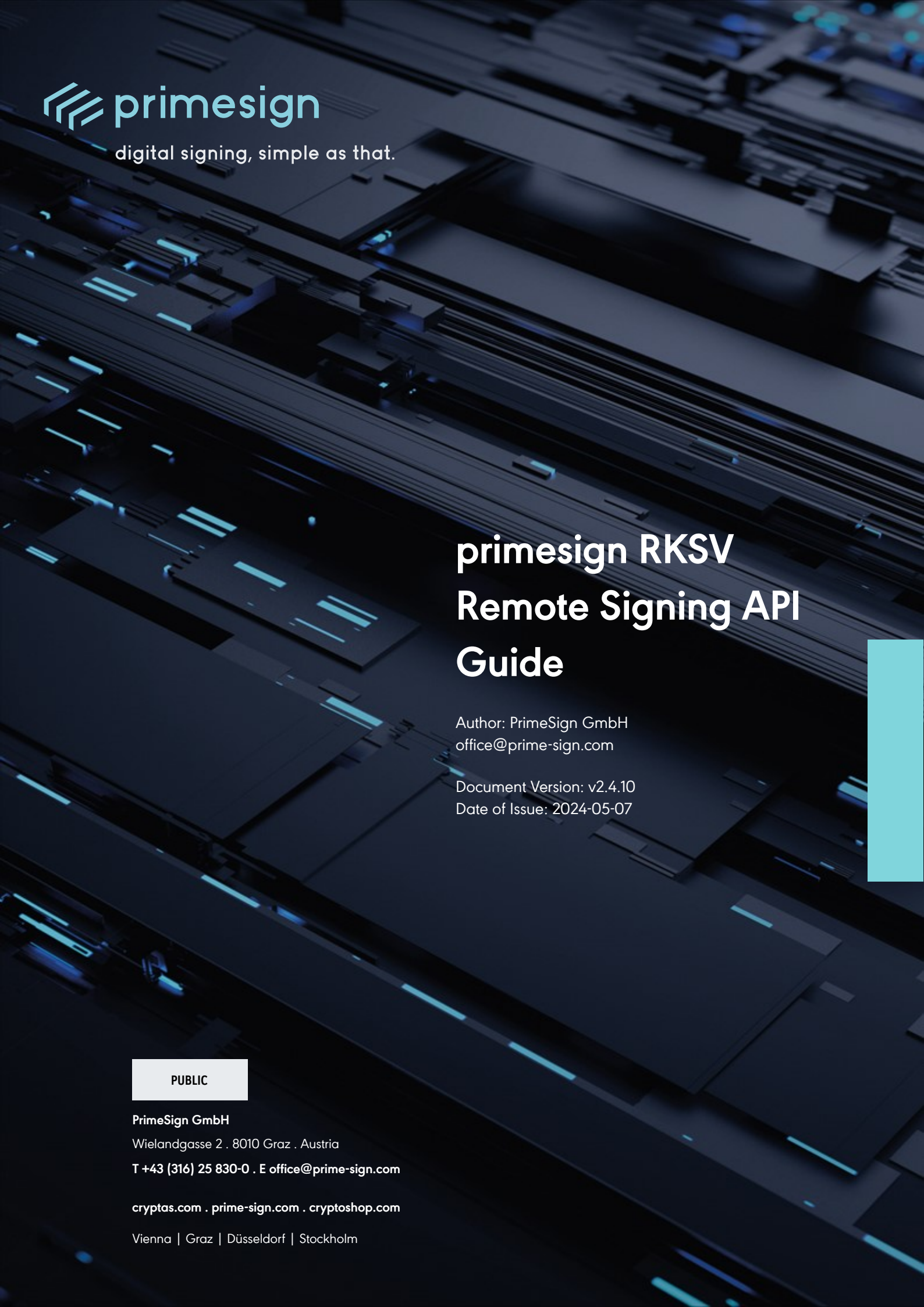

primesign

digital signing, simple as that.

primesign RKSV Remote Signing API Guide

Author: PrimeSign GmbH
office@prime-sign.com

Document Version: v2.4.10
Date of Issue: 2024-05-07

PUBLIC

PrimeSign GmbH
Wielandgasse 2 . 8010 Graz . Austria
T +43 (316) 25 830-0 . E office@prime-sign.com

cryptas.com . prime-sign.com . cryptoshop.com

Vienna | Graz | Düsseldorf | Stockholm

TABLE OF CONTENTS

1. Overview

primesign RKSV Remote Signing service is used to sign cash box receipts according to RKSV. See specifications in [RKSV-II] and [RKSV-Detail].

primesign RKSV Remote Signing allows to create signatures using dedicated keys residing within a hardware security module. Each signature key is associated with a certificate, that is being issued by primesign.

Furthermore, administrators are allowed to manage users, signature keys and request certificates for cash boxes. This document provides an in-depth documentation of the REST API for cash box signing provided by the primesign RKSV Remote Signing service.

1.1. System Overview

The primesign RKSV Remote Signing service is responsible for creating and managing signing keys and the associated certificates. Furthermore, primesign RKSV Remote Signing manages users and their credentials and provides authorization checks on whether a user is allowed to create signatures with the specified key. The primesign RKSV Remote Signing service itself does not issue certificates, but forwards certificate requests to the issuing Certificate Authority (CA) via the Registration Authority Gate (RA-Gate). The so-called Portal denotes an integrator-specific system that communicates with the RA-Gate and primesign RKSV Remote Signing service. Figure 1 outlines the involved components.

The RA-Gate holds the identity records of entities a primesign certificate has been issued to. The CA issues certificates based on certificate requests that have been authorized by a Registration Officer (RO) beforehand.

To get a primesign RKSV Remote Signing certificate a certificate request has to be sent to the primesign RKSV Remote Signing service. The certificate request is forwarded to the RA-Gate. Upon successful verification, whether the request has been authorized by an RO, the RA-Gate initiates certificate issuance at the CA. If an authorized RO has requested the certificate the request is forwarded immediately from the RA-Gate to the CA. If the request has been placed by a user without RO rights, the request is stored at the RA-Gate and only forwarded after an RO authorizes the request.

This documentation focuses on the REST API provided by primesign RKSV Remote Signing service (Figure 1, marked red and yellow). This document illustrates how certificates can be requested and users and signature keys can be managed via the Admin REST API (marked red) and documents the methods available to users via the User REST API (marked yellow).

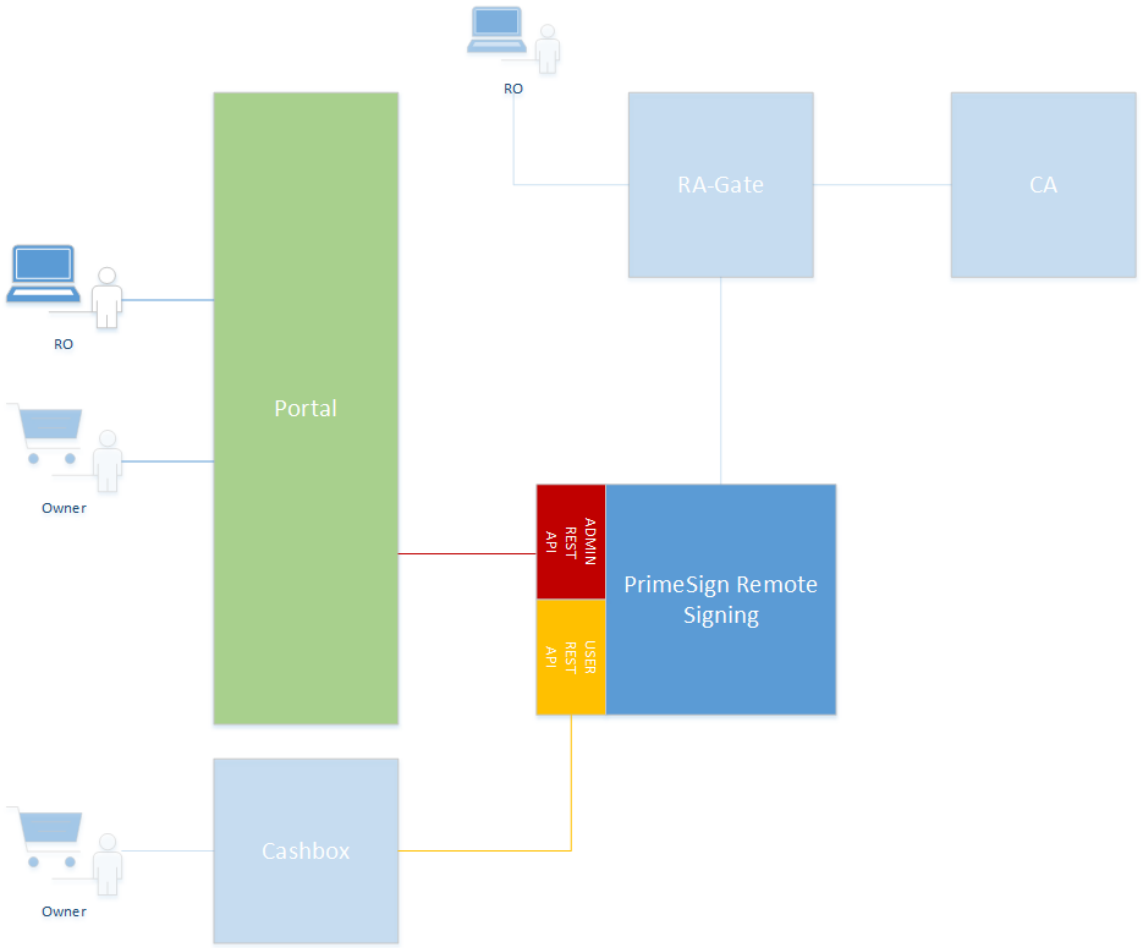

Figure 1: primesign RKSV Remote Signing System Overview

The following section shortly outlines the main entities that are managed within primesign RKSV Remote Signing service. Figure 2 provides an overview of the involved entities and their relations.

The primesign RKSV Remote Signing service manages users, signature keys and the associated certificates. Each user is identified by its unique userId. Users with the role ADMIN are allowed to manage other users, create signature keys and place certificate requests. Each user has both a password and a shared secret that can be used for authentication (see Admin REST API Authentication and User REST API Authentication).

A user can have one or more signature keys. A signature key is identified by its unique keyId. Each user can be associated with a defaultKey (the first key created for a specific user will automatically be regarded as default key if not otherwise set), which is automatically used for signature creation in cases where the keyId was not set explicitly. Furthermore, a signature key is

associated to the issued certificate. A certificate can only be associated to exactly one signature key.

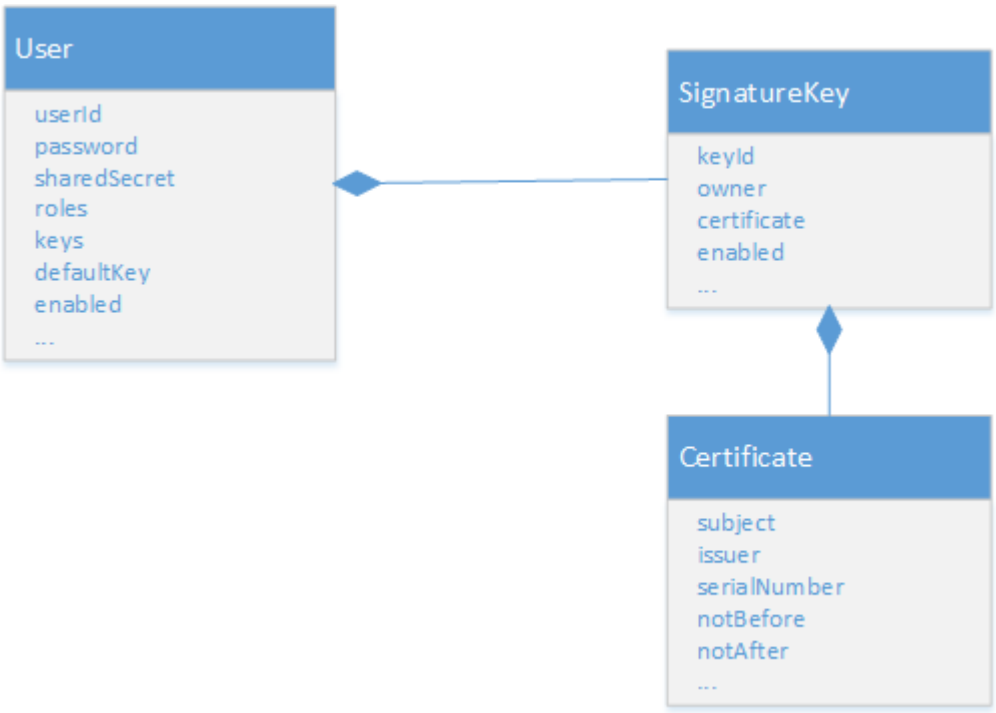

Figure 2: primesign RKSV Remote Signing Entities

In order to request a certificate at primesign RKSV Remote Signing service that can be used for RKSV Remote Signing, a user and a signature key have to be created and a certificate issuing request has to be placed. For convenience, these three steps have been combined into one single REST API method (see All-In-One Certificate Request). However, in addition primesign RKSV Remote Signing service provides a REST API method that allows to request a certificate for an existing signature key (see Request Certificate).

1.2. API Overview

The management REST API is split into methods accessible only to administrators (e.g. managing other users, signature keys and requesting certificates) and methods accessible to ordinary users (e.g. signature creation, certificate and configuration download).

Methods accessible only to administrators include the string **admin** within the request path, i.e. https://<host>/rs/**admin**/users to retrieve a list of all users.

1.3. Management GUI

Out-of-the-box primesign RKSV Remote Signing service offers a web client to manage users, signature keys and certificates.

The primesign RKSV Remote Signing Administration client can be accessed via https://<host>/rs/admin/ui/

2. API Documentation

2.1. Entities

This section describes the JSON data structures used within primesign RKSV Remote Signing service.

2.1.1. User

Entity representing a user.

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`defaultKey`	`STRING`	`true`	The user's default key.

Path	Type	Optional	Description
`roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Entitity used for creating a new user.

Path	Type	Optional	Description
`userId`	`STRING`	`true`	The user's unique id.
`password`	`STRING`	`true`	The user's password.
`roles`	`ARRAY`	`true`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.
`enabled`	`BOOLEAN`	`true`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.

2.1.2. Signature Key

Entity representing a signature key.

Path	Type	Optional	Description
`keyId`	`STRING`	`false`	The signature key's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.

Path	Type	Optional	Description
`creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).
`keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The certificate associated to this signature key.

Path	Type	Optional	Description
`owner`	`STRING`	`false`	The userId of the owner of this signature key.

Entity used to create a new signature key.

Path	Type	Optional	Description
`ownerId`	`STRING`	`false`	The user id of the owner of the newly created signature key.

2.1.3. Certificate

Entity representing the X509 Certificate. This entity contains a subset of fields parsed from the certificate file.

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).

2.1.4. Certificate Request

Entity representing a request to issue a new certificate.

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate should be issued for. When included within an attribute value (e.g. CN), the following characters have to be escaped by prefixing a backslash '\': '"', '#', '+', ',', ';', '<', '=', '>', and "(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.

Path	Type	Optional	Description
`templateId`	`STRING`	`false`	Id referencing the template that should be used for certificate issuance.
`regInfo`	`OBJECT`	`false`	The regInfo map that should be passed to the RA-Gate. The content of regInfo is customer-specific and defined by primesign. Typically it includes the fields `accountingId`, `contactInfo` and optionally `productId`, further fields are possible. For `contactInfo` the following formatting restrictions apply: The following characters have to be escaped by prefixing a backslash '\\': '\"', '#', '+', ',', ';', '<', '=', '>', and '\'(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.

For RKSV usage the `subjectDN` has to contain:

- VAT/UID number: Prefix of "UID ATU" followed by 8 digits, e.g. "CN=UID

ATU12345678,O=Company name,C=AT"

- Global location number: Prefix of "GLN " followed by 13 digits, e.g. "CN=GLN 1234567890123,O=Company name,C=AT"
- Austrian tax number: Prefix of "Steuernummer " followed by 9 digits, e.g. "CN=Steuernummer 123456789,O=Company name,C=AT"

primesign provides a documentation for each customer, which fields have to be included within `regInfo`. Typically `regInfo` includes the fields `accountingId`, `contactInfo` of the certificate subject and optionally `productId`, further fields are possible.

Entity used within the all-in-one certificate request call that creates a user, signature key and requests the certificate in one single REST call.

Path	Type	Optional	Description
`user.userId`	`STRING`	`true`	The user's unique id.
`user.password`	`STRING`	`true`	The user's password.
`user.roles`	`ARRAY`	`true`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.
`user.enabled`	`BOOLEAN`	`true`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`user`	`OBJECT`	`true`	The user that should be created.

Path	Type	Optional	Description
`certificateRequest.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate should be issued for. When included within an attribute value (e.g. CN), the following characters have to be escaped by prefixing a backslash '\': '"', '#', '+', ',', ';', '<', '=', '>', and "(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.
`certificateRequest.templateId`	`STRING`	`false`	Id referencing the template that should be used for certificate issuance.

Path	Type	Optional	Description
`certificateRequest.regInfo`	`OBJECT`	`false`	The regInfo map that should be passed to the RA-Gate. The content of regInfo is customer-specific and defined by primesign. Typically it includes the fields `accountingId`, `contactInfo` and optionally `productId`, further fields are possible. For `contactInfo` the following formatting restrictions apply: The following characters have to be escaped by prefixing a backslash '\\': '\"', '#', '+', ',', ';', '<', '=', '>', and '\'(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.
`certificateRequest`	`OBJECT`	`false`	The certificate request.

primesign provides a documentation for each customer, which fields have to be included within `regInfo`. Typically `regInfo` includes the fields `accountingId`, `contactInfo` of the certificate subject and optionally `productId`, further fields are possible.

Response entity returned by the all-in-one certificate request.

Path	Type	Optional	Description
`user.userId`	`STRING`	`false`	The user's unique id.
`user.enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`user.password`	`STRING`	`false`	The user's password.
`user.sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.
`user`	`OBJECT`	`false`	The newly created user.
`key.keyId`	`STRING`	`false`	The signature key's unique id.
`key`	`OBJECT`	`false`	The newly created signature key.
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.

Path	Type	Optional	Description
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The issued certificate.

2.1.5. Cashbox Configuration

Entity representing a Cashbox Configuration.

Path	Type	Optional	Description
`tspId`	`STRING`	`false`	Id of the trust service provider (Field: VDA). PrimeSign has the value "AT3".
`user.userId`	`STRING`	`false`	The user's unique id.
`user.enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.

Path	Type	Optional	Description
`user.signatureKeys[].keyId`	`STRING`	`false`	The signature key's unique id.
`user.signatureKeys[].keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`user.signatureKeys[].certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`user.signatureKeys[].certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`user.signatureKeys[].certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`user.signatureKeys[].certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`user.signatureKeys[].certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`user.signatureKeys[].certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`user.signatureKeys[].certificate`	`OBJECT`	`true`	The certificate associated to this signature key.

Path	Type	Optional	Description
`user.signatureKeys`	`ARRAY`	`false`	The signature keys this user is allowed to access.
`user.defaultKey`	`STRING`	`true`	The user's default key.
`user`	`OBJECT`	`false`	The user the cashbox configuration has been requested for.

2.2. Transaction Id

Most REST API requests are implicitely associated with an underlying transaction. The identifier of the respective transaction is returned in each HTTP response with a certain HTTP header `X-Transaction-ID`. The identifier value has no impact on the user or on the client software but may help us to track issues reported by users.

Clients are highly encouraged to integrate primesign transaction ids within their own logging.

Example HTTP Response (response on signature creation request):

```
HTTP/1.1 200 OK
X-Transaction-ID: a6d54088ilqid
Content-Type: text/plain;charset=UTF-8
Content-Length: 298
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

eyJhbGciOiJFUzI1NiJ9.X1IxLUFUMzAwX0NBU0hCT1gtREVNTy0xX0NBU0hCT1gtREVNTy0xLVJl
Y2VpcHQtSUQtMTE4XzIwMTYtMDktMDVUMTA6MTc6NThfMCwwMF8wLDAwXzAsMDBfMCwwMF8wLDAwX
3RBYktPQjBaVFhBPV8zOWQ1MzYzYTE2ZWFmYTBhXzRwamhUcUtmK2tnPQ.t0DWYCl4jtMT7736Pqs
ZH9WD-e3-79AHxko5OCi6vpcsgEB_0KTNBQh-HMwvoHsPmnUXOvRVYK-Gkat7U6T16Q
```

In case a specific REST API request results in an error the respective error response contains an explicit (but optional) field `transactionId` (see Error Messages) - in parallel to the above mentioned implicit HTTP response header (both values are identical).

There may be some API functions that are not associated with transactions (e.g. health check or sign check). Therefore the header is regarded **optional**.

2.3. Admin REST API

2.3.1. Authentication

The Admin REST API supports two authentication types:

- **Basic Authentication:** When using the Basic Authentication Scheme [RFC2617] for authentication **userId** and **password** are included within the HTTP Authorization request header. This authentication type is stateful. After login the credentials can be omitted for subsequent requests. Furthermore, handling of CSRF-Tokens is required.
- **API Token:** The header field X-AUTH-TOKEN has to contain the shared secret of the user. The shared secret is regarded as API Token. If this authentication type is used, the shared secret has to be included in every request as no session is established.

Basic Authentication (Stateful with Session)

With session-based authentication the caller has to include the user credentials within the HTTP Authorization Header using the Basic Authentication Scheme [RFC2617] (see Login method for an example). In case the provided credentials are valid, the primesign RKSV Remote Signing service returns a **Session Cookie** named **SESSION** to the caller. In subsequent requests the issued cookie has to be returned to primesign RKSV Remote Signing service. Web browsers automatically include the cookie within the subsequent request.

In case the provided user credentials are invalid, no Session Cookie has been provided or the session has expired, primesign RKSV Remote Signing service returns a HTTP status code **401 Unauthorized**.

For convenience, a dedicated Login method has been provided, however, each method of the Admin REST API is able to process the provided user credentials from the HTTP Authorization Header and return a Session Cookie to the caller. To invalidate the user session, primesign RKSV Remote Signing service provides a Logout method.

CSRF

primesign RKSV Remote Signing service provides protection against Cross Site Request Forgery (CSRF) by using so-called CSRF tokens. primesign RKSV Remote Signing service sets a randomly

generated CSRF token as cookie named **XSRF-TOKEN**. All subsequent requests have to contain both, the issued cookie and the CSRF token as value of the Header **X-XSRF-TOKEN**. An example for handling CSRF tokens is provided within the documentation for Login/Logout.

API Token (Stateless without Session)

With stateless authentication the user's shared secret is included within the **X-AUTH-TOKEN** header, see example below.

Example

```
GET /rs/admin/users HTTP/1.1
X-AUTH-TOKEN: VY4JP-ZZO1O-3K1P3-QEOPH-EIC95
```

The **X-AUTH-TOKEN** header has to be included with every request.

2.3.2. Login/Logout

Login and Logout are only used with session-based authentication. See session-based authentication for more information. With stateless authentication the credentials are included with every HTTP request and no login or logout has to be performed.

Login

This method should be used for login and to retrieve information on the logged in user. If the user is not authenticated, HTTP Status code **401 Unauthorized** is returned. In case the user is authenticated or the provided user credentials are correct, this method returns information on the currently authenticated user.

URL Structure: /rs/admin/login/user

Example:

HTTP Request:

```
GET /rs/admin/login/user HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=d349c21b-07df-4a05-889b-c212d7f8708c;path=/rs
SESSION=7d7f7095-3ee3-42c9-9e93-4d601b4e0546;path=/rs/admin;HttpOnly
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 254

{
  "userId" : "admin",,
  "registrationTimeStamp" : 1481112797000,
  "enabled" : true,
  "defaultKey" : null,
  "roles" : [ "USER", "ADMIN" ]
}
```

The **path** field within the Header Set-Cookie is set to the context path of the primesign RKSV Remote Signing service. The example above uses the context path **/rs**.

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`defaultKey`	`STRING`	`true`	The user's default key.

Path	Type	Optional	Description
`roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Logout

Logout the currently authenticated user.

URL Structure: /rs/admin/logout

Example:

HTTP Request:

```
POST /rs/admin/logout HTTP/1.1
Content-Type: application/json;charset=UTF-8
X-CSRF-TOKEN: f6099566-3a0f-4285-b41b-3013afc5ab01
Content-Length: 0
Cookie: SESSION=7d7f7095-3ee3-42c9-9e93-4d601b4e0546; XSRF-TOKEN=f6099566-
3a0f-4285-b41b-3013afc5ab01
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=;Max-Age=0;path=/rs SESSION=;Max-
Age=0;path=/rs/admin;HttpOnly
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 0
```

2.3.3. Managing Users

List All Users

Retrieves a list of all users within the primesign RKSV Remote Signing service.

URL Structure: /rs/admin/users

Example:

HTTP Request:

```
GET /rs/admin/users HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=bf0ab145-7175-4c2b-b7e2-0d9bbca0dad1; Path=/
X-Transaction-ID: z1tihg555eom6
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 301

[ {
  "userId" : "ps-t4p45dfo",
  "registrationTimeStamp" : 1715062701169,
  "enabled" : true,
  "defaultKey" : "3fp65zav",
  "roles" : [ "USER" ]
}, {
  "userId" : "ps-ff545aw3",
  "registrationTimeStamp" : 1715062701169,
  "enabled" : true,
  "defaultKey" : null,
  "roles" : [ "USER", "ADMIN" ]
} ]
```

Response Field Descriptions:

Path	Type	Optional	Description
`[].userId`	`STRING`	`false`	The user's unique id.
`[].registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.
`[].enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`[].defaultKey`	`STRING`	`true`	The user's default key.
`[].roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Create User

Create a new user. All parameters are optional.

Default values:

- **userId**: The userId will be autogenerated.
- **password**: The password will be autogenerated.
- **enabled**: true
- **roles**: USER

> The default pattern for userId allows only lowercase characters, digits and the special characters "-" and "_".

Create a new user without parameters:

URL Structure: /rs/admin/users

Example:

HTTP Request:

```
POST /rs/admin/users HTTP/1.1
X-CSRF-TOKEN: 4c786d23-fb4d-47e6-8e69-24a495d8467f
Content-Type: application/x-www-form-urlencoded
```

HTTP Response:

```
HTTP/1.1 201 Created
X-Transaction-ID: 7o2mias35zdk4
Location: http://localhost:8080/rs/admin/users/ps-t4p45dfo
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 133

{
  "userId" : "ps-t4p45dfo",
  "enabled" : true,
  "password" : "somePassword",
  "sharedSecret" : "TXZN5-WBBAR-6A3FQ-2KZUD-ROJSV"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`password`	`STRING`	`false`	The user's password.

Path	Type	Optional	Description
`sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.

Create a new user with parameters:

Example:

HTTP Request:

```
POST /rs/admin/users HTTP/1.1
Content-Type: application/json
Content-Length: 114
X-CSRF-TOKEN: d8380c89-85f9-4c3c-aa9b-b62fd0ab0216

{
  "userId" : "ps-t4p45dfo",
  "password" : "somePassword",
  "roles" : [ "USER", "ADMIN" ],
  "enabled" : true
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`true`	The user's unique id.
`password`	`STRING`	`true`	The user's password.
`roles`	`ARRAY`	`true`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Path	Type	Optional	Description
`enabled`	`BOOLEAN`	`true`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.

HTTP Response:

```
HTTP/1.1 201 Created
X-Transaction-ID: rwd6tsck46mdh
Location: http://localhost:8080/rs/admin/users/ps-t4p45dfo
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 133

{
  "userId" : "ps-t4p45dfo",
  "enabled" : true,
  "password" : "somePassword",
  "sharedSecret" : "TXZN5-WBBAR-6A3FQ-2KZUD-ROJSV"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.

Path	Type	Optional	Description
`password`	`STRING`	`false`	The user's password.
`sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.

Fetch a User

Retrieve a specific user identified by its unique userId.

URL Structure: /rs/admin/users/{userId}

Parameter	Description
`userId`	The userId of the user that should be fetched.

Example:

HTTP Request:

```
GET /rs/admin/users/ps-t4p45dfo HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=dfa769e4-ff34-4630-96c8-9dc953e6203c; Path=/
X-Transaction-ID: rgckytsvh47g1
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 146

{
  "userId" : "ps-t4p45dfo",
  "registrationTimeStamp" : 1715062701281,
  "enabled" : true,
  "defaultKey" : "3fp65zav",
  "roles" : [ "USER" ]
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`defaultKey`	`STRING`	`true`	The user's default key.
`roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Update User

Update a specific user. Currently only the fields **enabled**, **roles** and **defaultKey** can be updated.

The currently authenticated admin is not allowed to remove his role ADMIN. Furthermore, the currently authenticated admin is not allowed to disable himself. When updating the **defaultKey** the user has to be allowed to access the specified signature key.

URL Structure: /rs/admin/users/{userId}

Parameter	Description
`userId`	The userId of the user that should be updated.

Example:

HTTP Request:

```
PUT /rs/admin/users/ps-t4p45dfo HTTP/1.1
Content-Type: application/json
Content-Length: 146
X-CSRF-TOKEN: b80058af-5467-495e-b6ff-432e9951d8ae

{
  "userId" : "ps-t4p45dfo",
  "registrationTimeStamp" : 1715062701346,
  "enabled" : true,
  "defaultKey" : "3fp65zav",
  "roles" : [ "USER" ]
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.

Path	Type	Optional	Description
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`defaultKey`	`STRING`	`true`	The user's default key.
`roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: 7wfgpzi0ydjub
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 146

{
  "userId" : "ps-t4p45dfo",
  "registrationTimeStamp" : 1715062701346,
  "enabled" : true,
  "defaultKey" : "3fp65zav",
  "roles" : [ "USER" ]
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.

Path	Type	Optional	Description
`registrationTimeStamp`	`NUMBER`	`false`	The registration date of this user.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`defaultKey`	`STRING`	`true`	The user's default key.
`roles`	`ARRAY`	`false`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.

Delete User

Deletes a user identified by its unique userId.

When deleting a user, all signature keys of that user are deleted and the associated certificates are revoked and deleted as well. This action cannot be undone. When deleting a user, the associated signature keys are deleted irrevocably and it is not possible to create any more signatures with these signature keys.

URL Structure: /rs/admin/users/{userId}

Parameter	Description
`userId`	The userId of the user that should be deleted.

Example:

HTTP Request:

```
DELETE /rs/admin/users/ps-t4p45dfo HTTP/1.1
X-CSRF-TOKEN: 5043e58f-f82a-4e64-8594-583c5b4cdd01
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: f6lf57ko8c1qh
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

Update Password

Updates the user's password. The password is included as string within the request body.

URL Structure: /rs/admin/users/{userId}/password

Parameter	Description
`userId`	The userId of the user whose password should be updated.

Example:

HTTP Request:

```
PUT /rs/admin/users/ps-t4p45dfo/password HTTP/1.1
Content-Length: 11
X-CSRF-TOKEN: 7aabf04c-3353-429c-b308-00188097e29d

newPassword
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: zf4ypl06iezrp
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

Reset Shared Secret

Resets the user's shared secret. The shared secret is included as string within the response body.

URL Structure: /rs/admin/users/{userId}/sharedsecret

Parameter	Description
`userId`	The userId of the user whose shared secret should be reset.

Example:

HTTP Request:

```
PUT /rs/admin/users/ps-t4p45dfo/sharedsecret HTTP/1.1
Content-Length: 29
X-CSRF-TOKEN: 8daf7435-813a-445b-b19a-b33cf96b9c72

EJ2NN-FALQG-KZN68-9EANX-RMMSL
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: tki2fwrr5nmqx
Content-Type: text/plain
Content-Length: 29
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

EJ2NN-FALQG-KZN68-9EANX-RMMSL
```

List Owned Signature Keys

Retrieve a list of signature keys owned by the user with the given userId.

URL Structure: /rs/admin/users/{userId}/keys

Parameter	Description
`userId`	The userId of the user whose signature keys should be fetched.

Example:

HTTP Request:

```
GET /rs/admin/users/ps-t4p45dfo/keys HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: r3gooe6ckxtl8
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 433

[ {
  "keyId" : "3fp65zav",
  "enabled" : true,
  "creationTimeStamp" : 1484137584817,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
} ]
```

Response Field Descriptions:

Path	Type	Optional	Description
`[].keyId`	`STRING`	`false`	The signature key's unique id.
`[].enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.
`[].creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`[].keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`[].certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`[].certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`[].certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`[].certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`[].certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`[].certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`[].certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`[].owner`	`STRING`	`false`	The userId of the owner of this signature key.

2.3.4. Managing Signature Keys

List All Signature Keys

Retrieves a list of all signature keys within the primesign RKSV Remote Signing service.

URL Structure: /rs/admin/keys

Example:

HTTP Request:

```
GET /rs/admin/keys HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=f26ee157-68f2-42d7-8aac-a47aaab2ab55; Path=/
X-Transaction-ID: uh555d9uoirx1
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 867

[ {
  "keyId" : "3fp65zav",
  "enabled" : true,
  "creationTimeStamp" : 1715062705862,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
}, {
  "keyId" : "1twt1a6p",
  "enabled" : true,
  "creationTimeStamp" : 1715062705862,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU39163339,O=Shop XY GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "127864",
    "serialNumberHex" : "1f378",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
} ]
```

Response Field Descriptions:

Path	Type	Optional	Description
`[].keyId`	`STRING`	`false`	The signature key's unique id.
`[].enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.
`[].creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).
`[].keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`[].certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`[].certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`[].certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`[].certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`[].certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`[].certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`[].certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`[].owner`	`STRING`	`false`	The userId of the owner of this signature key.

Create Signature Key

Create a new signature key. Each signature key has to be associated to a user who owns that signature key.

URL Structure: /rs/admin/keys

Example:

HTTP Request:

```
POST /rs/admin/keys HTTP/1.1
Content-Type: application/json
Content-Length: 31
X-CSRF-TOKEN: aa9d22e2-9046-4703-a1d0-af276c593ef2

{
  "ownerId" : "ps-ff545aw3"
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`ownerId`	`STRING`	`false`	The user id of the owner of the newly created signature key.

HTTP Response:

```
HTTP/1.1 201 Created
X-Transaction-ID: 6l1fiv7v90yrq
Location: http://localhost:8080/rs/admin/keys/expected
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 166

{
  "keyId" : "expected",
  "enabled" : false,
  "creationTimeStamp" : 1715062705901,
  "keyAlgorithmType" : "EC",
  "certificate" : null,
  "owner" : "ps-ff545aw3"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`keyId`	`STRING`	`false`	The signature key's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.
`creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`owner`	`STRING`	`false`	The userId of the owner of this signature key.

Fetch a Signature Key

Retrieve a specific signature key identified by its unique keyId.

URL Structure: /rs/admin/keys/{keyId}

Parameter	Description
`keyId`	The keyId of the signature key that should be fetched.

Example:

HTTP Request:

```
GET /rs/admin/keys/3fp65zav HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=36eee055-8c9d-4706-bb22-3d01e1d06695; Path=/
X-Transaction-ID: q587wf032jbgj
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 429

{
  "keyId" : "3fp65zav",
  "enabled" : true,
  "creationTimeStamp" : 1715062705834,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`keyId`	`STRING`	`false`	The signature key's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.

Path	Type	Optional	Description
`creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).
`keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The certificate associated to this signature key.

Path	Type	Optional	Description
`owner`	`STRING`	`false`	The userId of the owner of this signature key.

Update Signature Key

Update a specific signature key. Currently only the fields **enabled** and **owner** can be changed. Changes to other fields will be ignored.

It is not possible to specify a null value for the field **owner**. Each signature key has to have an owner.

URL Structure: /rs/admin/keys/{keyId}

Parameter	Description
`keyId`	The keyId of the signature key that should be updated.

Example:

HTTP Request:

```
PUT /rs/admin/keys/3fp65zav HTTP/1.1
Content-Type: application/json
Content-Length: 430
X-CSRF-TOKEN: b28429b6-793d-4097-bf87-d7cfcec79623

{
  "keyId" : "3fp65zav",
  "enabled" : false,
  "creationTimeStamp" : 1715062705920,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`keyId`	`STRING`	`false`	The signature key's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.
`creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).
`keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).

Path	Type	Optional	Description
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`owner`	`STRING`	`false`	The userId of the owner of this signature key.

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: wf18hn9bdhdq4
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 430

{
  "keyId" : "3fp65zav",
  "enabled" : false,
  "creationTimeStamp" : 1715062705920,
  "keyAlgorithmType" : "EC",
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1484137584817,
    "notAfter" : 1515673584817
  },
  "owner" : "ps-t4p45dfo"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`keyId`	`STRING`	`false`	The signature key's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates whether this signature key is enabled or disabled. It is not possible to create signatures with a disabled signature key.
`creationTimeStamp`	`NUMBER`	`false`	The creation date of this signature key (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`owner`	`STRING`	`false`	The userId of the owner of this signature key.

Delete a Signature Key

Deletes a signature key identified by its unique keyId. If the deleted signature key is a user's default key, the user no longer has a default key. If any certificate is associated with the signature key it gets deleted as well.

This action cannot be undone.

URL Structure: /rs/admin/keys/{keyId}

Parameter	Description
`keyId`	The keyId of the signature key that should be deleted.

Example:

HTTP Request:

```
DELETE /rs/admin/keys/ps-t4p45dfo HTTP/1.1
X-CSRF-TOKEN: 10ab82a7-96bc-4218-952d-c6ae4ecb03e1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: 6nav1z97ho8yt
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

2.3.5. Managing Certificates

Each signature key is associated with a certificate issued by primesign. A signature key can only have one certificate and vice versa a certificate can only be associated to one single signature key.

Request Certificate

Requests a certificate for an existing signature key.

Within the Location Header of the HTTP response the URL to the generated certificate is included.

URL Structure: /rs/admin/keys/{keyId}/certificate

Parameter	Description
`keyId`	The keyId of the key this certificate was issued for.

Example:

HTTP Request:

```
POST /rs/admin/keys/3fp65zav/certificate HTTP/1.1
Content-Type: application/json
Content-Length: 244
X-CSRF-TOKEN: 5adfaf66-5c37-4abf-8656-d60db6f0dad1

{
  "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
  "templateId" : "rksv-r1",
  "regInfo" : {
    "contactInfo" : "cn=Max Mustermann,EmailAddress=max.mustermann@test.at",
    "productId" : "123456789",
    "accountingId" : "123456"
  }
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate should be issued for. When included within an attribute value (e.g. CN), the following characters have to be escaped by prefixing a backslash '\': '"', '#', '+', ',', ';', '<', '=', '>', and "(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.
`templateId`	`STRING`	`false`	Id referencing the template that should be used for certificate issuance.

Path	Type	Optional	Description
`regInfo`	`OBJECT`	`false`	The regInfo map that should be passed to the RA-Gate. The content of regInfo is customer-specific and defined by primesign. Typically it includes the fields `accountingId`, `contactInfo` and optionally `productId`, further fields are possible. For `contactInfo` the following formatting restrictions apply: The following characters have to be escaped by prefixing a backslash '\\': '\"', '#', '+', ',', ';', '<', '=', '>', and '\'(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.

For RKSV usage the `subjectDN` has to contain:

- VAT/UID number: Prefix of "UID ATU" followed by 8 digits, e.g. "CN=UID ATU12345678,O=Company name,C=AT"
- Global location number: Prefix of "GLN " followed by 13 digits, e.g. "CN=GLN 1234567890123,O=Company name,C=AT"
- Austrian tax number: Prefix of "Steuernummer " followed by 9 digits, e.g. "CN=Steuernummer

123456789,O=Company name,C=AT"

primesign provides a documentation for each customer, which fields have to be included within `regInfo`. Typically `regInfo` includes the fields `accountingId`, `contactInfo` of the certificate subject and optionally `productId`, further fields are possible.

HTTP Response:

```
HTTP/1.1 201 Created
X-Transaction-ID: fszq4g5iag78m
Location: http://localhost:8080/rs/admin/keys/3fp65zav/certificate
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 254

{
  "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
  "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
  "serialNumber" : "379109",
  "serialNumberHex" : "5c8e5",
  "notBefore" : 1715062704784,
  "notAfter" : 1715062704784
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.

Path	Type	Optional	Description
`serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).

Download Certificate (JSON)

Downloads the certificate as JSON data-structure.

URL Structure: /rs/admin/keys/{keyId}/certificate

Parameter	Description
`keyId`	The keyId of the key this certificate is associated with.

Example:

HTTP Request:

```
GET /rs/admin/keys/3fp65zav/certificate HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: hzl3ly3e9oc22
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 254

{
  "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
  "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
  "serialNumber" : "379109",
  "serialNumberHex" : "5c8e5",
  "notBefore" : 1715062704813,
  "notAfter" : 1746598704813
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).

Download Certificate (*.cer)

Downloads the certificate as DER-encoded *.cer file.

URL Structure: /rs/admin/keys/{keyId}/certificate.cer

Parameter	Description
`keyId`	The keyId of the key this certificate is associated with.

Example:

HTTP Request:

```
GET /rs/admin/keys/3fp65zav/certificate.cer HTTP/1.1
```

HTTP Response:

The file **certificate.cer** with Media Type **application/x-x509-ca-cert**.

Download Certificate (*.pem)

Downloads the certificate as PEM-encoded *.pem file.

URL Structure: /rs/admin/keys/{keyId}/certificate.pem

Parameter	Description
`keyId`	The keyId of the key this certificate is associated with.

Example:

HTTP Request:

```
GET /rs/admin/keys/3fp65zav/certificate.pem HTTP/1.1
```

HTTP Response:

The file **certificate.pem** with Media Type **application/x-x509-ca-cert**.

Delete Certificate

Deletes a certificate identified by its signature key's unique keyId. Deleting a certificate also deletes the associated signature key. The primesign TSP will be contacted to revoke the certificate. If the deleted signature key is a user's default key, the user no longer has a default key.

This action **cannot** be undone.

URL Structure: /rs/admin/keys/{keyId}/certificate

Parameter	Description
`keyId`	The keyId of the key this certificate is associated with.

Request parameter:

Name	Type	Optional	Description
`norevocation`	`BOOLEAN`	`true`	Indicates whether the certificate revocation should be skipped (true) or not (false). The default value, if not provided is false.

The parameter `norevocation` may only be used in consultation with the TSP.

Example:

HTTP Request:

```
DELETE /rs/admin/keys/3fp65zav/certificate?norevocation=false HTTP/1.1
X-CSRF-TOKEN: 6135df41-b20e-40c7-b075-ab9206bcc4ee
Content-Type: application/x-www-form-urlencoded
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: growxuw5w6b6y
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

2.3.6. All-In-One Certificate Request

A certificate is needed to generate a signature. In order to request a new RKSV signing certificate, first a user and a signature key have to be created. A certificate is associated to exactly one signature key.

For convenience, the Admin REST API offers a single REST method to create a new user, a new signature key and request a certificate for the newly generated signature key. In response, primesign RKSV Remote Signing service returns the userId and the credentials of the newly generated user, the keyId of the created signature key and a JSON representation of the issued certificate.

Within the Location Header of the HTTP response the URL to the generated certificate is included.

Create User, Key and Request Certificate

Creates a new user, creates a new signature key for this user and requests a certificate for the newly created signature key.

Example:

HTTP Request:

```
POST /rs/admin/certificate HTTP/1.1
Content-Type: application/json
Content-Length: 426
X-CSRF-TOKEN: 58fd0f5a-636a-41cf-91b0-066b27989e38

{
  "user" : {
    "userId" : "test-user-1",
    "password" : "somePassword",
    "roles" : [ "USER", "ADMIN" ],
    "enabled" : true
  },
  "certificateRequest" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "templateId" : "rksv-r1",
    "regInfo" : {
      "contactInfo" : "cn=Max
Mustermann,EmailAddress=max.mustermann@test.at",
      "productId" : "123456789",
      "accountingId" : "123456"
    }
  }
}
```

Request Field Descriptions:

Path	Type	Optional	Description
`user.userId`	`STRING`	`true`	The user's unique id.
`user.password`	`STRING`	`true`	The user's password.
`user.roles`	`ARRAY`	`true`	The roles of this user. Only users with role 'ADMIN' can access the Admin REST API.
`user.enabled`	`BOOLEAN`	`true`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.

Path	Type	Optional	Description
`user`	`OBJECT`	`true`	The user that should be created.
`certificateRequest.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate should be issued for. When included within an attribute value (e.g. CN), the following characters have to be escaped by prefixing a backslash '\': '"', '#', '+', ',', ';', '<', '=', '>', and "(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.
`certificateRequest.templateId`	`STRING`	`false`	Id referencing the template that should be used for certificate issuance.

Path	Type	Optional	Description
`certificateRequest.regInfo`	`OBJECT`	`false`	The regInfo map that should be passed to the RA-Gate. The content of regInfo is customer-specific and defined by primesign. Typically it includes the fields `accountingId`, `contactInfo` and optionally `productId`, further fields are possible. For `contactInfo` the following formatting restrictions apply: The following characters have to be escaped by prefixing a backslash '\\': '\"', '#', '+', ',', ';', '<', '=', '>', and '\'(U+0022, U+0023, U+002B, U+002C, U+003B, U+003C, U+003D, U+003E, U+005C, respectively). Leading or trailing spaces are prohibited within attribute values. See https://www.ietf.org/rfc/rfc4514.txt Section 2.4 for more information.
`certificateRequest`	`OBJECT`	`false`	The certificate request.

For RKSV usage the `subjectDN` has to contain:

- VAT/UID number: Prefix of "UID ATU" followed by 8 digits, e.g. "CN=UID ATU12345678,O=Company name,C=AT"

- Global location number: Prefix of "GLN " followed by 13 digits, e.g. "CN=GLN 1234567890123,O=Company name,C=AT"
- Austrian tax number: Prefix of "Steuernummer " followed by 9 digits, e.g. "CN=Steuernummer 123456789,O=Company name,C=AT"

primesign provides a documentation for each customer, which fields have to be included within `regInfo`. Typically `regInfo` includes the fields `accountingId`, `contactInfo` of the certificate subject and optionally `productId`, further fields are possible.

HTTP Response:

```
HTTP/1.1 201 Created
X-Transaction-ID: ofd3n9yo1z4gv
Location: http://localhost:8080/rs/admin/keys/3fp65zav/certificate
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 488

{
  "user" : {
    "userId" : "test-user-1",
    "enabled" : true,
    "password" : "somePassword",
    "sharedSecret" : "TXZN5-WBBAR-6A3FQ-2KZUD-ROJSV"
  },
  "key" : {
    "keyId" : "3fp65zav"
  },
  "certificate" : {
    "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
    "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
    "serialNumber" : "379109",
    "serialNumberHex" : "5c8e5",
    "notBefore" : 1715062706998,
    "notAfter" : 1715062706998
  }
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`user.userId`	`STRING`	`false`	The user's unique id.
`user.enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`user.password`	`STRING`	`false`	The user's password.
`user.sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.
`user`	`OBJECT`	`false`	The newly created user.
`key.keyId`	`STRING`	`false`	The signature key's unique id.
`key`	`OBJECT`	`false`	The newly created signature key.
`certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).

Path	Type	Optional	Description
`certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`certificate`	`OBJECT`	`true`	The issued certificate.

2.3.7. Cashbox Configuration

According to Section 2 of [RKSV-Detail] every receipt has to include an algorithm attribute ("Registrierkassenalgorithmuskennzeichen", German). Amongst other fields this attribute contains the id of the trust service provider that issued the certificate (Field "VDA"). This id consists of the country code of the country where the certificate authority resides in and the index of the certificate authority. In case of primesign the field "VDA" has the value "AT3".

Furthermore, the receipt has to contain the serial number of the issued certificate. Both values, "VDA" and serial number of the certificate can be retrieved via the REST API. In addition, the cashbox configuration includes information about the user (e.g., a cash box) as well as a list of all its associated signature keys and certificates.

Get Cashbox Configuration

Retrieves the cashbox configuration for the specified user.

URL Structure: /rs/admin/rk/config/{userId}

Parameter	Description
`userId`	The user the cashbox configuration has been requested for.

Example:

HTTP Request:

```
GET /rs/admin/rk/config/ps-t4p45dfo HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
Set-Cookie: XSRF-TOKEN=0eac11dd-fc9b-4ac6-80ef-a04cbb987bc4; Path=/
X-Transaction-ID: ta7fgk6s1zvmg
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 536

{
  "tspId" : "AT3",
  "user" : {
    "userId" : "ps-t4p45dfo",
    "enabled" : true,
    "signatureKeys" : [ {
      "keyId" : "3fp65zav",
      "keyAlgorithmType" : "EC",
      "certificate" : {
        "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
        "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign
GmbH,C=AT",
        "serialNumber" : "379109",
        "serialNumberHex" : "5c8e5",
        "notBefore" : 1483228860000,
        "notAfter" : 1577836680000
      }
    } ],
    "defaultKey" : "3fp65zav"
  }
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`tspId`	`STRING`	`false`	Id of the trust service provider (Field: VDA). PrimeSign has the value "AT3".
`user.userId`	`STRING`	`false`	The user's unique id.
`user.enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`user.signatureKeys[].keyId`	`STRING`	`false`	The signature key's unique id.
`user.signatureKeys[].keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`user.signatureKeys[].certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`user.signatureKeys[].certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`user.signatureKeys[].certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`user.signatureKeys[].certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).

Path	Type	Optional	Description
`user.signatureKeys[].certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`user.signatureKeys[].certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`user.signatureKeys[].certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`user.signatureKeys`	`ARRAY`	`false`	The signature keys this user is allowed to access.
`user.defaultKey`	`STRING`	`true`	The user's default key.
`user`	`OBJECT`	`false`	The user the cashbox configuration has been requested for.

2.4. User REST API

2.4.1. Authentication

The User REST API does not support sessions, therefore, the user credentials have to be included within every request. The User REST API supports two methods to include the user credentials.

API Token

When using this method the user's shared secret is included within the **X-AUTH-TOKEN** header, see example below. The shared secret is regarded as API Token.

Example

```
GET /rs/rk/config HTTP/1.1
X-AUTH-TOKEN: VY4JP-ZZO1O-3K1P3-QEOPH-EIC95
```

The **X-AUTH-TOKEN** header has to be included with every request.

Basic Authentication

The **userId** and the **sharedSecret** are included within the HTTP **Authorization** header, using the Basic Authentication Scheme [RFC2617].

Example

```
GET /rs/rk/config HTTP/1.1
Authorization: Basic dXNlcjEyMzp1c2VycGFzc3dvcmQxMjM=
```

The **Authorization** header has to be included with every request.

2.4.2. Signature Creation

The primesign RKSV Remote Signing service is responsible for signature creation only. Measures to provide chaining of cash box receipts and other requirements derived from [RKSV-Detail] have to be provided by the cash-box manufacturer.

primesign currently provides two types of API endpoints for signing:

- An API endpoint that takes **plain text** (e.g. a compact representation of a receipt) as input, creates a signature (using the referenced key) and returns it as **JWS signature** as defined in [RFC7515]. The currently implemented algorithm creates SHA256/ECDSA signatures.
- A second "raw" signature API endpoint that does **no digest calculation** at all. It just takes the provided data as is (should reflect a digest), creates a signature (current default algorithm is ECDSA) and returns the **raw signature** value encoded as Base64-URL string.

A user can have multiple signature keys, each identified by a unique keyId. In addition, each user can be associated with a default signature key (the first key created for a specific user will automatically be regarded as default key if not otherwise set) which is automatically used in cases where no certain keyId was set. Thus, the signature creation interface provides a short (no certain keyId, use default key) and a long (use certain keyId) URL scheme.

JSON Web Signature (JWS)

The signature service expects the text to be signed as plain text within the POST request body. The input is processed as input for a JWS signature as defined in [RFC7515] (a JWS header is created, transformed into Base64-URL string and concatenated with a Base64-URL string representation of the given plain text). The resulting JWS signature value consists of the Base64-URL encoded header, the Base64-URL encoded payload and the Base64-URL encoded signature value, each separated with "`."`.

The request body contains data to be signed as UTF-8 encoded string (see example request below). The data to be signed is sent as plain text corresponding to the compact representation of the receipt as defined in [RKSV-Detail] Section 5.

The resulting JWS signature value consists of the Base64-URL encoded header, the Base64-URL encoded payload and the Base64-URL encoded signature value, each separated with ".". The content-type is "text/plain".

Create a new JWS Signature and explicitly declare the signing key:

URL Structure: /rs/rk/keys/{keyId}/signatures/{algorithmId}

Parameter	Description
`keyId`	The keyId of the private signature key. The keyId is automatically generated during issuance of the certificate.
`algorithmId`	The algorithmId defines the used algorithm suite as specified in [RKSV-Detail] Section 2 as „Registrierkassenalgorithmuskennzeichen" (ger.).Currently the only available option is „r1" (SHA256withECDSA signature)

Example:

HTTP Request:

```
POST /rs/rk/keys/key1/signatures/r1 HTTP/1.1
Content-Type: text/plain;charset=UTF-8
Content-Length: 140

_R1-AT3_CASHBOX-DEMO-1_CASHBOX-DEMO-1-Receipt-ID-118_2016-09-
05T10:17:58_0,00_0,00_0,00_0,00_0,00_tAbKOB0ZTXA=_39d5363a16eafa0a_4pjhTqKf+k
g=
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: a6d54088ilqid
Content-Type: text/plain;charset=UTF-8
Content-Length: 298
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

eyJhbGciOiJFUzI1NiJ9.X1IxLUFUMzAwX0NBU0hCT1gtREVNTy0xX0NBU0hCT1gtREVNTy0xLVJl
Y2VpcHQtSUQtMTE4XzIwMTYtMDktMDVUMTA6MTc6NThfMCwwMF8wLDAwXzAsMDBfMCwwMF8wLDAwX
3RBYktPQjBaVFhBPV8zOWQ1MzYzYTE2ZWFmYTBhXzRwamhUcUtmK2tnPQ.t0DWYCl4jtMT7736Pqs
ZH9WD-e3-79AHxko5OCi6vpcsgEB_0KTNBQh-HMwvoHsPmnUXOvRVYK-Gkat7U6T16Q
```

Create a new JWS Signature using the default signing key:

URL Structure: /rs/rk/signatures/{algorithmId}

Parameter	Description
`algorithmId`	The algorithmId defines the used algorithm suite as specified in [RKSV-Detail] Section 2 as „Registrierkassenalgorithmuskennzeichen“ (ger.).Currently the only available option is „r1“ (SHA256withECDSA signature)

Example:

HTTP Request:

```
POST /rs/rk/signatures/r1 HTTP/1.1
Content-Type: text/plain;charset=UTF-8
Content-Length: 140

_R1-AT3_CASHBOX-DEMO-1_CASHBOX-DEMO-1-Receipt-ID-118_2016-09-
05T10:17:58_0,00_0,00_0,00_0,00_0,00_tAbKOB0ZTXA=_39d5363a16eafa0a_4pjhTqKf+k
g=
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: x0ecxinejwuti
Content-Type: text/plain;charset=UTF-8
Content-Length: 298
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

eyJhbGciOiJFUzI1NiJ9.X1IxLUFUMzAwX0NBU0hCT1gtREVNTy0xX0NBU0hCT1gtREVNTy0xLVJl
Y2VpcHQtSUQtMTE4XzIwMTYtMDktMDVUMTA6MTc6NThfMCwwMF8wLDAwXzAsMDBfMCwwMF8wLDAwX
3RBYktPQjBaVFhBPV8zOWQ1MzYzYTE2ZWFmYTBhXzRwamhUcUtmK2tnPQ.t0DWYCl4jtMT7736Pqs
ZH9WD-e3-79AHxko5OCi6vpcsgEB_0KTNBQh-HMwvoHsPmnUXOvRVYK-Gkat7U6T16Q
```

Raw Signature

This API endpoint for raw signatures expects the content-type to be plain text and UTF-8 charset ("Content-Type": "text/plain;charset=UTF8"). Note that no digest calculation will be performed on this data. It is the caller's responsibility to perform digest calculation of the data to be signed making sure that the (digest) data provided to the raw signature service can be signed with the chosen algorithm. Currently the one and only algorithm supported is R1 (= ECDSA signature). The resulting signature is a ECDSA signature consisting of the EC pair r and s. Both values are returned concatenated in Base64-URL encoded form as expected by [RKSV-Detail].

The request body contains the input data for raw signature. In practice the body should contain a binary digest value (the plain digest bytes, no specific encoding). Note that the digest must fit to the chosen signature algorithm.

The response is a Base64-URL string (content type "text/plain") which contains both signature values (EC pair r and s for the currently available ECDSA algorithm R1) concatenated as expected by [RKSV-Detail].

Create a new Raw Signature and explicitly declare the signing key:

URL Structure: /rs/rk/keys/{keyId}/signatures/{algorithmId}

Parameter	Description
`keyId`	The keyId of the private signature key. The keyId is automatically generated during issuance of the certificate.
`algorithmId`	The algorithmId defines the used algorithm suite as specified in [RKSV-Detail] Section 2 as „Registrierkassenalgorithmuskennzeichen“ (ger.).Currently the only available option is „r1“ (SHA256withECDSA signature)

Example:

HTTP Request:

```
POST /rs/rk/keys/key1/signatures/r1raw HTTP/1.1
Content-Length: 32

□□□□□□ @J□  Þ□□□,e□□+W □w⋆□□
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: izvh7wrmm9qge
Content-Type: text/plain;charset=UTF-8
Content-Length: 86
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

Yn7MPLOgGjOtWu4U0QleBxUFFiBkWFTDDWvKdU3hvh6oDrAoMBGZMfzqI2vTCGxcCB83H2vB-
XBBRQQBVYqtPg
```

Create a new Raw Signature using the default signing key:

URL Structure: /rs/rk/signatures/{algorithmId}

Parameter	Description
`algorithmId`	The algorithmId defines the used algorithm suite as specified in [RKSV-Detail] Section 2 as „Registrierkassenalgorithmuskennzeichen“ (ger.).Currently the only available option is „r1“ (SHA256withECDSA signature)

Example:

HTTP Request:

```
POST /rs/rk/signatures/r1raw HTTP/1.1
Content-Length: 32

[illegible]
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: tkn2otkgzxn03
Content-Type: text/plain;charset=UTF-8
Content-Length: 86
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

Yn7MPLOgGjOtWu4U0QleBxUFFiBkWFTDDWvKdU3hvh6oDrAoMBGZMfzqI2vTCGxcCB83H2vB-
XBBRQQBVYqtPg
```

2.4.3. Certificate Retrieval

Each signature key is associated with a certificate issued by primesign. The REST API offers methods to download the certificate. A certificate can either be downloaded as JSON data-structure, DER-encoded *.cer file or PEM-encoded *.pem file.

A user can only download certificates of signature keys he or she is allowed to access.

Download Certificate (JSON)

Downloads the certificate as JSON data-structure.

URL Structure: /rs/keys/{keyId}/certificate

Parameter	Description
`keyId`	The keyId associated with the certificate

Example:

HTTP Request:

```
GET /rs/keys/3fp65zav/certificate HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: 0j4jvkmfkl0un
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 254

{
  "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
  "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign GmbH,C=AT",
  "serialNumber" : "379109",
  "serialNumberHex" : "5c8e5",
  "notBefore" : 1715062728510,
  "notAfter" : 1746598728510
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).
`notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).

Download Certificate (*.cer)

Downloads the certificate as DER-encoded *.cer file.

URL Structure: /rs/keys/{keyId}/certificate.cer

Parameter	Description
`keyId`	The keyId associated with the certificate

Example:

HTTP Request:

```
GET /rs/keys/3fp65zav/certificate.cer HTTP/1.1
```

HTTP Response:

The file **certificate.cer** with Media Type **application/x-x509-ca-cert**.

Download Certificate (*.pem)

Downloads the certificate as PEM-encoded *.pem file.

URL Structure: /rs/keys/{keyId}/certificate.pem

Parameter	Description
`keyId`	The keyId associated with the certificate

Example:

HTTP Request:

```
GET /rs/keys/3fp65zav/certificate.pem HTTP/1.1
```

HTTP Response:

The file **certificate.pem** with Media Type **application/x-x509-ca-cert**.

2.4.4. Cashbox Configuration

According to Section 2 of [RKSV-Detail] every cash box receipt has to include an algorithm attribute (“Registrierkassenalgorithmuskennzeichen”, German). Amongst other fields this attribute contains the id of the trust service provider that issued the certificate (Field “VDA”). This id consists of the country code of the country where the certificate authority resides in and the index of the certificate authority. In case of primesign the field “VDA” has the value “AT3”.

Furthermore, the receipt has to contain the serial number of the issued certificate. Both values, “VDA” and serial number of the certificate can be retrieved via the REST API. In addition, the cashbox configuration includes information about the user (e.g., a cash box) as well as a list of associated signature keys and certificates.

Get Cashbox Configuration

Retrieves the cashbox configuration for the currently authenticated user.

URL Structure: /rs/rk/config

Example:

HTTP Request:

```
GET /rs/rk/config HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: 1qjp76d8an4o4
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 536

{
  "tspId" : "AT3",
  "user" : {
    "userId" : "ps-t4p45dfo",
    "enabled" : true,
    "signatureKeys" : [ {
      "keyId" : "3fp65zav",
      "keyAlgorithmType" : "EC",
      "certificate" : {
        "subjectDN" : "CN=UID ATU79261119,O=Test GmbH,C=AT",
        "issuerDN" : "CN=CRYPTAS-PrimeSign !TEST! RKSV CA,O=PrimeSign
GmbH,C=AT",
        "serialNumber" : "379109",
        "serialNumberHex" : "5c8e5",
        "notBefore" : 1483228860000,
        "notAfter" : 1577836680000
      }
    } ],
    "defaultKey" : "3fp65zav"
  }
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`tspId`	`STRING`	`false`	Id of the trust service provider (Field: VDA). PrimeSign has the value "AT3".
`user.userId`	`STRING`	`false`	The user's unique id.

Path	Type	Optional	Description
`user.enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`user.signatureKeys[].keyId`	`STRING`	`false`	The signature key's unique id.
`user.signatureKeys[].keyAlgorithmType`	`STRING`	`false`	Algorithm identifier specifying the key type (aka. the algorithm) of this signature key (e.g. EC).
`user.signatureKeys[].certificate.subjectDN`	`STRING`	`false`	The distinguished name of the subject this certificate has been issued for.
`user.signatureKeys[].certificate.issuerDN`	`STRING`	`false`	The distinguished name of the certificate issuer.
`user.signatureKeys[].certificate.serialNumber`	`STRING`	`false`	The serial number of this certificate (decimal).
`user.signatureKeys[].certificate.serialNumberHex`	`STRING`	`false`	The serial number of this certificate (hexadecimal).
`user.signatureKeys[].certificate.notBefore`	`NUMBER`	`false`	The date on which this certificate becomes valid (in milliseconds since 1.1.1970).

Path	Type	Optional	Description
`user.signatureKeys[].certificate.notAfter`	`NUMBER`	`false`	The date after which this certificate is no longer valid (in milliseconds since 1.1.1970).
`user.signatureKeys[].certificate`	`OBJECT`	`true`	The certificate associated to this signature key.
`user.signatureKeys`	`ARRAY`	`false`	The signature keys this user is allowed to access.
`user.defaultKey`	`STRING`	`true`	The user's default key.
`user`	`OBJECT`	`false`	The user the cashbox configuration has been requested for.

2.5. primesign RKSV Remote Signing Administration

For the administration of the primesign RKSV Remote Signing service several REST API methods are exposed.

2.5.1. Setup

Get Setup Status

Simple check if the primesign RKSV Remote Signing service is initialized.

URL Structure: `/rs/setup`

Example Request for initialized instance:

HTTP Request:

```
GET /rs/setup HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: b9z9s5bq0oa50
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 4

true
```

Example Request for NOT initialized instance:

HTTP Request:

```
GET /rs/setup HTTP/1.1
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: s4ms5uj83siyd
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 5

false
```

Initial Setup

Initalizing the first user with administrator rights of the primesign RKSV Remote Signing service with given credentials or with automatically generated credentials.

URL Structure: /rs/setup

Example for setup call with a automatically generated user account:

HTTP Request:

```
POST /rs/setup HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: zaaarv6fl6g5h
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 134

{
  "userId" : "ps-1er3h9",
  "enabled" : true,
  "password" : "ZQQSjCIr",
  "sharedSecret" : "410f8b68-1b47-4e86-9a05-d2f9ab7df327"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`password`	`STRING`	`false`	The user's password.

Path	Type	Optional	Description
`sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.

Example for setup call with given user credentials:

HTTP Request:

```
POST /rs/setup HTTP/1.1
Content-Type: application/json
Content-Length: 55

{
  "userId" : "admin",
  "password" : "somePassword"
}
```

The default pattern for userId allows only lowercase characters, digits and the special characters "-" and "_".

Request Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	true	The user's unique id.
`password`	`STRING`	true	The user's password.

HTTP Response:

```
HTTP/1.1 200 OK
X-Transaction-ID: ryby53dt3zaqg
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 134

{
  "userId" : "admin",
  "enabled" : true,
  "password" : "somePassword",
  "sharedSecret" : "4aa92938-eb45-4dc0-8a8f-f230210e8dbb"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`userId`	`STRING`	`false`	The user's unique id.
`enabled`	`BOOLEAN`	`false`	Indicates if this user is enabled or disabled. A disabled user is not allowed to authenticate, e.g. the user is not allowed to create signatures.
`password`	`STRING`	`false`	The user's password.
`sharedSecret`	`STRING`	`false`	The user's shared secret. The shared secret is always autogenerated.

Fetch Instance Certificate

During initialization of the primesign RKSV Remote Signing service a new instance key and certificate is created automatically. In order to be able to send certificate requests to the RA-Gate, the instance certificate must be sent to PrimeSign GmbH. PrimeSign GmbH uses the instance

certificate to verify the authenticity of requests from the primesign RKSV Remote Signing service.

Example for retrieving the instance certificate as *.pem file:

HTTP Request:

```
GET /rs/setup/instancecertificate.pem HTTP/1.1
```

HTTP Response:

The file **certificate.pem** with Media Type **application/x-x509-ca-cert**.

2.5.2. Health Check

Check if primesign RKSV Remote Signing service is running

Diagnosis check for the status of the primesign RKSV Remote Signing service.

URL Structure: /rs/actuator/health

Example:

HTTP Request:

```
GET /rs/actuator/health HTTP/1.1
Accept: application/json
```

HTTP Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 21

{
  "status" : "UP"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`status`	`STRING`	false	Status for the primesign RKSV Remote Signing service. UP if everything works fine, other possible values are DOWN, OUT_OF_SERVICE and UNKNOWN.

2.5.3. Sign Check

Check if signatures via primesign RKSV Remote Signing service are possible

Diagnosis check whether signatures are possible via primesign RKSV Remote Signing service. This check does not query the status of the primesign RKSV Remote Signing administration service.

URL Structure: /rs/actuator/sign

Example:

HTTP Request:

```
GET /rs/actuator/sign HTTP/1.1
Accept: application/json
```

HTTP Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 21

{
  "status" : "UP"
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`status`	`STRING`	false	Signature status for the primesign RKSV Remote Signing service. UP if signatures are possible, otherwise DOWN.

2.6. Error Codes & Responses

2.6.1. Http Status Codes

HTTP status codes that are used by the primesign RKSV Remote Signing service.

Code	Text	Description
200	OK	Success! The requested action has been performed successfully.
201	Created	A new resource has been created. The Location header includes the path to the newly created resource.

Code	Text	Description
202	Accepted	The request has been accepted for processing, but the processing has not been completed. Currently used when requesting a certificate, but the certificate was not issued immediately as the certificate request is still pending (e.g. due to missing RO approval).
400	Bad Request	The request was invalid. The accompanying error message will explain further.
401	Unauthorized	The user is not authenticated or the provided user credentials are invalid.
403	Forbidden	Access to a resource was denied. For example, within the Admin REST API this status code is returned if no CSRF Token has been included or the provided CSRF Token is invalid.
404	Not Found	The requested URI is invalid or the requested resource does not exist.
405	Method Not Allowed	A request to a resource was made using a request method not supported by that resource.
409	Conflict	The request could not be completed due to a conflict with the current state of the target resource.
500	Internal Server Error	An error occurred during processing the request. The accompanying error message will explain further.

Table 1. Table Http Status Codes

2.6.2. Error Messages

When the primesign RKSV Remote Signing service returns error messages, the following JSON structure is returned.

This example shows the error occurring when creating a new user with a userId that already exists within the primesign RKSV Remote Signing service.

```
HTTP/1.1 409 Conflict
X-Transaction-ID: 0omvejlfxpc6w
Content-Type: application/json
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 187

{
  "errorCode" : 302,
  "errorMessage" : "Failed to create user 'user-1'. Duplicate userId.",
  "subject" : "user-1",
  "transactionId" : "0omvejlfxpc6w",
  "timestamp" : 1715062703450
}
```

Response Field Descriptions:

Path	Type	Optional	Description
`errorCode`	`NUMBER`	`false`	The code identifying the error.
`errorMessage`	`STRING`	`false`	The error message, suitable for being presented to the user.
`transactionId`	`STRING`	`true`	An optional identifier referencing the transaction related to the error.

Path	Type	Optional	Description
`subject`	`STRING`	`false`	The field that is subject to the error.
`timestamp`	`NUMBER`	`false`	The date the error has occurred (in milliseconds since 1.1.1970).

In case no value has been provided for a response field (i.e. subject) the response field is not included within the return error message. Note that an error response may contain an extra field `transactionId` referencing the transaction that results in this error. The respective `transactionId` may be helpful when tracking the error up to the originating request.

2.6.3. Error Codes

primesign RKSV Remote Signing service specific error codes.

Error Code	Identifier	Description
-1	UNSPECIFIED	Access to a resource was denied.
1	INVALID_REQUEST	The sent request is invalid, e.g. parameters are missing.
2	UNKNOWN_RESOURCE	The requested resource does not exist.
3	ACCESS_DENIED	The user is not allowed to access the requested resource.
100	UNSUPPORTED_SIGNATURE_ALGORITHM	The chosen signature algorithm is not supported.
101	UNKNOWN_SIGNATURE_KEY	The referenced signature key does not exist.
102	INVALID_DATA_TO_BE_SIGNED	The received data to be signed is invalid.

Error Code	Identifier	Description
103	UNKNOWN_CERTIFICATE	The requested certificate does not exist.
104	SIGNATURE_KEY_DISABLED	The referenced signature key is disabled and cannot be used for signing.
200	INSTANCE_ALREADY_INITIALIZED	The primesign RKSV Remote Signing service is already initialized.
201	UNKNOWN_WRAPPING_KEY	The wrapping key of this primesign RKSV Remote Signing service could not be found.
300	UNKNOWN_USER	The referenced user does not exist.
301	UNIQUE_USERID_GENERATION_FAILED	An error occurred during generation of a new random userId.
302	DUPLICATE_USER_ID	The chosen userId already exists. Choose another userId for this user.
303	INVALID_USER_ID	The chosen userId does not comply with the policy of this primesign RKSV Remote Signing service.
304	UNSUPPORTED_USER_ROLE	The given user role does not exist.
305	INVALID_SUBJECT_DN	The given subject distinguished name is invalid.
306	USER_DISABLED_FAILED	Failed to disable user. A user is not allowed to disable himself.
307	ACCESS_TO_KEY_DENIED	The user is not allowed to access the requested signature key.

Error Code	Identifier	Description
308	UPDATE_USER_ROLES_FAILED	Failed to update the user's rule. E.g. the currently authenticated user with role ADMIN is not allowed to remove his ADMIN role.
309	INVALID_CERTIFICATE	The certificate is not valid.
310	CERTIFICATE_ALREADY_ISSUED	A certificate for the given signature key has already been issued. There can only be one certificate per signature key.
311	CERTIFICATE_ALREADY_REQUESTED	A certificate for the given signature key has already been requested. There can only be one certificate request.
312	DELETE_USER_FAILED	User could not be deleted as the currently authenticated user with role ADMIN is not allowed to delete himself.
400	ISSUE_CERTIFICATE_FAILED	No certificate could be issued because of an internal error.
401	ISSUE_CERTIFICATE_FAILED_USER_ERROR	No certificate could be issued because of an invalid/missing user input.

Table 2. Table primesign RKSV Remote Signing Error Codes

3. References

- [RKSV-II]: Registrierkassensicherheitsverordnung Teil II
- [RKSV-Detail]: RKSV Detailspezifikation
- [RFC2617]: HTTP Authentication
- [RFC7515]: JSON Web Signature (JWS)

4. Revision History

4.1. Version 1.0.0

Date	Author	Changes
13.01.2017	Sandra Kreuzhuber, Heimo Divis	Initial Version
01.02.2017	Sandra Kreuzhuber	Add notice on subjectDN encoding, update sample values for shared secret, add new error code 312
02.02.2017	Sandra Kreuzhuber	Update sample input data for signature

4.2. Version 1.1.0

Date	Author	Changes
17.02.2017	Heimo Divis	Add chapter PrimeSign Remote Signing administration (setup and health)
20.02.2017	Manuel Schallar	Add documentation for error code (401) in case of an invalid user input while requesting a certificate, updated documentation for error code (400)

4.3. Version 1.1.1

Date	Author	Changes
21.02.2017	Thomas Knall	Add documentation for error response field `transactionId` and response header `X-Transaction-ID`

Date	Author	Changes
22.02.2017	Heimo Divis	Add project version to revision history. Add examples for GLN and austrian tax number.
27.03.2017	Thomas Knall	Add content type `text/plain;charset=UTF-8` to signature creation example.

4.4. Version 1.1.2

Date	Author	Changes
06.04.2017	Manuel Schallar	Replaced Base64 encoding references with Base64-URL encoding.
06.04.2017	Manuel Schallar	Added chapter 'References'.
05.12.2017	Heimo Divis	Updated Chapter 'Raw Signature'.

4.5. Version 2.0.2

Date	Author	Changes
17.09.2019	Manuel Schallar	Update documentation to reflect the API change: Deleting a certificate also deletes the signature key.
02.10.2019	Cornelia Lahnsteiner, Heimo Divis	Add documentation and warning for the 'norevocation' parameter.

4.6. Version 2.0.4

Date	Author	Changes
08.11.2019	Gernot Schaberl	Updating URL structures and refining document.
08.11.2019	Sandra Kreuzhuber	Added API-Call for creating a new instance key and certificate during initialization process.

4.7. Version 2.1.0

Date	Author	Changes
09.03.2020	Manuel Schallar	Add documentation for the `/rs/actuator/sign` endpoint.

4.8. Version 2.3.0

Date	Author	Changes
08.03.2023	Sandra Kreuzhuber	Add notice on `regInfo` format and provide examples.
03.04.2023	Simon Roth	Use the primesign asciidoc template for the documentation.

4.9. Version 2.4.9

Date	Author	Changes
30.04.2024	Sandra Kreuzhuber	Update product name to primesign RKSV Remote Signing service.