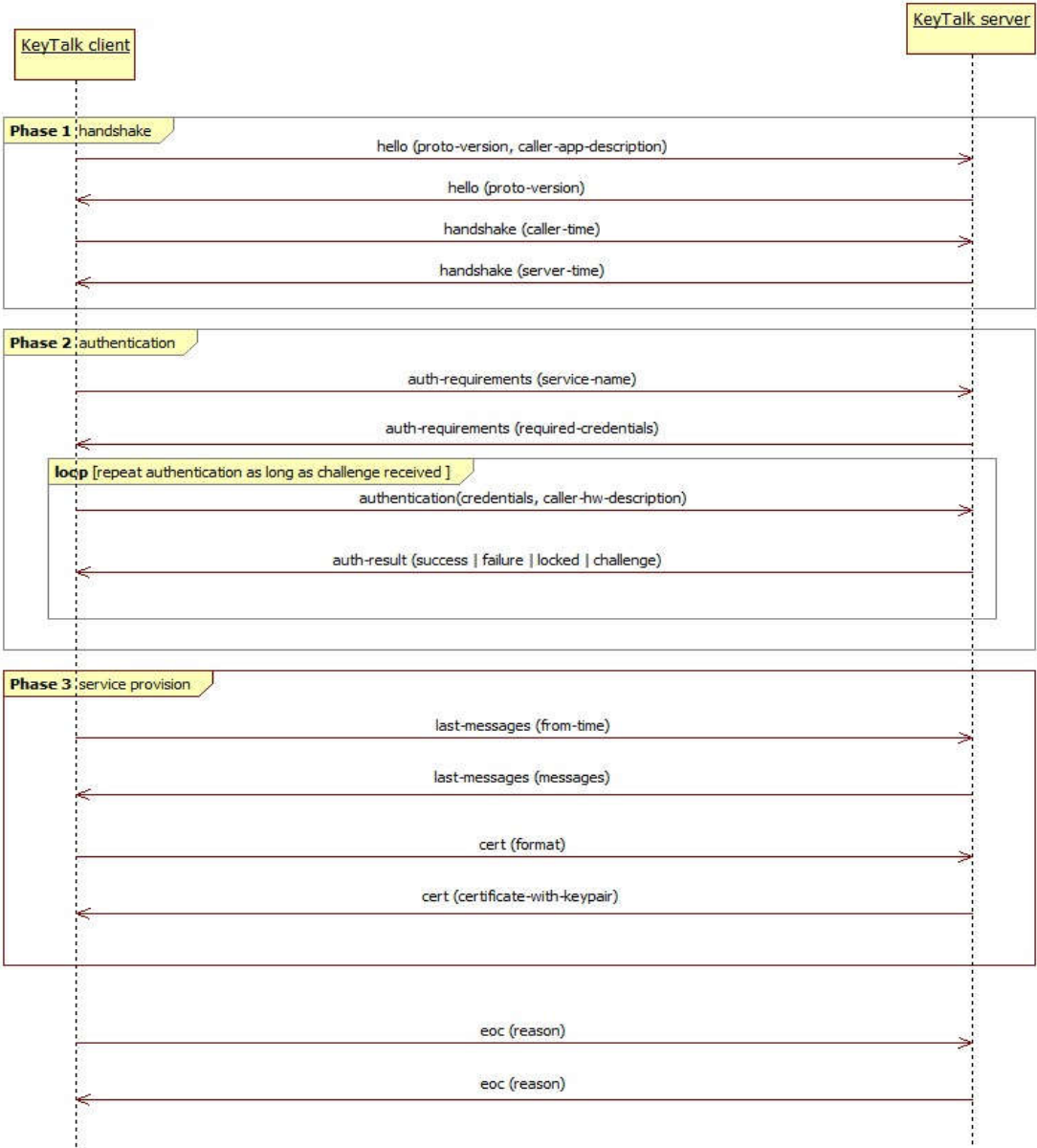
Content-type: application/json
Cache-Control: no-cache
Set-Cookie: keytalkcookie=a622bb821bec1f5315668c8f9a8e780f
{'status': 'auth-result', 'auth-status': 'OK'}

2.3 RCDPv2 communication phases

The complete RCDPv2 communication circle consists of 3 phases:

- Phase1:** handshake
- Phase 2:** authentication
- Phase 3:** service provision

5



Further we describe message semantics on each phase in detail.

2.4 Messages sent in all phases

2.4.1 End Of communication

Request

GET /rdp/<version>/eoc

Example:

/rdp/2.2.0/eoc
/rdp/2.2.0/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.

2.4.2 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

GET /rdp/<version>/error

Example:

/rdp/2.2.0/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.

2.5 Phase 1 (handshake)

2.5.1 Hello

Agree on RCDP protocol version and establish session ID.

Request

GET /rcdp/<version>/hello

Example:

/rcdp/2.2.0/hello
/rcdp/2.1.0/hello?caller-app-description=Demo+KeyTalk+client

Query parameters

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

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

Response

HTTP 200 - application/json

```
{
  "status": "hello",
  "version": proposed protocol 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

GET /rcdp/<version>/handshake

Example:

/rcdp/2.2.0/handshake?caller-utc=2016-04-22T10%3A44%3A35.746255Z

Query parameters

parameter	type	required	description
caller-utc	UTC string in ISO 8601 format including date and time	yes	caller UTC

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

Response

5

HTTP 200 - application/json

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

2.6 Phase 2 (authentication)

2.6.1 Request authentication requirements

Request authentication requirements from the server.

Request

GET /rcdp/<version>/auth-requirements

Example:

/rcdp/2.2.0/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": need to resolve service URIs? ,
  [optional] "calc-service-uris-digest": need to calculate service URIs digest?
}
```

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".

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-uri": ["https://demo.keytalk.com"],
  "resolve-service-uri" : "true"
}
```

2.6.2 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

GET /rdcp/<version>/authentication

Example:

/rdcp/2.2.0/authentication?service=DEMO_SERVICE&caller-hw-description=Windows+7%2C+BIOS+s%2Fn+1234567890&USERID=DemoUser&HWSIG=123456&PASSWD=change%21&resolved=%5B%7B%22ips%22%3A+%5B%2281.175.103.107%22%5D%2C+%22uri%22%3A+%22https%3A%2F%2Fdemo.keytalk.com%2F%22%7D%5D

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 1234567890iPAD: 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	<i>string</i>	if requested	User password. Required if <code>PASSWD</code> was previously set by the server in <code>auth-requirements</code> response.
PIN	<i>string</i>	if requested	User pincode. Required if <code>PIN</code> was previously set by the server in <code>auth-requirements</code> response.
resolved	<i>JSON array</i>	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 <code>resolve-service-uris</code> was previously set in <code>auth-requirements</code> response. Example: [{ "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]"] }]
digests	<i>JSON array</i>	if requested	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. Example: [{ "uri": "file://%Program Files%\\vpn\\vpn.exe", "digest": "01c7198fb614bf8746b46062aa551dff4506dd553ad96817622c76dafa8dc354" }, { "uri": "file://%Program Files%\\vpn\\vpn2.exe", "digest": "01c7198fb614bf8746b46062aa551dff4506dd553ad96817622c76dafa8dc355" }]

Response

5

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.

5

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"]
}
```

- 5 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.
 - 10 - 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.

2.6.2.1 Challenge-response authentication

Request

GET /rcdp/<version>/authentication

Example:

/rcdp/2.2.0/authentication?responses=%7B%22CK%22%3A+%22123%22%2C+%22RES%22%3A+%22456%22%2C+%22IK%22%3A+%22789%22%7D

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

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
}
```

5

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.

10

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:"}],
}
```

5

2.6.3 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

10 GET /rcdp/<version>/change-password

Example:

15 /rcdp/2.2.0/change-password?old-password=changeme&new-password=changed

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

20 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.

2.7 Phase 3 (service provision)

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

GET /rcdp/<version>/last-messages

Example:

/rcdp/2.2.0/last-messages
/rcdp/2.2.0/last-messages?from-utc=2018-04-26T06%3A49%3A55.614010Z

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

GET /rcdp/<version>/cert

Example:

/rcdp/2.2.0/cert?format=P12
/rcdp/2.2.0/cert?format=PEM&include-chain=True
/rcdp/2.2.0/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
include-chain	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.
}
```

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

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwGgAwIBAgIIWurOaAAAABywDQYJKoZIhvcNAQELBQAwYgXhZdBgkq\nhnhkiG9wOB
CQEWEGluZm9Aa2V5dGFSay5jb20xCzAUBgNVBAYTAk5MMRwwGgYDVQOQK\nnDBNL2X1UYWxrIEU1F
NlY3VyaXR5MRgwFgYDVQQQLDA9GYWN0b3J5IERlZmF1bHhQX\nnIDAeBgNVBAMMF0tleVRhbGsgRGVt
byBTAwduaW5nIENBMB4XDTE4MDUwMzA3NTUw\nnNFOxDTDE4MDUwMzA5NTUwNFowZGZAxETAPBgNVBA
MMCERlbW9Vc2VyMQswCQYDVQOQK\nnEwJOTDEWMBQGA1UECAwNtm9vcmtQmFyYmFudDESMBAGA1UE
BwwJRWluZGhvdmVu\nnMRQwEgYDVQQKDAwTaW9leCBHcm9lcDEMMMAoGA1UECwwDU0VTMR4wHAYJKo
ZIhvcN\nnAqKBFG90ZXN0dWlAc2lvdXguZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\n\nAo
IBAQDJGKTHSL16vsgxIjXvDOTKkL2q518JaIF9Q9ews88NmpVV9cDbOPRxswns\nnSdlkNAXEYi05
ScmIc5pGpIV8hyNjtz17t1oLV00ALkXgk7hG70w2z+baQzCdvS\nnMoJttzo6gZPYcQVlfg+ENMT
39ibLqfuAnMLjVpn44fwfgxQFEsd4do074E1bUXh7\nn7KZaoxsIdAYiITYZe5az90L47ffg3pR
Dtq\ /6IDYmr7x1BMOq+7QObKBu0pgwNkn\nn3JTgkBspXGEXok6S1qNBqJ199NjJdyjiWjHa\ /9vS
pHSN8RF2s9xrBanLM3S+fnr6\nnBx34P6cBoTcc11Z9Dpr8IYNJWkanAgMBAAGjFTB7MAKGA1UdEw
QCMAAwHQYDVVR01\nnBBYwFAYIKwYBBQUHAWIGCCsGAQUFbWMBMBAsGA1UdDwQEAwID+DAqBg1ghkgB
hvhC\nnAQIEHRYbQ1VTVF9QQVNTV0RfSU5URVJQOUxfVEVTVFVJMBYGA1UdEQQPMA2CC25z\nnLnNp
b3V4LmV1MA0GCSqGSIb3DQEBQCUAA4ICAQCYKf1OTJqL3eg1JgJdbLpDe74\nnlfqZdEBpNkeBFe6
nQ6calHJRZSG857GSGbVDFKESorkGHmdSN1\ /0XM+yS1pcNOWQf\nnM9o9rxKQigk4n\ /tvjNCiVX
Ra125t5pUR1ZSyu11SWQAJYc2nPjzas15B8SwJOIet\nnJV80z1pgLFh2GU7hGn1WVqJLJ\ /UO\ /t
+xZ11WlsZ64i1h49owTsLt9CL06pD6KPN6\nnWvmzLNOk\ /ouEeRnYgkyWXv1ahGY5N2bPwlq+7+s
3BOYRo3APL4N6iVEOUfYDE78K\nn05g5zdhVbn717CMx1sQpXggyF5X\ /ztQLkrUB5kLT9D7eCBnL
DvdjELz112KJar\ /b\nny9eumcCg+Y9PCZN2513o1zU1DLGAh9\ /9KdCf6yEca3D3NvnbfCmrDvx1
0AN+Ht3L\nn4XU2L5Rx2rqwB9tj3rZy8i6BK7\ /A+ARfg6Tqki5FQ9k667q2hBRPtr69bLeML5at\
nyn\ /beKjnYnzCrcfXDNjIKZdfk2tPBm71h508Hn6aaRZUfHBKHxjMxwuXNMdq9m\nnHk6+H8rb
RipV\ /4xCzEYFvaqlpY0310KZL1rW8AohRLhUzX7UFGM1dbpn3G2ge1kD1Z\nnhySYTxbmjXEODVnPL
X05+MR08Eq3hC6QDYs3gBZgP3nILVfEZLi0ax4fqbT3ijJ9\nnoxMI+OJswZMG0u0w==\n-----
END CERTIFICATE-----\n-----BEGIN ENCRYPTED PRIVATE KEY-----
\nMIIFDjBAGBgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIQi9o+wzvbxQQCaggA\nnMBQGCGcG
S1b3DQMHBAipsAoCJT4gVgSCBMhTb\ /8ws1tw9uhH12t9mozccMQjQeSAe\nnIDxu86RaxgbaMchJ2
GnfQjFPou1Ik28eU4Pbi60Epd1GSBatrRTK9ZsIOcv+26vN\nnjrh4gFsLqa9LC\ /RB6T7gQFK6ns
j+9332d+icJr4tKBUIJvSu6hmTGTORaePhb8ic8B\nn1Shphmz91N91M311qYKMzhW\ /MZg043u2TBJ
zx1LdsFicIH\ /KJ8LXkYqNmY0G663y\nnqWpncyjWzL7oL5Zrh5p7ygFTYbC\ /1akD03inuC8
fn3\ /Zy1374IHeAk4V\ /hQK\nnC7Fmpf1q5FTZAYICuKQQsLTZUKOd+90qlq8YrbcPbHrcMH43UteaJ
zjklc3R5K\ /mQk\nn6a2ggjPc2z4LoFOYEtoPUointBLnRetk7QEHWQdWWW5WfFGRjrbK2t0jZLLV
zXuS\nnZ0QYBoHeGzFYH0AeYB01DacT8OC9PAB4r\ /vEFdKyXD85OdYdIp4cAbYm5IBB8bYd\nnnf9
JIV8iifIHy38of6FpHI3AwPZqZTTDaR+arLTtjpmP6d9bRfMNYWUWnJsv0W0o\nnd1YuWU\ /\ /OE0
tdvVQKnU1T9FdhbjyW6nQpR8uwHyLi\ /BIjpvCUK6ZAE\ /+1lik0Z2+nnCXnlbU225MoA2Y2LS3B
iZUxUkMcQ0A4JE5tEj9vMsEa4VHvt9zcsfpT4vZIGMG2h\nnU9UoY2XGhZ4jIEVtq02ihz7Vlow+k
07eD6H1HMHws9CPZkKh03294FK1V\ /SF536\ndnR1sAmuu15HJroXYyX6N5LgvsNwyvOWRPu
UjgWPZrfvLzndp6IFPils7L4\nn2fRIDEHwe\ /VV0StF31CV6N88KRYGN+gWBrgvKKGJ8EozhEz2
qToqLBU0CLQ+FVO1E\nnuYS30hejXc8wYKFupwS0lhpJUP2B4zC4EbsmTnn7sS55Yk+9NCetE\ /k0
VMf\ /PVVN\nnWG0kFhq5CCmtkx8fVvq0nnnNuZS4Hy+tbLEeqMvRvQQ62eRCR94msYG2LCVxRUiB\
nnRkQvBM3\ /RbxjQFVULr6Wjw9I8dLenjffjou47JLSMShax1DeAG5iBb0GzLZP6Wlh\nnOyXIYusR
ePxv40GPZsCBRQd2C6fdk52U3Bgk7asctplL9Y1qP711bJwnuFtygt+7\nnZ+7b38PLltxMRyMCoL
D78kugFAP2St0iGgdzUEWoiP\ /IZT5mMo578CPum3RSht\nnu31CthfzrMIq2o1uTGv+HdSwTr
LwZt\ /VDcaZUP9a6Vyfdz83jqRXCKfEbK2udm\nnHDo5TC6EvLAv9cXqGRW8VSxkJ1WdyxhIdjNS
CN+CrECX\ /PTbmV5MP9gydnqDSJDq\nnpCHXZr6dca6vAUGYN5ouQuhrTjsSRsk4M5ZhwgYt9xwCc
fNE+juVeweWEJm1GnxP\nnmEW3fFSE+NNDfYoPWEA5XEGPr3xF7g9Bj51T4Yk0XVX\ /ED3hTx0VI8
g2IZGrvt40\nnyh+ \ /OxyxB9zUzsleQVDitmzQnqti3NReHwyenO0p9frC5J\ /o4ibYKkPF91H9\
/UK\nnh8SCSLpWBi1\ /8RBQ8kD0Pms5G\ /Z2TNS6dnwRXZU+solpl+Kk+T+TTjKKDp8U1xkv\nnWCL
AUsbs8gO0289SjGjhPge0c4UWRiKLElj6jDx0g3yHoJU8bi6pMnJzVeg7IhLF\nnxK8=\n-----
END ENCRYPTED PRIVATE KEY-----\n"
```

Example when certificate download URL is returned:

```
{
  "status": "cert",
  "cert-url-templ": "
http://$(KEYTALK_SVR_HOST):8000/cert/?cbf498dc683c4e0499fd7e2d27640917"
}
```

5

2.7.3 [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.

10

Request

GET /rcdp/<version>/csr-requirements

Example:

/rcdp/2.2.2/csr-requirements

15

Response

HTTP 200 - application/json

20

```
{
  "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": "Amersfoort",
    "o": "KeyTalk",
    "ou": "Development",
    "e": "test@keytalk.com",
  }
}
```

25

2.7.4 [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 desribed 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-----
%0AMIIC1jCCAb4CAQAwgZAx CzA JBgNVBAYTAk5MMRIwEAYDVQQHDA1FaW5kaG92ZW4x%0ADDAKBgN
VBAsMA1NFUzEUMBIGA1UECgwLU21vdXggR3JvdXAxFjAUBgNVBAGMDU5v%0Ab3JkLUJhcmJhbnQxE
TAPBgNVBAMMCERlbW9Vc2VYMR4wHAYJKoZIhvcNAQkBFg90%0AZXN0dWlAc21vdXguZXUwgGEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDG%0AfyCCKM7cbVhpBCSx1Nf%2BFDqa9banKf9sPRW
5VwBFYP5sildsywNkNqrFYcV0w6ss%0Ath21qK9bkjZoyiKpbzvzQw08NlbBmJfj700O18HUn2xL
vp2z6J6q3Z4rAR4d8jx%0ApwcdRlPeJO5b3OtBaURKILaJTjtsUVyCXr%2B6u%2FgiuaD0DGBKsIQ
ccyAWGy%2B1zNer%0AsmUib%2FsnWHEaAPJtvg7T2amaWACKcqIOppR%2BHDJUUNSYyju9xZqCLjx
6Y2%2B2ZXHK%0AMpFcFsP%2F8GCYGG2%2FAilWtsVzKSaRWmTVJfBsy50gW3YmwI0QYghl52NIDQu
BJeoT%0AmQFxsKXpqcWjpP3KTOS5AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAbUVCaYm%2F%0A
wlotZaLgtCP2mIVVH%2FgHvTeVFs1436Lz%2FaKT5q1QRee81C2uslz9G7h3PG%2BM6w1N%0AUJau
wqQ2mR2c1VAidROdT52syNPR4jXeRl1%2F7a%2FmsZFqaw3%2FLlwVtBJHEfOA6apU%0AjSVWi6%2
F3kUjD0FhYHAufKm2nJ10qGnwC5xpzuvYOQsUFFobLZoyGq5NNEgnSpK8X%0A9A9j5kKGBom9eQOr
Wxw%2F0UlwRqLpt6l76Gt5%2B1Mp5BtTCPK2uboHvJiPu4aJUuHh%0Afx9ZjKox73V%2BleOEmNSY
fesuQPE5AwifkE988NFixGXOHw7uQdWc9SFsYFRFZG2p%0AYb%2Bm9iFyUY8AHw%3D%3D%0A-----
END+CERTIFICATE+REQUEST-----%0A" -X POST
https://test.keytalk.com/rcdp/2.2.0/cert
```

Request POST parameters:

parameter	type	required	default value	description
csr	string	yes	n/a	Base64 encoded PKCS#10 certificate signing request
include-chain	boolean	no	false	Request the entire certificate chain including subordinate and root CAs.
out-of-band	boolean	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": PEM-encoded certificate 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.
}

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

```
{
  "status": "cert",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIFGTCCAwwGawIBAgIIWurNEwAAABUwDQYJKoZIhvcNAQELBQAwGygxHZAAdBgkq\n\nhkiG9w0B\nCQEWEGluZm9Aa2V5dGFsay5jb20xCzAJBgNVBAYTAk5MMRwwGgYDVQQK\n\nQnDBNLZX1UYWxrIEl1U\nNlY3VyaXR5MRgwFgYDVQQLDA9GYWN0b3J5IERlZmF1bHJx\n\nIDAEBgNVBAMMF0tleVRhbGsgRGVt\nbyBTAwduaW5nIENBMB4XDTE4MDUwMzA3NDky\n\nnM1oXDTE4MDUwMzA5NDkyM1owGZA\n\nxZAJBgNVBA\nYTAk5MMRiwEAYDVQQHDA1FaW5k\n\nnaG92ZW4xDDAKBgNVBAsMA1NFUzEUMBI\n\nGA1UECgwLU21vdXgg\nR3JvdXAxFjAUBGNV\n\nnBAGMDU5vb3JkLUJhcmJhbnQx\n\nETAPBgNVBAMMCERlbW9Vc2VyMR4wHAYJKo\n\nZIhvcN\n\nnAQkBFg90ZXN0dWlAc2lvdXguZ\n\nXUwggEiMA0GCSqGSIb3DQEB\n\nAQUAA4IBDwAwggEK\n\nnAoIBAQDGfyCCKM7cbVhpBCSx1Nf\n\n+FDqa9banKf9sPRW5VwBFYP5siLdsy\n\nwNkNqrF\n\nnYcV0w6ssth21\nqK9bkjZoyiKpbzvzgQw08N1bBmJfj\n\n700018HUn2xLvp2z6J6q3Z4\n\nnrAR4d8jxpwc\n\nRlPeJO5b30\n\nBaURKILaJTjtsUVyC\n\nXr+6u\n\n/giuaD0DGBKsIQccyAW\n\nnGy+1zNersmUib\n\n/snWHEaAPJtv\n\nG7T2amaWACKcq\n\nIoppR+HDJUUNSYYju9xZqCLjx6\n\nnY2+2ZXHKMpFcFsP\n\n/8GCYgz2\n\n/AilWtsVzKSaRWm\n\nTVJfBsy50gW3Ymw\n\nIOQYghl52NI\n\nnDQuBJeoTmQFxsKXpqcWj\n\npP3KTOS5AgMBAAGj\n\nfTB7MAkGA1Ud\n\nEwQCMAAwHQYDVR01\n\nnBBYwFAYIKwYBBQUH\n\nAwIGCCsGAQUFBwMB\n\nMAsGA1UdDwQEAwID\n\n+DAqBg1ghkgBh\n\nvhC\n\nnAQIEHRYbQ1VT\n\nVF9QQVNTV0RfSU5URV\n\nJQOUxfVEVTVFVJMBYGA1Ud\n\nEQQPMA2CC25z\n\nnLnNpb3V4LmV1MA0GCSqGSIb3DQEB\n\nCwUAA4ICAQCCca0C\n\n1I9Dw+io7IIqMZ8UKzhq\n\nn8MWcbpthcgFHPdxqFYI\n\nfTWYOzXCN8FVq96oHH2e09anBYopGyHW\n\n+a5oMbY8bKbP\n\nnvGD6\n\n/Cs1C8nFFqkQfRTH6\n\nnanDSq18S\n\n/4uc3bMaIQvWzv5mEYpiTKtKCSUMf\n\nv7FLN\n\nnS64I\n\n/UQNg1EhHMul1UyL0NM3xU8QY\n\nmz+k6qnkw2C3M5Y9eprUT9iZ\n\nxXCm4XGJo7j\n\nnUPBIRBXUCsaPz+UdK0Syq2H1\n\n/IsREt5iPRJIU\n\n/B4FjduJlD1R68Z\n\nAyNnyOeDQI7f\n\nnEJWUeBYC2QwdlXW3FqKd\n\nwki928wksRpY4x3Fyz9\n\n/f32chZQoihee378HP9PDiT\n\nZQ\n\nnFCIWSsrO+WUUjToehK2Ergq\n\nwCrH0Ydw5ZuIV1vVivGzlgmD\n\nHmIQY6uPnYasa1kQw\n\nnspY2JyvlZA\n\n/9mhCvfupwB6L4QIA8y\n\njwNoM3MasZgq4fvk1kxm\n\n/k1pRMPB2bSGy4u\n\nnFLyMoodTAYJf\n\npzh\n\n/gCwWnrYowqw2T67HsPqBBi\n\nOnsuaA0h4k\n\n/m88i4ypcv5f48wJ\n\nnzcxaXqRqWqxzw\n\n/efkYg5m4HdncAPU05N\n\nxwJmP17n77188MzvKc0wVbA\n\n+22vCBgCi\n\nnMaOYWhnkTuBN90AoaYAJwelbkL\n\nlbTFMZJjsNPvvS5sAk119NihCrXS8Z\n\nWtZRfGyz\n\nngPkm+UPWboYdQbKCRg==\n\nn-----END CERTIFICATE-----\n"
```

5

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"
}
```

10