Kofax SignDoc Standard

Developer's Guide

Version: 3.0.0

Date: 2021-06-15

KOFAX

Table of Contents

Preface

This programming guide provides information on how to work with the Kofax SignDoc Standard REST API. It covers the user roles defined by SignDoc Standard and how they are reflected on the REST API as well as how to easily integrate SignDoc Standard in your own application.

There also is a complete REST API description on each individual REST request in the chapters that follow.

The SignDoc Standard REST API can be consumed by any programming language able to use HTTP requests. It requires nothing more than a valid API authentication to enable a user to interact with Kofax SignDoc Standard programmatically.

Related documentation

The full documentation set for Kofax SignDoc Standard is available at the following location:

https://docshield.kofax.com/Portal/Products/SD/3.0.0-7s9x4v5c5f/SD.htm

In addition to this guide, the documentation set includes the following items:

- *Help for Kofax SignDoc Standard*
- *Help for Kofax SignDoc Standard Administration Center*
- *Help for Signing Documents with Kofax SignDoc*
- *Kofax SignDoc Standard Administrator's Guide*
- *Kofax SignDoc Standard Installation Guide*
- *Kofax SignDoc Technical Specifications*

Training

Kofax offers both classroom and online training to help you make the most of your product. To learn more about training courses and schedules, visit the Kofax Education Portal on the Kofax website.

Getting help with Kofax products

The Kofax Knowledge Base repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base, go to the Kofax website and select **Support** on the home page.

Note The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).

  Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).

  Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.
- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Chapter 1

REST interface

The SignDoc Standard REST interface provides a simplified possibility to work with signing packages, accounts and teams from different clients via the HTTP(S) protocol. The information exchanged between client and application server is typically in JSON format, but could be also XML.

REST URL

RESTful web services can be accessed via a REST path which is appended to the SignDoc Standard context. It is followed by the SignDoc Standard REST version number and the requested resource with any necessary parameters.

The URL to the SignDoc Standard API has the following syntax:

```
scheme://domain:port/path?query_string
```

whereas `path` is divided in the parts

```
context/rest/version-number/resource
```

Part	Description
`context`	The `context` is `cirrus` by default.
`rest`	The `rest` part is fixed.
`version-number`	The `version-number` is `v8`.
`resource`	The `resource` part identifies the requested resource, for example `package`.

Example

```
http://localhost:6611/cirrus/rest/v8/package
```

REST error response

HTTP is based on the exchange of representations, and that applies to errors as well.

When a server encounters an error, either because of problems with the request that a client submitted or because of problems within the server, always return a representation that reflects the state of the error condition. This includes the response status code, response headers, and a body containing the description of the error.

For errors due to client inputs, return a representation with a 4xx status code. For errors due to server implementation or its current state, return a representation with a 5xx status code.

In case of an error the response body contains a RestMsgList element.

RestMsgList

- **list** (Array[RestMsg], optional): List of messages (info, warnings, errors)

RestMsg

- **code** (integer): Message code
- **message** (string): Message description
- **type** (string): Message type. Possible values: ERROR, WARNING, INFO

Example response body (JSON)

```
{
  "list" : [ {
   "code" : 1100,
   "message" : "The requested signing package does not exist",
   "type" : "ERROR"
  } ]
}
```

HTTP status code information

The SignDoc Standard REST APIs return either 200 (OK) or 201 (Created) when an API call successfully runs to completion, and status codes in the 400-500 range for failures.

Status Code	Description
400 (bad request)	Some of the data of the request was not valid or could not be processed in the current context. The response contains an error list with a unique identifier and a localized error message.
401 (unauthorized)	The authentication or authorization required for the API call did not pass.
403 (forbidden)	The authentication token or password is expired.
404 (not found)	The resource being accessed does not exist. This might occur on GET or DELETE requests.
405 (method not allowed)	The HTTP method (GET, PUT, POST or DELETE) is not valid for this resource URI.
415 (unsupported media type)	The data type of some data in the request is not supported.
500 (internal server error)	An internal server-side error has occurred.
503 (service unavailable)	The server is currently busy to process multiple requests at the same time. If a status code 503 occurs try again in a few seconds.

Authentication token

For each request on the SignDoc Standard REST API the user can authenticate via the Create an authentication token request (/rest/v8/users/authentication). This request takes user id and password as input to create an authentication token that enables a user to make further requests on the API. By default the authentication token is valid for 4 hours.

The following example represents an authentication token:


```
ew0KICAiYWNjb3VudElEIiA6ICJhY2NvdW50MyIsDQogICJhY2NvdW50TmFtZSIgOiAiYWNjaXVudD
MiLA0KICAidXNlcklkIiA6ICJ1c2VyMjQiLA0KICAidXNlck5hbWUiIDogInVzZXIxMzQiLA0KICAi
ZU1haWwiIDogInVzZXIyM0Brc2QuY29tIiwNCiAgInJvbGVzIiA6IFsgIlVTRVIiLCAiVEVBTU1HUi
IsICJBRE1JTiIgXSwNCiAgImV4cCIgOiAxNDQ3MDg5NzQ5NDA1LA0KICAiaWF0IiA6IDE0NDcwNzUz
NDk0MDUNCn0=.64HScbedftEqYvMWXyiLT9a/oajzv97wQTbrvOS97e4=
```


In the example the following part of the authentication token


```
ew0KICAiYWNjb3VudElEIiA6ICJhY2NvdW50MyIsDQogICJhY2NvdW50TmFtZSIgOiAiYWNjaXVudD
MiLA0KICAidXNlcklkIiA6ICJ1c2VyMjQiLA0KICAidXNlck5hbWUiIDogInVzZXIxMzQiLA0KICAi
ZU1haWwiIDogInVzZXIyM0Brc2QuY29tIiwNCiAgInJvbGVzIiA6IFsgIlVTRVIiLCAiVEVBTU1HUi
IsICJBRE1JTiIgXSwNCiAgImV4cCIgOiAxNDQ3MDg5NzQ5NDA1LA0KICAiaWF0IiA6IDE0NDcwNzUz
NDk0MDUNCn0=.
```


represents a Base64-encoded JSON structure containing the most important information of the authenticated user, the associated account as well as an expiration date.

This structure can be decoded and viewed at any time:

```
{
  "accountID" : "account3",
  "accountName" : "account3",
  "userId" : "user24",
  "userName" : "user134",
  "eMail" : "user23@ksd.com",
  "roles" : [ "USER", "TEAMMGR", "ADMIN" ],
  "exp" : 1447089749405,
  "iat" : 1447075349405
}
```

Separated by a full stop (.) the access token additionally contains a hash


```
64HScbedftEqYvMWXyiLT9a/oajzv97wQTbrvOS97e4=
```


ensuring that no changes can be made on the token.

Every further REST request needs to have an X-Auth-Token header containing the complete and valid authentication token. Each generated authentication token possesses the exact same permissions as the user it is associated with. This means, that a user with ADMIN role can only use his token with the exact same permissions.

In the same manner the signer can authenticate via Create a signing authentication token. Signer's authentication token will be returned as X-S-Auth-Token response header.

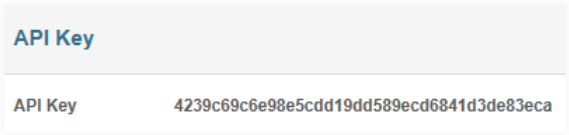
The signer's authentication data, decoded from the authentication token, has the following structure:

```
{
  "exp" : 1461964793414,
  "iat" : 1461950393414,
  "hst" : 270409440,
  "sst" : "c",
  "aid" : "account1",
  "pid" : "5b1bdcf5-71a8-4435-bdbf-56e37b2f96a3"
}
```

Note If a request to a SignDoc Standard REST API contains more than one authentication token, SignDoc Standard Server takes the first one according to the following order: X-S-Auth-Token, X-Auth-Token, API-key

Authentication via API key

For each request on the SignDoc Standard REST API the user can authenticate with an API key generated in the SignDoc Standard UI. This API key has to be either added to the request header (api-key: *API key*) or typed in text field on the upper right corner of the Swagger UI. To find the API key generated by SignDoc Standard the user has to navigate to the preference page of his individual account. There he can find a separate section for API keys.



Each API key generated possesses the exact same permissions as the user it is associated with. This means, that a user with account administrator role can only use his API key with the exact same permissions. With the API key an account administrator is able to use the REST API to create users, teams and signing packages by using the appropriate REST request. But as an account administrator he will be unable to create another account because this request requires a different permission level. The different user roles and permissions will be covered in more detail in the chapters that follow.

An authentication via API key is not possible if LDAP authentication is used on the system. In this case only the authentication via an authentication token can be used, where the LDAP authentication is used to generate the authentication token.

Representation formats

SignDoc Standard supports JSON and XML. The returned format is determined

- by an extension postfixed to the URL (".json" or ".xml"), if using a simple web browser without the option to manipulate the Accept-Header suffix the URL with an extension ".json" or ".xml" to receive the preferred format
- by the accept header found in the HTTP request
- by default as JSON

Swagger UI

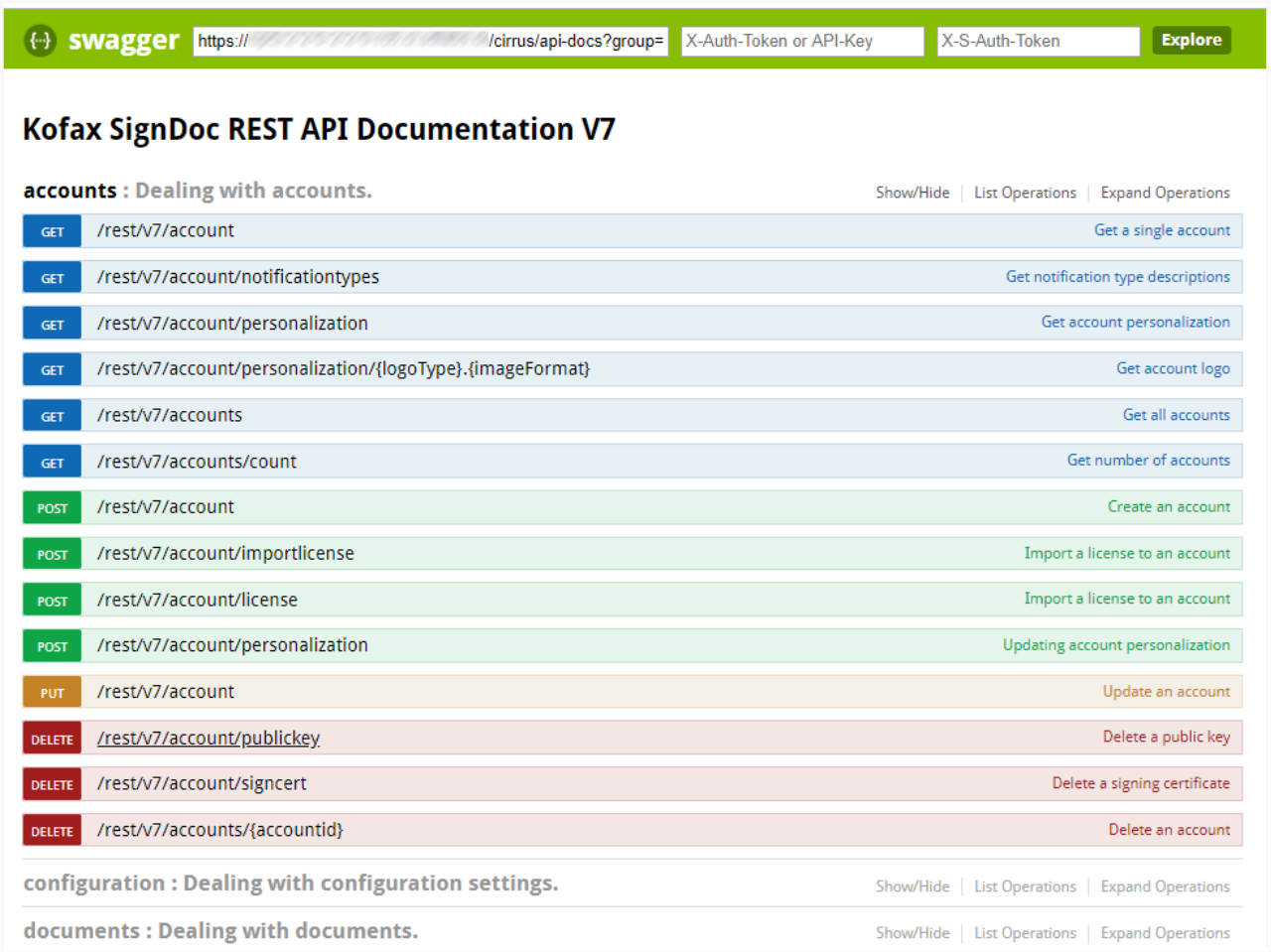
Swagger is a framework for describing, producing, consuming, and visualizing RESTful web services. The overarching goal of Swagger is to document methods, parameters, and models and tightly integrate into the server code. To experiment with the SignDoc Standard REST API a Swagger implementation is provided and can be accessed using the following path:

```
servername:port/cirrus/swagger/
```

Swagger visualizes the REST API and each request on one page and enables the user to easily send a request with all its parameters for test purposes. It additionally provides documentation for each request and its parameters.

The Swagger UI can be enabled by adding "swagger" to the list of active profiles in cirrus.properties:

```
cirrus.spring.profiles.active=swagger
```



User roles and permissions

SignDoc Standard is a role-based application. The functionality and tasks that are performed are determined by roles assigned to the user. SignDoc Standard provides functionality according to the user's role, therefore REST API requests are also depending on the role a user possesses.

The following roles are currently defined for SignDoc Standard:

Role	Enum Value	Description	Authorized Tasks
Server administrator	SUPERUSER	Server administrator is the first account present on a SignDoc Standard distribution to enable a customer to create accounts.	With the server administrator role you can: • Create accounts • Update accounts • Delete accounts • Query accounts
Account administrator	ADMIN	Will initiate the SignDoc Standard sequence to create and edit signing packages. Fulfills administrate roles for users and teams within the account.	Same privileges as user, team member and team manager. The account administrator is allowed to: • Create users associated with the account • Change business settings for the whole account • Create teams
Team manager	TEAMMGR	Assigned team manager for a team. Administrates a team.	The team manager: • Invites users to become a team member
User	USER	Will initialize the SignDoc Standard sequence to create and edit signing packages. This is the basic role to interact and work with signing packages.	The User: • Can manage his own created tasks and signing packages • Can be part of a team • Maintains dashboard

New requests in v7

SignDoc REST API v7

The SignDoc REST API is extended with new functionality and changed input and output structures. Therefore, it is necessary to implement a new SignDoc REST API v7.

Extension of signer information structures

The RestSignerListEntry structure can be returned in the following requests:

- Get a list of packages
  GET /rest/v7/packages
- Get a single signing package
  GET /rest/v7/packages/{packageid}
- Signer search
  GET /rest/v7/signers

The list entry structure contains a subset of signer specific attributes. It was extended with the new attributes esignConsentRequired, gdprConsentRequired and relatedUser.

- esignConsentRequired
  The boolean value indicates whether the e-sign consent must be displayed and agreed by the signer before he can continue signing documents.
- gdprConsentRequired
  The boolean value indicates whether the EU specific GDPR (General Data Protection Regulation) statement must be displayed and agreed by the signer before he can continue signing documents.
- relatedUser
  The string value is an identifier for a SignDoc user which is related to the signer. This relatedUser attribute is available if the SignDoc user was added as a signer to the signing package.

The signer search (GET /rest/v7/signers) includes a filter option "resfield", where you can specify the signer fields you want to have in the result for the found signers. The three new fields esignconsentrequired, gdprconsentrequired and relatedUser can be specified in the comma separated list for field filtering.

According extension of RestSignerListEntry, also the RestSignerInput and RestSignerOutput were extended in v7 by the fields esignConsentRequired, gdprConsentRequired and relatedUser which can be updated and retrieved by SignDoc users with role USER.

Support of new, EU specific GDPR (General Data Protection Regulation) statement

Like the changed e-sign consent handling there is new GDPR text that can be displayed to the signer after authentication. As well as for the e-sign consent the signer must confirm the GDPR text before he can continue with document signing or reviewing.

Here we have also three new configuration settings:

- client.signing.gdpr.text
  The account-specific GDPR text to be displayed in the Signing Client
- client.signing.gdpr.required
  The account-specific flag which controls whether the GDPR text is displayed or not (default is false, because it is only EU specific)
- client.signing.gdpr.url
  The account-specific external URL which is displayed as clickable link in the Signing Client with further GDPR information (optional)

The settings can be viewed by persons which have one of the following roles: USER, SIGNER, ADMIN and SUPERUSER. It can be edited by SignDoc users with either role ADMIN or SUPERUSER.

The account-specific setting client.signing.gdpr.required is the default for all the signers which are assigned to the account-related signing packages. This setting can be overruled by a signer-specific flag "gdprConsentRequired" (RestSignerInput structure) which can be set by a 'normal' SignDoc user (role USER), either in the SignDoc Manage Client or directly in the REST API.

Extension of signature field signing modes

The signature field structures RestSignatureFieldInput and RestSignatureFieldOutput have the attribute signingModeOptions. This element describes which signing methods are allowed for signature capture.

The signing modes which can be set are: HW for handwritten (Tablet or HTML5), PH for photo, captured by camera, C2S for Click2Sign, and now also IMG for sign with (up-)loaded image. The default is HW, PH, C2S, IMG.

Signature image

A new signing method "Signature image" is introduced with Kofax SignDoc Standard 2.2.

For a signer it is now possible to sign a document with a signature image which was loaded on client side during signing session. Once loaded in SignDoc the signature image can be re-used from the signer on the same client machine with the same browser also for other documents and packages as signature without reloading.

This signing method is enabled by default now for all inserted signature fields during package creation. The package creator can also disable the signing method. If it is enabled it can be used by the related signer during signing in a ‘remote’ signing session.

It cannot be selected by a signer in the ‘in-person’ signing session, even if it is enabled for the signature field. An exception here is the signer related SignDoc user (see also "SignDoc user as signer"), which is also a new feature in Kofax SignDoc Standard 2.2.

SignDoc user as signer

Since Kofax SignDoc Standard 2.2 it is possible to associate a SignDoc user with a signer. This is useful if a SignDoc user wants to act also as signer for a signing package. This can be achieved by setting the userid of the SignDoc user in the relatedUser attribute in the RestSignerInput structure during signer creation. The signer’s name and email address are then inherited from the SignDoc user. These attributes cannot be changed afterwards in this case. A signer cannot be associated with a SignDoc user afterwards, if the signer was already created without relatedUser. It is also not possible to remove the relation from a SignDoc user and a signer.

A signer related SignDoc user can use the signing method “Signature image” (if enabled for the related signature field) even in the ‘in-person’ signing session.

With the REST API it is possible to associate also other SignDoc users as signers for a signing session. But, the signing method "Signature image" can be enabled only for one SignDoc user which is associated with a signer. This can be done during preparation of the ‘in-person’ signing session (POST /rest/v7/packages/{packageid}/signingsession/{signtype}). This REST API was extended with the optional attribute ‘userid’. This specifies the userid for which the 'Signature image' signature capture method should be enabled in an 'in-person' signing session. Usually the userid of the package creator is used here.

Retrieving the display name of a locale

A new general system service is implemented in SignDoc REST API v7 which provides the display name of a locale, specified by a language tag. The display language can be specified by another language tag as query parameter.

Example

The request URL is

```
http://localhost:6611/cirrus/rest/v7/locale/fr/info?locale=pt-BR
```

The response body is

```
{
   "languageTag": "fr",
   "displayName": "francês"
}
```

Add new signer attribute preferred language

The new attribute preferred language can contain the language tag of the preferred language of a signer.

This must be IETF BCP 47 language tag string, defined in RFC 5646 "Tags for Identifying Languages", for example

- de (German)
- en-US (English as used in the United States)
- ru-RU (Russian for Russia)

By default email notifications are sent in the account specific communication language (via setting "cirrus.communication.locale"). If the SignDoc user chooses another ‘preferred’ language for a signer then emails are sent in this ‘preferred’ language to the signer. This ‘preferred’ language is also used for the ‘Signing Client’ if the signer starts a signing session. If the ‘preferred’ language is not supported by the ‘Signing Client’ then the browser language is used. If also the browser language is not supported by the ‘Signing Client’ then English is used as fallback.

The new attribute preferredLanguage was added to SignDoc REST beans (v7) RestSignerInput, RestSignerOutput, and RestSignerListEntry.

Language support for document types

The REST interface v7 was extended in order to allow specification of language specific document types.

It is possible to create language specific document type definition.

As of yet it was only possible to specify a document type with a name, a description and the maximum files number, which are allowed for uploading appropriate supplemental documents by the signer.

Now it is possible to create language specific definitions for one document type. You can define another name, description and even a different number of maximum files number (which could be useful for different countries) for another language tag.

The following REST APIs have a new optional locale query parameter which must have an IETF BCP 47 language tag string as value:

- Get available document types
  GET /rest/v7/account/doctypes

- Get a specified document type
  GET /rest/v7/account/doctypes/{doctypeid}
- Delete a specified document type
  DELETE /rest/v7/account/doctypes/{doctypeid}

The RestDocumentType structure has a new attribute locale with the related language tag string.

The following APIs are changed indirectly because of the new attribute locale in RestDocumentType:

- Create a document type
  POST /rest/v7/account/doctype
- Update a specified document type
  PUT /rest/v7/account/doctypes/{doctypeid}

Note The document types so far have got the default language tag ‘en’. The migration to the extended document types includes also the creation of additional definitions for the already existing document types, one for each supported signing client language, which are en,de,fr,nl,it,es,pt-BR,ja. The values for name, description and the maximum files number are unchanged in the copied definitions for customer specific document types, only the language information is different.

Language support for document type instances

The language support in the document types allows also the display of different, language-specific values for name, description and maximum files number.

The GET requests for retrieving document type instances was extended by the new optional query parameters locale and effective=true/false (default: true).

- Get a document type instance for a single signer
  GET /rest/v7/packages/{packageid}/signers/{signerid}/doctypeinstances
- Get single document type instance
  GET /rest/v7/packages/{packageid}/signers/{signerid}/doctypeinstances/{doctypeinstid}

The parameter 'effective' controls whether only the explicit values for name and description from the document type instance are returned (effective=false) or if also the implicit and language specific default values could be returned in the name and description attribute (effective=true, default), if no explicit values are set.

If explicit entered values are available for a document type instance (only possible for type GENERIC) then the locale parameter has no effect, because only the explicit values are returned. The locale parameter is only important if no explicit values are available.

The account specific values for name and description are returned for the specified locale.

If the language for the specified locale is not available, then the predefined default locale 'en' is assumed.

If 'en' is also not available in the account-specific settings then a predefined general text in English, like "Upload supplemental documents" for name and "Please submit copies of supplemental documents to complete information." as description is returned.

Changed requests in v7

Changed e-sign consent handling

The account attribute esignConsent was deleted and the value was moved (via automatic migration) to the appropriate, account specific configuration setting client.signing.esign.consent.text, which can be found under configuration category "Advanced signing setting". Only users with role ADMIN or SUPERUSER can change the account specific e-sign consent text via configuration. The system configuration setting can be changed only by a user with role SUPERUSER.

As consequence the esignConsent attribute is no longer returned in the RestAccountOutput structure and in the RestSigningPackageOutput structure.

This esignConsent of the account object contained the e-sign consent text, which is displayed by default to the signer before he can sign or review a document of a signing package. The signer must agree the e-sign consent before he can continue processing.

It is now possible to configure whether the e-sign consent is displayed for confirmation or not. This can be configured with the advanced signing boolean setting client.signing.esign.consent.required by a user with role SUPERUSER as system default or account specific by a user with role SUPERUSER or ADMIN. The default is true, if nothing is configured.

The account specific value overwrites the system setting, but also the account setting can be overruled by a signer specific setting. This signer specific setting can be set as recipient property during package preparation in the SignDoc Manage Client or via REST interface.

In addition to the e-sign consent text you can configure an e-sign consent URL with the advanced signing setting client.signing.esign.consent.url (system or account specific). If defined, this e-sign consent URL is displayed as link in the SignDoc Signing Client. It can be used for displaying additional information according consent for electronic signing for the recipient. Only users with role ADMIN or SUPERUSER can change the account specific e-sign consent URL via configuration. The system configuration setting can be changed only by a user with role SUPERUSER.

All these settings can be viewed by persons which have one of the following roles: USER, SIGNER, ADMIN and SUPERUSER.

Default mail subject and mail message are moved from user settings to configuration

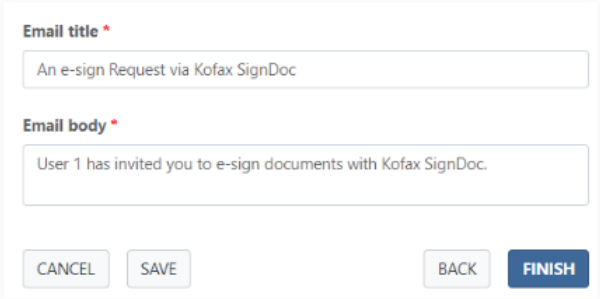
Until SignDoc REST API v6 the following settings were relevant for the entries of email subject and message if a signer or reviewer should be notified with a request for signing (or reviewing) documents in a package:

1. The email subject is configured with the setting mail.message.signing.subject or mail.message.reviewing.subject, dependent from the recipient type. The default value for this setting is %%PACKAGEMAILSUBJECT%%.

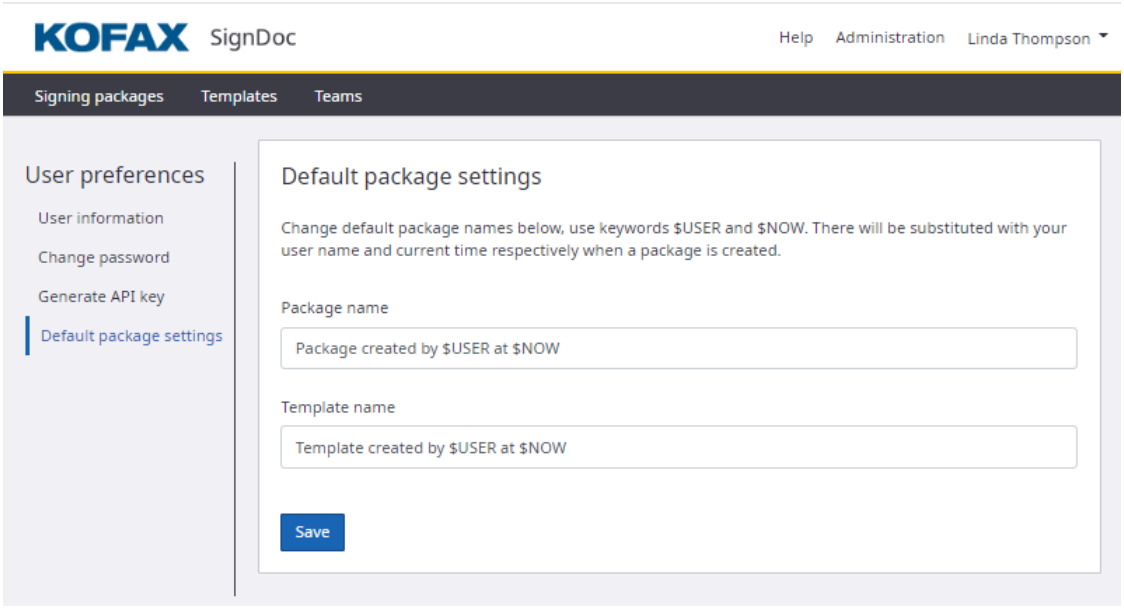
   %%PACKAGEMAILSUBJECT%% is a placeholder which is replaced during mail creation.

   The email body is configured with the setting mail.message.signing.body or mail.message.reviewing.body. The default value for this setting is a HTML string which includes the placeholder %%PACKAGEMAILBODYTEXT%%.

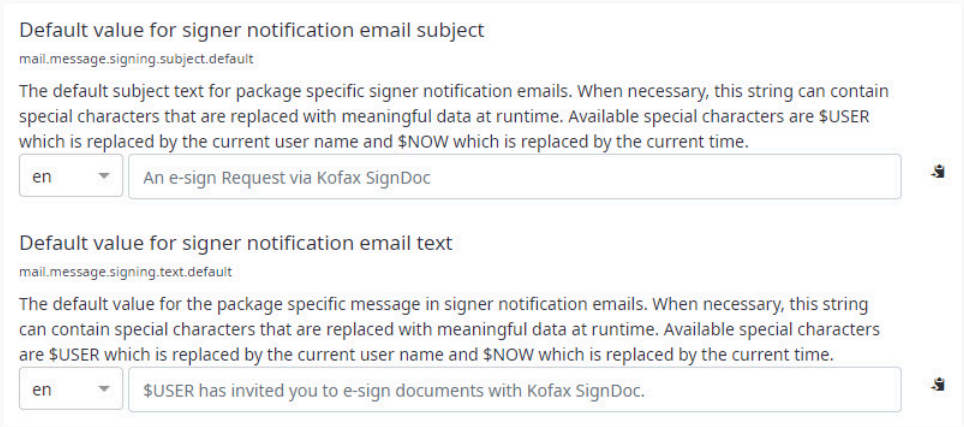
2. The actual value which replaces the placeholder %%PACKAGEMAILSUBJECT%% comes from the package attribute mailSubject. This attribute can be set via REST API or via 'SignDoc Manage Client' in the 'Package settings' dialog in the text entry field with label 'Email title' during package creation or editing. The value for the placeholder %%PACKAGEMAILBODYTEXT%% comes from the package attribute mailMessage. It can be set via REST API or via 'SignDoc Manage Client' in the 'Package settings' dialog in the text entry field 'Email body'.



3. The default text displayed in 'Email title' and 'Email body' comes from the user preferences. These attributes can be set via REST API in RestUserInput / settings / defaultMailSubject and defaultMailMessage or via 'SignDoc Manage Client' under 'Preferences' / 'Default package settings' / 'Email subject' and 'Email message'.



Since SignDoc REST API v7 the default settings for 'Email title' and 'Email body' come from the account specific configuration settings mail.message.signing.subject.default and mail.message.signing.text.default instead of from the user preferences.



These new configuration settings are localizable. The displayed text in the ‘Package settings’ dialog depends therefore on the account specific communication language which can be configured with the setting cirrus.communication.locale.

The subject and the email message for the signer notification are used only from the signing package specific ‘Email title’ and ‘Email body’ if these values are explicitly entered in the ‘Manage Client’ (or REST API directly). If these values are not set explicitly then the default values from the (account specific) configuration settings mail.message.signing.subject.default and mail.message.signing.text.default are used. The locale for these values comes from the signer ‘preferred’ language if set or as fallback from the account specific ‘communication’ language.

The attributes defaultMailSubject and defaultMailMessage of the user preferences are no longer supported in REST API v7. The attributes can be set and retrieved in older REST APIs for compatibility reasons but without impact to the default values of ‘Email title’ and ‘Email body’.

The replaced value for the email placeholder $NOW has been changed from American format "MM/dd/yyyy HH:mm XXX" to ISO 8601 format "yyyy-mm-ddThh:mm:ss.nnnnnn+|-hh:mm", e.g. '2011-12-03T10:15:30+01:00'.

Definition of a field position

Until now (before SignDoc REST API v7) the position of the visible part of a document field was described with the attributes page, posx, posy, width and height.

With SignDoc REST API v7 the position is defined in a RestWidget structure. Theoretically one field can have several widgets with different positions, even on different pages in the document. This is necessary at the latest for radio button groups, which are not yet supported in SignDoc. Currently (with v7) SignDoc supports only one widget per field (index=0).

```
RestWidget {
bottom (number): The bottom coordinate. The origin is in the bottom left corner of the
 page.
index (integer, optional): The index of the widget within the field (is not considered
 if a new field is added).
left (number): The left coordinate. The origin is in the bottom left corner of the
 page.
pageNumber (integer): The index of the page on which this field occurs (1 for the first
 page).
```

```
right (number): The right coordinate. The origin is in the bottom left corner of the
 page
tabIndex (integer, optional): The tab index of the field (page-based output attribute).
top (number): The top coordinate. The origin is in the bottom left corner of the page.
}
```

Example for "Get a single signature field" with SignDoc REST API v6

Request URL

```
http://localhost:6611/cirrus/rest/v6/packages/d0c37e30-
e1dd-45f7-9edf-92adfcc9d6fd/documents/document-1/signaturefields/signature-1
```

Response body

```
{
  "id": "signature-1",
  "name": "3e7c5916-2788-4804-81e1-c1997df5476f",
  "signerId": "signer-1",
  "label": "Signature Field 1",
  "required": false,
  "readOnly": false,
  "page": 1,
  "posx": 100,
  "posy": 300,
  "width": 300,
  "height": 100,
  "state": "PENDING",
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ]
}
```

The same example for "Get a single signature field" with SignDoc REST API v7

Request URL

```
http://localhost:6611/cirrus/rest/v7/packages/d0c37e30-
e1dd-45f7-9edf-92adfcc9d6fd/documents/document-1/signaturefields/signature-1?
resolution=72
```

Response Body

```
{
  "id": "signature-1",
  "name": "3e7c5916-2788-4804-81e1-c1997df5476f",
  "signerId": "signer-1",
  "alternateName": "Signature Field 1",
  "required": false,
  "readOnly": false,
  "signed": false,
  "widgets": [
    {
      "index": 0,
      "pageNumber": 1,
      "top": 400,
      "left": 100,
      "right": 400,
      "bottom": 300
```

```
    }
  ],
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ]
}
```

Extend allowed characters

The id of a resource and the user api key is now allowed to contain a "." character in addition to the existing allowed characters, so the allowed characters now are a-z, A-Z, 0-9, '_', '.' and '-'. This id should not end with a special character and should not contain only special characters.

New requests in v8

Staged implementation

As part of REST API v8 a new signing package processing type 'STAGED' was introduced. Users can create, view and update a signing package in 'STAGED' processing type only from REST API v8 onwards.

The 'STAGED' processing type allows signers to be assigned to a stage, each stage can have multiple signers. The signers are notified as per stages. All signers belonging to one stage are notified at the same time. Once a stage is complete the next stage is notified until the signing package is completed.

All document fields in a 'STAGED' processing type can be assigned to a stage and as well to a signer. The document field can be assigned to either 'stage' or a 'signer' and not both at the same time.

A document field assigned to a stage can be signed by any signer belonging to a particular stage.

For example, if there are two document fields that are assigned to a stage and that particular stage has 3 signers the stage can be marked as complete when only two signers complete their signing process and the stage document fields, the third signer has no mandatory fields assigned and hence is marked as 'STAGED_COMPLETE'.

Similarly if a particular stage has more than one reviewer and all signers have signed, if one reviewer completes the reviewer process then the stage is marked as complete and the second reviewer does not have any pending action.

If the signing package is updated and all the signers of the particular stage are removed or only reviewers are available in a stage then all the document fields for that particular stage are also removed from the document.

If a role of a signer is changed to 'REVIEWER' and there are no signers in the stage with role 'SIGNER' all the document fields assigned to that particular stage are removed from the document.

In the signing session if a document field is assigned to a stage the fields are mandatory only for the signer entering last to sign his signing session, others can decide not to sign it.

Delegate signing session to a new recipient

This feature allows a recipient with role SIGNER to delegate its signing request to a new recipient. It is accessible to both signer and reviewers. Any recipient who has the ‘delegate’ flag set to true is allowed to delegate the signing session to a new recipient else an error is thrown. Delegation is allowed only if the package is in STARTED state. A recipient whose state is not complete and is either in ASSIGNED, INFORMED or ERROR state is allowed to delegate the signing session. When delegating the signing session the request must contain only name, first name, last name, email and delegated message, if any other fields are present in the delegate session request an error is thrown.

A parent recipient with 2F authentication method is not allowed to delegate the signing session. The delegated recipient inherits all its field values from the parent recipient with exceptions for name, first name, last name and email. The parent recipient which has first name set needs to supply first name for the delegated recipient as well, otherwise an error is thrown, similarly the last name. On the other hand the name is mandatory. The parent recipient can supply an email and a delegated message optionally. If an email and a delegated message is supplied the delegated message is prefixed to the actual signing request message body and the new recipient is notified. If no delegated message is provided an email is sent without any changes to the actual message body. It is not allowed to delegate a signing session if a TSP plugin registered to a signer.

Create express package

This new REST endpoint allows a user to create a simple express package for one document and one signer. It returns a common signing session URL in the response along with resource URL and id. If the signer email URL is specified, a notification with a link to the remote signing session is sent to the signer.

Changed requests in v8

The changes to REST API v7 also apply to REST API v8. See Changed requests in v7 for more details.

Overview of breaking API changes

The following table lists the breaking changes of the REST API for the SignDoc versions.

Since SignDoc version	Change	Links
2.1.0.1	Changing the email address of a user or server administrator requires the password of the authenticated user.	Update a user Update a server administrator

Since SignDoc version	Change	Links
2.2.0	The REST API v2 is discontinued with the following exception: • Create a new signing package The REST API v3 is discontinued with the following exceptions: • Add document to package • Create a new signing package • Schedule a signing package • Delete a signing package **Note** We recommend to use the REST API v7 as substitute for all v2 and v3 requests The field position attributes are moved from the field definition to the RestWidget bean. The following APIs are affected by this change: • Create/update signing package/template • Create/update/get document • Create/update/get field	
3.0.0	A user can create a new package with processing type 'STAGED' from REST API v8 onwards. This is not allowed for earlier API versions. A package created using 'STAGED' processing type can be viewed only using REST API v8 version, older versions will throw an error for a single GET request using the package id and will exclude it from the final response if all the packages are requested.	
3.0.0	A user can now save a binary configuration along with a password for type 'ENCRYPTED' and subtype 'PKCS12'. This is mostly a case for #PKCS12 certificates. A new optional query parameter of type string was added to POST /rest/v8/configuration/binary/{configid} from REST API v8 onwards. If a user is adding a configuration with type 'ENCRYPTED' and subtype 'PKCS12', the query parameter 'confdata_password' is mandatory otherwise it is optional.	
3.0.0	The id of a resource and the user api key is now allowed to contain a "." character in addition to the existing allowed characters, so the allowed characters now are a-z, A-Z, 0-9, '_', '.' and '-'. This id should not end with a special character and should not contain only special characters. This change is effective in v8 and all older api versions as well.	

Since SignDoc version	Change	Links
3.0.0	A new signing method "Sign with a stamp" is introduced as part of 3.0 release. This new signing method is similar to the existing signing method "Sign with image". The signer can upload a (stamp) image from his computer to sign a signature field with a (stamp) image. The stamp image is cached in local storage of the browser so that the signer can re-use it for other signature fields.	
3.0.0	As part of SignDoc 3.0 release API version v8, account and user will not contain the timezone field. The "Get time zones" API from system resource http://*host_server:port_number*/cirrus/rest/v8/system/timezones is also removed from v8 API.	
3.0.0	A new signing method ‘TSP’ is introduced as part of release 3.0.0. The signer can now sign a signature field using ‘TSP’ using a TSP plugin that supports the same. When user attempts to sign a signature field using this mode in the signing client, the signer is taken to an external ‘TSP’ service to complete the signing process. Depending on the outcome of the ‘TSP’ signing session, the signer will have relevant message displayed on the screen. The signature returned from the ‘TSP’ service is applied to the field on success.	
3.0.0.1	Rate limiting access to the rest endpoints is introduced as part of this release. An administrator can control the number of rest access allowed per authentication token in a given time frame using pre defined configuration options.	
3.0.0.1	From SignDoc 3.0.0.1 release, older api’s v2, v3, v5 are no longer supported.	

Chapter 2

Version-independent REST API

This API does not need an Authorization Token or an API key and is not bound to a REST API version.

API requests

Get the latest REST API version

Returns the latest (i.e. current) API version of the REST API along with the REST API's Base URL.

Note For this request no authentication is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/system/version/rest

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/system/version/rest
```

Example request headers

```
Accept: application/json
```

Response class

Status is 200 (OK): The latest REST API version was queried successfully.

RestID

- **commonSigningUrl** (string, optional): The URL to access the common signing session of the created resource

- **id** (string): The id of the created resource
- **messages** (RestMsgList, optional): Any informational messages
- **url** (string): The URL to access the created resource

RestMsgList

- **list** (Array[RestMsg], optional): List of messages (info, warnings, errors)

RestMsg

- **code** (integer): Message code
- **message** (string): Description
- **type** (string): Message type. Possible values: ERROR, WARNING, INFO

Example response body (JSON)

```
{
"id": "v1",
"url": "https://xxx/cirrus/rest/v1"
}
```

Get all active APIs

This request returns a list of all a available APIs. This enables the client to find out if a particular API is available or not.

URL

http://*host_server:port_number*/cirrus/rest/api/all

Produces

application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/api/all
```

Example request headers

```
Accept: application/json
```

Example response body (JSON)

```
{
  "v2": http://localhost:6611/cirrus/rest/V2,
  "v3": http://localhost:6611/cirrus/rest/v3,
  "v5": http://localhost:6611/cirrus/rest/v5,
  "v6": http://localhost:6611/cirrus/rest/v6,
  "v7": http://localhost:6611/cirrus/rest/v7,
  "v8": http://localhost:6611/cirrus/rest/v8
}
```

Chapter 3

REST API reference v8

Account requests

Get a single account

This request returns a single account specified by a given account id. By default a flat account object with only essential information is returned. The accountFilter parameter allows to customize the response body of the request. Depending on a comma separated list of filter options the response body can return additional information.

Note For this request a valid authentication with a USER, ADMIN or SUPERUSER role is necessary. If the requesting user has only role USER then only account details and teams in which the user is a member can be retrieved. Users cannot be retrieved with only role USER.

URL

http://*host_server:port_number*/cirrus/rest/v8/account

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Example response body (JSON)

```
{
 "id": "account",
 "name": "account",
 "company": "account",
 "state": "ACTIVE",
 "notificationsEnabled": true,
 "publicKeyInfo": {
 "algorithm": "RSA",
 "format": "X.509",
 "bitLength": 2048
 },
 "signingCertificateInfo": {
 "type": "X.509",
 "version": 3,
 "subject": "EMAILADDRESS=MyEmail@MyOrganisation.org, CN=MyName, OU=MyOrganizationUnit,
 O=MyOrganisation, L=MyCity, ST=MyLand, C=MyCountry",
 "validityDateNotBefore": "2015-08-28T13:58:52Z",
 "validityDateNotAfter": "2017-03-29T01:02:20Z"
 },
  "users": [
    {
      "id": "user1",
      "name": "user1",
      "email": "user1@ksd.com",
      "state": "ACTIVE",
      "url": "http://localhost:6611/cirrus/rest/v8/users/user1",
      "roles": [
        "USER",
        "TEAMMGR",
        "ADMIN"
      ],
  "teams": [
    {
      "id": "team1",
      "name": "Team_1 of Tenant_1 2015/10/21 11:47:29",
      "url": "http://localhost:6611/cirrus/rest/v8/teams/5cade8f8-
f8e3-40a8-93a1-0560a6e2fdbe"
    },
  "creationTime": 1445325618798,
  "lastUpdateTime": 1446542631245,
  "licenseInfo": {
    "expiryDate": 1451558150004,
    "licensedUsers": 10,
    "currentUsers": 7,
    "licensedPackages": 100,
    "processedPackages": 5,
    "licensedDocuments": 100,
    "processedDocuments": 5,
    "accountInformation": "some tenant-specific information",
    "state": "VALID"
  },
  "url": "http://localhost:6611/cirrus/rest/v8/account?accountid=account3"
}
```

Query parameters

- **accountid** (string, optional): The account id. This parameter is required if the user is not bound to an account. This is the case for users with the role SUPERUSER.

- **accountFilter** (Array[string]), optional]: The account filter option. Possible values: USERS, TEAMS, NONE
- **useIntId** (boolean, optional): The provided accountId is the internal id which was returned by get audit trail method. Default value: false.

Response class

Status 200 (OK): The account was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAccountOutput

- **company** (string, optional): The company of the account
- **contactInformation** (string, optional): The contact information of the account.
- **creationTime** (string, optional): The creation time of the account. Format: date-time with a time-zone in UTC, such as '2020-12-3T10:15:30Z' (ISO-8601)
- **dnsLabel** (string, optional): The DNS label of the account.
- **id** (string, optional): The account id
- **lastUpdateTime** (string, optional): The last update time of the account. Format: date-time with a time-zone in UTC, such as '2020-12-3T10:15:30Z' (ISO-8601)
- **licenseInfo** (RestAccountLicenseOutput, optional): The current account license information
- **name** (string, optional): The name of the account
- **notificationsEnabled** (boolean, optional): Whether notifications are send or not
- **publicKeyInfo** (RestPublicKeyInfo, optional): The public key information for the account
- **signingCertificateInfo** (RestSigningCertificateInfo, optional): The signing certificate information for the account
- **state** (string, optional): The state of the account. Possible values: INACTIVE, ACTIVE
- **teams** (Array[RestTeamListEntry], optional): A list of all teams of the account
- **url** (string, optional): The URL to query the account
- **users** (Array[RestUserListEntry], optional): A list of all users of the account

RestAccountLicenseOutput

- **accountInformation** (string, optional): Account information
- **currentUsers** (integer, optional): Current number of users
- **expiryDate** (string, optional): License expiry date. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **licensedPackages** (integer, optional): Number of licensed signing packages
- **licensedUsers** (integer, optional): Number of licensed users
- **processedPackages** (integer, optional): Number of processed signing packages
- **renewableLicensePeriod** (string, optional): The renewal period of the license (if renewable license). Possible values: MONTHLY, YEARLY
- **state** (string, optional): The state of the license. Possible values: VALID, MAXIMUM_REACHED, EXPIRED, NO_LICENSE, INVALID

RestPublicKeyInfo

- **algorithm** (string, optional): The algorithm of the public key
- **bitLength** (integer, optional): The bit length of the public key. This value is available if the public key algorithm is RSA
- **format** (string, optional): The format of the public key

RestSigningCertificateInfo

- **subject** (string, optional): The subject (subject distinguished name) value from the certificate as an X500Principal, for example EMAILADDRESS=info@company.de, CN=SignDoc, O=Company, L=Boeblingen, ST=BW, C=DE
- **type** (string, optional): The type of the signing certificate
- **validityDateNotAfter** (string, optional): The notAfter date from the validity period of the certificate. Format: datetime with a time-zone in UTC (ISO-8601)
- **validityDateNotBefore** (string, optional): The notBefore date from the validity period of the certificate. Format: datetime with a time-zone in UTC (ISO-8601)
- **version** (integer, optional): The version of the signing certificate

RestTeamListEntry

- **id** (string, optional): The id of the team
- **name** (string, optional): The name of the team
- **teamManagers** (Array[string], optional): A list of all team managers the team possesses
- **teamMembers** (Array[string], optional): A list of all team members the user possesses
- **url** (string, optional): The request URL to query the complete team

RestUserListEntry

- **email** (string, optional): The email of the user
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-3T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **url** (string, optional): The request URL to query the complete user

Get notification type descriptions

This request retrieves notification type descriptions of loaded plugins which support NotificationParametersEvent.

Notification types can be retrieved for globally enabled plugins (without specifying an account id), or for account specific plugins (by specifying the account id). Users with the role USER or ADMIN can only query their own account id, while users with the role SUPERUSER can query the setting for any account id.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/notificationtypes

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/notificationtypes
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **accountid** (string, optional): The account id. If omitted, notification type descriptions for plugins that are enabled account independent are returned.
- **languagetag** (string, optional): Language tag for retrieving the locale-specific configuration setting description (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en-US will be used.

Response class

Status 200 (OK): The notification type descriptions were successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestNotificationTypeDescription

- **maxMessageSize** (integer, optional): The maximum size of the notification message
- **notificationType** (string, optional): The notification type (e.g. SMS)
- **pluginId** (string, optional): The notification plugin id
- **targetParameterList** (Array[RestNotificationTargetParameterDescription], optional): The notification target parameters (e.g. phone number for SMS)

RestNotificationTargetParameterDescription

- **description** (string, optional): The description of the parameter. This description will be displayed by the GUI when the user enters a notification target,
- **helpText** (string, optional): The help text to be displayed as a tool tip for the parameter

- **name** (string, optional): The name of the parameter
- **placeholder** (string, optional): Placeholder to be displayed in the parameter field, can be null
- **validationRegExp** (string, optional): The Java regular expression used by the GUI to validate the parameter value entry

Get account personalization

This request returns the account-specific or the default personalization data with client layout information.

Note For this request no authentication is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/personalization

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/personalization
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **accountid** (string, optional): The account id. Either accountid or the query parameter usedefaultaccount must be specified for retrieving account specific personalization data.
- **defaultvalues** (boolean, optional): With defaultvalues=true the request returns the initial default values for the account independent personalization data. The parameters accountid and usedefaultaccount are ignored in this case. Default value: false
- **usedefaultaccount** (boolean, optional): If only one account is available in the current installation this parameter can be set to true in order get the personalization data from this account without to specify the accountid. Default value: false

Response class

Status 200 (OK): The account was successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages. In case of a successful query the response body contains the following information regarding the requested personalization object (RestPersonalization).

RestPersonalization

- **footerBackground** (string, optional): The footer background color as RGB hex value
- **footerForeground** (string, optional): The footer foreground color as RGB hex value
- **footerLinks** (Array[RestFooterLink], optional): A list with footer links
- **headerBackground** (string, optional): The header background color as RGB hex value, e.g. f1f1f1
- **headerForeground** (string, optional): The header foreground color as RGB hex value
- **headerLogoType** (string, optional): The login logo image type
- **headerLogoUrl** (string, optional): The header logo URL
- **headingsTitleForeground** (string, optional): The heading title foreground color as RGB hex value
- **loginLogoType** (string, optional): The login logo image type
- **loginLogoUrl** (string, optional): The login logo URL

RestFooterLink

- **url** (string, optional): The URL which can be selected in the SignDoc Standard Client footer
- **urlText** (string, optional): The displayed text for the selectable footer link

Get account logo

This request returns the account logo. This request uses server caching with entity tags, as described here: https://www.w3.org/2005/MWI/BPWG/techs/CachingWithETag.html

This request extends retrieving personalization data with the possibility to load the actual logo images which are referenced in the headerLogoUrl and the logonLogoUrl in the response from Get account personalization.

Note For this request no authentication is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/personalization/{logotype}.{imageFormat}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

application/octet-stream, JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/personalization/loginlogo.png
```

Query parameters

- **accountid** (string, optional): The account id. Either accountid or the query parameter usedefaultaccount must be specified for retrieving an account specific logo.
- **usedefaultaccount** (boolean, optional): If only one account is available in the current installation this parameter can be set to true in order to get the logo image for this account without specifying the accountid. Default value: false
- **defaultimage** (boolean, optional): With defaultimage=true the request returns the initial, account independent default image for the requested logo type. The parameters accountid and usedefaultaccount are ignored in this case. Default value: false

Path parameters

- **logoType** (string, required): The logo type. Possible values: loginlogo, headerlogo
- **imageFormat** (string, required): The logo image format suffix. Possible values: png, jpg, bmp, gif. The requested image format suffix must match the actual stored image type for the requested logo type. The actual request URL can be requested with the Get account personalization request (response attributes headerLogoUrl and loginLogoUrl). Example: png

Response class

Status 200 (OK): The logo could be retrieved for download, otherwise a SignDoc Standard status code is returned together with the explaining messages. The response body returns the binary content of the logo.

Get status information of account entities

This request retrieves status information of the following account entities: SIGNING_CERTIFICATE, BIOMETRIC_KEY, S/MIME_CERTIFICATE, SMTP_CONNECTION. The SIGNING_CERTIFICATE is tested for validity, usability and expiry dates. The BIOMETRIC_KEY is tested for validity and usability. The S/MIME_CERTIFICATE is tested for validity, usability and expiry dates. The SMTP_CONNECTION is tested for a valid configuration and if it can connect to the configured SMTP Service.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/status

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/status
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **accountid** (string, optional): The account id. A user with role SUPER can provide the account id in order to get also account specific notification types (enabled) in addition. He gets only the account independent notification type descriptions (enabled) if the account id is omitted.
- **locale** (string, optional): ETF BCP 47 language tag. If value is unknown or invalid, English will be used.

Response class

Status 200 (OK): The account entities were successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestEntityStatus

- **id** (string, optional): The id of the information. Possible values: SIGNING_CERTIFICATE, SMTP_CONNECTION, BIOMETRIC_KEY, SMIME_CERTIFICATE
- **statusClass** (string, optional): The status class of the information. Possible values: INFO, OK, WARN, PROBLEM
- **statusClassString** (string, optional): The localized description of the status class value.
- **statusCode** (string, optional): The status code of the information: Possible values: SMTP_ERROR, EXPIRED, NOT_YET_VALID, EXPIRES_SOON, NOT_SET, IS_VALID, INVALID, GENERAL_ERROR, MESSAGE
- **statusCodeDescription** (Array[string], optional): The localized description of the status code value. Consists usually of multiple lines.
- **statusIdString** (string, optional): The localized description of the entity name
- **timestamp** (string, optional): Timestamp in ISO format

Get all accounts

This request returns all accounts of the system. The resulting list contains only necessary information for further queries sorted by last update time descending. For a complete data set use the Get a single account request (/rest/account/{id}).

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/accounts

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/accounts
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)


```
[
  {
    "id": "account1",
    "name": " account1",
    "company": " account1",
    "state": "ACTIVE",
    "licenseInfo": {
      "expiryDate": 1451550974321,
      "licensedUsers": 10,
      "currentUsers": 8,
      "licensedPackages": 100,
      "processedPackages": 5,
      "licensedDocuments": 100,
      "processedDocuments": 5,
      "accountInformation": "some tenant-specific information",
      "state": "VALID"
    },
    "url": "http://localhost:6611/cirrus/rest/v8/account?accountid= account1"
  },
  {
    "id": " account2",
    "name": " account2",
    "state": "ACTIVE",
    "licenseInfo": {
      "expiryDate": 1451550975192,
      "licensedUsers": 100,
      "currentUsers": 1,
      "processedPackages": 0,
      "processedDocuments": 0,
      "accountInformation": "Enter customer number",
      "state": "VALID"
    },
```

```
    "url": "http://localhost:6611/cirrus/rest/v8/account?accountid= account2"
  }
]
```

Response class

Status 200 (OK): The accounts were successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAccountListEntry

- **company** (string, optional): The company of the account
- **id** (string, optional): The id of the account
- **licenseInfo** (RestAccountLicenseOutput, optional): The license information of the account
- **name** (string, optional): The name of the account
- **state** (string, optional): The state of the account. Possible values: INACTIVE, ACTIVE
- **url** (string, optional): The URL of the account

RestAccountLicenseOutput

- **accountInformation** (string, optional): Account information
- **currentUsers** (integer, optional): Current number of users
- **expiryDate** (string, optional): License expiry date. Format: date-time with a time-zone in UTC, such as '2007-12- 03T10:15:30Z' (ISO-8601)
- **licensedPackages** (integer, optional): Number of licensed signing packages
- **licensedUsers** (integer, optional): Number of licensed users
- **processedPackages** (integer, optional): Number of processed signing packages
- **renewableLicensePeriod** (string, optional): The renewal period of the license (if renewable license). Possible values: MONTHLY, YEARLY
- **state** (string, optional): The state of the license. Possible values: VALID, MAXIMUM_REACHED, EXPIRED, NO_LICENSE, INVALID

Get number of accounts

This request returns the number of accounts.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/accounts/count

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/accounts/count
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
{
  "count": 2
}
```

Response class

Status 200 (OK): The number of accounts was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestCount

- **count** (integer, optional): The result of request

Create an account

To create an account, an input JSON or XML string specifying the details of the account has to be provided as a request parameter. The specification has to provide at least an account name with more than two characters and a valid license.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/account
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id": "account1",
    "name": "KSD default users account",
    "license": "base64 encoded license",
    "users": [{
        "id": "user1",
        "roles": ["ADMIN"],
        "name": "user1",
        "email": "user1@ksd.com",
        "password": "Password1!"
   }]
}
```

Body parameters

- **newAccount** (RestAccountInput, required): Represents a RestAccountInput object either in JSON or XML.

RestAccountInput

- **company** (string, optional): Company of the account
- **contactInformation** (string, optional): The contact information of account
- **dnsLabel** (string, optional): The DNS label of account
- **id** (string, optional): The id of the account
- **license** (string, optional): The current license for the account
- **name** (string, optional): Name of the account
- **notificationsEnabled** (boolean, optional): Whether notifications are send or not
- **publicKey** (string, optional): The public key of account
- **signingCertificate** (string, optional): The signing certificate of account
- **signingCertificatePassword** (string, optional): The password of the signing certificate
- **state** (string, optional): The state of the account. Possible values: INACTIVE, ACTIVE
- **teams** (Array[RestTeamInput], optional): A list of all teams of the account
- **users** (Array[RestUserInput], optional): A list of all users of the account

RestTeamInput

- **id** (string, optional): The id of the team
- **managers** (Array[string], optional): A list of all team managers specified by id or email
- **members** (Array[string], optional): A list of all team members specified by id or email
- **name** (string, optional): The name of the team

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a user's email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **name** (string, optional): The name of the user
- **password** (string, optional): [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

Response class

Status 201 (Created): The account was successfully created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestID

- **id** (string): The id of the created resource
- **messages** (RestMsgList, optional): Any informational messages
- **url** (string): The URL to access the created resource

RestMsgList

- **list** (Array[RestMsg], optional): List of messages (info, warnings, errors)

RestMsg

- **code** (integer): code
- **message** (string): description
- **type** (string): Message type. Possible values: ERROR, WARNING, INFO

Import a license to an account

This request imports a license to an existing account identified by id.

Note For this request a valid authentication with ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/license

or

http://*host_server:port_number*/cirrus/rest/v8/account/importlicense

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/account/license
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Body parameters

- **restLicense** (RestAccountLicenseInput, required): Input XML or JSON string of account license

RestAccountLicenseInput

- **accountId** (string, optional): The account id. This parameter can be omitted if the user is bound to an account.
- **license** (string): Account license as base64 encoded string

Response class

Status 201 (Created): The license was successfully imported. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAccountLicenseOutput

- **accountInformation** (string, optional): Account information
- **currentUsers** (integer, optional): Current number of users
- **expiryDate** (string, optional): License expiry date. Format: date-time with a time-zone in UTC, such as '2020-12- 03T10:15:30Z' (ISO-8601)
- **licensedPackages** (integer, optional): Number of licensed signing packages
- **licensedUsers** (integer, optional): Number of licensed users
- **processedPackages** (integer, optional): Number of processed signing packages
- **renewableLicensePeriod** (string, optional): The renewal period of the license (if renewable license). Possible values: MONTHLY, YEARLY
- **state** (string, optional): The state of the license Possible values: VALID, MAXIMUM_REACHED, EXPIRED, NO_LICENSE, INVALID

Update account personalization

This request updates account-specific personalization information.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/personalization

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/account/personalization
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Form parameters

- **accountid** (string, required): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **headerLogo** (file, optional): The Manage Client header logo. This can be an image file from type png, jpg, gif or bmp with a maximum size of 740000 bytes.
- **loginLogo** (file, optional): The Manage Client login logo. This can be an image file from type png, jpg, gif or bmp with a maximum size of 740000 bytes.
- **headingsTitleForeground** (string, optional): The header RGB headings title foreground color as RGB hex value, e.g. 31708f
- **loginLogoType** (string, optional): Login logo type as an image media type. Example: image/png
- **headerBackground** (string, optional): The header background color as RGB hex value, e.g. 0079C1
- **headerForeground** (string, optional): The header foreground color as RGB hex value, e.g. ffffff
- **footerBackground** (string, optional): The footer background color as RGB hex value, e.g. f5f5f5
- **footerForeground** (string, optional): The footer foreground color as RGB hex value, e.g. 3D6897
- **footerLinks** (Array[string], optional) The footer links. One or more string pairs which define each a link. The first part of the string pair is the URL text, the second part is the URL. The strings are separated with commas. Example for two links: Contact Us here, http://www.kofax.com/contact/contact-kofax or here, http://www.kofax.com/contact/alternate-contact-kofax

Response status

Status 200 (OK): The account personalization was successfully updated, otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update an account

This request updates an existing account identified by id with a given input JSON or XML string. Each attribute of the existing account can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled. If an update request was successful, and the account state equals SENDCONF a separate confirmation message will be sent.

Note For this request a valid authentication with ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/account
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id": "account1",
    "name": "KSD default users account",
    "license": "base64 encoded license",
    "users": [{
        "id": "user1",
        "roles": ["ADMIN"],
        "name": "user1",
        "email": "user1@ksd.com",
        "password": "Password1!"
   }]
}
```

Query parameters

- **accountid** (string, required): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.

Body parameters

- **newAccount** (RestAccountInput, required): Input JSON or XML string of account

RestAccountInput

- **company** (string, optional): Company of the account
- **contactInformation** (string, optional): The contact information of account
- **dnsLabel** (string, optional): The DNS label of account
- **id** (string, optional): The id of the account
- **license** (string, optional): The current license for the account
- **name** (string, optional): Name of the account
- **notificationsEnabled** (boolean, optional): Whether notifications are send or not
- **publicKey** (string, optional): The public key of account
- **signingCertificate** (string, optional): The signing certificate of account
- **signingCertificatePassword** (string, optional): The password of the signing certificate
- **state** (string, optional): The state of the account. Possible values: INACTIVE, ACTIVE
- **teams** (Array[RestTeamInput], optional): A list of all teams of the account
- **users** (Array[RestUserInput], optional): A list of all users of the account

RestTeamInput

- **id** (string, optional): The id of the team
- **managers** (Array[string], optional): A list of all team managers specified by id or email
- **members** (Array[string], optional): A list of all team members specified by id or email
- **name** (string, optional): The name of the team

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a user's email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user, name (string, optional): The name of the user
- **password** (string, optional): [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

Response class

Status 200 (OK): The account was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestID

- **id** (string): The id of the created resource
- **messages** (RestMsgList, optional): Any informational messages
- **url** (string): The URL to access the created resource

RestMsgList

- **list** (Array[RestMsg], optional): List of messages (info, warnings, errors)

RestMsg

- **code** (integer): Message code
- **message** (string): Description
- **type** (string): Message type. Possible values: ERROR, WARNING, INFO

Delete a public key

This request deletes the existing public key of an account specified by id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/publickey

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/account/publickey
```

Example header

```
Accept: application/json
```

Query parameters

- **accountid** (string, optional): The account id. This parameter is required if the user is not bound to an account. This is the case for users with the role SUPERUSER.

Response status

Status 200 (OK): The public key was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a signing certificate

This request deletes the existing signing certificate and associated password of an account specified by id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/signcert

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/account/signcert
```

Example header

```
Accept: application/json
```

Query parameters

- **accountid** (string, optional): The account id. This parameter is required if the user is not bound to an account. This is the case for users with the role SUPERUSER.

Response status

Status 200 (OK): The signing certificate was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete an account

This request deletes the existing account specified by id.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/rest/v8/accounts/{accountid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/accounts/account1
```

Example header

```
Accept: application/json
```

Path parameters

- **accountid**: (string, required): The id of the account. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

Response status

Status 200 OK: The account was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Configuration requests

Get configuration settings

Global system settings and also account-specific settings can be retrieved as server administrator, as account administrator, as user or as signing session (depending from the necessary authorization in the appropriate setting description).

```
GET http(s)://domainOrIPAddress:port/context/rest/v8/configuration/[?]
[accountid={accountid}][&][startswith={prefix}][&][endswith={suffix}][&]
[cache={true|false}][&]effective={true|false}][&]language={languagetag}]
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is requested. Users with role SUPERUSER can request any account and the global value. Users with role ADMIN can only request their own account, or null, for the global value
- **startswith** (Array[string], optional): The prefix of the requested configuration keys. Multiple startswith parameters can be specified.
- **endswith** (Array[string], optional): The suffix of the requested configuration keys. Multiple endswith parameters can be specified.
- **cache** (boolean, optional): Indicator whether configuration data should be retrieved from the cache.

- **effective** (boolean, optional): Indicator whether effective configuration data should be retrieved (default) or only request specific. Without effective flag you retrieve only the configuration value which was set for the specific account the user belongs to or in case of a SUPERUSER the global, account independent value if the request does not include an account identifier. With effective flag equals true the request returns the first available setting in the following sequence: The account specific value (if available in the request) then the globally (account independent) value and last the default value if it is specified in the configuration description.
- **language** (string, optional): Locale for retrieving the locale-specific audit logs (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47. If omitted, default language tag 'en' will be used.

Get all configuration settings starting with the value of {prefix} and/or ending with the value of {suffix}. Omitting the startswith and endswith query parameters returns all configuration settings. Prerequisite for retrieving configuration values is that the requesting user must have the (role) permission to view the requested configuration keys.

The {prefix} (also {suffix}) can contain the complete configuration key or the prefix of any keys, for example ‘plugins’. In this case all configuration settings (which the caller is allowed to read) whose key start with ‘plugins’ are returned.

A RestConfiguration object with an empty list means that either no configuration setting exists for the requested key or that the requesting user does not have the permission to read the value(s).

Omitting the startswith and endswith query parameters returns all configuration settings for which the user is allowed to read.

The configuration settings are stored in a local application server cache which is refreshed (if necessary after an update) after a short time period.

The optional cache parameter should be set on ‘true’ if it is sufficient to retrieve the configuration settings from this cache which is not guaranteed to have the latest changes immediately available. The usage of the cache prevents unnecessary database queries. The cache parameter should be set to ‘false’ (default) if the settings must be necessarily up-to-date, for example for editing purposes.

The effective flag indicates whether an effective value should be retrieved, or strictly what has been requested. Configuration values are available on three levels: an account specific value, a global (system wide) value and a default value (if no configuration has been explicitly set). An account specific value is requested by specifying the account id. If no account id is specified, the global (system wide) value will be returned. If effective=false is passed with the request, the exact value requested will be returned. This is useful if editing the values in a configuration editor. With effective=true, the request will return the next available value from the hierarchy. If you request an account specific value, the system will return the account specific setting, if available. If not, the global value will be returned. If that is not available, the default value will be returned instead. This setting is usually the most common for actually using a configuration value.

The account is determined by the account assignment of the requesting user. In case of a requesting server administrator the accountid can be specified in the query parameter accountid if an account-specific setting is needed.

The request body contains RestConfiguration with a list of RestConfigEntry objects.

A RestConfigEntry has a key "k" and a value "v".

```
RestConfiguration {
```

```
  list (Array[RestConfigEntry], optional):
  List of configuration settings (as key-value pairs)
}
RestConfigEntry {
  k (string): The key of the configuration entry ,
  v (string): The value of the configuration entry
}
```

Example (JSON output)

```
GET .../rest/v8/configuration?endswith=loadlist&cache=true&effective=true
{
  "list": [
    {
      "k": "plugin.loadlist",
      "v": "de.softpro.cirrus.plugins.notification.NotificationSMSClickatell"
    }
  ]
}
```

In case of an error a RestMsgList object is returned (see declaration above).

Error conditions:

Reason	Error Code
Server administrator provides an invalid accountid	4001
At least one of the requested configuration settings could not be read	37

Example (JSON output)

```
{
  "list": [
    {
      "code": 4001,
      "type": "ERROR",
      "message": "The requested account does not exist"
    }
  ]
}
```

Get a binary configuration

This request returns a binary configuration.

URL

http://*host_server:port_number*/cirrus/rest/v8/configuration/binary/{configid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **configid** (string, required): The id of the setting.

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is updated. Users with role SUPERUSER can update any account, or null for the global value. Users with role ADMIN can only update their own account.
- **effective** (boolean, optional): Indicator whether effective configuration data should be retrieved (default) or only account specific. Without effective flag you retrieve only the configuration value which was set for the specific account the user belongs to or in case of a SUPERUSER the global, account independent value if the request does not include an account identifier. With effective flag equals true the request returns the first available setting in the following sequence: The account specific value (if available in the request) then the globally (account independent) value and last the default value if it is specified in the configuration description. Default value: true
- **locale** (string, optional): Requests locale specific configuration data. Format must be compatible with Locale.languageTag.
- **includeoriginalcontenttype** (boolean, optional): Includes the original content type of the binary data in the response.

Response status

Status 200 (OK): The binary configuration was successfully returned.

Get configuration setting descriptions

A user can retrieve configuration setting descriptions if he has the (role) permission to view or edit the appropriate configuration setting.

```
GET http(s)://domainOrIPAddress:port/context/rest/v8/configuration/
descriptions[?][accountid={accountid}][&][startswith={prefix}][&]
[endswith={suffix}][&][locale={locale}]
```

Query parameters

- **accountid** (string, optional): The account id. This parameter is used to determine if global or account specific descriptions are to be used. It is currently not validated or used to refer to a specific account.
- **startswith** (string, optional): The prefix of the requested configuration keys
- **endswith** (string, optional): The suffix of the requested configuration keys
- **locale** (string, optional): Locale for retrieving the locale-specific configuration setting description (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en-US will be used.

Get descriptions of all configuration settings starting with the value of {prefix} and/or ending with the value of {suffix}. Omitting the startswith and endswith query parameters returns all configuration descriptions. Multiple startswith and endswith parameters can be specified.

An account based request returns only descriptions which can have account-specific configuration values. The account is determined by the account assignment of the requesting user. In case of a requesting server administrator any existing account id can be specified in the query parameter accountid if descriptions of configurations are needed which can be set account-specific. If the accountid is omitted (by a server administrator) the request returns additionally configuration descriptions of settings which cannot be set for a specific account.

The parameter locale specifies the language (if available) for the description text (IETF BCP 47 language tag). If omitted, en-US will be used.

Prerequisite for retrieving a configuration description is that the requesting user (role) must be enabled to edit the appropriate configuration setting.

Returns a list of RestConfigurationDescription objects.

```
RestConfigDescriptions{
   list (Array [RestConfigurationDescription], optional): List of configuration
 descriptions
}

RestConfigurationDescription{
   accountSpecific (boolean, optional): Specifies whether the configuration setting can
 have account-specific values,
   defaultValue (string, optional): The default value of the configuration setting,
   description (string, optional): The description of the configuration setting,
   editRoles (integer, optional): Defines the roles which are allowed to edit the
 setting,
   id (string, optional): The key of the configuration setting,
   provider (string, optional): The configuration provider of the configuration setting
 =
   [
      'CIRRUS',
      'PLUGIN'
   ],
   rangeMax (integer, optional): Either the highest allowed number or the maximum
 number of characters if the value is a string,
   rangeMin (integer, optional): Either the lowest allowed number or the minimum number
 of characters if the value is a string,
   regExp (string, optional): The regular expression pattern for the value validation,
   type (string, optional): The type of the configuration setting =
   [
      'STRING',
      'INTEGER',
      'DATE',
      'PASSWORD',
      'BINARY',
      'BOOLEAN',
      'ENCRYPTED'
   ],
   viewRoles (integer, optional): Defines the roles which are allowed to view the
 setting
}
```

Example (JSON output)

```
GET .../v8/configuration/descriptions?startswith=plugins
```

returns all descriptions for configuration settings starting with key plugin like

```
{
  "list": [
    {
      "id": "plugin.cfg.NotificationSMSClickatell.maxparts",
      "description": "Maximum number of message parts to be used (1-3, default 3)",
      "type": "STRING",
      "provider": "PLUGIN",
      "accountSpecific": true,
      "editRoles": 12,
      "viewRoles": 12,
      "regExp": "[1-3]"
    },
    {
```

```
      "id": "plugin.cfg.NotificationSMSClickatell.userparm",
      "description": "Additional parameters to be sent to the service (use URL
 encoding)",
      "type": "STRING",
      "provider": "PLUGIN",
      "accountSpecific": true,
      "editRoles": 12,
      "viewRoles": 12,
      "regExp": "[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]"
    },
   …
  ]
}
```

In case of an error a RestMsgList object is returned:

```
RestMsgList {
  list (Array[RestMsg], optional):
  List of messages (info, warnings, errors).
  }
RestMsg {
  code (integer): code ,
  message (string): description ,
  type (string): type = ['ERROR', 'WARNING', 'INFO']
}
```

Error conditions:

Reason	Error Code
Server administrator provides an invalid accountid	4001
The locale string is not a valid IETF BCP 47 language tag	21

If an invalid locale is requested the error code 21 is returned.

Example (JSON output)

```
{
  "list": [
    {
      "code": 21,
      "type": "ERROR",
      "message": "Invalid locale specified."
    }
  ]
}
```

Export configuration settings and (or) document types

This request exports configurations and (or) document types. With successful response, the server returns the configuration file settings.json compressed in a zip file settings.zip. To trigger browser download, the response headers have to include:

```
Content-Disposition: ATTACHMENT; filename=settings.zip

Content-Type: application/octet-stream;charset=UTF-8
```

URL

http://*host_server:port_number*/cirrus/rest/v8/configuration/export

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is exported.
- **effective** (boolean, optional): Indicator whether effective configuration data should be retrieved or only account specific. Without effective flag you retrieve only the configuration value which was set for the specific account the user belongs to or in case of a SUPERUSER the global, account independent value if the request does not include an account identifier. With effective flag equals true the request returns the first available setting in the following sequence: The account specific value (if available in the request) then the globally (account independent) value. Default value: true
- **readonly** (boolean, optional): Indicator whether a hash of the content must be calculated or not. Default value: false
- **plugin** (boolean, optional): Indicator whether plugin configurations must be exported. Default value: true
- **types** (Array[string], required): Defines what kind of settings are exported. Valid types are CONFIGURATION and DOCTYPE. Multiple types are comma separated.

Response message

Status 200 (OK): The export configuration request was successful.

Set configuration settings

Set one or more configuration values.

```
POST http(s)://domainOrIPAddress:port/context/rest/v8/configuration/[?
accountid={accountid}]
```

The request body contains a RestConfiguration object (see declaration above) with a list of RestConfigEntry objects. A RestConfigEntry has a key "k" and a value "v".

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is updated. Users with role SUPERUSER can update any account, or null for the global value. Users with role ADMIN can only update their own account.

Body parameters

- **restConfigEntries** (string, required): Input JSON or XML string of RestDataList.

Example (JSON input)

```
POST .../rest/v8/configuration
```

with body:

```
{
  "list": [
```

```
    {
      "k": "plugin.cfg.NotificationSMSClickatell.maxparts",
      "v": "2"
    },
    {
      "k": "plugin.cfg.NotificationSMSClickatell.utf16",
      "v": "true"
    }
  ]
}
```

The requesting user must have the permission to change the configuration settings. The request is rejected if at least one of the settings must not be changed by the user.

A server administrator can set explicitly an accountid if he wants to change an account-specific value. For all other users the accountid parameter is ignored.

In case of an error a RestMsgList object is returned (see declaration above):

Error conditions:

Reason	Error Code
Server administrator provides an invalid accountid	4001
Edit of at least one of the specified configuration settings is not allowed	33
Edit of at least one of the specified configuration settings is not allowed for a specific account	34
Validation of at least one of the specified configuration settings failed	35
At least one of the specified configuration settings could not be saved	38

Example (Validation failed with error code 35)

```
{
  "list": [
    {
      "code": 35,
      "type": "ERROR",
      "message": "The validation of the value failed."
    }
  ]
}
```

Set a binary configuration

This request sets a binary configuration.

URL

http://*host_server:port_number*/cirrus/rest/v8/configuration/binary/{configid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **configid** (string, required): The key of the setting.

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is updated. Users with role SUPERUSER can update any account, or null for the global value. Users with role ADMIN can only update their own account.
- **locale** (string, optional): Set for locale specific configuration data. Format must be compatible with Locale.languageTag.
- **confdata_password** (string, optional): The password of the configuration binary data, in case the type is 'ENCRYPTED' and subType is 'PKCS12'.

Form data

- **confdata_bin** (file, required): The binary configuration data.

Response status

Status 201 (Created): The binary configuration value was stored successfully.

Import configuration settings and (or) document types

This request imports existing configuration settings. The request body contains compressed *.zip file. The accountid parameter is necessary if the requesting user is a server administrator which wants to import account specific configuration settings. The accountid parameter is omitted for account administrators because a user with role ADMIN can only import settings into his own account. If not specified for role SUPERUSER, system configuration is imported. Only users with role SUPERUSER can import system configuration.

URL

http://*host_server:port_number*/cirrus/rest/v8/configuration/import

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is imported. Users with role SUPERUSER can import any account, or null for the global value. Users with role ADMIN can only import their own account.

Form data

- **file** (file, required): File

Response status

Status 200 (OK): The configuration settings have been imported successfully.

Delete configuration settings

This request deletes one or more configuration values.

```
DELETE http(s)://domainOrIPAddress:port/context/rest/v8/configuration/[?
accountid={accountid}]
```

The request body contains a RestConfiguration object (see declaration above) with a list of RestConfigEntry objects. A RestConfigEntry has a key "k" and a value "v". The value "v" needs not be set, it is ignored in the delete request.

Query parameters

- **accountid** (string, optional): The account id. This parameter can be null if global configuration is updated. Users with role SUPERUSER can update any account, or null for the global value. Users with role ADMIN can only update their own account.

Body parameters

- **restConfigEntries** (string, required): Input JSON or XML string of RestDataList

Example (JSON input)

```
POST .../rest/v8/configuration
```

with body:

```
{
  "list": [
    {
      "k": "plugin.cfg.NotificationSMSClickatell.maxparts"
    },
    {
      "k": "plugin.cfg.NotificationSMSClickatell.utf16"
    }
  ]
}
```

The requesting user must have the permission to edit the configuration settings. The request is rejected if at least one of the settings must not be deleted by the user.

A server administrator can set explicitly an accountid if he wants to delete an account-specific value. For all other users the accountid parameter is ignored. The account of the authorized user will be used.

In case of an error a RestMsgList object is returned (see declaration above).

Error conditions:

Reason	Error Code
Server administrator provides an invalid accountid	4001
Delete of at least one of the specified configuration settings is not allowed	33
Delete of at least one of the specified configuration settings is not allowed for a specific account	34
At least one of the specified configuration settings could not be deleted	36

Example (Insufficient permissions to delete the specified configuration setting: error code 33):

```
{
  "list": [
    {
```

```
      "code": 33,
      "type": "ERROR",
      "message": "The current credentials do not permit changing the configuration
 value."
    }
  ]
}
```

Document requests

Download all documents as zip

This request downloads all documents of a signing package specified by a given signing package id as zip.

Note For this request a valid authentication with USER role is necessary. It additionally is required, that the logged in user either has read permissions on the account or is a member of the account.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

application/octet-stream, JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package to query

Query parameters

- **filename** (string, optional): The new download name of the zip. Extension needs to be included. If the file name is missing the signing package name is used with the "zip" extension.

Response status

Status 200 (OK): The archived documents could be retrieved for download. Otherwise a SignDoc Standard status code is returned together with the explaining messages. The response body returns the binary content of the zip file.

Get a single document

This request returns a single document specified by a given document id.

Note For this request a valid session and a valid authentication with USER role is necessary. It additionally is required, that the logged in user either has read permissions on the account or is a member of the account.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002
```

Example header

```
Accept: application/json
```

Path parameters

- **documentid** (string, required): The id of the document
- **packageid** (string, required): The id of the signing package

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.
- **fields** (string, optional): Specifies the field type(s) that should be returned. Possible field types values are: signature, text, checkbox, radiobutton, all (default) or none. The default value 'all' means all fields (of supported field types) are included in the response. You can define also subset of specific field

types, separated by commas. You could also provide the value 'none' (as single value) if no field information is required. Example 'signature, text' (without quotes).

- **pages** (string, optional): Specifies the page numbers for which information should be returned. The default value is 'all' and means all pages of the document. You can define a single page number, a list of page numbers (separated by commas) or a range of page numbers (separated by a minus sign '-') or '0' if no page info is required. Examples are '1, 2,4' or '2-5' or also '1-3,5' (without quotes).
- **content** (boolean, optional): Whether the document content (as base64 string) is returned or not.
- **thumbnail** (boolean, optional): Whether the thumbnail image of the first document page (as base64 string) is returned or not.
- **useIntId** (boolean, optional): The provided ids for package and document are the internal ids which were returned by get audit trail method.

Response class

Status 200 (OK): The document was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages. In case of a successful query the response body contains the following information regarding the requested document (RestDocumentOutput).

RestDocumentOutput

- **checkboxFields** (Array[RestCheckboxFieldOutput], optional): A list of all checkboxes contained in the document
- **content** (string): The actual document in Base64 format
- **custom** (string): Custom field
- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The message displayed when the recipient opens the document
- **fileName** (string, optional): The filename of the document
- **format** (string, optional): The document format. Possible values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **id** (string, optional): The id of the document
- **name** (string): The name of the document
- **order** (integer, optional): The order of the document
- **pageTotalNumber** (integer, optional): The total number of pages
- **pages** (Array[RestPageInfo, optional): A list of all requested pages information
- **signatureFields** (Array[RestSignatureFieldOutput], optional): A list of all signature fields contained in the document
- **textFields** (Array[RestTextFieldOutput], optional): A list of all text fields contained in the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string

RestCheckboxFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field

- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestPageInfo

- **height** (number): The alternate name of the field (used as label value in SignDoc)
- **imageURL** (string): The URL for retrieving the image of the page (png format)
- **number** (integer): The index of the page within the document (1 for the first page
- **width** (number): The page width in document coordinates

RestSignatureFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signed** (boolean): Whether the signature field is signed or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingMode** (string, optional): The signing mode of the signature field if signed. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' to sign with a trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default. Possible values: UNKNOWN, HW, PH, C2S, IMG, STAMP, TSP
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' to sign with a trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestTextFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The max length of the field
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field

- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page).
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page

A widget describes the appearance of a field. A REST field object has an array of widgets, because a pdf field can have multiple appearances (e.g. radio buttons). Currently only one widget is supported in SignDoc.

Download a single document

This request downloads a single document specified by a given document id.

Note For this request a valid authentication with USER role is necessary. It additionally is required, that the logged in user either has read permissions on the account or is a member of the account.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/content

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

application/pdf, JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/content
```

Example header

```
Accept: application/json
```

Path parameters

- **documentid** (string, required): The id of the document
- **packageid** (string, required): The id of the signing package

Query parameters

- **disposition_type** (string, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog in browser. This could (browser dependent) be reached by setting the response header Content-Disposition: attachment; filename="<file name.ext>". The query parameter content_disposition sets this Content-Disposition header value in the response. Usually ATTACHMENT and INLINE are supported by a browser (see also http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html - 19.5.1 Content-Disposition)
- **filename** (string, optional): The new download name of the document. Extension needs to be included. If the file name is missing the document file name is used with the "pdf" extension.

Response

Status 200 (OK): The requested document could be retrieved for download. Otherwise a SignDoc Standard status code is returned together with the explaining messages. The response body returns the binary document.

Get a document page image

This request returns an image of a document page or a snippet within the page specified by a given document id and page number.

Note For this request a valid authentication with role USER or SIGNER is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/pages/{pageno}/image

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **documentid** (string, required): The id of the document
- **packageid** (string, required): The id of the signing package
- **pageno** (string, required): The page number

Query parameters

- **format** (string, optional): The requested image output format, can be png (default) or jpeg.
- **resolution** (float, optional): The resolution in dpi for image rendering. Default value: 72 dpi
- **left** (double, optional): The left horizontal coordinate of the image if only a snippet from the page is requested. Origin is in the bottom left corner of the page.
- **right** (double, optional): The right horizontal coordinate of the image if only a snippet from the page is requested. Origin is in the bottom left corner of the page.

- **bottom** (double, optional): The lower vertical coordinate of the image if only a snippet from the page is requested. Origin is in the bottom left corner of the page.
- **top** (double, optional): The upper vertical coordinate of the image if only a snippet from the page is requested. Origin is in the bottom left corner of the page.
- **disposition_type** (string, optional): The disposition type of the downloaded image
- **filename** (string, optional): The file name of the downloaded image

Response class

Status 200 (OK): The document image could be rendered successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Find text in a single document

Find text occurrences in a document.

Note Only the lower vertical coordinate (bottom) is returned, the upper vertical coordinate cannot be evaluated.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/text

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **documentid** (string, required): The id of the document
- **packageid** (string, required): The id of the signing package

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of field coordinates to document coordinates, default: 72 dpi.
- **searchtext** (string, optional): The text to find and locate
- **natural_order** (boolean, optional): Defines the sort order, when multiple matches are found. true is natural sort order, false is reversed sort order.

Response class

Status 200 (OK): The request was successful. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestFindTextResult

- **bottom** (number, optional): The bottom coordinate
- **left** (number, optional): The left coordinate
- **page** (integer): The page of the result
- **resolution** (number, optional): The resolution (dpi) of the coordinates

- **right** (number, optional): The right attribute contains the rough (calculated) horizontal coordinate value from the right margin of the found text

Add document to signing package

This request adds a new document to an existing signing package.

An input JSON or XML string specifying the details of the document has to be provided as a request parameter. The specification has to provide at least one document content including the content attribute (the document bytes encoded as Base64) of the document and a document name and a file name. The order is by default the current document size plus one.

Note For this request a valid authentication with USER role is necessary.

If uploaded files contain slash (/) or backslash (\) characters in the document name or file name, these characters will be replaced by an underscore (_) character.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/document

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/document
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "name": "PDF example document",
  "fileName": "document.pdf",
  "format": "PDF",
  "content": " base64 encoded PDF",
  "signatureFields": [
```

```
  {
    "id": "signature-1",
    "required": "true",
    "label": "Signature field 1",
    "signingModeOptions": [
      "PH",
      "C2S"
    ],
     "widgets": [
        {
          "bottom": 300,
          "index": 0,
          "left": 100,
          "pageNumber": 1,
          "right": 200,
          "top": 100
        }
  }
 ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package the user wants to add a reminder

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.
- **autoprep** (boolean, optional): Autoprepare feature places signature fields in the document for each of the signers that have been defined.. Default value: false

Body parameters

- **restDocument** (RestDocumentInput, required): Represents a RestDocument object either in JSON or XML. Default value: null

RestDocumentInput

- **checkboxFields** (Array[RestCheckboxFieldInput], optional): A list of all checkboxes contained in the document
- **content** (string): The actual document in Base64 format
- **custom** (string): Custom field
- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The message displayed when the recipient opens the document
- **fileName** (string, optional): The filename of the document
- **format** (string, optional): The document format. Possible values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **id** (string, optional): The id of the document
- **name** (string): The name of the document
- **order** (integer, optional): The order of the document
- **signatureFields** (Array[RestSignatureFieldInput], optional): A list of all signature fields contained in the document
- **textFields** (Array[RestTextFieldInput], optional): A list of all text fields contained in the document

- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' sign with trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestTextFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The max length of the field
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Response class

Status 201 (Created): The document was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestDocumentListEntry

- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The document message
- **fileName** (string, optional): The file name of the document
- **id** (string, optional): The id of the document
- **name** (string, optional): The name of the document
- **order** (integer, optional): The order of the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string
- **url** (string, optional): The request URL for the entire document

Update a document

This request updates an existing document within an account using an input JSON or XML string. Each attribute of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary. There is only a limited selection of fields available for the signer (content of the document, state of signature fields and checkbox fields and content of text fields). Other fields are ignored.

The document content cannot be replaced when it does not have the same number of pages or not the same page sizes as the original document.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
   "id": "document1",
   "name": "PDF example document",
   "format": "PDF",
   "content": "base64 encoded PDF"
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.

Body parameters

- **restDocument** (RestDocumentInput, required): Represents a RestDocument object either in JSON or XML. Default value: null

RestDocumentInput

- **checkboxFields** (Array[RestCheckboxFieldInput], optional): A list of all checkboxes contained in the document
- **content** (string): The actual document in Base64 format
- **custom** (string): Custom field
- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The message displayed when the recipient opens the document
- **fileName** (string, optional): The filename of the document
- **format** (string, optional): The document format. Possible values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF

- **id** (string, optional): The id of the document
- **name** (string): The name of the document
- **order** (integer, optional): The order of the document
- **signatureFields** (Array[RestSignatureFieldInput], optional): A list of all signature fields contained in the document
- **textFields** (Array[RestTextFieldInput], optional): A list of all text fields contained in the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' sign with trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestTextFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The max length of the field
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not

- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Delete a document

This request deletes a document and all fields of the document.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document the user wants to delete

Response status

Status is 200 (OK): The document was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Download the final document

This request downloads the final document of a signing package.

Note For this request a valid authentication with USER role is necessary. It additionally is required, that the logged in user either has read permissions on the account or is a member of the account.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/finaldocument

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/finaldocument
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package to query

Query parameters

- **filename** (string, optional): The new download name of the final document. Extension needs to be included. If the file name is missing the signing package name is used with the ".pdf" extension
- **disposition_type** (string, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog in browser. This could (browser dependent) be reached by setting the response header Content-Disposition: attachment;

filename="<file name.ext>". The query parameter content_disposition sets this Content-Disposition header value in the response. Usually ATTACHMENT and INLINE are supported by a browser (see also http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html - 19.5.1 Content-Disposition). Default value: INLINE

Response status

Status 200 (OK): The final document could be retrieved for download, otherwise a SignDoc Standard status code is returned together with the explaining messages. The response body returns the binary document.

Document type requests

The REST API provides CRUD functionality for document types. Document type is a template for a set of supplemental documents, which can be defined by an account administrator within an account.

An owner of a signing package can create document type instances based on the document types and assign them to package's recipients.

Document type has name, description and max number of supplemental documents.

Note Maximum number of files of a document type cannot be more than a value of account-specific configuration setting 'cirrus.document.prepare.supplemental.file.max-number'.

There are predefined document types, which are assigned to each account:

ID	Name	Description	Max number of files
GENERIC	Upload supplemental documents	Submit copies of supplemental documents to complete information.	25
DRIVER-LICENSE	Driver's license	Provide a copy of your driver's license. Provide front and reverse page of the card or unfolded document	2
NATIONAL-ID-CARD	Identity card	Provide a copy of your national identification card. Provide front and reverse page of the card.	2
PASSPORT	Passport	Provide a copy of your passport. Provide at least the identification page and the following page.	4
IDENTIFICATION	Identification	Provide an identification document. This can be a driver license, passport or identity card.	4

Note GENERIC document type cannot be deleted and its id cannot be changed.

Get available document types

This request returns document types available for the account.

Note For this request a valid authentication with SUPERUSER or USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/doctypes

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/doctypes
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **locale** (string, optional): Locale for retrieving the locale-specific document type(s) (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, all available document types for all languages will be returned.

Response class

Status 200 (OK): The document types were queried successfully, otherwise a SignDoc Standard status code is returned together with the explaining messages.

In case of a successful query the response body contains the following information regarding the document types (array of RestDocumentType).

RestDocumentType

- **description** (string, optional): The description of the document type
- **id** (string, optional): The id of the document type
- **locale** (string, optional): The locale of the document type

- **maxFilesNumber** (integer, optional): The maximum number of files allowed for the document type
- **name** (string, optional): The name of the document type

Example response


```
[
  {
    "id": "GENERIC",
    "locale": "en",
    "name": "Upload supplemental documents",
    "description": "Please submit copies of supplemental documents to complete
 information.",
    "maxFilesNumber": 25
  },
  {
    "id": "DRIVER-LICENSE",
    "locale": "en",
    "name": "Driver's license",
    "description": "Please provide a copy of your driver's license. The copy should
 include front and reverse page of the card or unfolded document.",
    "maxFilesNumber": 2
  },
  {
    "id": "NATIONAL-ID-CARD",
    "locale": "en",
    "name": "Identity card",
    "description": "Please provide a copy of your national identification card. The
 copy should include front and reverse page of the card.",
    "maxFilesNumber": 2
  },
  {
    "id": "PASSPORT",
    "locale": "en",
    "name": "Passport",
    "description": "Please provide a copy of your passport. Provide at least the
 identification page and the following page.",
    "maxFilesNumber": 4
  },
  {
    "id": "IDENTIFICATION",
    "locale": "en",
    "name": "Identification",
    "description": "Please provide an identification document. This can be a driver
 license, passport or identity card.",
    "maxFilesNumber": 4
  }
]
```


Get a specified document type

This request returns the requested document type.

Note For this request a valid authentication with SUPERUSER or USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/doctypes/{doctypeid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/account/doctypes/PASSPORT
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **locale** (string, optional): Locale for retrieving the locale-specific document type(s) (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en will be used.

Path parameters

- **doctypeid** (string, required): The document type id

Response class

Status 200 (OK): The document type was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

In case of a successful query the response body contains the following information regarding the document type (RestDocumentType).

- **description** (string, optional): The description of the document type
- **id** (string, optional): The id of the document type
- **locale** (string, optional): The locale of the document type
- **maxFilesNumber** (integer, optional): The maximum number of files allowed for the document type
- **name** (string, optional): The name of the document type

Create a document type

This request creates a document type using an input JSON or XML data.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/doctype

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Content-Type: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/account/doctype
```

Example headers

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "description": "Payroll accounting",
  "id": "PAYROLL",
  "locale": "en",
  "maxFilesNumber": 4,
  "name": "Payroll"
}
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.

Body parameters

- **documenttype** (RestDocumentType, required): Document type data

RestDocumentType

- **description** (string, optional): The description of the document type
- **id** (string, optional): The id of the document type
- **locale** (string, optional):
  The locale of the document type. It must be a valid IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47. Example: en for English or pt-BR for Brazilian Portuguese. If omitted, en will be used.
- **maxFilesNumber** (integer, optional): The maximum number of files allowed for the document type
- **name** (string, optional): The name of the document type

Response status

Status 201 (Created): The document type was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a specified document type

This request updates a document type using an input JSON or XML data.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/doctypes/{doctypeid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Content-Type: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/account/doctypes/PASSPORT
```

Example headers

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "description": "This value will be changed"
}
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.

Path parameters

- **doctypeid** (string, required): The document type id

Body parameters

- **documenttype** (RestDocumentType, required): Input XML or JSON string with serialized document type information. Note: The locale attribute of the RestDocumentType input cannot be changed. It must be set, in order to identify the locale specific document type definition which should be updated. If ommited the locale 'en' is assumed

RestDocumentType

- **description** (string, optional): The description of the document type
- **id** (string, optional): The id of the document type
- **locale** (string, optional): The locale of the document type. It must be a valid IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47. Example: en for English or pt-BR for Brazilian Portuguese. If omitted, en will be used.
- **maxFilesNumber** (integer, optional): The maximum number of files allowed for the document type
- **name** (string, optional): The name of the document type

Response status

Status 200 (OK): The document type was updated, otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a specified document type

This request deletes a document type with the specified id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/account/doctypes/{doctypeid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/account/doctypes/PASSPORT
```

Example headers

```
Accept: application/json
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **locale** (string, optional): Locale for retrieving the language-specific Locale for retrieving the locale-specific document type(s) (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en will be used.

Path parameters

- **doctypeid** (string, required): The document type id

Response class

Status 200 (OK): The document type was deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestDocumentType

- **description** (string, optional): The description of the document type
- **id** (string, optional): The id of the document type
- **locale** (string, optional): The locale of the document type. It must be a valid IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47. Example: en for English or pt-BR for Brazilian Portuguese. If omitted, en will be used.
- **maxFilesNumber** (integer, optional): The maximum number of files allowed for the document type
- **name** (string, optional): The name of the document type

Document type instance requests

Get document type instance of a single signer

This request returns a document type instances of a single signer.

Note For this request a valid authentication with USER role is and read access permitted to the signing package is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/signer-1/
doctypeinstances
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The signing package id
- **signerid** (string, required): The signer id

Query parameters

- **effective** (boolean, required): The parameter effective controls whether only the explicit values for name and description from the document type instance are returned (effective=false) or if also the implicit and language specific default values could be returned in the name and description attribute (effective=true, default), if no explicit values are set. Default value: true
- **locale** (string, optional): Locale for retrieving the language specific document types (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, language tag 'en' will be used. If explicit entered values are available for a document type instance (only possible for type GENERIC) then the locale parameter has no effect, because only the explicit values are returned. The locale parameter is only relevant if no explicit values are available. The account specific values for name and description are returned for the specified locale. If the language for the specified locale is not available then the locale 'en' is assumed. If 'en' is also not available in the account specific settings then a predefined general default text in English for name and description is returned.

Response class

Status 200 (OK): The document type instances were retrieved successfully, otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestDocumentTypeInstanceOutput

- **description** (string, optional): The description of the instance
- **docTypeId** (string): The id of the document type. An error is returned if this document type id does not exist for the account
- **id** (string, optional): The id of the instance, must be unique for a signer. If omitted a random id is created at the server
- **maxFilesNumber** (integer, optional): The maximum number of files permitted for this document type
- **name** (string, optional): The name of the instance
- **required** (boolean, optional): This flag indicates that supplemental documents for the referenced document type are required, defaults to TRUE
- **supplementalDocuments** (Array[RestSupplementalDocumentInfo], optional): The list of supplemental documents in this instance.

RestSupplementalDocumentInfo

- **contentUrl** (string, optional): The URL for retrieving document content
- **creationTime** (string, optional): The creation time of the supplemental document
- **fileName** (string, optional): The file name of the supplemental document
- **id** (string, optional): The ID of the supplemental document
- **order** (integer, optional): The order of the supplemental document

Download supplemental document content

This request downloads a single supplemental document content.

Note For this request a valid authentication with USER or SIGNER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{doctypeinstanceid}/documents/{supplementaldocid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

application/pdf

Header

-

Method

GET

Query parameters

- **dispositiontype** (string, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog in browser. This could (browser dependent) be reached by setting the response header Content-Disposition: attachment; filename="<file name.ext>". The query parameter content_disposition sets this Content-Disposition header value in the response. Usually ATTACHMENT and INLINE are supported by a browser (see also http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html - 19.5.1 Content-Disposition). Default value: ATTACHMENT
- **filename** (string, optional): The new download name of the supplemental document. Extension needs to be included. If the file name is missing the signing package name is used with the '.pdf' extension.

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer
- **doctypeinstanceid** (string, required): The id of the document instance
- **supplementaldocid** (string, required): The id of the supplemental document

Response class

Status 200 (OK): The supplemental document was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages. In case of a successful query the response body contains binary data with document content providing with proper HTTP headers.

Get a single document type instance

This request returns the requested document type instance.

Note For this request a valid authentication with USER role and write access permitted to the signing package is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{doctypeinstid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/signer-1/doctypeinstances/doctype-1
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The signing package id.
- **signerid** (string, required): The signer id.
- **doctypeinstid** (string, required): The document type instance id.

Query parameters

- **effective** (boolean, required): The parameter effective controls whether only the explicit values for name and description from the document type instance are returned (effective=false) or if also the implicit and language specific default values could be returned in the name and description attribute (effective=true, default), if no explicit values are set. Default value: true

- **locale** (string, optional): Locale for retrieving the language specific document types (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, default language tag 'en' will be used. If explicit entered values are available for a document type instance (only possible for type GENERIC) then the locale parameter has no effect, because only the explicit values are returned. The locale parameter is only relevant if no explicit values are available

Response class

Status 200 (OK): The document type instance was retrieved successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestDocumentTypeInstanceOutput

- **description** (string, optional): The description of the instance
- **docTypeId** (string): The id of the document type. An error is returned if this document type id does not exist for the account
- **id** (string, optional): The id of the instance, must be unique for a signer. If omitted a random id is created at the server
- **maxFilesNumber** (integer, optional): The maximum number of files permitted for this document type
- **name** (string, optional): The name of the instance
- **required** (boolean, optional): This flag indicates that supplemental documents for the referenced document type are required, defaults to TRUE
- **supplementalDocuments** (Array[RestSupplementalDocumentInfo], optional): The list of supplemental documents in this instance

RestSupplementalDocumentInfo

- **contentUrl** (string, optional): The URL for retrieving document content
- **creationTime** (string, optional): The creation time of the supplemental document
- **fileName** (string, optional): The file name of the supplemental document
- **id** (string, optional): The ID of the supplemental document
- **order** (integer, optional): The order of the supplemental document

Create a document type instance for a signer

This request creates a document type instance for a signer.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstance

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Content-Type: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/signer-1/
doctypeinstance
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example body

```
{
  "id": "doctypeinstance-1",
  "docTypeId": "DRIVER-LICENSE",
  "name": "Driver's license",
  "description": "Please provide a copy of your driver's license. Provide front and
 reverse page of the card or unfolded document.",
  "required": true,
  "maxFilesNumber": 2
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer

Body parameters

- **restDocumentTypeInstanceInput** (RestDocumentTypeInstanceInput, required): JSON or XML string of RestDocumentTypeInstanceInput

RestDocumentTypeInstanceInput

- **description** (string, optional): The description of the instance. Only allowed for document type GENERIC
- **docTypeId** (string, optional): The id of the document type. On creation, an error is returned if this document type id does not exist for the account. On update, this field cannot be changed. An error is returned if docTypeId in the update request does not match the document type which was entered during creation.
- **id** (string, optional): The id of the instance, must be unique for a signer. If omitted a random id is created at the server
- **maxFilesNumber** (integer, optional): The maximum number of files permitted for this document type. Only allowed for document type GENERIC
- **name** (string, optional): The name of the instance. Only allowed for document type GENERIC
- **required** (boolean, optional): This flag indicates that supplemental documents for the referenced document type are required, defaults to TRUE

Response status

Status 201 (Created): The document type instance was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Add supplemental document

This request adds a supplemental document for a specified signer. The content type of the body is multipart/form-data.

Note For this request a valid authentication with SIGNER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{instanceid}/document

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/GTFH-345SDFG/signers/TYUJ-FGH-RTG/doctypeinstances/XCEWR-76YJU/document
```

Example header

```
Accept: application/json

Content-Type: multipart/form-data
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer
- **instanceid** (string, required): The id of the document type instance

Form parameters

- **id** (string, optional): The id of the supplemental document. If not present, the server will generate one.
- **filename** (string, optional): The name of the supplemental document. If not present, the server will generate one.

- **content** (file, required): The supplemental document. The following formats are supported: JPEG (JPG), PNG, PDF, DOC, DOCX. Other formats may throw an error

Response class

Status 201 (Created): The supplemental document was successfully added, otherwise a SignDoc Standard status code is returned together with the explaining messages. In case of a successful request the response body contains the following information as RestSupplementalDocumentInfo structure:

RestSupplementalDocumentInfo

- **contentUrl** (string, optional): The URL for retrieving document content
- **creationTime** (string, optional): The creation time of the supplemental document
- **fileName** (string, optional): The file name of the supplemental document
- **id** (string, optional): The ID of the supplemental document
- **order** (integer, optional): The order of the supplemental document

Update a document type instance

This request updates a document type instance.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{doctypeinstid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/signer-1/
doctypeinstance/{doctypeinstance-1}
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example body

```
{
  "name": "Driver's license",
  "description": "Please provide the front and reverse page of your driver's license.
 ",
  "required": false
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer
- **doctypeinstid** (string, required): The id of the document type instance

Body parameters

- **restDocumentTypeInstanceInput** (RestDocumentTypeInstanceInput, required): JSON or XML string of RestDocumentTypeInstanceInput.

RestDocumentTypeInstanceInput

- **description** (string, optional): The description of the instance. Only allowed for document type GENERIC
- **docTypeId** (string, optional): The id of the document type. On creation, an error is returned if this document type id does not exist for the account. On update, this field cannot be changed. An error is returned if docTypeId in the update request does not match the document type which was entered during creation
- **id** (string, optional): The id of the instance, must be unique for a signer. If omitted a random id is created at the server
- **maxFilesNumber** (integer, optional): The maximum number of files permitted for this document type. Only allowed for document type GENERIC
- **name** (string, optional): The name of the instance. Only allowed for document type GENERIC
- **required** (boolean, optional):This flag indicates that supplemental documents for the referenced document type are required, defaults to TRUE

Response status

Status 200 (OK): The document type instance has been successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a document type instance

This request deletes a document type instance.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{doctypeinstid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/{packageid}/signers/
{signerid}/doctypeinstances/{doctypeinstid}
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer
- **doctypeinstid** (string, required): The id of the document type instance

Response status

Status is 200 (OK): The document type instance was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete supplemental document

This request deletes a supplemental document for a specified signer.

Note For this request a valid authentication with SIGNER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/doctypeinstances/{instanceid}/document/{docid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/NRT3456/signers/XCVB-345-FGH/doctypeinstances/DFSG-45-FGH/document/A54-GH
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer
- **instanceid** (string, required): The id of the document type instance
- docid (string, required): The id of the supplemental document

Response status

Status is 200 (OK): The supplemental document was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Event requests

Get an event

Long polling request which returns an event within a specified (configurable, account-specific) time frame. Any requested event is provided via SignDoc-Events header. The call returns a RestEvent object with a list of RestEventEntry elements which can include action, subject, product and parameters. The returned event can be a client-specific event or an event that indicates that no action is needed (action=NONE). The event with action=NONE is returned if the request times out w/o returning any client-specific event.

URL

http://*host_server:port_number*/rest/v8/event

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/event
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Response class

Status 200 OK: An event is returned successfully.

Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Trigger an event

This request triggers an event to the system.

Note For this request a valid authentication with USER or SIGNER role is necessary.

URL

http://*host_server:port_number*/rest/v8/event

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/event
```

Example header

```
Accept: application/json
```

Example request body (JSON)

```
{
   "list": [{
      "k": "action",
      "v": "END"
   }, {
      "k": "subject",
      "v": "SIGNING_SESSION"
   }, {
      "k": "product",
      "v": "CIRRUS"
   }]
}
```

An event is presented as a list of event's parameters with the following keys:

- **action**: (string, required): Action name which should be triggered. It can be applied to specified subject.
- **subject**: (string, optional): Subject for specified action.
- **product**: (string, optional): If not specified, the event should be handled by SignDoc Standard. Possible value: CIRRUS

Structure of the event input object:

RestEvent

- **list** (Array[RestEntry], optional): List of properties (as key-value pairs) of the event

RestEventEntry

- **k** (string): The key of the pair
- **v** (string): The value of the pair

The following list of events can be processed by SignDoc Standard:

Action	Subject	Product	Parameters	Description
END	SIGNING_SESSION	CIRRUS	n/a	End the existing Common Signing Session. Signing host can finish the signing session any time triggering the event. The signing session will be reset, authentication token will be invalidated. Can only be triggered by a signing host.
	SIGNER	CIRRUS	n/a	End a signing package signing by a signer. The signer has to complete signing/reviewing of all required documents before sending the event. The signing package owner will be notified, the proper message will be added to the audit trail. Can only be triggered by a signer.

Action	Subject	Product	Parameters	Description
AGREE_ESIGN_CONSENT	SIGNER	CIRRUS	n/a	Accept the e-sign consent by an authenticated signer. The consent must be accepted once by each signer. The proper message will be added to the audit trail. Can only be triggered by a signer.
DECLINE	SIGNER	CIRRUS	DECLINE_REASON (mandatory) DECLINE_COMMENT (optional)	Decline a signing package by a signer. The proper message will be added to the audit trail. Can only be triggered by a signer.
DELIVERED	SIGNER	CIRRUS	DOCUMENT_ID (mandatory) PAGE_NUMBER (mandatory)	A document page was delivered to the signer. The proper message will be added to the audit trail. Can only be triggered by a signer.
DONE	SIGNER_DOCUMENT	CIRRUS	DOCUMENT_ID (mandatory)	A signer has completed a document. The proper message will be added to the audit trail. Can only be triggered by a signer.

Body parameters

- **event**: (RestEvent, required): Event as list of event's properties (key-value pairs) presented as input JSON/XML data

Response class

Status 200 OK: The event was triggered successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Field requests

Get a single checkbox

This request returns a single checkbox specified by a given checkbox id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/checkboxes/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
checkboxes/1003
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "1003",
  "name": "checkbox1",
  "description": "Please select the checkbox",
  "signerId": "signer-1",
  "alternateName": "Checkbox 1",
  "required": true,
  "readOnly": false,
  "checked": false,
  "widgets": [
    {
      "index": 0,
      "pageNumber": 1,
      "top": 448.99,
      "left": 333.75,
      "right": 348.75,
      "bottom": 433.99
    }
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Query parameters

- **resolution** (float): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.

- **useIntId** (boolean, optional): The provided ids for signing package, document and field are the internal ids which were returned by the get audit trail method. Default value: false

Response class

Status 200 (OK): The checkbox was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestCheckboxFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Get all fields

This request returns all fields of a specified document. The returned fields only contain absolute necessary information like name, id or assigned signer. To access the complete data set a separate request URL is provided as an additional parameter.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/fields

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/fields
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
[
  {
    "id": "cb1",
    "label": "Checkbox 1",
    "type": "CheckBox",
    "signerID": "signer-1",
    "url": "http://localhost:6611/cirrus/rest/v8/packages/documents/checkboxes/403"
  },
  {
    "id": "sig1",
    "label": "Signature 1",
    "type": "SignatureField",
    "signerID": "signer-1",
    "url": "http://localhost:6611/cirrus/rest/v8/packages/documents/
signaturefields/401"
  },
  {
    "id": "t1",
    "label": "Text 1",
    "type": "TextField",
    "signerID": "signer-1",
    "url": "http://localhost:6611/cirrus/rest/v8/packages/documents/textfields/402"
  }
]
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document

Query parameters

- **fieldFilter** (string, optional): Whether the returned list should be filtered to a specific field type or not. Possible values: SignatureField, CheckBox, TextField

Response class

Status 200 (OK): The list was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestDocumentFieldListEntry

- **id** (string, optional): The id of the field

- **label** (string, optional): The label of the field
- **signerID** (string, optional): The signer id of the field
- **type** (string, optional): The type of the field. Possible values: SignatureField, CheckBox, TextField
- **url** (string, optional): The request URL for the entire field

Get a single signature field

This request returns a single signature field specified by a given signature field id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages{packageid}/documents/{documentid}/signaturefields/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
signaturefields/1003
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "signature-1",
  "name": "signatureField1",
  "signerId": "signer-1",
  "alternateName": "Signature Field 1",
  "required": false,
  "readOnly": false,
  "signed": false,
  "widgets": [
    {
      "index": 0,
      "pageNumber": 1,
```

```
      "top": 400,
      "left": 100,
      "right": 400,
      "bottom": 300
    }
  ],
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.
- **useIntId** (boolean, optional): The provided ids for signing package, document and field are the internal ids which were returned by the get audit trail method. Default value: false

Response class

Status 200 (OK): The signature field was queried successfully, otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestSignatureFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signed** (boolean): Whether the signature field is signed or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingMode** (string, optional): The signing mode of the signature field if signed. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' sign with trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default. Possible values: UNKNOWN, HW, PH, C2S, IMG, STAMP, TSP
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, ‘TSP’ sign with trusted service provider. Default value: HW,

PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.

- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page).
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page

Get a single text field

This request returns a single text field specified by a given text field id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/textfields/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
textfields/1003
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "f30d5193-6823-46d3-b33e-908d93f21318",
  "name": "TextField1",
  "description": "This is an important text field",
  "signerId": "signer-1",
  "alternateName": "Text field 1",
  "required": true,
  "readOnly": false,
  "multiLine": false,
  "maxLength": 1024,
  "widgets": [
    {
      "index": 0,
      "pageNumber": 1,
      "top": 454.9999999999999,
      "left": 99.75,
      "right": 324.75,
      "bottom": 432.4999999999999
    }
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates, default: 72 dpi.
- **useIntId** (boolean, optional): The provided ids for signing package, document and field are the internal ids which were returned by the get audit trail method. Default value: false

Response class

Status 200 (OK): The text field was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestTextFieldOutput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The max length of the field
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page).
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page

Add checkbox to document

This request adds a new checkbox to an existing document. An input JSON or XML string specifying the details of the signature field has to be provided as a request parameter. The input data structure must contain one RestWidget with position attributes in the widgets list.

To immediately assign a signer, the user has to provide a signer id as a request parameter. The attribute required will be set to false by default if nothing further is specified.

Note For this request a valid authentication with USER role is necessary. The page, coordinates and dimensions of the new checkbox have to be in the bounds of the document.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/checkbox

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
checkbox
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "alternateName": "Check Box 2",
  "description": "This is my new checkbox",
  "id": "cbox2",
  "name": "CheckBox2",
  "readOnly": false,
  "required": false,
  "signerId": "signer-1",
  "widgets": [
    {
      "bottom": 300,
      "index": 0,
      "left": 200,
      "pageNumber": 1,
      "right": 220,
      "top": 320
    }
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of field coordinates to document coordinates, default: 72 dpi.

Body parameters

- **checkBox** (RestCheckBoxFieldInput, required): Input XML or JSON string of checkbox field

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget], optional): A list of all widgets of the field

RestWidget

- **bottom** (number, optional): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number, optional): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer, optional): The index of the page on which this field occurs (1 for the first page)

- **right** (number, optional): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number, optional): The top coordinate. The origin is in the bottom left corner of the page.

Response class

Status 201 (Created): The checkbox was added to the document. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Add signature field to document

This request adds a new signature field to an existing document. An input JSON or XML string specifying the details of the signature field has to be provided as a request parameter. The input data structure must contain one RestWidget with position attributes in the widgets list.

To immediately assign a signer, the user has to provide a signer id as a request parameter. The attribute required will be set to false by default if nothing further is specified. By default the signingModeOptions are set to HW, PH, C2S, IMG. If a TSP plugin is registered to a signer and if the TSP plugin supports signing the signature field using 'TSP', 'TSP' mode is also added by default.

It is possible to add a signature field with explicit position attributes via RestWidget bean or you can add a signature field in relative position to a specific text in the document.

In the latter case it is not necessary to provide a RestWidget structure in the widgets attribute of RestSignatureFieldInput. A search text can be specified as query parameter. The signature field is placed relative to the origin of the found text. The relative offsets for another position of the signature field can be set via query parameters offset_horizontal and offset_vertical. The desired width and height of the new inserted signature field can be set via query parameters desired_width and desired_height.

The desired size of the signature field could be reduced if the field position would exceed a page boundary.

Note For this request a valid session and USER role is necessary. The page, coordinates and dimensions of the new signature field have to be in bounds of the document.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/signaturefield

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request with explicit position attributes

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
signaturefield
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "alternateName": "Signature Field 2",
  "description": "This is my new signature field",
  "id": "sig2",
  "name": "Signature2",
  "readOnly": false,
  "required": true,
  "signerId": "signer-1",
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ],
  "widgets": [
    {
      "bottom": 500,
      "index": 0,
      "left": 150,
      "pageNumber": 1,
      "right": 300,
      "top": 600
    }
  ]
}
```

Example request with search text query parameters without explicit position attributes

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
signaturefield?resolution=72&searchtext=Hello&offset_horizontal=0&offset_vertical=
 -50&select_match=0&natural_order=true&desired_width=-1&desired_height=60&min_height=
 30&min_width=100
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "alternateName": "Signature Field 3",
  "description": "Signature field, inserted via searchText",
  "id": "sig3",
  "name": "Signature3",
  "readOnly": false,
```

```
  "required": true,
  "signerId": "signer-1",
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of field coordinates to document coordinates, default: 72 dpi.
- **searchtext** (string, optional): The text to find for locating the new field.
- **offset_horizontal** (double, optional): Move the found text position value horizontally by this value as left horizontal coordinate of the new signature field. Negative values go left, positive values go right.
- **offset_vertical** (double, optional): Move the found text position value vertically by this value as lower vertical coordinate of the new signature field. Negative values go down, positive values go up.
- **select_match** (integer, optional): If multiple text matches are found, define which match to use. 0 defines the first match, 1 the second, and so on. If the value is bigger than the results, the last result will be selected.
- **natural_order** (boolean, optional): Defines the sort order, when multiple text matches are found. true is natural sort order, false is reversed sort order.
- **desired_width** (double, optional): Defines the width of the created signature field if searchtext is used for locating. If not set or < 0, the width of the found text will be used. If this should not be possible, a fixed default value is used.
- **desired_height** (double, optional): Defines the height of the created signature field if searchtext is used for locating. If not set, a fixed default value is used.
- **min_height** (double, optional): Defines the minimum height of the created signature field if searchtext is used for locating. This minimum height is used if the evaluated field's height is less than this value.
- **min_width** (double, optional): Defines the minimum width of the created signature field if searchtext is used for locating. This minimum width is used if the evaluated field's width is less than this value.

Body parameters

- **signaturefField** (RestSignatureFieldInput, required): Input XML or JSON string of signature field

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field

- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' sign with trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidget], optional): A list of all widgets of the field

RestWidget

- **bottom** (number, optional): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number, optional): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer, optional): The index of the page on which this field occurs (1 for the first page)
- **right** (number, optional): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number, optional): The top coordinate. The origin is in the bottom left corner of the page.

Response status

Status 201 (Created): The signature field was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Get the biometric data of a signature field

This request returns the biometric data of a signature field that was signed using asymmetrically encrypted biometric signature data. A signature contains encrypted biometric data, if the biometric encryption key was set in the account settings when signing the field.

Note For this request a valid session and USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/signaturefield/{fieldid}/biodata

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

type: multipart/form-data

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
signaturefield/1003/biodata
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): Id of field

Form data

- **privkey** (file, required): The private key to decrypt the biometric data.

Query parameters

- **privkeypassword** (string, optional): The password for the private key.

Response status

Status 201 (Created): The requested data could be retrieved.

RestBiometricDataOutput

- **dataBase64** (string, optional): The binary data encode in BASE64

Add text field to document

This request adds a new text field to an existing document. An input JSON or XML string specifying the details of the signature field, has to be provided as a request parameter. The input data structure must contain one RestWidget with position attributes in the widgets list.

To immediately assign a signer, the user has to provide a signer id as a request parameter. The attribute required will be set to false by default if nothing further is specified.

Note For this request a valid authentication with USER role is necessary. The page, coordinates and dimensions of the new text field have to be in bounds of the document.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/textfield

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
textfield
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "alternateName": "Text field 1",
  "description": "This is a new text field",
  "id": "text1",
  "maxLength": 256,
  "multiLine": false,
  "name": "Text1",
  "readOnly": false,
  "required": false,
  "signerId": "signer-1",
  "value": "This is the default text",
  "widgets": [
    {
      "bottom": 200,
      "index": 0,
      "left": 50,
      "pageNumber": 1,
      "right": 150,
      "top": 225
    }
  ]
}
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of field coordinates to document coordinates. Default: 72 dpi.

Body parameters

- **textField** (RestTextFieldInput, required): Represents a RestTextFieldInput object either in JSON or XML. Default value: null

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field

- **maxLength** (integer, optional): The maximum length of the text value
- **multiLine** (boolean, optional): Whether the field is a multi-line text field or not
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget], optional): A list of all widgets of the field

RestWidget

- **bottom** (number, optional): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number, optional): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer, optional): The index of the page on which this field occurs (1 for the first page)
- **right** (number, optional): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number, optional): The top coordinate. The origin is in the bottom left corner of the page.

Response status

Status 201 (Created): The text field was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a checkbox field

This request updates an existing checkbox field. Selected attributes of the existing checkbox field can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/checkboxes/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
checkboxes/1003
```

Example header

```
Accept: application/json

Content-Type: multipart/form-data
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field
- **resolution** (float, required): The resolution in dpi for conversion of field coordinates to document coordinates. Default value: 72 dpi

Body parameters

- **checkBox** (RestCheckboxFieldInput, required): Input JSON or XML string of checkbox field

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Response status

Status 200 (OK): The checkbox was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a signature field

This request updates an existing signature field. Selected attributes of the existing signature fields can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/signaturefields/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
signaturefields/1003
```

Example header

```
Accept: application/json

Content-Type: multipart/form-data
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Body parameters

- **signatureField** (RestSignatureFieldInput, required): Input JSON or XML string of signature field

Query parameters

- **resolution** (float, optional): The resolution in dpi for conversion of field coordinates to document coordinates. Default value: 72 dpi

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Response status

Status 200 (OK): The signature field was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a text field

This request updates an existing text field. Selected attributes of the existing text field can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/textfields/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/
textfields/1003
```

Example header

```
Accept: application/json

Content-Type: multipart/form-data
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field
- **resolution** (float, required): The resolution in dpi for conversion of field coordinates to document coordinates. Default value: 72 dpi

Body parameters

- **fieldName** (RestTextFieldInput, required): Input JSON or XML string of user

RestTextFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The max length of the field
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is readonly or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field

- **widgets** (Array[RestWidget): A list of all widgets of the field

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page.
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Response status

Status 200 (OK): The text field was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a field

This request deletes a field from a document. The request is applicable for signature fields, checkbox fields or text fields.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{docuemntid}/fields/{fieldid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/1001/documents/1002/fields/1003
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Response status

Status 200 (OK): The field was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Remove a signer from field

This request removes a signer from the field specified by its id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/documents/{documentid}/fields/{fieldid}/signer

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/kuy4fh67h/
documents/234g5t/fields/asdf3e/signer
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the field

Response status

Status 200 (OK): The signer was successfully removed. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Sign a signature field

With this request a signature field can be signed.

URL

http://*host_server:port_number*/cirrus/rest/v8/documents/{documentid}/{fieldid}/signature

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **documentid** (string, required): The id of the document
- **fieldid** (string, required): The id of the Field (can be the id of the field or the PDF field name). See also fieldid_is_pdf_fieldname.

Query parameters

- **sigtype** (string, required): The signature type. Valid values: SIGNWARE, SIGNATURE_B, IMAGE, C2S
- **sigdata** (string, optional): The signature data as String. Either sigdata or sigdata_bin must be set. Usually encoded. Encoding must be one of [BASE64, BASE64RAW, NIBBLEHEX].
- **encoding** (string, optional): The encoding if the signature data. Valid values: BASE64, BASE64RAW, NIBBLEHEX, BINARY
- **fieldid_is_pdf_fieldname** (boolean, optional): If set to true, the fieldid specified is the PDF field name (and must be URL encoded). If unset or false, the fieldid is the id of the field.
- **signer_name** (string, optional): The signer's name used and must be set when sigtype is set to C2S
- **signing_certificate** (string, optional): BASE64 encoded, unencrypted signing certificate
- **signing_certificate_encrypted** (string, optional): BASE64 encoded, encrypted signing certificate (has precedence over signing_certificate)
- **signing_certificate_key** (string, optional): The AES session key, RSA-encrypted and Base64-encoded. Used for signing_certificate_encrypted and signing_certificate_pass_encrypted. RSA is used with OAEP, SHA-1, and MGF1.
- **signing_certificate_pass** (string, optional): The password for the PKCS #12 blob passed for signing_certificate, unencrypted, Base64-encoded.
- **signing_certificate_pass_encrypted** (string, optional): The password for the PKCS #12 blob passed for signing_certificate_encrypted, encrypted, Base64-encoded.
- **signing_certificate_name** (string, optional): Optional name of the certificate
- **signing_certificate_description** (string, optional): Optional description of the certificate

- **locale** (string, optional): ETF BCP 47 language tag. Default: en
- **tz** (string, optional): Time zone. Default: UTC

Form data parameters

- **sigdata_bin** (file, required): The signature data as binary data (file blob). Either sigdata or sigdata_bin must be set. Encoding must be set to BINARY.

Response status

Status 201 (Created): The signature field was successfully signed. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAddSignatureResult

- **fieldsToUpdate** (Array[RestField], optional): The fields to be updated
- **resultCode** (string, optional): The result code. Possible values: SUCCESS, SIGNATURE_TOO_SIMPLE_OR_NOT_USABLE

RestField

- **alternateName** (string, optional)
- **captureFieldSubtype** (string, optional): Possible values: CFST_UNKNOWN, CFST_SIGNATURE, CFST_IMAGECAPTURE, CFST_C2SSIGNATURE, CFST_SIGNATUREIMG
- **captureFieldSubtypeChoice** (Array[string], optional)
- **clickedIndex** (integer, optional)
- **custom** (string, optional)
- **editable** (boolean, optional)
- **imageUrlOriginal** (string, optional)
- **imgFormat** (string, optional): Possible values: JPEG, PNG, TIFF
- **imgHeight** (integer, optional): A list of all widgets of the field
- **imgWidth** (integer, optional)
- **items** (Array[RestListEntry], optional)
- **mandatory** (boolean, optional)
- **maxLength** (integer, optional)
- **multiLine** (boolean, optional)
- **multiSelection** (boolean, optional)
- **name** (string, optional)
- **noToggleToOff** (boolean, optional)
- **readOnly** (boolean, optional)
- **role** (string, optional)
- **sameValueUnison** (boolean, optional)
- **signatory** (string, optional)
- **signatureColor** (RestColorRGB, optional)
- **signed** (boolean, optional)
- **signerName** (string, optional)
- **signerText** (string, optional)

- **sort** (boolean, optional)
- **tooltip** (string, optional)
- **topIndex** (integer, optional)
- **type** (string, optional): Possible values: FT_UNKNOWN, FT_TEXT, FT_CHECKBOX, FT_LISTBOX, FT_COMBOBOX, FT_RADIOBUTTON, FT_CAPTURE
- **validateMessage** (string, optional)
- **validateRegEx**
- **value** (string, optional)
- **widgets** (Array[RestWidget], optional

RestListEntry

- **exportValue** (string, optional)
- **itemValue** (string, optional):
- **selected** (integer, optional)

RestColorRGB

- **blue** (integer, optional)
- **green** (integer, optional):
- **red** (integer, optional)

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page.
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page.
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page)
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page.

Clear a signature

This request clears a signature from a signature field.

URL

http://*host_server:port_number*/cirrus/rest/v8/documents/{documentid}/{fieldid}/signature

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **documentid** (string, required): The id of the document
- **fieldid** (string, required):The id of the Field (can be the id of the field or the PDF field name). See also fieldid_is_pdf_fieldname.

Query parameters

- **fieldid_is_pdf_fieldname** (boolean, required): If set to true, the fieldid specified is the PDF field name (and must be URL encoded). If unset or false, the fieldid is the id of the field.

Response status

Status 200 (OK): The signature was successfully cleared from the signature field. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Signing package requests

Get a list of packages

This request returns a list of signing packages or templates owned by the authenticated user or by a team member of a shared team. Filters can be applied to retrieve more specific signing packages. The result list can be limited by pagination parameters and contains only necessary information for further queries, because a complete list of signing packages can become very large. For a complete data set use the Get a single signing package request (/rest/v8/packages/{packageid}).

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages
```

Example header

```
Accept: application/json
```

Query parameters

- **packageTypeFilter** (string, optional): Whether to filter the list for the types signing package or template or not. Possible values: PACKAGE, TEMPLATE
- **startdate** (string, optional): Start date filter with yyyy-MM-dd format and user-specific time zone
- **enddate** (string, optional): End date filter with yyyy-MM-dd format and user-specific time zone
- **useddate** (string, optional): Target of the date filters. Possible values: CREATION, LASTUPDATE, COMPLETION, EXPIRATION, STARTDATE. Default value: LASTUPDATE
- **searchtext** (string, optional): Search text of signing package name or description. For information on case sensitivity, see *SignDoc Standard Installation Guide*, chapter "Database installation"
- **state** (Array[string, optional): The states filter of the signing package. More values can be provided. Possible values: draft, started, complete, rejected, expired, canceled, archived
- **allteams** (boolean, optional): Includes signing packages owned by members of all teams where the current user is a member too. Defaults to false which means that only the user's own signing packages are returned. Default value: false
- **team** (Array[string], optional): Includes packages owned by members of all provided teams where the current user is a member too. If the 'allteams' flag is set to true, it has higher priority.
- **page** (integer, optional): The page number to return (first=1)
- **limit** (integer, optional): The number of results to be returned in one page

Response class

Status 200 (OK): The signing packages were queried successfully.

Status 206 (PARTIAL_CONTENT): The signing packages were queried successfully but the number of returned signing packages was reduced to the allowed limit.

Status 404 (NOT_FOUND): The signing packages were queried successfully but the result is empty.

The response header contains more information about pagination:

- 'x-total-count' contains the total amount of signing packages available
- 'link' provides information about further navigation according to RFC5888

RestPackageListEntry

- **currentSigner** (string, optional): The ID of the signer which is currently locking the package
- **description** (string, optional): The description of the package
- **documentEntries** (Array[RestDocumentListEntry], optional): A list of all documents contained in the package (short info)
- **expirationDate** (string, optional): The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-3T10:15:30Z' (ISO-8601) id (string, optional): The id of the package lastUpdateTime (string, optional): The last update time of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **id** (string, optional): The id of the package
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package
- **name** (string, optional): The name of the package
- **owner** (string, optional): The owner id of the package
- **signerEntries** (Array[RestSignerListEntry], optional): A list of all signers contained in the package

- **state** (string, optional): The state of the package. Possible values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED
- **type** (string, optional): The type of the package. Possible values: PACKAGE, TEMPLATE

RestDocumentListEntry

- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The document message
- **fileName** (string, optional): The file name of the document
- **id** (string, optional): The id of the document
- **name** (string, optional): The name of the document
- **order** (integer, optional): The order of the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string
- **url** (string, optional): The request url for the entire document

RestSignerListEntry

- **accessCodeDeliveryChannel** (string, optional): The channel name by which authentication code will be sent to the signer
- **authenticationMode** (string, optional): The id authentication mode the signer. Possible values: NONE, CODE, EXTAUTH
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **email** (string): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **name** (string, optional): The name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **relatedUser** (string, optional): The identifier of the related SignDoc user
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (integer, optional): The stage of the signer
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE'
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **unreadDocuments** (Array[string], optional): List of documents ids needed to be reviewed by the signer
- **url** (string, optional): The request URL for the entire signer info

RestTSPSignatureDocument

- **documentId** (string): Id of the document
- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

Get a single signing package

This request returns a single signing package specified by a given account id. Because a signing package usually contains multiple documents and can become quite large, a separate list with RestDocumentListEntry objects was introduced to limit the needed size. A RestDocumentListEntry contains only the necessary information about a document as well as a separate rest url parameter to query the entire document in a separate request.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "ebab2946-0abb-46c0-9a18-c15c7eed6060",
  "name": "Hello Template without TSP",
  "description": "A simple template",
  "type": "PACKAGE",
  "processingType": "PAR",
  "state": "DRAFT",
  "auditTrailOptions": 3,
  "mailSubject": "MailSubject - An request via Kofax SignDoc",
  "mailMessage": "MailMessage - You are invited to sign",
```

```
  "lastUpdateTime": "2019-05-08T13:50:03.269Z",
  "creationTime": "2019-05-08T13:50:03.23Z",
  "auditTrailUrl": "http://localhost:6611/cirrus/rest/v8/packages/
ebab2946-0abb-46c0-9a18-c15c7eed6060/audittrail",
  "documentEntries": [
    {
      "id": "document-1",
      "name": "Main Doc",
      "fileName": "Hello.pdf",
      "url": "http://localhost:6611/cirrus/rest/v8/packages/ebab2946-0abb-46c0-9a18-
c15c7eed6060/documents/document-1",
      "thumbnail": " ... base64 encoded string ...",
      "order": 1
    }
  ],
  "signerEntries": [
    {
      "id": "signer-1",
      "name": "Important Signer",
      "email": "john@doe.com",
      "order": 1,
      "role": "SIGNER",
      "state": "ASSIGNED",
      "authenticationMode": "NONE",
      "url": "http://localhost:6611/cirrus/rest/v8/packages/ebab2946-0abb-46c0-9a18-
c15c7eed6060/signers/signer-1",
      "gdprConsentRequired": false,
      "esignConsentRequired": true,
      "tspSignatureDocuments": []
    }
  ]
}
```


Path parameters

- **packageid** (string, required): The id of the signing package to query

Query parameters

- **useIntId** (boolean, optional): The provided packageid is the internal id which was returned by get audit trail method. Default value: false

Response status

Status 200 (OK): The signing package was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

In case of a successful query the response body contains the following information as RestSigningPackageOutput structure.

RestSigningPackageOutput

- **auditTrailOptions** (integer, optional): The audit trail options of the package
- **auditTrailUrl** (string, optional): The URL to retrieve audit trail for the package
- **completionTime** (string, optional): The completion date-time of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **currentSigner** (RestSignerOutput, optional): The current signer of the package

- **custom** (string, optional): Custom field
- **description** (string, optional): The description of the package
- **documentEntries** (Array[RestDocumentListEntry], optional): A list of all documents contained in the package (short info)
- **expirationDate** (string, optional): The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **id** (string, optional): The ID of the package
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package
- **lastUpdateTime** (string, optional): The last update time of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **mailMessage** (string, optional): The mail message of the package
- **mailSubject** (string, optional): The mail subject of the package
- **name** (string, optional): The name of the package
- **owner** (RestUserInfo, optional): The owner's information
- **processingType** (string, optional): The processing type of the package. Possible values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default), STAGED - sequential and parallel notification of signers based on staged value
- **reminderEntries** (Array[RestReminderOutput], optional): A list of all reminders contained in the package
- **signerEntries** (Array[RestSignerListEntry], optional): A list of all signers contained in the package
- **startDate** (string, optional): The start date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **state** (string, optional): The state of the package. Possible values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED
- **timeStarted** (string, optional): The time when the package started. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **type** (string, optional): The type of the package. Possible values: PACKAGE, TEMPLATE

RestSignerOutput

- **accessCodeDeliveryChannel** (string, optional): The channel type by which authentication code will be sent to the signer
- **accessCodeDeliveryParameters** (Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationTime** (string, optional): The authentication time of the signer. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer

- **completionTime** (string, optional): The completion time of the signer. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignerInfo** (RestTSPSignerInfoOutput, optional): TSP-related info for the signer to sign documents using TSP

RestDocumentListEntry

- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The document message
- **fileName** (string, optional): The file name of the document
- **id** (string, optional): The id of the document
- **name** (string, optional): The name of the document
- **order** (integer, optional): The order of the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string
- **url** (string, optional): The request url for the entire document

RestUserInfo

- **company** (string, optional): The company name of the user
- **email** (string, optional): The email of the user
- **id** (string, optional): The ID of the user

- **name** (string, optional): The name of the user

RestReminderOutput

- **days** (integer): The days of the reminder
- **id** (string, optional): The id of the reminder
- **sendDate** (string, optional): Date when the reminder shall be send. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **type** (string, optional): The type of the reminder. Possible values: AFTER_SEND, BEFORE_EXPIRES

RestSignerListEntry

- **accessCodeDeliveryChannel** (string, optional): The channel name by which authentication code will be sent to the signer
- **authenticationMode** (string, optional): The id authentication mode the signer. Possible values: NONE, CODE, EXTAUTH
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **email** (string): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **name** (string, optional): The name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **relatedUser** (string, optional): The identifier of the related SignDoc user
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (string, optional): The stage of the signer
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **unreadDocuments** (Array[string], optional): List of documents ids needed to be reviewed by the signer
- **url** (string, optional): The request URL for the entire signer info

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoOutput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP

- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestTSPSignatureDocument

- **documentId** (string): Id of the document
- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Get audit trail

This request returns the audit trail for a signing package specified by a given signing package id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/audittrail

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/audittrail
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
[
  {
```

```
   "message": "The signing package Hello Template without TSP owned by
user01@kofax.com has been created.",
   "workflowEvent": "PKG_CREATED",
   "creationTime": "2019-05-13T06:44:46.147Z",
   "intUserId": 4,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "Recipient Important Signer in signing package Hello Template without
TSP owned by user01@kofax.com has been notified.",
   "workflowEvent": "SIG_NOTIFIED",
   "creationTime": "2019-05-13T06:44:56.216Z",
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The signer Important Signer in signing package Hello Template without
TSP has been manually authenticated by passport.",
   "workflowEvent": "SIG_MANUALY_AUTHENTICATED",
   "creationTime": "2019-05-13T06:44:59.904Z",
   "intUserId": 4,
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "A common signing session has been authenticated successfully for
signing package Hello Template without TSP owned by Sandor Clegane .",
   "workflowEvent": "SIG_COMMON_SESSION_AUTHENTICATION_SUCCEEDED",
   "creationTime": "2019-05-13T06:45:02.209Z",
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The signing package Hello Template without TSP owned by
user01@kofax.com has been started.",
   "workflowEvent": "PKG_STARTED",
   "creationTime": "2019-05-13T06:45:02.21Z",
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The signer Important Signer has been authenticated successfully for
the common signing session of signing package Hello Template without TSP owned by
user01@kofax.com.",
   "workflowEvent": "SIG_COMMON_SESSION_SIGNER_AUTHENTICATION_SUCCEEDED",
   "creationTime": "2019-05-13T06:45:04.287Z",
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "Recipient Important Signer in signing package Hello Template without
TSP owned by user01@kofax.com has agreed to the e-sign consent of the package.",
   "workflowEvent": "SIG_AGREE_ESIGN_CONSENT",
   "creationTime": "2019-05-13T06:45:06.099Z",
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "Recipient Important Signer in signing package Hello Template without
TSP owned by user01@kofax.com has visited page 1.",
```

```
   "workflowEvent": "SIG_VISITED_PAGE",
   "creationTime": "2019-05-13T06:45:09.343Z",
   "intSignerId": 13695,
   "intDocumentId": 14129,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "Recipient Important Signer in signing package Hello Template without
TSP owned by user01@kofax.com has saved the document.",
   "workflowEvent": "SIG_SAVE_DOCUMENT",
   "creationTime": "2019-05-13T06:45:14.709Z",
   "intSignerId": 13695,
   "intDocumentId": 14129,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The signer Important Signer in signing package Hello Template without
TSP owned by user01@kofax.com has completed the signing package.",
   "workflowEvent": "REC_COMPLETED",
   "creationTime": "2019-05-13T06:45:16.179Z",
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The signing package Hello Template without TSP owned by
user01@kofax.com has been completed.",
   "workflowEvent": "PKG_COMPLETED",
   "creationTime": "2019-05-13T06:45:16.182Z",
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The user Sandor Clegane received an email concerning the fact that
signer Important Signer has completed their package part.",
   "workflowEvent": "USR_MAIL_SIGNER_COMPLETE",
   "creationTime": "2019-05-13T06:45:16.258Z",
   "intUserId": 4,
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "The user Sandor Clegane received an email that signing package Hello
Template without TSP is complete.",
   "workflowEvent": "USR_MAIL_PACKAGE_COMPLETE",
   "creationTime": "2019-05-13T06:45:16.723Z",
   "intUserId": 4,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
   "message": "An email was send to the signer Important Signer in signing package
Hello Template without TSP owned by Sandor Clegane due to package complete state.",
   "workflowEvent": "SIG_MAIL_PACKAGE_COMPLETE",
   "creationTime": "2019-05-13T06:45:16.86Z",
   "intSignerId": 13695,
   "intPackageId": 14313,
   "intAccountId": 51
 },
 {
```

```
    "message": "A signing session was closed for the signing package Hello Template
 without TSP owned by user01@kofax.com.",
    "workflowEvent": "PKG_SIGNING_SESSION_CLOSED",
    "creationTime": "2019-05-13T06:45:17.61Z",
    "intPackageId": 14313,
    "intAccountId": 51
  }
]
```


Path parameters

- **packageid** (string, required): The id of the signing package to query the audit trail from

Query parameters

- **locale** (string, optional): Locale for retrieving the locale-specific audit logs (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en-US will be used
- **names** (string, optional): The flag indicates whether performed by / document names should be returned

Response class

Status 200 (OK): With list of RestAuditTrailOutput entries if the signing package audit trail was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAuditTrailOutput

- **creationTime** (string, optional): The creation date of the audit trail. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **documentName** (string, optional): The name of the document
- **intAccountId** (integer, optional): The internal ID of the account
- **intDocumentFieldId** (integer, optional): The internal ID of the document field
- **intDocumentId** (integer, optional): The internal ID of the document
- **intPackageId** (integer, optional): The internal ID of the package
- **intSignerId** (integer, optional): The internal ID of the signer
- **intUserId** (integer, optional): The internal ID of the user
- **message** (string, optional): The message of the audit trail
- **performedBy** (string, optional): The person who performed this action
- **workflowEvent** (string, optional): The workflow event of the audit trail. Possible values: PKG_CREATED, PKG_STARTED, PKG_EXPIRED, PKG_COMPLETED, PKG_DELETED, PKG_NAME_CHANGED, PKG_REMINDER_ADDED, PKG_REMINDER_DELETED, PKG_SIGNING_SESSION_CLOSED, SIG_NOTIFIED, SIG_REMINDED, REC_COMPLETED, SIG_OPEN_PKG, SIG_DECLINED, SIG_AUTHENTICATION_FAILED, SIG_AUTHENTICATION_SUCCEEDED, SIG_COMMON_SESSION_AUTHENTICATION_SUCCEEDED, SIG_COMMON_SESSION_SIGNER_AUTHENTICATION_SUCCEEDED, SIG_REMOTE_SESSION_AUTHENTICATION_SUCCEEDED, SIG_REMOTE_SESSION_CODE_AUTHENTICATION_SUCCEEDED, SIG_REMOTE_SESSION_EXTAUTH_STARTED, SIG_REMOTE_SESSION_EXTAUTH_SUCCEEDED, SIG_REMOTE_SESSION_EXTAUTH_FAILED, SIG_REMOTE_SESSION_EXTAUTH_ERROR, SIG_REMOTE_SESSION_EXTAUTH_CANCEL, SIG_AGREE_ESIGN_CONSENT, SIG_AGREE_GDPR_CONSENT, SIG_OPEN_DOCUMENT,

SIG_MANUALY_AUTHENTICATED, SIG_SIGNED, SIG_CLEARED_SIGNATURE, SIG_CHECKBOX_CHECKED, SIG_CHECKBOX_UNCHECKED, SIG_TEXTBOX_CHANGED, SIG_VISITED_PAGE, SIG_SAVE_DOCUMENT, SIG_TSP_DOCUMENT_SIGNED, SIG_TSP_DOCUMENT_SIGNATURE_FAILED, SIG_TSP_DOCUMENT_SIGNATURE_CANCELLED, SIG_SUPPLEMENTAL_DOCUMENT_ADD, SIG_SUPPLEMENTAL_DOCUMENT_DELETE, USR_MAIL_PASSWORD_FORGOTTEN, USR_MAIL_ERR_PASSWORD_FORGOTTEN, USR_MAIL_RESET_PASSWORD, USR_MAIL_ERR_RESET_PASSWORD, USR_MAIL_CHANGE_PASSWORD, USR_MAIL_ERR_CHANGE_PASSWORD, USR_MAIL_ACCOUNT_INVITED, USR_MAIL_ERR_ACCOUNT_INVITED, USR_MAIL_ADD_TO_TEAM_INVITED, USR_MAIL_ERR_ADD_TO_TEAM_INVITED, USR_MAIL_SIGNER_COMPLETE, USR_MAIL_ERR_SIGNER_COMPLETE, USR_MAIL_REVIEWER_COMPLETE, USR_MAIL_ERR_REVIEWER_COMPLETE, USR_MAIL_PACKAGE_COMPLETE, USR_MAIL_ERR_PACKAGE_COMPLETE, SIG_MAIL_ERR_NOTIFY, SIG_MAIL_ERR_REMINDER, SIG_MAIL_ERR_REMIND_MAIL, SIG_MAIL_PACKAGE_COMPLETE, SIG_MAIL_ERR_PACKAGE_COMPLETE, REC_MAIL_DOCUMENTS_AS_ARCHIVE, REC_MAIL_ERR_DOCUMENTS_AS_ARCHIVE, REC_MAIL_REMIND, REC_MAIL_ERR_REMIND, MAIL_REMIND, MAIL_ERR_REMIND, SIG_EMAIL_CHANGED, PKG_STATE_PLUGIN_STARTED, PKG_STATE_PLUGIN_ENDED, SIG_STATE_PLUGIN_STARTED, SIG_STATE_PLUGIN_ENDED

Create a new express signing package

This request allows to create an express signing package for a single document with one signer. To create an express package a user needs to provide a document to upload with or without document fields. Optionally the user can supply the signer name, signer email, package id and a document name. If a package already exists with the package id, an error is thrown if 'cirrus.rest.expresspackage.delete-existing' is set to false.

If the user does not supply a signer name the signer is created using a default location-specific signer name from the configuration 'cirrus.rest.expresspackage.signer.default.name'. All the fields are then assigned to this signer.

The response contains the package id, the location URL of the package created and the common signing session URL. The package created has state 'PREPARED', the user can edit the package from the SignDoc Manage Client as long as it is not started. If the signer email is supplied in the request, a notification is sent to the signer with a link to the remote signing session.

Note For this request a valid authentication with USER role is necessary.

The following configurations are used internally when creating the express package and should be set from the SignDoc Manage Client.

- **cirrus.rest.expresspackage.auto-prepare**
  Default auto-prepare option for documents.
- **cirrus.rest.expresspackage.delete-existing**
  Default delete-existing option for signing package.
- **cirrus.rest.expresspackage.signaturefield.required.default**
  Default 'required' flag for signature fields.
- **cirrus.rest.expresspackage.signer.authentication.default**
  Default signer authentication method.

- **cirrus.rest.expresspackage.signer.default.name**
  Signer default name.

URL

http://*host_server:port_number*/cirrus/rest/v8/expresspackage

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

multipart/form-data

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/expresspackage
```

Example header

```
Accept: application/json
```

Query parameters

- **document** (formData, required): The document to be added to the signing package.
- **id** (string, optional): The id of the signing package.
- **documentName** (string, optional): The name of the document.
- **signerName** (string, optional): The name of the signer to whom the document fields will be assigned.
- **signerEmail** (string, optional): Email address of the signer.
- **Locale** (string, optional): Locale for retrieving the locale-specific signer name configuration setting (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en will be used. This setting will only be used if signer name is not provided in the request.

Response status

Status 201 (Created): The signing package was created, otherwise a SignDoc Standard status code is returned together with the explaining messages.

Example response body (JSON)

```
{ "id": "5dd81cb7-ae92-4092-8510-41245c51fe58" ,
```

```
"url": "http://localhost:6611/cirrus/rest/v8/packages/5dd81cb7-
ae92-4092-8510-41245c51fe58" ,
"commonSigningUrl": "http://localhost:6611/cirrus/signing-client?pid=
5dd81cb7-ae92-4092-8510-41245c51fe58&auth=c1c91fac-975b-4ec0-
b6ff-6b7b6775623c&signtype=COMMON" }
```

Example response body (XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:RestID xmlns:ns2="http://www.kofax.com/ksd/cirrus/rest/v8">
<ns2:ID>dd6c8382-1dab-49f3-8e2b-1ba9a5d3bddb</ns2:ID>
<ns2:url>http://localhost:6611/cirrus/rest/v8/packages/
dd6c8382-1dab-49f3-8e2b-1ba9a5d3bddb</ns2:url>
<ns2:commonSigningUrl>http://localhost:6611/cirrus/signing-client?
pid=dd6c8382-1dab-49f3-8e2b-1ba9a5d3bddb&amp;
auth=c2566fff-2e36-40ec-929e-ba8bb61384ff&amp;signtype=COMMON</ns2:commonSigningUrl>
</ns2:RestID>
```

Create a new signing package

It is possible to create a signing package from scratch or as a copy of another already existing signing package.

If you want to create a signing package on base of another signing package you have to define either the source signing package identifier (query parameter) src_id, which is returned as id in a RestPackageListEntry object returned by a Get a list of packages request.

To create a signing package from scratch, an input JSON or XML string specifying the details of the signing package, has to be provided in a request body. This request prepares a default signing package if nothing further is specified.

To immediately add documents to the signing package a RestDocumentInput object has to be added (see Example body).

To immediately assign a signer to a field use the id attribute of the signer to make a cross reference (see Example body).

In Example body a signer with id "signer1" is created and later cross-referenced in the signer attribute of the signature field. The signing package is still created without assigning a signer to the signature field, if the cross reference is incorrect.

Note For this request a valid authentication with USER role is necessary.

If schedule=true is set as query parameter then the signing package is scheduled directly after successful creation if all criteria for scheduling are satisfied. In this case the state of the package is set to PREPARED and all signers with an email address will be notified.

URL

http://*host_server:port_number*/cirrus/rest/v8/package

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/package
```

Example header

```
Accept: application/json
```

Query parameters

- **src_id** (string, optional): The id of the source signing package which is used as base for the new signing package.
- **schedule** (boolean, optional): Schedules the signing package immediately after creation if all start criteria are fulfilled.
- **clean_fields** (boolean, optional): Clears the content of defined textfields and checkboxes. Only valid in conjunction with src_id.
- **delete_existing** (boolean, optional): Deletes existing signing package with the same id before creating a new signing package.
- **autoprepare** (boolean, optional): Whether the signing package should be auto prepared or not.
- **prepare** (boolean, optional): Whether the signing package should be set in PREPARED state or stay in DRAFT (default) state.
- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi

Body parameters

- **restSigningPackageInput** (restSigningPackageInput, required): Represents a RestSigningPackageInput object either in JSON or XML. See Signing package properties overview.

Response status

Status 201 (Created): The signing package was successfully added, otherwise a SignDoc Standard status code is returned together with the explaining messages.

Example response body (JSON)

```
{
  "id": "539be1b9-3025-456b-9d1f-5e78caeed289",
  "url": "http://localhost:6611/cirrus/rest/v8/
packages/539be1b9-3025-456b-9d1f-5e78caeed289"
}
```

Example response body (XML)

```
?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:RestID xmlns:ns2="http://www.kofax.com/ksd/cirrus/rest/v8">
  <ns2:ID>240bc490-1254-41c5-861d-2321b5ababd6</ns2:ID>
  <ns2:url>http://spci-2-2-0-test.ci.sdlabs.de/cirrus/rest/v8/
packages/240bc490-1254-41c5-861d-2321b5ababd6</ns2:url>
</ns2:RestID>
```

Example body (JSON)

```
{
   "expirationDate": "2025-12-03T10:15:30Z",
   "description": "description",
   "processingType": "SEQ",
   "mailSubject": "Insurance application form",
   "documents": [
      {
         "content": "... pdf document as base64 encoded string ..."",
         "signatureFields": [
            {
               "signerId": "signer-1",
               "id": "signature-1",
               "signingModeOptions": [
                  "PH",
                  "C2S"
               ],
               "widgets": [
                  {
                     "top": 690,
                     "left": 199,
                     "pageNumber": 1,
                     "right": 348,
                     "bottom": 644
                  }
               ]
            }
         ],
         "id": "document-1",
         "textFields": [
            {
               "signerId": "signer-1",
               "maxLength": 1024,
               "widgets": [
                  {
                     "top": 692,
                     "left": 198,
                     "pageNumber": 1,
                     "right": 348,
                     "bottom": 644
                  }
               ],
               "multiLine": false,
               "id": "textfield-3",
               "required": "true"
            },
            {
               "signerId": "signer-1",
               "maxLength": 1024,
               "widgets": [
                  {
                     "top": 693,
                     "left": 198,
                     "pageNumber": 1,
                     "right": 348,
                     "bottom": 644
```

```
                }
             ],
             "multiLine": false,
             "id": "textfield-2",
             "required": "true"
          },
          {
             "signerId": "signer-1",
             "maxLength": 1024,
             "widgets": [
                {
                   "top": 694,
                   "left": 198,
                   "pageNumber": 1,
                   "right": 348,
                   "bottom": 644
                }
             ],
             "multiLine": false,
             "id": "textfield-1",
             "required": "true"
          }
       ],
       "fileName": "Insurance_application_form.pdf",
       "format": "PDF",
       "name": "Insurance application form",
       "checkboxFields": [
          {
             "signerId": "signer-1",
             "id": "checkbox-3",
             "widgets": [
                {
                   "top": 691,
                   "left": 198,
                   "pageNumber": 1,
                   "right": 348,
                   "bottom": 644
                }
             ],
             "required": "false"
          },
          {
             "signerId": "signer-1",
             "id": "checkbox-2",
             "widgets": [
                {
                   "top": 692,
                   "left": 197,
                   "pageNumber": 1,
                   "right": 348,
                   "bottom": 644
                }
             ],
             "required": "false"
          },
          {
             "signerId": "signer-1",
             "id": "checkbox-1",
             "widgets": [
                {
                   "pageNumber": 1,
                   "top": 693,
                   "left": 196,
                   "right": 348,
```

```
                    "bottom": 644
                }
             ],
             "required": "false"
          }
       ]
    }
 ],
 "startDate": "2024-04-21T00:00:00Z",
 "signingModeOptions": [
    "PH",
    "C2S"
 ],
 "custom": "custom",
 "type": "PACKAGE",
 "signers": [
    {
       "role": "SIGNER",
       "id": "signer-1",
       "name": "Laura Wilson",
       "email": "laura.wilson@email.com"
    }
 ],
 "auditTrailOptions": "3",
 "reminders": [
    {
       "days": 3,
       "id": "aaa",
       "type": "AFTER_SEND"
    }
 ],
 "mailMessage": "Your insurance application form is ready for signing",
 "name": "Insurance Application"
}
```

Create signing package detailed description

To create a signing package via the SignDoc Standard REST API a user has several different options.

It is possible to create a signing package

- from scratch, which can be done by specifying a new signing package data structure in the request body, see Create signing package from scratch
- from an already existing template, see Create signing package from template
- from a template with metadata, see Create signing package from template with metadata
- by using a Word document with already aligned signature lines, see Create signing package from Word document

The following chapter will cover the most important approaches on how to create a signing package via the "Create signing package" request.

Parameters overview

Parameters in more detail

The **src_id** query parameter specifies the id of a template, a new signing package is supposed to be based on. This parameter enables a complete different approach to create a signing package and uses a signing package specified in the body parameter purely as metadata. A template usually has all documents signers and fields predefined. And a new signing package created with this option is an exact

copy of this template (except the id). The signing package data structure specified in the body is only applied as an update on the new signing package. This body parameter is only used to make intermediate changes on the new signing package. For example a signing package name can be changed via the body parameter. It also is possible to adjust a signer via the body. The body parameter is used to specify the signers with a predefined id in the template to directly map the email addresses to the signers. For a more detailed explanation of the general workflow consult Create signing package from template with metadata in this chapter.

The **schedule** query parameter (default false) causes the signing package to be scheduled immediately. The recipients (defined signers) are notified to sign the documents directly after creation. The signing package is evaluated before the actual scheduling. This means the specified signing package has to be in a consistent and valid state.

If a signing package exists with the same id passed as an attribute in the signing package structure a "Create signing package" request will fail. This is the case, because signing packages with the same id are not allowed within SignDoc Standard. With the **delete_existing** option a user is able to automatically delete any signing package associated to the same id as specified in signing package data structure immediately. SignDoc Standard deletes the signing package with the same id (if exists) before creating the new signing package in one request. If no id is specified in the signing package input data structure the query parameter delete_existing is ignored.

The **autoprepare** parameter is used to automatically prepare a specified document, by adding assigned signature fields for each provided signer to the document.

The body parameter is used to specify a complete signing package as either a JSON or XML structure. A new signing package is created according to the defined values. This parameter is additionally used to map data directly to already defined Signers in a signing package template. For more details consult Create signing package from template with metadata of this chapter.

Parameter use cases

Parameter	Create signing package from scratch	Create signing package from template	Create signing package with Word document
src_id	no	yes	yes
schedule	yes	yes	yes
delete_existing	yes	yes	yes
autoprepare	yes	yes	yes

Signing package properties overview

RestSigningPackageInput

- **auditTrailOptions** (integer, optional): The audit trail options of the final compound pdf document after package is completed. 0 - no audit trail is added, 1 - add signing package audit trail, 2 - add document audit trail, 3 - both (default)
- **custom** (string, optional): Custom field
- **description** (string, optional): The description of the package
- **documents** (Array[RestDocumentInput], optional): The list of all documents contained in the package
- **expirationDate** (string, optional): The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)

- **id** (string, optional): The ID of the package
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package
- **mailMessage** (string, optional): The mail message of the package for signer notification
- **mailSubject** (string, optional): The mail subject of the package for signer notification
- **name** (string, optional): The name of the package
- **processingType** (string, optional): The processing type of the package. Possible values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default), STAGED - sequential and parallel notification of signers based on staged value
- **reminders** (Array[RestReminderInput], optional): The list of package specific email reminders for the signers
- **signers** (Array[RestSignerInput], optional): The list of all package specific signers
- **signingModeOptions** (Array[string], optional): List of the default signing modes for signature fields created automatically on base of signature lines in Word documents. The parameter is only valid on signing package creation for Word documents included in the same package
- **startDate** (string, optional): The start date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **state** (string, optional): The state of the package. Possible values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED or ARCHIVED. Note: For package creation it can be set either to DRAFT (default) or PREPARED (if all requirements are fulfilled).
- **type** (string, optional): The type of the package, can be PACKAGE (default) or TEMPLATE

RestDocumentInput

- **checkboxFields** (Array[RestCheckboxFieldInput], optional): A list of all checkboxes contained in the document
- **content** (string, required): The actual document in Base64 format
- **custom** (string, optional): Custom field
- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The message displayed when the recipient opens the document
- **fileName** (string, optional): The filename of the document. Note: If no file name is provided during package creation a file name is built from the document name and a UUID with pdf extension (e.g. 'Insurance application form c5bee274-129a-4160-902c-3813aec53d4b.pdf'). If also no document name is provided the file name is a UUID with pdf extension (e.g. '942bcdc8-65f1-4194-812e-406a338c9256.pdf').
- **format** (string, optional): The document format. Possible values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **id** (string, optional): The id of the document
- **name** (string): The name of the document
- **order** (integer, optional): The order of the document. The documents will be displayed in the SignDoc clients in ascending order.
- **signatureFields** (Array[RestSignatureFieldInput], optional): A list of all signature fields contained in the document
- **textFields** (Array[RestTextFieldInput], optional): A list of all text fields contained in the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string

RestReminderInput

- **days** (integer, required): The days of the reminder
- **id** (string, optional): The id of the reminder
- **type** (string, optional): The type of the reminder. Possible values: = ['AFTER_SEND', 'BEFORE_EXPIRES']

RestSignerInput

- **accessCodeDeliveryParameters**(Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided.
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries. Note: The access code delivery plugin can be used for sending an access code to a recipient for a 2-factor authentication.
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationParam** (string, optional): The authentication code for the signer
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer. The signer will be notified in ascending order if sequential package processing is selected. Note: Sequential processing for a package can be set via processingType=SEQ in the RestSigningPackageInput structure.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47) Note: The preferred signer language overwrites the account specific communication language. This setting determines the used language in the emails to the signer.
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5

  Explanation:

  R1: There is a problem with the document(s)

  R2: I do not recognize the sender

  R3: I do not want to sign online

  R4: I do not agree with the terms of the e-sign consent

  R5: I do not agree with the terms of the GDPR statement

- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input. Note: If a SignDoc user is a signer within a signing package it is possible to allow him to use a locally stored image as signature input. In this case the relatedUser attribute must contain the SignDoc userid during creation of a signer. If the signer is requested later then the attribute relatedUser contains another value than the entered userid (for security reason).
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plugin it is possible to let the signer sign with a qualified certificate.

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox, true is checked, false is unchecked (default)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingModeOptions** (Array[string], optional): A list of the signing modes allowed for the signer. Signing modes can be HW for a handwritten signature with a tablet or using HTML5, PH for a photo captured by camera, C2S for typing in the name using Click-to-Sign, IMG for an (up-)loaded image, STAMP for an (up-)loaded stamp image, 'TSP' sign with trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plugin is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestTextFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field

- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The maximum length of the text value
- **multiLine** (boolean, optional): Whether the field is a multi-line text field or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoInput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP
- **tspSignatureDocuments** (Array[string], optional): List of Ids of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page).
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **selected** (boolean, optional): Whether the field allows multiple lines or not
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Create signing package scenarios

Create signing package from scratch

This scenario will create an easy signing package with one signer and signature field. The only thing needed is to specify a complete signing package in the body parameter. To create a signing package use the sample signing package structure provided below.

Body parameter (JSON)

```
{
   "documents": [
      {
         "content": "... pdf document as base64 encoded string ...",
         "signatureFields": [
            {
               "signerId": "signer-1",
               "id": "signature-1",
               "signingModeOptions": [
                  "PH",
                  "C2S"
               ],
               "widgets": [
                  {
                     "top": 690,
                     "left": 199,
                     "pageNumber": 1,
                     "right": 348,
                     "bottom": 644
                  }
               ]
            }
         ],
         "id": "document-1",
         "fileName": "Insurance_application_form.pdf",
         "format": "PDF",
         "name": "Insurance application form"
      }
   ],
   "signingModeOptions": [
      "PH",
      "C2S"
   ],
   "type": "PACKAGE",
   "signers": [
      {
         "role": "SIGNER",
         "id": "signer-1",
         "name": "Laura Wilson",
         "email": "laura.wilson@email.com"
      }
   ],
   "name": "Insurance Application"
}{
```

Make sure you add a Base64-encoded document in the document content. To immediately send the created signing package it is only needed to specify the schedule query parameter as true. If it is needed to delete an already existing signing package with the same id you can do that by specifying the delete_existing query parameter as true.

Create signing package from template

In this scenario a signing package will be created from an already existing template. To do that the src_id parameter has to be specified with an existing template id.

Example for creating a template

```
POST http://localhost:6611/cirrus/rest/v8/package?
schedule=false&delete_existing=false&autoprepare=false&resolution=72
```

Body parameter (JSON)

```
{
  "id": "template-1",
  "name": "Sample Template",
  "description": "A simple template",
  "type": "TEMPLATE",
  "documents": [
    {
      "id": "document-1",
      "name": "Test Doc",
      "content": "... pdf document as base64 encoded string ...",
      "signatureFields": [
          {
             "signerId": "signer-1",
             "id": "signature-1",
             "signingModeOptions": [
                "PH",
                "C2S",
    "HW",
    "IMG"
             ],
             "widgets": [
                {
                   "top": 690,
                   "left": 199,
                   "pageNumber": 1,
                   "right": 348,
                   "bottom": 644
                }
             ]
          }
       ],
      "fileName": "Test.pdf"
    }
  ],
 "signers": [
    {
       "role": "SIGNER",
       "id": "signer-1",
       "name": "Signer 1"
    }
 ]
}
```

Make sure you add a Bas64-encoded document in the document content. To immediately send the created signing package it is only needed to specify the schedule parameter as true. It additionally is possible to delete an already existing signing package by specifying the delete_existing parameter as true.

Create signing package from template with metadata

For this scenario a signing package will be created from template with specified metadata in the body. First a template with a document and signer has to be created.

After the successful creation of the template use the returned id in our next "Create signing package" request as the src_id parameter.

This new created signing package is a copy of the source template. The only difference is the new created id, because the package identifiers must be unique within an account, independent whether it is

a template or a signing package. You can modify the copy directly in the create package call in request body.

Example

The name of the package should be changed from "Sample Template" to "New package with signer email". The signer should get another name and a specific email address.

```
{
  "name": "New package with signer email",
  "signers": [
    {
     "id": "signer-1",
     "name": "John Doe",
     "email": "John@Doe.com"
    }
  ]
}
```

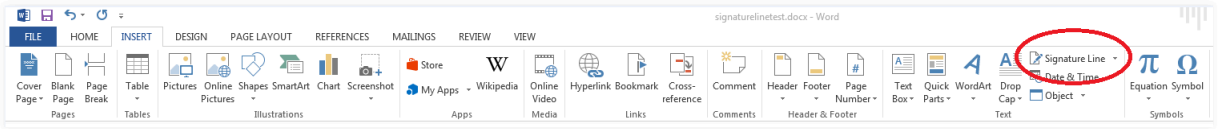
The mapping of signer name and email address is accomplished via id. In the previous signing package template the signer id as signer-1 was specified. This principal applies to every other properties of the signing package a user is allowed to change (see Signing package properties overview).

Also with create package on base of a template it is possible to schedule the package immediately (schedule=true) and to delete an already existing package with the same id before package creation (delete_existing=true).

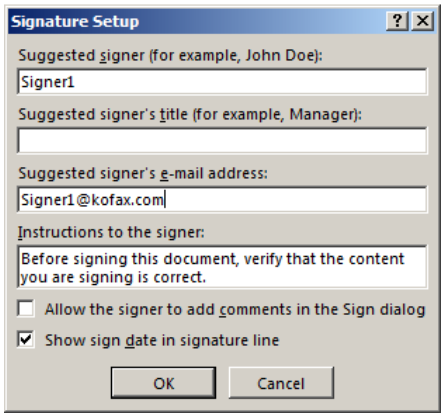
Create signing package from Word document

This scenario will use a Word document to create a signing package.

SignDoc Standard uses a separate Word feature to align signature fields and assign signers. Word provides a so called Signature Line Feature that enables a user to align a signature line to a certain position in the Word document and to embed metadata to it. This can be helpful, because aligning a signature field by coordinates alone can be quite a nuisance. The Signature Line Feature can be accessed in the Insert menu of Word on the right side of the submenu pane.



After clicking on the menu entry "Signature Line" a separate window pops up where a user can specify additional information like signer name, email address and instructions. The user should specify the suggested signer field (Signer1) and the suggested signer's email address (signer1@kofax.com).



After clicking the OK button a new signature line is added to the Word document.

A user who wants to prepare a document for SignDoc Standard has several different approaches.

1. Set signer name for a new SignDoc Standard signer. The specified signer name will be the id of the signer.
2. Set signer name for an existing signer (used for already existing signer in templates). The specified signer name has to match with the id of the signer in the template.
3. Don't set the signer name to generate an automatic and random id.

Additionally, a user can predefine a signer email address in the Signature Line. This can be used to map an email address to an already existing signer or to immediately assign an email address to a new signer. For more details, see C reate signing package from Word document .

MS WORD specific configuration

SignDoc offers currently two possibilities to set the OID of the signer which must be assigned to the new signature field.

The account specific configuration setting **cirrus.document.prepare.msword.signatureline.signerid.source** can contain two values, **field_name** (default) and **signer_name**.

If signer_name is provided the signer OID is set from the "Suggested signer" name.

Note The entered signer name must be conform to the allowed characters for an OID and must not contain spaces! Conform means characters a-z, A-Z, 0-9, '-' and '_'.

If field_name is provided the signer OID is set from the signature field name which is derived from the signature line tag id.

If a calling program can read the signature line tag id it is possible to reference the signer with this tag id (field name) directly.

In prior releases this setting value comes from the entry winword.field.identification.key in cirrus-(db).properties (with default signer_name).

The following tables will give an overview of the metadata provided with a signature line and the SignDoc Standard behavior if **cirrus.document.prepare.msword.signatureline.signerid.source** has value **signer_name**:

Signature line (MS Word)	SignDoc Standard Standard behavior
tag id	id of signer
Suggested signer	name of signer
Suggested signer's title	not used
Suggested signer's email address	email of signer

cirrus.document.prepare.msword.signatureline.signerid.source has value **field_name (default)**:

Signature line (MS Word)	SignDoc Standard Standard behavior
tag id	not used
Suggested signer	id of signer
Suggested signer's title	not used
Suggested signer's email address	email of signer

Principle process

1. The signer object id is created based on the value of the **cirrus.document.prepare.msword.signatureline.signerid.source**. If the value is field_name, the signer id becomes the name of the signature line which is the same as the signature line tag id. Otherwise, the signer id becomes the signature line assignee name. Note that the signature line assignee name can be empty, while the signature line tag id always has a value.
2. If the signer id and assignee email is not empty after the first step, a signer with that id is retrieved from the signers of the signing package.
    a. If there is a signer with the same id, the signature line will be bound to this signer.
    b. If there is no such signer, SignDoc Standard attempts to create a new signer in the signing package based on the signature line assignee email. If there are more signers with such an email, the one with assigned fields will be taken, otherwise the first one found. In case there are no signers found based on the signature line assignee email, a new signer is created and assigned to the signature line.
3. If the signer id is empty and there is a signature line assignee email, a signer is retrieved from the signing package based on the signature line assignee email.
    a. If there are more signers with such an email, the one with assigned fields will be taken, otherwise the first one found.
    b. In case there are no signers found based on the signature line assignee email, a new signer is created and assigned to the current signature line.

4. If the signer object id is empty and there is no signature line assignee email, a new signer with a random id is created and assigned to the current signature field.

In the next step the prepared Word document has to be converted to a Base64-encoded string and added to the following signing package structure.

Body parameter (JSON)

```
{
    "id": "package-1",
    "name": "Package 1",
    "type": "PACKAGE",
    "documents": [
      {
        "id": "document-1",
        "name": "Word Doc",
          "content": "... MS WORD document with signature line as base64 encoded
 string ...",
        "fileName": "Word document with signature line.docx"
      }
    ]
  }
```

After running the "Create signing package" request a new signing package is created with the given signer information in the signature line.

To immediately send the created signing package it is only needed to specify the schedule parameter as true. It additionally is possible to delete an already existing signing package by specifying the delete_existing parameter as true.

Schedule a signing package

This request evaluates the specified signing package and schedules it if it is in in an acceptable state.

The signing package must fulfill some conditions before it can be scheduled:

The most important conditions are:

1. The package must be from type PACKAGE (not TEMPLATE)
2. The current state of the package must be either DRAFT, PREPARED or STARTED
3. The package must have at least one document
4. At least one signer with a specified name must be defined for the package
5. A signer with role SIGNER must be assigned to at least one signature field in a document
6. Each defined signature field must be assigned to a signer
7. Each 'required' field must be assigned to a signer

Afterwards it triggers the signing package process and sends out the invitations for the signing package. Note that the time of the send out invitations depends on the specified start time of the signing package. A signing package with no start time will start immediately, a signing package with a start date will be set on hold until the start date has been reached. Note that a signing package has to be in a consistent state to be scheduled, otherwise validation errors will occur. In such cases all validation errors and detailed information on the problem will be returned in form of a list.

Note For this request a valid authentication with USER role is necessary. The request will be denied if there is at least one signer with role SIGNER who does not have any assigned signature field.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/scheduler

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/scheduler
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package

Response status

Status 200 (OK): The signing package was successfully started.

Send email to all signers

This request sends an email to all signers of a signing package which have a defined email address. If "include a link" to the signing package is not requested, a single email is composed for all signers. A separate email is composed for each signer if including a link is requested, because the link is signer-specific. The link can be clicked by a signer from the email in order to call the signing client for signing the documents of the package.

Note For this request a valid authentication with USER role is necessary.

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/email

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/signers/email
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example body (JSON)

```
{
  "subject":"Important documents to sign",
  "message":"Dear Mr. Doe, please sign the documents earliest possible."
}
```

Path parameters

- **packageid** (string, required): The id of the signing package for which the email shall be sent

Query parameters

- **includelink** (boolean, optional): Include a link to the signing package. Default value: false

Body parameters

- **subject** (string, required): The subject of the email to be sent.
- **message** (string, required): The message (body) of the email to be sent.

Response status

Status code 200 (OK): Is returned on success without further content. This only means that the mail has been successfully queued for delivery because mail delivery is an asynchronous process.

Send email to one signer

This request sends an email to one signer of a signing package. An optional link to the signing client for signing the package documents may be included with the mail.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/email

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/signers/email/1051
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example body (JSON)

```
{
  "subject":"Reminder",
  "message":"Hello Mr. Doe, please do not forget to sign the documents."
}
```

Path parameters

- **packageid** (string, required): The id of the signing package for which the email shall be sent.
- **signerid** (string, required): The id of the signer for which the email shall be sent.

Query parameters

- **includelink** (boolean, optional): Include a link to the signing package. Default value: false

Body parameters

- **subject** (string, required): The subject of the email to be sent.
- **message** (string, required): The message (body) of the email to be sent.

Response status

Status code 200 (OK): Is returned on success without further content. This only means that the mail has been successfully queued for delivery because mail delivery is an asynchronous process.

Prepare signing session

This request prepares a signing session. In case of signtype **common** it returns the link and optional the appropriate QR code for starting a 'common' (or 'in-person') signing session. All signers with an email are notified with an invitation for a 'remote' signing session if signtype **remote** is used.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signingsession/{signtype}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/signingsession/common
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package the user wants to create a signing session for
- **signtype** (string, required): Signing type of the signing session. Possible values: common, remote

Query parameters

- **userid** (string, optional): The id of the user which should be enabled to use the capture method 'Signature image' in an 'in-person' signing session. The user must be related to the signer (see also method for adding a signer Add signer to signing package).

Body parameters

- **commonSigningSessionInput** (RestCommonSigningSessionInput, optional): Represents a RestCommonSigningSessionInput object either in JSON or XML. Default value: null

RestCommonSigningSessionInput

- **manualSignerAuthentications** (Array[RestManualSignerAuthentication]): A list of manual signer authentications
- **qrCodeSpecifications** (RestQRCodeSpecifications, optional): QR code specification of the common signing session
- **unlockPackage** (boolean, optional): A flag whether the signing package must be unlocked. Note: A signing package is locked as soon as a signer starts a signing session. It is unlocked again if the signer finishes his signing session.

RestManualSignerAuthentication

- **signerId** (string): Id of the signer
- **passport** (boolean, optional): Passport identification
- **driverLicense** (boolean, optional): Driver license identification
- **visualVerification** (boolean, optional): Visual verification identification
- **other** (string, optional): Other methods of identification. An additional explaining text for a manual signer authentication can be added here.

RestQRCodeSpecifications

- **imageType** (string): Image type of the QR code. Possible values: JPG, GIF, PNG
- **width** (integer): Width of the QR code image (50 - 1000)
- **height** (integer): Height of the QR code image (50 - 1000)

Example body (JSON)

```
{
  "manualSignerAuthentications": [
    {
      "signerId": "signer-1",
      "passport": true,
      "visualVerification": true,
      "other": "Signer is personally known"
    }
  ],
  "qrCodeSpecifications": {
    "imageType": "JPG",
    "width": 200,
    "height": 200
  }
}
```

Response class

Status 200 (OK): The signing session was successfully prepared. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

In case of a successful query the response body contains the following information as RestCommonSigningSessionOutput structure:

RestCommonSigningSessionOutput

- **qrcode** (string, optional): Base64-encoded QR code of the common signing session
- **url** (string, optional): URL of the common signing session

Example response body (JSON)

```
{"url":"http://localhost:6611/cirrus/signing-client?pid=5d683948-8384-4267-
a35a-73b4a97355a5&auth=3f1465ef-436a-4a1a-a664-dde541e8c14d&signtype=COMMON",
"qrcode":"Base64 encoded QR-code string"}
```

Example response body (XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RestCommonSigningSessionOutput xmlns="http://www.kofax.com/ksd/cirrus/rest/v8">
<url>
http://localhost:6611/cirrus/signing-client?pid=5d683948-8384-4267-
a35a-73b4a97355a5&auth=3f1465ef-436a-4a1a-a664-dde541e8c14d&signtype=COMMON
</url>
<qrcode>
Base64 encoded QR-code string
</qrcode>
</RestCommonSigningSessionOutput>
```

Update a signing package

This request updates an existing signing package within an account using an input JSON or XML string. Each attribute of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id": "package-1",
  "name": "new package name",
  "type": "PACKAGE",
  "processingType": "SEQ"
}
```

Path parameters

- **packageid** (string, required): The id of the signing package the user wants to update
- **resolution** (float, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi

Body parameters

- **updPackage** (RestSigningPackageInput, required): Represents a RestSigningPackageInput object either in JSON or XML

RestSigningPackageInput

- **auditTrailOptions** (integer, optional): The audit trail options of the final compound pdf document after package is completed. 0 - no audit trail is added, 1 - add signing package audit trail, 2 - add document audit trail, 3 - both (default)
- **custom** (string, optional): Custom field
- **description** (string, optional): The description of the package
- **documents** (Array[RestDocumentInput], optional): The list of all documents contained in the package
- **expirationDate** (string, optional): The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **id** (string, optional): The ID of the package
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package
- **mailMessage** (string, optional): The mail message of the package for signer notification
- **mailSubject** (string, optional): The mail subject of the package for signer notification
- **name** (string, optional): The name of the package
- **processingType** (string, optional): The processing type of the package. Possible values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default)
- **reminders** (Array[RestReminderInput], optional): The list of package specific email reminders for the signers
- **signers** (Array[RestSignerInput], optional): The list of all package specific signers
- **signingModeOptions** (Array[string], optional): List of the default signing modes for signature fields created automatically on base of signature lines in Word documents. The parameter is only valid on signing package creation for Word documents included in the same package
- **startDate** (string, optional): The start date of the package. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **state** (string, optional): The state of the package. Possible values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED or ARCHIVED. Note: For package creation it can be set either to DRAFT (default) or PREPARED (if all requirements are fulfilled).
- **type** (string, optional): The type of the package, can be PACKAGE (default) or TEMPLATE

RestDocumentInput

- **checkboxFields** (Array[RestCheckboxFieldInput], optional): A list of all checkboxes contained in the document
- **content** (string, required): The actual document in Base64 format
- **custom** (string, optional): Custom field
- **description** (string, optional): The description of the document
- **documentMessage** (string, optional): The message displayed when the recipient opens the document
- **fileName** (string, optional): The filename of the document. Note: If no file name is provided during package creation a file name is built from the document name and a UUID with pdf extension (e.g. 'Insurance application form c5bee274-129a-4160-902c-3813aec53d4b.pdf'). If also no document name is provided the file name is a UUID with pdf extension (e.g. '942bcdc8-65f1-4194-812e-406a338c9256.pdf').
- **format** (string, optional): The document format. Possible values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **id** (string, optional): The id of the document
- **name** (string): The name of the document
- **order** (integer, optional): The order of the document. The documents will be displayed in the SignDoc clients in ascending order.
- **signatureFields** (Array[RestSignatureFieldInput], optional): A list of all signature fields contained in the document
- **textFields** (Array[RestTextFieldInput], optional): A list of all text fields contained in the document
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a base64 encoded string

RestReminderInput

- **days** (integer, required): The days of the reminder
- **id** (string, optional): The id of the reminder
- **type** (string, optional): The type of the reminder. Possible values: = ['AFTER_SEND', 'BEFORE_EXPIRES']

RestSignerInput

- **accessCodeDeliveryParameters**(Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided.
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries. Note: The access code delivery plugin can be used for sending an access code to a recipient for a 2-factor authentication.
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationParam** (string, optional): The authentication code for the signer
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **email** (string, optional): The email of the signer

- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **processingType** (string, optional): The processing type of the package. Possible values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default), STAGED - sequential and parallel notification of signers based on staged value
- **order** (integer, optional): The order of the signer. The signer will be notified in ascending order if sequential package processing is selected. Note: Sequential processing for a package can be set via processingType=SEQ in the RestSigningPackageInput structure.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47) Note: The preferred signer language overwrites the account specific communication language. This setting determines the used language in the emails to the signer.
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5

  Explanation:

  R1: There is a problem with the document(s)

  R2: I do not recognize the sender

  R3: I do not want to sign online

  R4: I do not agree with the terms of the e-sign consent

  R5: I do not agree with the terms of the GDPR statement
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input. Note: If a SignDoc user is a signer within a signing package it is possible to allow him to use a locally stored image as signature input. In this case the relatedUser attribute must contain the SignDoc userid during creation of a signer. If the signer is requested later then the attribute relatedUser contains another value than the entered userid (for security reason).
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plugin it is possible to let the signer sign with a qualified certificate.

RestCheckboxFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **checked** (boolean, optional): The state of the checkbox, true is checked, false is unchecked (default)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field

- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **widgets** (Array[RestWidget]): A list of all widgets of the field

RestSignatureFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **name** (string, optional): The name of the field (is only considered if the field is added, it cannot be updated!)
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **signingModeOptions** (Array[string], optional): A list of the signing modes which should be allowed for the signer can be HW for handwritten (Tablet or HTML5), PH for photo captured by camera, C2S for Click2Sign, IMG for signing with (up-)loaded image. Default value: HW, PH, C2S, IMG
- **widgets** (Array[RestWidgetl): A list of all widgets of the field

RestTextFieldInput

- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc)
- **description** (string, optional): The description of the field
- **id** (string, optional): The id of the field
- **maxLength** (integer, optional): The maximum length of the text value
- **multiLine** (boolean, optional): Whether the field is a multi-line text field or not
- **name** (string, optional): The name of the field
- **readOnly** (boolean, optional): Whether the field is read only or not
- **required** (boolean, optional): Whether the field is required or not
- **signerId** (string, optional): The id of the signer assigned to the field
- **value** (string, optional): The value of the field
- **widgets** (Array[RestWidget): A list of all widgets of the field

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoInput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP
- **tspSignatureDocuments** (Array[string], optional): List of Ids of documents needed to be signed by the signer using the required TSP

- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestWidget

- **bottom** (number): The bottom coordinate. The origin is in the bottom left corner of the page
- **index** (integer, optional): The index of the widget within the field (is not considered if a new field is added)
- **left** (number): The left coordinate. The origin is in the bottom left corner of the page
- **pageNumber** (integer): The index of the page on which this field occurs (1 for the first page).
- **right** (number): The right coordinate. The origin is in the bottom left corner of the page
- **selected** (boolean, optional): Whether the field allows multiple lines or not
- **tabIndex** (integer, optional): The tab index of the field (page-based output attribute)
- **top** (number): The top coordinate. The origin is in the bottom left corner of the page

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Response status

Status 200 (OK): The signing package was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a signing package

This request deletes a signing package as well as all signers, documents and fields associated with the specified signing package.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/1001
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package the user want to delete

Response status

Status 200 (OK): The signing package was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete expiration date of a signing package

This request deletes the expiration date of a signing package.

Note For this request a valid authentication with USER role is necessary. Any reminders which are from type "before expires" are not removed but the sendDate (in RestReminderOutput, retrievable by "GET single package") is reset to null.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/expirationdate

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/4711/expirationdate
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package whose expiration date should be deleted

Response status

Status 200 (OK): The expiration date of a signing package was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete start date of a signing package

This request deletes the start date of a signing package.

Note For this request a valid authentication with USER role is necessary. Any reminders which are from type "after sent" are not removed but the sendDate (in RestReminderOutput, retrievable by "GET single package") is reset to null.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/startdate

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/4711/startdate
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package whose start date should be deleted.

Response status

Status 200 (OK): The start date of a signing package was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Plugin requests

Get a list of enabled plugins

This request returns a list (array) of plugins that are enabled for the specified account and implement the specified type.

Note A user with role SUPER must specify the accountid parameter.

URL

http://*host_server:port_number*/cirrus/rest/v8/plugins

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/plugins
```

Example header

```
Accept: application/json
```

Query parameters

- **accountid** (string, optional): The account to use to list the enabled plugins. A user with role SUPER must specify the account id.
- **type** (string, required): Filter for plugins of the specified type. Possible value: TSP

Response class

Status 200 (OK): The plugins were queried successfully.

RestPluginInfo

- **id** (string): The ID of the plugin
- **name** (string): The name of the plugin

Reminder requests

Add reminder to signing package

This request adds a new reminder to an existing signing package. An input JSON or XML string specifying the details of the reminder, has to be provided as a request parameter.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/reminder

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/reminder
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
   "id" : "reminder1",
   "type": "BEFORE_EXPIRES",
   "days": "3"
}
```

Example request body (XML)

```
<RestReminderInput>
  <id>reminder1</>
  <type>BEFORE_EXPIRES</type>
  <days>3</days>
</RestReminderInput>
```

Path parameters

- **packageid** (string, required): The id of the signing package the user wants to add a reminder to

Body parameters

- **reminder** (RestReminderInput, required): Represents a RestReminderInput object either in JSON or XML

RestReminderInput

- **days** (integer): The days of the reminder
- **id** (string, optional): The id of the reminder
- **type** (string, optional): The type of the reminder. Possible values: AFTER_SEND, BEFORE_EXPIRES

Response status

Status 201 (Created): The reminder was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a reminder

This request updates an existing reminder within an account using an input JSON or XML string. Each attribute of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/reminders/{reminderid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/reminders/1002
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id" : "reminder1",
  "type": "BEFORE_EXPIRES",
  "days": "3"
}
```

Example request body (XML)

```
<RestReminderInput>
  <id>reminder1</id>
  <type>BEFORE_EXPIRES</type>
  <days>3</days>
</RestReminderInput>
```

Path parameters

- **packageid** (string, required): The id of the signing package containing the reminder the user wants to update
- **reminderid** (string, required): The id of the reminder the user wants to update

Body parameters

- **reminder** (RestReminderInput, required): Represents a RestReminderInput object either in JSON or XML

RestReminderInput

- **days** (integer): The days of the reminder
- **id** (string, optional): The id of the reminder
- **type** (string, optional): The type of the reminder. Possible values: AFTER_SEND, BEFORE_EXPIRES

Response status

Status 200 (OK): The reminder was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a reminder

This request deletes a reminder specified by id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/reminders/{reminderid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/1001/reminders/1002
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package containing the reminder the user wants to delete.
- **reminderid** (string, required): The id of the reminder the user wants to delete.

Response status

Status 200 (OK): The reminder was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Signer requests

Get a remote session signing URL for the specified signer

This request returns the signing package URL for a signer for a remote session. The request checks if a similar request was already made in the past. If yes, the request is reused and it returns the old URL. Otherwise a new access token is created for the signer for authorization.

Note For this request a valid authentication with role USER is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/signingurl

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/signers/1002/signingurl
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer

Response class

Status 200 (OK): The remote session signing URL was successfully created for the signer. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Get a single signer

This request returns a signer specified by its id and signing package id.

Note For this request a valid authentication with USER or SIGNER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/1001/signers/1002
```

Example header

```
Accept: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, optional): The id of the signer

Query parameters

- **useIntId** (boolean, optional): The provided ids for signing package and signer are the internal ids which were returned by get audit trail method. Default value: false

Response class

Status 200 (OK): The signer was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages. In case of a successful query the response body contains the following information regarding the requested document as RestSignerOutput structure:

- accessCodeDeliveryChannel
- accessCodeDeliveryParameters
- accessCodeDeliveryPluginId
- authenticationMode (NONE, CODE)
- authenticationTime
- clientCertificateRequest (DISABLED, REQUIRED)
- commentForDecline
- completionTime
- email
- esignConsentRequired
- externalAuthenticationUrl
- firstName
- gdprConsentRequired
- id
- lastName
- name
- order
- preferredLanguage
- reasonForDecline
- relatedUser
- role (SIGNER, REVIEWER)
- state (ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR)
- tspSignerInfo
- tspCredentials (a list of RestEntry)
- k
- v
- tspPluginId
- tspSignatureDocuments (a list of RestTSPSignatureDocument)
- documentId
- state (REQUIRED, DONE)
- tspSignatureType (BASIC, ADVANCED, QUALIFIED)

RestSignerOutput

- **accessCodeDeliveryChannel** (string, optional): The channel type by which authentication code will be sent to the signer
- **accessCodeDeliveryParameters** (Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationTime** (string, optional): The authentication time of the signer. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **completionTime** (string, optional): The completion time of the signer. Format: date-time with a time-zone in UTC such as '2020-12-03T10:15:30Z' (ISO-8601)
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignerInfo** (RestTSPSignerInfoOutput, optional): TSP-related info for the signer to sign documents using TSP

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoOutput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an intercation with the required TSP
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

RestTSPSignatureDocument

- **documentId** (string): Id of the document
- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

Get a list of signers

This request returns a list of signers which are found by the defined SignerSearchEvent plugin. The SignDoc default implementation of the plugin performs a distinct search in all packages of the whole user account. The result list contains name and email address of the signer. The signers can be filtered by name and/or email. It is also possible to limit the number of result entries.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/signerlist

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/signerlist
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **searchname** (string, optional): The signer name for searching. A signer is included only if his name starts with the provided text. This is case-insensitive. searchname and searchemail cannot be set at the same time.
- **searchemail** (string, optional): The signer email for searching. A signer is included only if his email address starts with the provided text. This is case-insensitive. searchname and searchemail cannot be set at the same
- **limit** (integer, optional): The number of results to be returned. If 0 or no limit is set then the value from configuration entry 'cirrus.rest.resultset.size.max.signers.autocomplete' is used as default limit. Default value: configuration value
- **custom** (string, optional): Any custom character data which is passed through to the called plugin.

Response class

Status code 200 (OK): The request was successful. The body content contains a list of RestSignerListEntry elements with the required fields.

RestSignerListEntry

- **accessCodeDeliveryChannel** (string, optional): The channel name by which authentication code will be sent to the signer
- **authenticationMode** (string, optional): The id authentication mode the signer: Possible values: NONE, CODE, EXTAUTH
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **email** (string): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **name** (string, optional): The name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **relatedUser** (string, optional): The identifier of the related SignDoc user
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (integer, optional): The stage of the signer

- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **unreadDocuments** (Array[string], optional): List of documents ids needed to be reviewed by the signer
- **url** (string, optional): The request URL for the entire signer info

RestTSPSignatureDocument

- **documentId** (string): Id of the document
- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

Signer search

This request returns a list of signers from signing packages where the current user is the owner. The signers can be filtered by name and email and extended by one or more teams. It is also possible to reduce the content of the result to one or more fields.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/signers

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/signers
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **searchname** (string, optional): The signer name for searching. A signer is included only if his name starts with the provided text. This is case-insensitive. searchname and searchemail cannot be set at the same
- **searchemail** (string, optional): The signer email for searching. A signer is included only if his email address starts with the provided text. This is case-insensitive. searchname and searchemail cannot be set at the same time.
- **team** (Array[string], optional): A list of comma delimited teams to be included in the selection. The result contains signers from packages where the current user is the owner and additionally signer from packages where the owner and the current user are member of one of the named teams. The list is ignored if parameter allteams is also set.
- **resfield** (Array[string], optional): A list of comma delimited fields to be included in the response. If this parameter is omitted all fields are returned. Otherwise the return is filtered and only the requested fields are returned. Valid names are id, name, email, order, role, state, authenticationmode, accesscodedeliverychannel, externalauthenticationurl, clientcertificaterequest, unreaddocuments, esignconsentrequired, gdprconsentrequired, relateduser and preferredlanguage. The filter list is case insensitive. Filter names not matching the given list are ignored.
- **distinct** (boolean, optional): If resField has been specified, only return distinct entries (ignoring the id and url). The most recently changed entry is returned. Default value: according to description
- **allteams** (boolean, optional): If this parameter is set to true, the result contains signers from packages where the current user is the owner and additionally signer from packages where the owner and the current user are both members of the same team.
- **page** (integer, optional): The page number to return (first=1)
- **limit** (integer, optional): The number of results to be returned in one page

Response class

Status code 200 (OK): The request was successful. The body content contains a list of RestSignerListEntry elements with the required fields.

RestSignerListEntry

- **accessCodeDeliveryChannel** (string, optional): The channel name by which authentication code will be sent to the signer
- **authenticationMode** (string, optional): The id authentication mode the signer: Possible values: NONE, CODE, EXTAUTH
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **email** (string): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **name** (string, optional): The name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)

- **relatedUser** (string, optional): The identifier of the related SignDoc user
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer
- **stage** (integer, optional): The stage of the signer
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP
- **unreadDocuments** (Array[string], optional): List of documents ids needed to be reviewed by the signer
- **url** (string, optional): The request URL for the entire signer info

RestTSPSignatureDocument

documentId (string): Id of the document

- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

Get a single signer by email

This request returns a signer specified by its email.

URL

http://*host_server:port_number*/cirrus/rest/v8/signers/{email}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/signers/johndoe%40kofax.com
```

Note The @ sign within the email address must be encoded with %40.

Example header

```
Accept: application/json
```

Path parameters

- **email** (string, optional): The email of the signer

Response class

Status 200 (OK): The signer was successfully returned. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestSignerOutput

- **accessCodeDeliveryChannel** (string, optional): The channel type by which authentication code will be sent to the signer
- **accessCodeDeliveryParameters** (Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationTime** (string, optional): The authentication time of the signer. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **completionTime** (string, optional): The completion time of the signer. Format: date-time with a time-zone in UTC such as '2020-12-03T10:15:30Z' (ISO-8601)
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **externalAuthenticationUrl** (string, optional): The external authentication URL
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **signerColor** (string, optional): The display color assigned to the signer

- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **tspSignerInfo** (RestTSPSignerInfoOutput, optional): TSP-related info for the signer to sign documents using TSP

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value
- **RestTSPSignerInfoOutput**
- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interacation with the required TSP
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP

tspSignatureType (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

RestTSPSignatureDocument

- **documentId** (string): Id of the document
- **state** (string): Current state of the document in TSP signing process. Possible values: REQUIRED, DONE

Get TSP info

This request provides information about the configured TSP (trusted service provider).

Note For this request a valid authentication with role SIGNER is necessary.

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

URL

http://*host_server:port_number*/cirrus/rest/v8/tsp/info

Query parameters

- **locale** (string, optional): Locale for retrieving the locale-specific TSP information (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47). If omitted, en-US will be used.

Response class

Status 200 (OK): The TSP information is returned successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestTSPInfo

- **helpText** (string, optional)
- **name** (string, optional)
- **registrationURL** (string, optional)
- **signingText** (string, optional)
- **validationFields** (Array[RestTSPValidationEntryField], optional)
- **validationText** (string, optional)

RestTSPValidationEntryField

- **description** (string, optional)
- **id** (string, optional)
- **label** (string, optional)
- **placeholder** (string, optional)
- **regex** (string, optional)
- **type** (string, optional): Possible values: STRING, BOOLEAN

Process TSP result

This request processes the TSP result.

URL

http://*host_server:port_number*/cirrus/rest/v8/tsp/result/{result}/{tspkey}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Path parameters

- **result** (string, optional): The tsp result.
- **tspkey** (string, optional): The tsp key.

Query parameters

- **did** (string, required): The document id
- **curl** (string, required): The client URL for redirection with the TSP result

Add signer to signing package

This request adds a new signer to an existing signing package. An input JSON or XML string specifying the details of the signer, has to be provided as a request parameter. The specification has to provide at least a signer name with more than two characters or an email address. Note that a valid signer must have at least a name and role (by default SIGNER).

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signer

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/1001/signer
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
      "order": 3,
      "email": "cersei@llannister.com",
      "name": "Cersei Lannister",
      "role": "SIGNER",
      "authenticationMode": "CODE",
      "authenticationParam": "qgzCVa",
      "accessCodeDeliveryPluginId": "NotificationSMSClickatell",
      "esignConsentRequired": true,
      "gdprConsentRequired": true,
      "preferredLanguage": "en",
      "accessCodeDeliveryParameters": [
        {
          "key": "NotificationSMSClickatell.phonenumber",
          "value": "+1-62293520272454"
        }
      ],
      "id": "signer-1"
    }
```

Example request body (XML)

```
<RestSignerInput>
  <id>signer-1</id>
```

```
  <name>Cersei Lannister</name>
  <email>cersei@llannister.com</email>
  <role>SIGNER</role>
  <order>1</order>
  <authenticationMode>CODE</authenticationMode>
  <authenticationParam>qgzCVa</authenticationParam>
  <accessCodeDeliveryPluginId>NotificationSMSClickatell</accessCodeDeliveryPluginId>
  <esignConsentRequired>true<esignConsentRequired>
  <gdprConsentRequired>true</gdprConsentRequired>
  <preferredLanguage>en</preferredLanguage>
  <accessCodeDeliveryParameters>
    <key>NotificationSMSClickatell.phonenumber</key>
    <value>+1-62293520272454</value>
  </accessCodeDeliveryParameters>
</RestSignerInput>
```

Path parameters

- **packageid** (string, required): The id of the signing package the user wants to add a signer

Body parameters

- **signer** (RestSignerInput, required): Represents a RestSignerInput object either in JSON or XML. Default value: null

RestSignerInput

- **accessCodeDeliveryParameters**(Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided.
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries. Note: The access code delivery plugin can be used for sending an access code to a recipient for a 2-factor authentication.
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationParam** (string, optional): The authentication code for the signer
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer. The signer will be notified in ascending order if sequential package processing is selected. Note: Sequential processing for a package can be set via processingType=SEQ in the RestSigningPackageInput structure.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47) Note: The preferred signer language overwrites the account

specific communication language. This setting determines the used language in the emails to the signer.

- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5

  Explanation:

  R1: There is a problem with the document(s)

  R2: I do not recognize the sender

  R3: I do not want to sign online

  R4: I do not agree with the terms of the e-sign consent

  R5: I do not agree with the terms of the GDPR statement
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input. Note: If a SignDoc user is a signer within a signing package it is possible to allow him to use a locally stored image as signature input. In this case the relatedUser attribute must contain the SignDoc userid during creation of a signer. If the signer is requested later then the attribute relatedUser contains another value than the entered userid (for security reason).
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plugin it is possible to let the signer sign with a qualified certificate.

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoInput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP
- **tspSignatureDocuments** (Array[string], optional): List of Ids of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Response status

Status 201 (Created): The signer was successfully added to the signing package. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Create a signing authentication token

This request generates a time-limited authentication token for signer representing a RestSigningAuthentication JSON object and a corresponding hash value.

The JSON object and hash are both Base64-encoded and separated by a dot. For example BASE64 string(RestSigningAuthentication) + '.' + BASE64 string(hash value of RestSigningAuthentication).

The RestSigningAuthentication object can be used to access important information like the accountId, the packageId or the expiry date of the authentication token. The generated signing authentication token is returned as X-S-AUTH-TOKEN response header.

To authenticate via the generated authentication token a signer has to send the authentication token as part of the X-S-AUTH-TOKEN header for each new request. The content type of the body is application/x-www-form-urlencoded.

URL

http://*host_server:port_number*/cirrus/rest/v8/signers/authentication

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/signers/authentication
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **token** (string, required): The signing session token
- **signtype** (string, required): The signing type. Possible values: REMOTE, COMMON
- **signerid** (string, optional): The id of the signer. It should be provided in case of Common Signing Session for document processing requests.
- **accesscode** (string, optional): The access code requested for two-factor authentication of a remote signer

Response class

Status 200 (OK): The signing authentication token was successfully created, otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestSigningAuthentication

- **aid** (string, optional): The account id of the signing package
- **exp** (integer, optional): The expiration date of the authentication token in number of milliseconds from the epoch
- **hac** (integer, optional): The hash code of the 'Signer Authentication Code' (in case of Two-factor authentication for remote signers)
- **hst** (integer, optional): The hash code of the 'Signing Session Token' value
- **iat** (integer, optional): The 'issued At' date of the authentication token in number of milliseconds from the epoch
- **pid** (string, optional): The signing package id for which the authentication is valid
- **sid** (string, optional): The signer id for which the authentication is valid
- **sst** (string, optional): Signing session type. Possible values: c, r (common, remote)

Update a signer

This request updates an existing signer within an account using an input JSON or XML string. Each attribute of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/signers/1002
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
   "id" : "signer11",
   "name": "Signer11",
   "email": "signer11@test.com",
   "role": "SIGNER",
   "order": 1
}
```

Example request body (XML)

```
<RestSignerInput>
  <id>signer11</id>
  <name>Signer11</name>
  <email>signer11@test.com</email>
  <role>SIGNER</role>
  <order>1</order>
</RestSignerInput>
```

Path parameters

- **packageid** (string, required): The id of the signing package containing the signer to be updated
- **signerid** (string, required): The id of the signer the user wants to update

Body parameters

- **signer** (string, required): Represents a RestSignerInput object either in JSON or XML

RestSignerInput

- **accessCodeDeliveryParameters**(Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided.
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries. Note: The access code delivery plugin can be used for sending an access code to a recipient for a 2-factor authentication.
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE, CODE, EXTAUTH
- **authenticationParam** (string, optional): The authentication code for the signer
- **clientCertificateRequest** (string, optional): The setting specifies whether to use client certificate for signing. Possible values: DISABLED, REQUIRED
- **commentForDecline** (string, optional): The comment for decline of the signer
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed
- **firstName** (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed

- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer. The signer will be notified in ascending order if sequential package processing is selected. Note: Sequential processing for a package can be set via processingType=SEQ in the RestSigningPackageInput structure.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47) Note: The preferred signer language overwrites the account specific communication language. This setting determines the used language in the emails to the signer.
- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5

  Explanation:

  R1: There is a problem with the document(s)

  R2: I do not recognize the sender

  R3: I do not want to sign online

  R4: I do not agree with the terms of the e-sign consent

  R5: I do not agree with the terms of the GDPR statement
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input. Note: If a SignDoc user is a signer within a signing package it is possible to allow him to use a locally stored image as signature input. In this case the relatedUser attribute must contain the SignDoc userid during creation of a signer. If the signer is requested later then the attribute relatedUser contains another value than the entered userid (for security reason).
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plugin it is possible to let the signer sign with a qualified certificate.

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoInput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP
- **tspSignatureDocuments** (Array[string], optional): List of Ids of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Response status

Status 200 (OK): The signer was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delegate a signing session

This request delegates the signing session of a particular signer to a new signer. Only a signer with its delegate flag set to 'true' is allowed to delegate the signing session to a new signer, else an error is thrown.

Note For this request a valid authentication with role SIGNER is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/delegate

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/packages/1001/signers/1002/delegate
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
   "name": "Signer11",
   "email": "signer11@test.com",
   "delegateMessage": "Signer requst from a signer"
 }
```

Example request body (XML)

```
<RestSignerInput>
  <name>Signer11</name>
  <email>signer11@test.com</email>
  <delegateMessage>SIGNER</delegateMessage>
</RestSignerInput>
```

Path parameters

- **packageid** (string, required): The id of the signing package containing the signer who wants to perform the delegate operation.
- **signerid** (string, required): The id of the signer the user wants to delegate the signing session.

Body parameters

- **signer** (string, required): Represents a RestSignerInput object either in JSON or XML

RestSignerInput

- **accessCodeDeliveryParameters** (Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plugin for the access code delivery. The parameters are only considered if the id of the plugin for the access code delivery is also provided.
- **accessCodeDeliveryPluginId** (string, optional): The id of the plugin for the access code delivery. An empty string in RestSignerInput deletes the plugin id entry and the related RestAccessCodeDeliveryChannelParameter entries. Note: The access code delivery plugin can be used for sending an access code to a recipient for a 2-factor authentication.
- **authenticationMode** (string, optional): The authentication mode of the signer. Possible values: NONE,CODE, EXTAUTH
- **authenticationParam** (string, optional): The authentication code for the signer
- **clientCertificateRequired** (boolean, optional): The setting specifies whether to use client certificate for signing
- **commentForDecline** (string, optional): The comment for decline of the signer
- **delegate** (boolean, optional): The flag to indicate if the signer can delegate the signing session to some other recipient
- **delegateMessage** (string, optional): The optional message from the original recipient to the delegated recipient
- **email** (string, optional): The email of the signer
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed firstName (string, optional): The given name of the signer
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed
- **id** (string, optional): The id of the signer
- **lastName** (string, optional): The surname of the signer
- **name** (string, optional): The display name of the signer
- **order** (integer, optional): The order of the signer. The signer will be notified in ascending order if sequential package processing is selected. Note: Sequential processing for a package can be set via processingType=SEQ in the RestSigningPackageInput structure.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)

  Note The preferred signer language overwrites the account specific communication language. This setting determines the used language in the emails to the signer.

- **reasonForDecline** (string, optional): The reason for decline of the signer. Possible values: R1, R2, R3, R4, R5

  Explanation:

  R1: There is a problem with the document(s)

  R2: I do not recognize the sender

  R3: I do not want to sign online

  R4: I do not agree with the terms of the e-sign consent

  R5: I do not agree with the terms of the GDPR statement
- **relatedUser** (string, optional): The identifier of the related SignDoc user. A valid account specific user id is expected as input. The output can differ from the input. Note: If a SignDoc user is a signer within a signing package it is possible to allow him to use a locally stored image as signature input. In this case the relatedUser attribute must contain the SignDoc userid during creation of a signer. If the signer is requested later then the attribute relatedUser contains another value than the entered userid (for security reason).
- **role** (string, optional): The role of the signer. Possible values: SIGNER, REVIEWER
- **stage** (integer, optional): The stage value a signer belongs, mandatory when package processing type is 'STAGED' optional in case of 'PAR' (by default is equal to '1', must be '1' if supplied in the rest input) and 'SEQ' (by default is equal to 'signer order', must be equal to 'signer order' if supplied in the rest input). Signer will be notified in ascending order of stage value.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plugin it is possible to let the signer sign with a qualified certificate.

RestAcessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key
- **value** (string, optional): The parameter value

RestTSPSignerInfoInput

- **tspCredentials** (Array[RestEntry]): The list of credentials needed for an authentication in the required TSP
- **tspPluginId** (string): The Id of the plugin needed for an interaction with the required TSP
- **tspSignatureDocuments** (Array[string], optional): List of Ids of documents needed to be signed by the signer using the required TSP
- **tspSignatureType** (string): The type of signature which will be applied by the required TSP. Possible values: BASIC, ADVANCED, QUALIFIED

RestEntry

- **k** (string): The key of the entry
- **v** (string): The value of the entry

Response status

Status 200 (OK): The signing session was successfully delegated to a new signer. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a signer

This request deletes a signer specified by id.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/signers/1002
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer the user wants to delete

Response status

Status 200 (OK): The signer was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete signer email

This request deletes the email address of a signer specified by signerid.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/email

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/
signer-1/email
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer

Response status

Status 200 (OK): The email of the signer was successfully deleted, otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete signer TSP info

This request deletes the TSP info of a signer for the specified signerid and packageid.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/packages/{packageid}/signers/{signerid}/tspSignerInfo

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/package-1/signers/
signer-1/tspSignerInfo
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **packageid** (string, required): The id of the signing package
- **signerid** (string, required): The id of the signer

Response status

Status 200 (OK): The tsp info of the signer was successfully deleted, otherwise a SignDoc Standard status code is returned together with the explaining messages.

System requests

Get the global license information

This request retrieves information about an installed global SignDoc license.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/license

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/license
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
{
  "expiryDate": "2020-12-31T23:59:59.999Z",
  "licensedUsers": 1000,
  "currentUsers": 23,
  "licensedPackages": 1000000,
  "processedPackages": 56332
  "accountInformation": "Wonka industries",
  "state": "VALID",
  "renewableLicensePeriod": "MONTHLY"
}
```

Response class

Status 200 (OK): The global license could be retrieved successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAccountLicenseOutput

- **accountInformation** (string, optional): Account information
- **currentUsers** (integer, optional): Current number of users
- **expiryDate** (string, optional): License expiry date. Format: date-time with a time-zone in UTC, such as '2020-12- 03T10:15:30Z' (ISO-8601)
- **licensedPackages** (integer, optional): Number of licensed signing packages
- **licensedUsers** (integer, optional): Number of licensed users
- **processedPackages** (integer, optional): Number of processed signing packages
- **renewableLicensePeriod** (string, optional): The renewal period of the license (if renewable license). Possible values: MONTHLY, YEARLY
- **state** (string, optional): The state of the license. Possible values: VALID, MAXIMUM_REACHED, EXPIRED, NO_LICENSE, INVALID

Get locale display name

This request returns the display name for a specific locale. The locale is specified as language tag in the IETF BCP 47 language tag format, see https://tools.ietf.org/html/bcp47.

The display name language can be defined by providing another language tag (IETF BCP 47 language tag) as query parameter ‘locale’.

Note For this request a valid authentication with USER, SIGNER, ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/locale/{language_tag}/info

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
http://localhost:6611/cirrus/rest/v8/locale/fr/info?locale=pt-BR
```

Displays the name of the French locale in Brazilian Portuguese, you could request it like the following example.

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "languageTag": "fr",
  "displayName": "francês"
}
```

Path parameters

- **language_tag** (string, required): The language tag which describes the locale for which the name should be displayed in IETF BCP 47 language tag format, see https://tools.ietf.org/html/bcp47

Query parameters

- **locale** (string, optional): The language tag which describes the locale of the display language in IETF BCP 47 language tag format, see https://tools.ietf.org/html/bcp47

Response class

Status 200 (OK): The display name could be retrieved successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestLocaleInfo

- **displayName** (string, optional): The name for the locale that is appropriate for display to the user.
- **languageTag** (string, optional): The language tag (IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47)

Get the version of the Kofax SignDoc modules

This request returns the modules versions of the of Kofax SignDoc distribution. For this request authentication is required.

Note For this request a valid authentication with USER, SUPER or SIGNER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/system/modules

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/system/modules
```

Example header

```
Accept: application/json
```

Response class

Status is 200 (OK): The version of the modules was queried successfully.

RestModulesVersion

- **rest_api** (string, optional)
- **signdoc_sdk** (string, optional)
- **signdoc_standard** (string, optional)

Example response body (JSON)

```
{
"signdoc_standard": "version-x",
"signdoc_sdk": "version-y",
"rest_api": "v1"
}
```

Get status information of system entities

This request retrieves status information of these system entities: S/MIME_CERTIFICATE, SMTP_CONNECTION. The S/MIME_CERTIFICATE is tested for validity, usability and expiry dates. The SMTP_CONNECTION is tested for a valid configuration and if it can connect to the configured SMTP Service.

URL

http://*host_server:port_number*/cirrus/rest/v8/system/status

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/system/status
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **locale** (string, optional): ETF BCP 47 language tag. If value is unknown or invalid, English will be used.

Response class

Status 200 (OK): The system entities were successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestEntityStatus

- **id** (string, optional): The id of the information. Possible values: SIGNING_CERTIFICATE, SMTP_CONNECTION, BIOMETRIC_KEY, SMIME_CERTIFICATE

- **statusClass** (string, optional): The status class of the information. Possible values: INFO, OK, WARN, PROBLEM
- **statusClassString** (string, optional): The localized description of the status class value.
- **statusCode** (string, optional): The status code of the information: Possible values: SMTP_ERROR, EXPIRED, NOT_YET_VALID, EXPIRES_SOON, NOT_SET, IS_VALID, INVALID, GENERAL_ERROR, MESSAGE, NO_DATA
- **statusCodeDescription** (Array[string], optional): The localized description of the status code value. Consists usually of multiple lines.
- **statusIdString** (string, optional): The localized description of the entity name
- **timestamp** (string, optional): Timestamp in ISO format

Get the version of SignDoc distribution

This request returns the versions of the of SignDoc distribution. For this request no authentication is necessary. Returns the versions of the used modules.

The response to this endpoint is controlled by the environment variable ‘cirrus.rest.hide.app.version’, a valid response is returned only when this value is set to ‘false’.

URL

http://*host_server:port_number*/cirrus/rest/v8/system/version

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/system/version
```

Example header

```
Accept: application/json
```

Response class

Status is 200 (OK): The versions were queried successfully.

RestModulesVersion

- **signdoc_standard** (string, optional)

- **signdoc_sdk** (string, optional)
- **rest_api** (string, optional)

Example response body (JSON)

```
{
"signdoc_standard": "version-x",
"signdoc_sdk": "version-y",
"rest_api": "v1"
}
```

Import a global license

This request imports a global SignDoc license.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/license

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/license
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
RestAccountLicenseInput {

"license":"77u/aDpTUExNMiA0LjEwDQppOklEOjk5MjMzNzkNCmk6UHJvZHVjdDpLb2ZheFNpZ25
Eb2MNCmk6TWFudWZhY3R1cmVyOktvZmF4IERldXRzY2hsYW5kIEdtYkggDQppOkN1c3RvbWVyOkR1b
W15IGNvcnBvcmF0aW9uDQppOlZlcnNpb246OTkNCmk6T1M6YWxsDQphOnByb2R1Y3Q6dW5saW1pdGV
kDQphOnNpZ253YXJlOnVubGltaXRlZA0KYTpzaWduOjIwMTgtMTItMzENCmE6Y2FwdHVyZTp1bmxpb
```

Wl0ZWQNCmE6cmVuZGVyOnVubGltaXRlZA0KcHM6QUNDT1VOVF9JTkZPUk1BVElPTjpGNEQyMldPNjV
GIA0KcHM6Q3VzdG9tZXJDb21wYW55OkR1bW15IGNvcnBvcmF0aW9uIA0KcGk6TElDRU5TRV9UWVBFO
jENCnBpOk1BWF9TSUdOSU5HX1BBQ0tBR0VTOjEwMDAwMA0KcGk6TUFYX1VTRVJTOjEwMA0Kczo3Mzc
yZTQxMDI4NWYyMWZjZDdjZWNkNTY2OTE5MDM5NGY5NTFmYTAzMjg4OWU5N2JlOGY3YmRhZDJjODA5M
WVmDQpzOjdjNDE0MzM4MzI4YzEzZDBhY2UwZTU1NTAxZmE2NDBlOTdlZDMyMmI1Zjg5YTk0MTM1NTI
xMjIyM2Y0ZThjODQNCnM6MzkxNGUzNzQ5ZmZkNWU4MGM0OWM1OTJlZjlmODk4MTlmMTE3NzRiODQzN
TVkMjg1Y2EwZjUwYzhkNTRkNzMyZQ0Kczo3OWFlNjU1NWQ5NDBkMGM3ZjA2YmZlYzY1NzE0ZTMzOTg
1ZGI1ZjMwNmU4OTdmNmUwNWU5MTUxNmM1NDZiMjVk”

}

Body parameters

- **restLicense** (RestAccountLicenseInput, optional): Input JSON or XML string of license

RestAccountLicenseInput

- **accountId** (string, optional): The account id. This parameter can be omitted if the user is bound to an account.
- **license** (string): Account license as base64 encoded string

Response class

Status 200 (OK): The global license could be imported successfully, otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAccountLicenseOutput

- **accountInformation** (string, optional): Account information
- **currentUsers** (integer, optional): Current number of users
- **expiryDate** (string, optional): License expiry date. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **licensedPackages** (integer, optional): Number of licensed signing packages
- **licensedUsers** (integer, optional): Number of licensed users
- **processedPackages** (integer, optional): Number of processed signing packages
- **renewableLicensePeriod** (string, optional): The renewal period of the license (if renewable license). Possible values: MONTHLY, YEARLY
- **state** (string, optional): The state of the license: Possible values: VALID, MAXIMUM_REACHED, EXPIRED, NO_LICENSE, INVALID

Team requests

Get a single team

This request returns a single team of an account specified by a team id.

Note For this request a valid authentication with USER role is necessary. A user which has not the role ADMIN can retrieve only a team where he is a member.

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json
```

Path parameters

- **teamid** (string, required): The id of the team

Response class

Status 200 (OK): The team was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestTeamOutput

creationTime (string, optional): The creation time of the team. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)

- **id** (string, optional): The id of the team
- **lastUpdateTime** (string, optional): The last update time of the team. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **managers** (Array[RestUserListEntry], optional): A list of all team managers
- **members** (Array[RestUserListEntry], optional): A list of all team members
- **name** (string, optional): The name of the team
- **url** (string, optional): The url to query the team

RestUserListEntry

- **email** (string, optional): The email of the user
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC such as '2020-12-03T10:15:30Z' (ISO-8601)

- **name** (string, optional): The name of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **url** (string, optional): The request URL to query the complete user

Example response body (JSON)


```
{
  "id": "team1",
  "name": "Team 1",
  "managers": [
    {
      "id": "jdo",
      "name": "John Doe",
      "email": "john@doe.com",
      "state": "ACTIVE",
      "url": "http://localhost:6611/cirrus/rest/v8/users/jdo",
      "roles": [
        "USER",
        "TEAMMGR",
        "ADMIN"
      ],
      "lastSignInTime": "2019-05-14T16:16:10.915Z"
    }
  ],
  "members": [
    {
      "id": "ble",
      "name": "Big Lebowski",
      "email": "big@lebowski.com",
      "state": "ACTIVE",
      "url": "http://localhost:6611/cirrus/rest/v8/users/ble",
      "roles": [
        "USER",
        "TEAMMGR",
        "ADMIN"
      ],
      "lastSignInTime": "2019-05-14T13:54:57.687Z"
    },
    {
      "id": "tla",
      "name": "Tyrion Lannister",
      "email": "tyrion@lannister.com",
      "state": "ACTIVE",
      "url": "http://localhost:6611/cirrus/rest/v8/users/tla",
      "roles": [
        "USER"
      ],
      "lastSignInTime": "2019-05-14T16:16:10.915Z"
    },
    {
      "id": "jcu",
      "name": "Jesse Custer",
      "email": "jesse@custer.com",
      "state": "ACTIVE",
      "url": "http://localhost:6611/cirrus/rest/v8/users/jcu",
      "roles": [
        "USER"
      ],
      "lastSignInTime": "2019-05-14T13:54:55.479Z"
    }
  ],
```

```
  "creationTime": "2019-05-14T16:15:30.556Z",
  "url": "http://localhost:6611/cirrus/rest/v8/teams/team1"
}
```

Add team to account

This request adds a team to an account. An input JSON or XML string specifying the details of the team, has to be provided as a request parameter. The specification has to provide at least a team name with more than two characters.

Note For this request a valid authentication with ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/team

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/team
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
"id": "team1",
"name": "Team 1",
"managers": [
    "jdo", "ble"
],
"members": [
    "tla", "jcu"
]
```

Body parameters

- **team** (RestTeamInput, required): Input JSON or XML string team. Default value: null

RestTeamInput

- **id** (string, optional): The id of the team
- **managers** (Array[string], optional): A list of all team managers specified by id or email
- **members** (Array[string], optional): A list of all team members specified by id or email
- **name** (string, optional): The name of the team

Response Staus

Status 201 (Created): The team was created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Add user to team

This request adds an existing user to a team. It is possible to declare the added user as the team manager of the team, by setting the asManager property.

Note For this request a valid authentication with ADMIN or TEAMMGR role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}/users/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/teams/team1/users/user1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **teamid** (string, required): The id of the team
- **userid** (string, required): The id of the user who should be added

Query parameters

- **as_team_manager** (boolean, optional): Whether the added user should be team manager or not. Default value: false

Response class

Status 200 (OK): The user was successfully added to the team. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a team

This request updates an existing team within an account using an input JSON or XML string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with ADMIN or TEAMMGR role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "name": " Team "
}
```

Path parameters

- **teamid** (string, required): The id of the team the user wants to update

Body parameters

- **team** (RestTeamInput, required): Represents a RestTeam object either in JSON or XML.

RestTeamInput

- **id** (string, optional): The id of the team
- **managers** (Array[string], optional): A list of all team managers specified by id or email
- **members** (Array[string], optional): A list of all team members specified by id or email
- **name** (string, optional): The name of the team

Response status

Status 200 (OK): The team was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete team from account

This request deletes an existing team from the account.

Note For this request a valid authentication with ADMIN or TEAMMGR role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **teamid** (string, required): The id of the team the user wants to delete

Response status

Status 200 (OK): The team was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Remove user from team

This request removes an existing user from a team. Note that the actual user is not deleted. Only the relation between specified user and team is deleted.

Note For this request a valid authentication with ADMIN or TEAMMGR role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}/users/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/teams/team1/users/user1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **teamid** (string, required): The id of the team
- **userid** (string, required): The id of the user who should be removed from the team

Query parameters

- **manager** (boolean, optional): Whether the user has the team manager role or not. This is required to specify a different manager to the team. By default the user who sent the request is assigned as team manger. Default: false

Response status

Status is 200 (OK): The user was successfully removed from the team. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

User requests

Get current user

This request returns the current user specified by the current authentication.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/user

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/user
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Response class

Status 200 (OK): The user was successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserOutput

- **accessTokenExpires** (string, optional): The expiration date of the access token of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)

- **creationTime** (string, optional): The creation time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a user's email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastSignOutTime** (string, optional): The last sign out time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastUpdateTime** (string, optional): The last update time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values ACTIVE, SUSPENDED, INVITED
- **teamManager** (Array[RestTeamListEntry], optional): A list of all teams the user is manager of
- **teamMember** (Array[RestTeamListEntry], optional): A list of all teams the user is member of
- **url** (string, optional): The url to query the user

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

RestTeamListEntry

- **id** (string, optional): The id of the team
- **name** (string, optional): The name of the team
- **teamManagers** (Array[string], optional): A list of all team managers the team possesses
- **teamMembers** (Array[string], optional): A list of all team members the user possesses
- **url** (string, optional): The request url to query the complete team

Get all users

This request returns all users of an account. The resulting list (RestUserListEntry) contains only necessary information for further queries sorted by last update time descending. For a complete data set use the Get a single user request (/rest/users/{userid}).

Note For this request a valid authentication with TEAMMGR, ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
[
  {
    "id": "user1",
    "name": "user1",
    "email": "user1@ksd.com",
    "state": "ACTIVE",
    "url": "http://localhost:6611/rest/v8/users/user1",
    "roles": [
      "USER",
      "TEAMMGR",
      "ADMIN"
    ],
        "lastSignInTime": "2019-02-11T12:23:51.027Z"
  },
  {
    "id": "user2",
```

```
    "name": "user2",
    "email": "user2@ksd.com",
    "state": "ACTIVE",
    "url": "http://localhost:6611/rest/v8/users/user2",
    "roles": [
      "USER",
      "TEAMMGR"
    ],
        "lastSignInTime": "2019-04-10T13:12:35.123Z"
  },
  {
    "id": "user3",
    "name": "user3",
    "email": "user3@ksd.com",
    "state": "ACTIVE",
    "url": "http://localhost:6611/rest/v8/users/user3",
    "roles": [
      "USER",
      "TEAMMGR",
      "ADMIN"
    ],
        "lastSignInTime": "2019-05-02T14:21:57.121"
  }
]
```


Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.

Response status

Status 200 (OK): The users were queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserListEntry

- **email** (string, optional): The email of the user
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-3T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **url** (string, optional): The request URL to query the complete user

Get number of users

This request returns the number of users inside an account.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/count

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/count
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
{
  "count": 86
}
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account and has ADMIN role. Users with the role SUPERUSER have to provide

Response class

Status 200 (OK): The RestCount object was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestCount

- **count** (integer, optional): The result of request

Refresh an authentication token

This request creates a new authentication token based on an already existing valid authentication token. Use this request when your current authentication token is about to expire. The new authentication token is returned as part of X-AUTH-TOKEN response header.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/refreshtoken

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/refreshtoken
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Response status

Status 200 (OK): The authentication token was successfully refreshed. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Get current server administrator

This request returns the current server administrator.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmin

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/serveradmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@kofax.com",
  "state": "ACTIVE",
  "settings": {
    "defaultTemplateName": "Template created by Tenants Administrator at 11/17/2015
 11:17 AM",
    "defaultPackageName": "Package created by Tenants Administrator at 11/17/2015 11:17
 AM",
    "defaultMailSubject": "An e-signing request via Kofax SignDoc",
    "defaultMailMessage": "Tenants Administrator has invited you to sign documents with
 Kofax SignDoc"
  },
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1444743172375,
  "url": "/rest/v8/users/serveradmins/ksdadmin"
  }
```

Response class

Status 200 (OK): The server administrator was successfully queried. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserOutput

- **accessTokenExpires** (string, optional): The expiration date of the access token of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastSignOutTime** (string, optional): The last sign out time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastUpdateTime** (string, optional): The last update time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)

- **name** (string, optional): The name of the user
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user = ['ACTIVE', 'SUSPENDED', 'INVITED']
- **teamManager** (Array[RestTeamListEntry], optional): A list of all teams the user is manager of
- **teamMember** (Array[RestTeamListEntry], optional): A list of all teams the user is member of
- **url** (string, optional): The url to query the user

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

RestTeamListEntry

- **id** (string, optional): The id of the team
- **name** (string, optional): The name of the team
- **teamManagers** (Array[string], optional): A list of all team managers the team possesses
- **teamMembers** (Array[string], optional): A list of all team members the user possesses
- **url** (string, optional): The request url to query the complete team

Get all server administrators

This request returns all server administrators of the system. The resulting list contains only necessary information for further queries sorted by last update time descending. For a complete data set use the Get a single server administrator request (/rest/account/{id}).

Note For this request a valid session and SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/serveradmins
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)


```
[
  {
    "id": "ksdadmin",
    "name": "Tenants Administrator",
    "email": "ksdadmin@kofax.com",
    "state": "ACTIVE",
    "url": "/rest/v8/users/serveradmins/ksdadmin",
    "roles": [
      "SUPERUSER"
    ]
  },
  {
    "id": "user1",
    "name": "user1",
    "email": "user1@ksd.com",
    "state": "ACTIVE",
    "url": "/rest/v8/users/serveradmins/user1",
    "roles": [
      "SUPERUSER"
    ]
  }
]
```


Response class

Status 200 (OK): The server administrators were queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserListEntry

- **email** (string, optional): The email of the user
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **url** (string, optional): The request URL to query the complete user

Get a single server administrator

This request returns a single server administrator specified by a given id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/serveradmins/ksdadmin
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@kofax.com",
  "state": "ACTIVE",
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1448268442365,
  "url": "/rest/v8/users/serveradmins/ksdadmin"
  }
```

Path parameters

- **userid** (string, required): The id of user

Query parameters

- **useIntId** (boolean, optional): The provided userid is the internal id which was returned by get audit trail method.

Response class

Status 200 (OK): The server administrator was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserOutput

- **accessTokenExpires** (string, optional): The expiration date of the access token of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastSignOutTime** (string, optional): The last sign out time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastUpdateTime** (string, optional): The last update time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **teamManager** (Array[RestTeamListEntry], optional): A list of all teams the user is manager of
- **teamMember** (Array[RestTeamListEntry], optional): A list of all teams the user is member of
- **url** (string, optional): The URL to query the user

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

RestTeamListEntry

- **id** (string, optional): The id of the team
- **name** (string, optional): The name of the team
- **teamManagers** (Array[string], optional): A list of all team managers the team possesses

- **teamMembers** (Array[string], optional): A list of all team members the user possesses
- **url** (string, optional): The request URL to query the complete team

Get a single user

This request returns a single user specified by a given user and account id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/user1
```

Example header

```
Accept: application/json
```

Example response body (JSON)

```
{
  "id": "user1",
  "name": "user1",
  "email": "user1@ksd.com",
  "state": "ACTIVE",
  "roles": [
    "USER",
    "TEAMMGR"
  ],
  "lastSignInTime": 1447315855466,
  "creationTime": 1445853879453,
  "url": "/rest/v8/users/user1"
}
```

Path parameters

- **userid** (string, required): The id of the user to query

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account and has ADMIN role. Users with the role SUPERUSER have to provide it.
- **useIntId** (boolean, optional): The provided userid is the internal id which was returned by get audit trail method. Default value: false

Response class

Status 200 (OK): The user was queried successfully. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserOutput

- **accessTokenExpires** (string, optional): The expiration date of the access token of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastSignOutTime** (string, optional): The last sign out time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **lastUpdateTime** (string, optional): The last update time of the user. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **name** (string, optional): The name of the user
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED
- **teamManager** (Array[RestTeamListEntry], optional): A list of all teams the user is manager of
- **teamMember** (Array[RestTeamListEntry], optional): A list of all teams the user is member of
- **url** (string, optional): The URL to query the user

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

RestTeamListEntry

- **id** (string, optional): The id of the team
- **name** (string, optional): The name of the team
- **teamManagers** (Array[string], optional): A list of all team managers the team possesses
- **teamMembers** (Array[string], optional): A list of all team members the user possesses
- **url** (string, optional): The request URL to query the complete team

Add user to account

To add a user to an existing Account specified by id, an input JSON or XML string specifying the details of the user, has to be provided as a request parameter. The specification has to provide at least a user name with more than two characters and an email address. To specify an initial state the following choices are available: ACTIVE, SUSPENDED and INVITED. If no state is specified the default state will depend on if a password has been specified. If a password has been specified, the default state will be ACTIVE, if not, INVITED. If the specified user state is ACTIVE or SUSPENDED, a password has to be specified too. If the specified state is INVITED, the password has to be missing.

Note For this request a valid authentication with ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/user

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/user
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "email": "use1@kofax.com",
```

```
  "id": "use1",
  "name": "use1",
  "password": "2beUserU!",
  "roles": [
    "USER",
    "TEAMMGR",
    "ADMIN"
  ],
  "state": "ACTIVE"
}
```

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.

Body parameters

- **user** (RestUserInput, required): Represents a RestUserInput object either in JSON or XML.

Response class

Status 201 (Created): The user was successfully added to the account. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **name** (string, optional): The name of the user
- **password** (string, optional): [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special charactersare $USER which is replaced by the current user name and $NOW which is replaced by the current time

Set user API key

This request sets the API key for the authenticated user. The valid password of the user must be provided to set the API key. The API key must consist of the following allowed characters: a-z, A-Z, 0-9, '_', '.', and '-'. The API key must be at least 12 characters long. The content type of the body is application/x-www-form-urlencoded.

Note For this request a valid authentication with USER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/user/apikey

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/apikey
```

Example header

```
Accept: application/json

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

Form parameters

- **password** (string, required): The password of the user
- **apikey** (string, required): The new API key for the user

Response status

Status 200 (OK): The API key was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Set a user password

This request sets the password of a user identified by the provided token.

The user will get an email with the link for setting a new password if

- the user is created
- the password was reset by an administrator
- the user has selected 'Password forgotten' in the login dialog

The invitation link (in the email) calls the Manage Client with the query parameter aptoken (access token for server-admins) or uptoken (access token for other users).

Example

```
http://localhost:6611/cirrus/static/cirrus-client/?contexturl=http://
localhost:6611&usedefaultaccount=false&lang=en&country=#/set-password;
uptoken=3925d199-dc31-4324-8c79-5b3ba0cef728;pattern=...
```

This access token identifies and authenticates a user for setting a (new) password with the following REST API.

```
POST http://localhost:6611/cirrus/rest/v8/user/pw
```

The body contains the form parameters password (to be set) and the access token of the user which is passed to the client as value of query parameter aptoken or uptoken.

Form parameters

- **token** (string, required): The token which identifies the user.
- **password** (string, required): The password which should be set for the user.

Both body form parameters are required otherwise an error is returned:

- Error code 5086 - The token body parameter is required
- Error code 5087 - The password body parameter is required

The provided password must match the current password pattern, which is:

1. Length between 8 and 100 chars
2. At least one lowercase char
3. At least one uppercase char
4. At least one digit
5. At least one of the special signs !@#$%^&*()-_+:;'/\=,.~<>"[]{}?

The following two conditions have to be fulfilled:

- The user must not be suspended, otherwise:
  Error code 5090 - Not allowed to set the password of a suspended user
- The account of the user must be active, otherwise:
  Error code 5091 - Not allowed to set the password of a suspended user.

Note The configuration entry mail.enabled.password.changed determines whether a notification email is sent to the user if his password has been changed (default: false).

URL

http://*host_server:port_number*/cirrus/rest/v8/user/pw

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Produces

JSON, XML

Header

Accept: application/json, application/xml

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

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/user/pw
```

Example request body

```
token=3925d199-dc31-4324-8c79-5b3ba0cef728&password=cw!cdZVf*%2BdbE8r*
```

Example header

```
Accept: application/json

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

Create an authentication token

This request creates a time-limited authentication token based on basic authentication parameter specified in the request body. The returned token represents a RestAuthentication JSON object and a corresponding hash value.

The JSON object and hash are both Base64-encoded and separated by a dot. For example BASE64(RestAuthentication) + '.' + BASE64(hash value of RestAuthentication).

The RestAuthentication object can be used to access important information like the expiry time of the authentication token. The generated authentication token is returned in the X-AUTH-TOKEN response header.

To authenticate via the generated authentication token a user has to send the authentication token as part of the X-AUTH-TOKEN header for each new request. The content type of the body is application/x-www-form-urlencoded.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/authentication

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/authentication
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Query parameters

- **accountid** (string, optional) The account id. Account users have to provide it. Users with the role SUPERUSER must leave this field empty
- **authid** (string): Authentication id, needed for autologoff feature.
- **credentials** (string, required): Either the userid or email address of the user
- **newpassword** (string, optional): This field can be set to specify a new password for the user.
- **password** (string, required): The password of the user
- **usedefaultaccount** (boolean, optional): Server uses sole account id implicitly, applicable if SignDoc Standard system has only one account configured. Default value: false

Response class

Status 200 (OK): The authentication token was successfully created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

RestAuthentication

- **accountID** (string, optional)
- **accountName** (string, optional)
- **apikey** (string, optional)
- **eMail** (string, optional)
- **exp** (integer, optional): The expiration date of the authentication token in number of milliseconds from the epoch
- **globalLicense** (boolean, optional)
- **iat** (integer, optional): The 'issued At' date of the authentication token in number of milliseconds from the epoch

- **roles** (Array[string], optional)
- **userId** (string, optional)
- **userName** (string, optional)

Create server administrator

To create a server administrator, an input JSON or XML string specifying the details of the user, has to be provided as a request parameter. The specification has to provide at least a user name with more than two characters and an email address. To specify an initial state the following choices are available: ACTIVE, SUSPENDED and INVITED.If no state is specified the default state will depend on if a password has been specified. If a password has been specified, the default state will be ACTIVE, if not, INVITED. If the specified user state is ACTIVE or SUSPENDED, a password has to be specified too. If the specified state is INVITED, the password has to be missing.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmin

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/serveradmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id": "user1",
  "name": "user1",
  "password": "Test123!",
  "email": "user1@ksd.com",
  "roles": ["SUPERUSER"]
}
```

Body parameters

- **user** (RestUserInput, required): Input JSON or XML string of user

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password.
- **id** (string, optional): The id of the user
- **name** (string, optional): The name of the user
- **password** (string, optional): [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

Response status

Status 201 (Created): The server administrator was successfully created. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Invite or reinvite a server administrator

This request sends an invitation or a reinvitation to a server administrator specified by id.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}/invitation

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/serveradmins/admin1/invitation
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the user

Response status

Status 200 (OK): The server administrator was successfully invited or reinvited. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Reset password of a server administrator

This request resets the password of a server administrator identified by the id or email address (@ is encoded as %40).

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}/pwreset

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/serveradmins/kssdadmin/pwreset
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id or email of the user

Response status

Status 200 (OK): The password was successfully reset. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Invite or reinvite a user

This request sends an invitation or a reinvitation to a user specified by id.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/invitation

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/invitation
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the user

Query parameters

- **accountid** (string, optional): The id of the account. It is only necessary if a user with role SUPERUSER wants to invite or reinvite an account administrator.

Response status

Status 200 (OK): The user was successfully invited or reinvited. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Create 'password recovery' notification for a user

This request is used, when a user has forgotten his password. An email notification is sent to the user enabling him to set a new password.

Note For this request no authentication is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/pwrecovery

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/pwrecovery
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): Either the userid or email address of the user

Query parameters

- **accountid** (string, optional): This parameter can be omitted if the user is bound to an account
- **usedefaultaccount** (string, optional): Server uses sole account id implicitly, applicable if the SignDoc Standard system has only one account configured. Default value: false

Response status

Status 200 (OK): The recovery notification was sent successfully or the notification was not sent because of id or e-mail of a user is not correct. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Reset password of a user

This request resets the password of a user identified by the id or email address (@ is encoded as %40). The account id has to be specified if the requestor is not associated with the account (has the role SUPERUSER). The client should send the account id as form parameter in the body. Beside that anybody provides the accountid as query parameter.

Note For this request a valid authentication with ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/pwreset

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/pwreset
```

Example header

```
Accept: application/json

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

Path parameters

- **userid** (string, required): The id or email of the user

Form parameters

- **accountid** (string, optional): The id of the account. This parameter can be omitted if the user is bound to an account and has ADMIN role. Users with the role SUPERUSER have to provide it.

Response status

Status 200 (OK): The password was successfully reset. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Set API key for a specified user

This request sets the API key for the specified user. The valid password of the current user must be provided in order to set the API key. The API key must consist of the following allowed characters: a-z, A-Z, 0-9, '_', '.', and '-'. The API key must be at least 12 characters long.

Note For this request a valid authentication with ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/apikey

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/apikey
```

Example header

```
Accept: application/json

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

Path parameters

- **userid** (string, optional): The id of a user

Form parameters

- **password** string, required): The password of the user
- **apikey** (string, required): The new API key for the user

Response status

Status 200 (OK): The API key was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Add role to user

This request adds a role to a user specified by id. Valid roles are ADMIN or TEAMMGR.

Note For this request a valid authentication with SUPERUSER or ADMIN role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/role

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/role
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the user

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account and has ADMIN role. Users with the role SUPERUSER have to provide it.
- **role** (string, optional): Name of the new role. Possible values: ADMIN, TEAMMGR

Response status

Status 200 (OK): The role was successfully added. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a server administrator

This request updates a server administrator using an input JSON or XML string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with SUPERUSER role is necessary.

Breaking REST API change since SignDoc Standard 2.1.0.1

To change the email of a server administrator, the password attribute of the authenticated user must also be set in the request payload. This is the new default behavior. This is a breaking change to the previous REST APIs. This new behavior can be disabled by setting the configuration property cirrus.rest.user.email_change.require_password to "false". This configuration property can also be disabled in the Account Administration section.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/users/serveradmins/ksdadmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

```
{
  "password": "Test123!",
  "email": " ksdadmin@ksd.com ",
```

```
  "name": "ksdadmin"
}
```

Path parameters

- **userid** (string, required): The id of the user to update

Body parameters

- **user** (RestUserInput, required): Input JSON or XML string of user

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password attribute.
- **id** (string, optional): The id of the user
- **name** (string, optional): The name of the user
- **password** (string, optional): [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise. See email attribute.
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

Response status

Status 200 (OK): The server administrator was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Update a user

This request updates an existing user within an account using an input JSON or XML string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON or XML string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

Note For this request a valid authentication with USER or SUPERUSER role is necessary.

Breaking REST API change since SignDoc Standard 2.1.0.1

To change the email of a user, the password attribute of the authenticated user must also be set in the request payload. This is the new default behavior. This is a breaking change to the previous REST APIs. This new behavior can be disabled by setting the configuration property cirrus.rest.user.email_change.require_password to "false". This configuration property can also be disabled in the Account Administration section.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/users/user1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "id": "user1",
  "name": "user1",
  "email": "user1@ksd.com",
  "state": "ACTIVE",
  "roles": [
      "USER",
      "TEAMMGR",
      "ADMIN"
    ]
}
```

Path parameters

- **userid** (string, required): The id of the user to update

Query parameters

- **accountid** (string, optional): The account id. This parameter can be omitted if the user is bound to an account and has role ADMIN. Users with the role SUPERUSER have to provide it.

Body parameters

- **user** (RestUserInput, required): Represents a RestUserInput object either in JSON or XML.

RestUserInput

- **currentContext** (string, optional): The current context of the user
- **email** (string, optional): The email of the user. Changing a users email address may require to also set the authenticated user's password. See password attribute.
- **id** (string, optional): The id of the user
- **name** (string, optional): The name of the user
- **password** (string, optional):
  [POST]: The password of the new user. [PUT]: Password of the authenticated user, in the case that the email address of a user should be changed and the configuration property cirrus.rest.user.email_change.require_password is set to "true". Ignored otherwise. See email attribute.
- **phone** (string, optional): The phone of the user
- **roles** (Array[string], optional): A list of all roles the user possesses
- **settings** (RestUserSettings, optional): Contains user specified settings. When necessary, the strings in this class can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **state** (string, optional): The state of the user. Possible values: ACTIVE, SUSPENDED, INVITED

RestUserSettings

- **defaultPackageName** (string, optional): The default package name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time
- **defaultTemplateName** (string, optional): The default template name used for this user. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time

Response status

Status 200 (OK): The user was successfully updated. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a server administrator

This request deletes an existing server administrator specified by user id. The currently authenticated user cannot delete himself.

Note For this request a valid authentication with SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/serveradmins/ksdadmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the server administrator who should be deleted

Response status

Status 200 (OK): The server administrator was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Delete a user

This request deletes the existing user specified by account id and user id.

Note For this request a valid authentication with ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/user1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the user who should be deleted

Query parameters

- **accountid** (string, optional): The id of the account. This parameter can be omitted if the user is not bound to an account. Users with the role SUPERUSER have to provide it.
- **force** (boolean, optional): If this parameter is set to true, than the user will be deleted although he is owner of any signing packages. In this case all assigned signing packages are also deleted.

  Important Use this option only if you are sure, that you want to delete the user and all the resources which are attached to his assigned signing packages.

Response status

Status 200 (OK): The user was successfully deleted. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Remove role from user

This request removes a role from a user specified by id. Valid roles are ADMIN or TEAMMGR.

Note For this request a valid authentication with ADMIN or SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/role

Note *host_server* is the host domain name or IP address, and *port_number* is the host port number (if applicable).

Consumes and produces

JSON, XML

Header

Accept: application/json, application/xml

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/user1/role
```

Example header

```
Accept: application/json
```

```
Content-Type: application/json
```

Path parameters

- **userid** (string, required): The id of the user

Query parameters

- **accountid** (string, optional): The id of account. It can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **role** (string, optional): Name of the role. Possible values: ADMIN, TEAMMGR

Response status

Status 200 (OK): The role was successfully removed. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Chapter 4

Support GDPR

This chapter explains methods to access personal data according to the EU Regulation 2016/679 (General Data Protection Regulation, GDPR) specially to support the right to erasure (‘right to be forgotten’).

When SignDoc Standard is integrated in other business process management (BPM) systems personal data are usually provided by them. By itself SignDoc Standard does not transfer personal data to third countries or international organizations. Transferring finalized documents and audit trails is in the responsibility of the system or account owner.

Inform where personal data are collected

Server administrator

Help for SignDoc Standard Administration Center informs about where personal data are collected from the data subject. See Related documentation.

Account administrator, team manager, SignDoc user

Help for SignDoc Standard informs about where personal data are collected from the data subject. See Related documentation.

Signer or reviewer

The e-sign consent form allows customers to provide information which personal data are used, processed, or collected during the signing process. When they agree with the information, the signing process will continue. Signers and reviewers have also the right to decline the process. Actions are documented in the audit trail.

Audit trail

For each signing package an audit trail is generated when defined during creation of a package. In the audit trail all actions of SignDoc users, signers and reviewers performed for this specific signing package is logged.

Provide information about personal data (right of access by the data subject)

Server administrator

SignDoc REST API call

```
GET /rest/v8/users/serveradmins/{userid}
```

returns information about the collected data (id, name, email address, phone) for a server administrator.

Account administrator, team manager, SignDoc user

SignDoc REST API call

```
GET /rest/v8/users/{userid}
```

returns information about the collected data (id, name, email address, phone) for an account administrator or a team manager, or a SignDoc user.

SignDoc REST API call

```
GET /rest/v8/packages/{packageid}/audittrail
```

returns the audit trail of a specified signing package. The audit trail informs about workflow events initiated by a SignDoc user.

Signer or reviewer

SignDoc REST API call

```
GET /rest/v8/packages/{packageid}/signers/{signerid}
```

returns information about the collected data (id, email, name, user credentials, tspSignerInfo) of a signer or reviewer according to the specified signing package.

SignDoc REST API call

```
GET /rest/v8/signers/{email}
```

returns a signer based on the email.

SignDoc REST API call

```
GET /rest/v8/packages/{packageid}/audittrail
```

returns the audit trail of a specified signing package. The audit trail informs about workflow events initiated by a signer or reviewer.

Update personal data (right of rectification)

Server administrator

SignDoc REST API call

```
PUT /rest/v8/users/serveradmins/{userid}
```

updates an existing server administrator.

Account administrator, team manager, SignDoc user

SignDoc REST API call

```
PUT /rest/v8/users/{userid}
```

updates an existing account administrator or a team manager, or a SignDoc user.

Signer or reviewer

SignDoc REST API call

```
PUT /rest/v8/packages/{packageid}/signers/{signerid}
```

updates a signer or reviewer of a specified signing package.

Delete personal data (right of erasure, ‘right to be forgotten’)

Server administrator

The SignDoc REST API call

```
DELETE /rest/v8/users/serveradmins/{userid}
```

deletes an existing server administrator.

Account administrator, team manager, SignDoc user

The SignDoc REST API call

```
DELETE /rest/v8/users/{userid}
```

deletes an existing account administrator or a team manager, or a SignDoc user.

Signer or reviewer

The SignDoc REST API call

```
DELETE /rest/v8/packages/{packageid}/signers/{signerid}
```

deletes a signer or reviewer of a specified signing package.

The SignDoc REST API call

```
DELETE /rest/v8/packages/{packageid}
```

deletes a signing package as well as all signers, documents and fields associated with the specified signing package. Also the audit trail entries related to the package are removed by default (configurable).