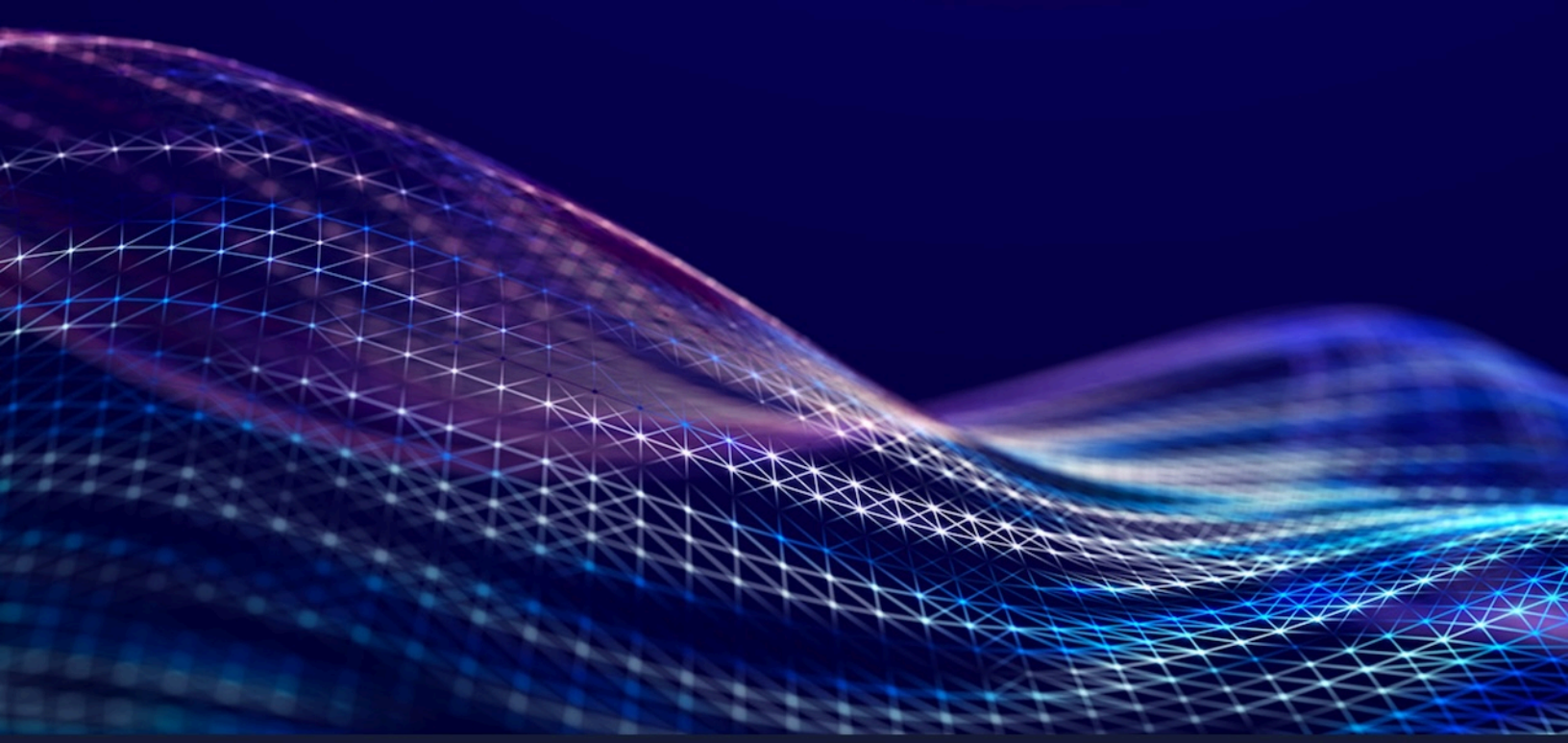


Tungsten SignDoc Standard

Developer's Guide

2025.3

TUNGSTEN AUTOMATION

Table of Contents

Preface

This programming guide provides information on how to work with the Tungsten 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 Tungsten SignDoc Standard programmatically.

Related documentation

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

https://docshield.tungstenautomation.com/Portal/Products/SD/2025.3-u5pe54ws45/SD.htm

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

Release notes

- *Tungsten SignDoc Release Notes*

Technical specifications

- *Tungsten SignDoc Technical Specifications*

Guides

- *Tungsten SignDoc Standard Administrator's Guide*
- *Tungsten SignDoc Standard Installation Guide*

Help

- *Tungsten SignDoc Standard Help*
- *Tungsten SignDoc Standard Administration Center Help*
- *Tungsten SignDoc Assistant Help*
- *Tungsten SignDoc Help - Signing Documents*
- *Tungsten SignDoc Device Connector Help*
- *Tungsten SignAlyze Help*

Software development kit

- *Tungsten SignDoc Browser Capture Help*
- *Tungsten SignDoc SDK API Documentation (C)*

- *Tungsten SignDoc SDK API Documentation (C++)*
- *Tungsten SignDoc SDK API Documentation (.NET with exceptions)*
- *Tungsten SignDoc SDK API Documentation (.NET without exceptions)*
- *Tungsten SignDoc SDK API Documentation (Java)*

Training

Tungsten Automation offers both on-demand and instructor-led training to help you make the most of your product. To learn more about training courses and schedules, visit the Tungsten Automation Learning Cloud.

Getting help with Tungsten Automation products

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

To access the Tungsten Automation Knowledge Portal, go to:

https://knowledge.tungstenautomation.com/

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

The Knowledge Portal 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 select the search icon.
- Product information, configuration details and documentation, including release news.
  To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or select the View All Products button.

From the Knowledge Portal home page, you can:

- Access the Tungsten Automation Community (for all customers).
  On the Tungsten Automation Resources menu, select the **Community** link.
- Access the Tungsten Automation Customer Portal (for eligible customers).
  Go to the Support Portal Information page and select **Log in to the Tungsten Automation Customer Portal**.
- Access the Tungsten Automation Partner Portal (for eligible partners).
  Go to the Support Portal Information page and select **Log in to the Tungsten Automation Partner Portal**.
- Access support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.
  Go to the Support Details page and select the appropriate article.

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.

API usage guidelines

The following guidelines help ensure reliable and consistent use of the REST API by minimizing the risk of adverse side effects through proper request serialization.

Concurrent requests

Concurrent state altering requests (POST/PUT/DELETE) on the same REST resource (e.g., a signing package) or any of its sub-entities are not supported.

Example: Any state-altering requests to document(s), signer(s) or the signing package object itself of a signing package must not be performed concurrently (in parallel) but sequential and in order.

It's recommended that REST clients interact with the REST API either using a single thread or synchronize all requests to prevent unpredictable behaviors such as stale data reads or race conditions.

Sequential operation

For consistent data integrity and application state, clients should ensure API calls are executed sequentially. Each API request should be fully processed and committed before initiating another.

Concurrency advisory

Issuing parallel requests can lead to side effects such as data overwrites or transaction conflicts due to the unordered nature of multi-threaded processing.

Best practices

Implement a client-side queue to serialize API requests effectively, ensuring stable and predictable interaction with the API.

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
```

The Current API version and REST Base URL can also be found out dynamically by:

```
GET /rest/api/current
```

Example response:

```
{ "id": "v8", "url": http://localhost:6611/cirrus/rest/v8 }
```

REST error response

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

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 input, 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): A list of messages (info, warnings, errors).

RestMsg

- **code** (integer): The code of the message.
- **message** (string): The description of the message.
- **type** (string): The type of the message. Valid 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 processing 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 to 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"
}
```

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 specific text field on the header of the Swagger UI. To generate the API key by SignDoc Standard the user has to navigate to the preference page of his individual account. There he can find a separate section for generating API keys. After generating an API key, you should store it in some safe place for further usage.

Each API key generated possesses the exact same permissions as the user it is associated with. This means that a user with the account administrator role can only use his API key with the exact same permissions. With the API key, an account administrator can use the REST API to create users, teams and signing packages by using the appropriate REST request. But as an account administrator, he cannot 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. The returned format is determined

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

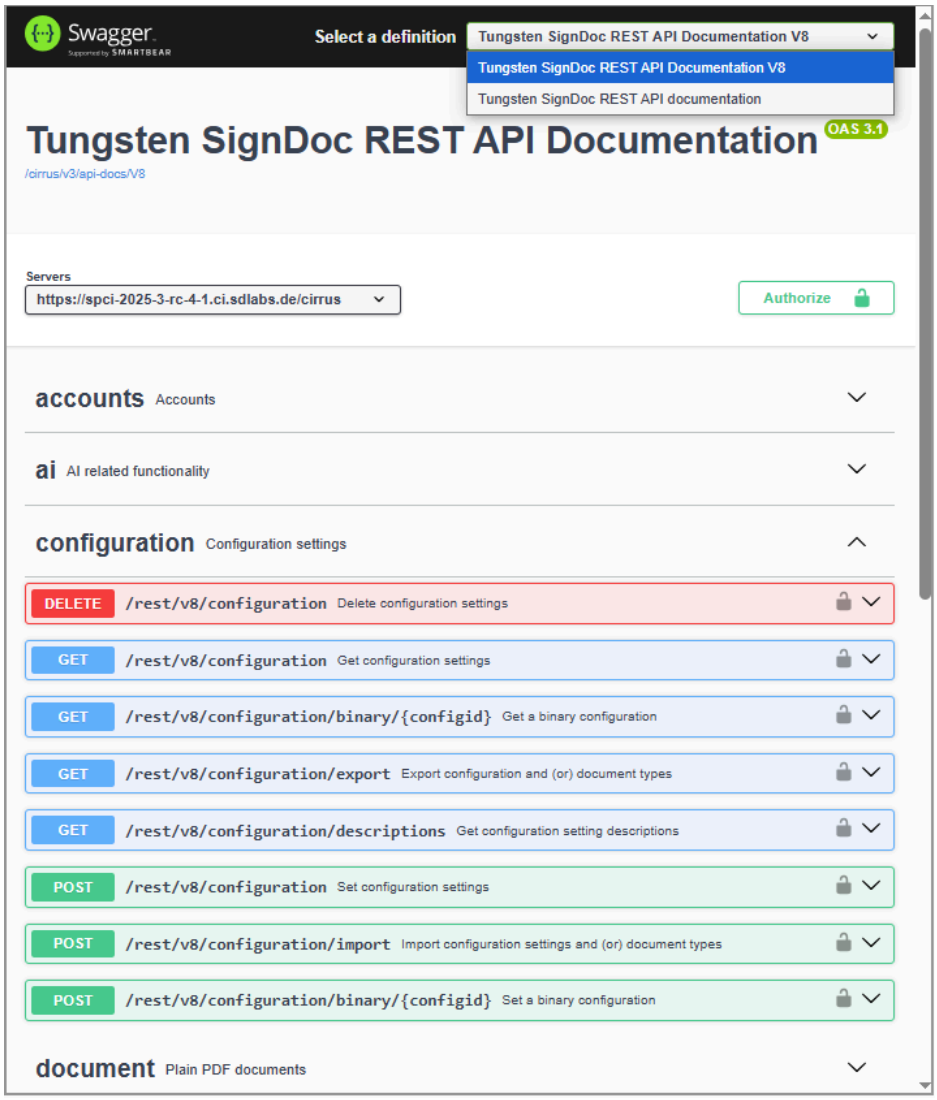
Swagger UI

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 is enabled by default and does not require any more a `spring.profiles.active=swagger` entry in the `application.properties` file.



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 also depend 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 • Make business settings for all the accounts • Get statistics for all the accounts and users
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.	The account administrator is allowed to: • Create users associated with the account • Change business settings for the whole account • Create teams • Get statistics for all users of the account
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 • Get statistics for themselves

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 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 STAGE_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 the 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 reviewer. 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 the 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. When a recipient delegates a signing session, a notification via email is sent to the owner of the signing package.

 A parent recipient with access code 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. A recipient with a TSP plug-in registered and who has TSP info set can delegate a signing session since SignDoc 3.2 release, however the TSP info of the original recipient cannot be changed.

Create express package

POST /rest/v8/expresspackage

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 is specified, a notification with a link to the remote signing session is sent to the signer.

Extension of signature field signing modes

The signature field structures `RestSignatureFieldInput` and `RestSignatureFieldOutput` have the attribute `restSigningModeOption`. 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, IMG for sign with (up-)loaded image and now also STAMP for adding a stamp with (up-)loaded image and TSP as external trust service provider for signature. The default is HW and IMG.

Stamp image

A new signing method "Stamp image" is introduced with SignDoc Standard 3.0.0.

For a signer it is now possible to add a stamp to a document with a stamp image which was loaded on the client side during the signing session. Once loaded in SignDoc the stamp 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.

TSP signature

A new signing method "TSP" is introduced as part of SignDoc 3.0.0 release. The signer can now sign a signature field using a TSP plug-in. When a 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 a relevant message displayed on the screen. The signature returned from the TSP service is applied to the field on success.

SignDoc statistics data

GET rest/v8/statistics

This new endpoint allows users to fetch SignDoc statistics data in a given time frame. A system administrator can fetch the statistics data for all its users or a specific user belonging to all its accounts. An account administrator on the other hand can view statistics data for all its users or a specific user in its own account. A user with the role USER can only view statistics for themselves and is not allowed to view statistics of other users.

Initials field

SignDoc 3.3.0 release supports a new document field named as initials field. The initials can be added by both the signer and a reviewer. The new endpoints support include: Add, update and get an initials field, along with add and clear initials.

Use TotalAgility Licensing server

SignDoc 3.4.0 introduces the capability for users to leverage SignDoc licenses via the TotalAgility Licensing server. This functionality can be enabled or disabled through a configuration option,

allowing users to revert to traditional SignDoc account-specific or global licenses. Administrators have the authority to enable this feature at the system level, making it accessible to all accounts, or at the account level.

For communication between SignDoc and the TotalAgility Licensing server, administrators must furnish the server's URL and a unique API key. The API key enables secure communication and can be shared across multiple accounts.

To facilitate user creation and signing package preparation, a "SignDoc Base" license is mandatory within the license list of the licensing server. Furthermore, customers are required to procure user-related and signing package-related licenses to create users and prepare signing packages respectively, ensuring they remain active and valid and activated in the licensing server.

Support Microsoft and Google as authentication providers for recipients

In SignDoc 3.4.0, users can designate Microsoft and Google as authentication providers for recipients. A package owner has the flexibility to choose Microsoft, Google, or both as authentication providers for recipients, along with the option of "EXTAUTH". Additionally, package owners can select and use access code authentication ("CODE") alongside these authentication providers. In versions 3.3.0 and earlier, users were able to utilize either "CODE" or "EXTAUTH" and not both for the same recipient. To summarize, package owners have the flexibility to assign one or more, or all, authentication providers simultaneously, and recipients can then select any one of the providers from the list in the signing client and complete the external authentication. The outcome of the authentication is logged in the audit trails of the respective signer. To utilize Microsoft or Google authentication options, the SignDoc application must be installed with the configuration settings outlined in the *Tungsten SignDoc Standard Administrator's Guide*.

Maintain and use signed documents

SignDoc 3.4.0.1 allows and maintains the signatures in an already signed document. The Manage Client does not allow editing such fields. The SignDoc user can upload/prepare an already signed PDF document. New fields are added only if the document's integrity and the signed fields can be maintained as original.

Enable polling for TSP plug-in status

Starting with SignDoc 2025.3, users can implement TSP plug-ins that operate asynchronously. Instead of returning an immediate result, the plug-in returns a request ID upon initiation. The status of the TSP operation can then be monitored by polling the /rest/v8/tsp/signing/request/{requestId} endpoint using the provided request ID.

For scenarios requiring immediate results, users can still implement TSP plug-ins using the existing synchronous mode available in versions prior to SignDoc 2025.3.

Decline/Withdraw signing package documents

SignDoc 2025.3 allows both package owners and signers to decline/withdraw individual documents within a signing package providing flexibility to reject specific documents without having to decline the entire package.

To enable this feature, the account administrator must set the configuration `cirrus.package.document.rejection.enabled = true`. Once enabled, the document rejection feature is available only for signing packages in the STARTED state.

A document is rejected in either a user or a signer context. The document can be rejected by using the Post event request /rest/v8/event, and passing the event type as 'REJECTED_DOCUMENT'.

Sample event requests look like below.

Example:

```
Signer Request { "list": [ { "k": "action", "v": "REJECTED_DOCUMENT" },
 { "k": "product", "v": "CIRRUS" }, { "k": "subject", "v": "SIGNER" }, { "k":
 "REJECTED_DOCUMENT_REASON", "v": "R2" }, { "k": "REJECTED_DOCUMENT_COMMENT", "v":
 "Other" }, { "k": "DOCUMENT_ID", "v": "79dcb1f0-af01-4f1f-b9de-ee9a6e941889" } ] }
```

Example:

```
User Request { "list": [ { "k": "action", "v": "REJECTED_DOCUMENT" }, { "k": "product",
 "v": "CIRRUS" }, { "k": "subject", "v": "USER" }, { "k": "REJECTED_DOCUMENT_REASON",
 "v": "R2" }, { "k": "REJECTED_DOCUMENT_COMMENT", "v": "Other" }, { "k": "DOCUMENT_ID",
 "v": "79dcb1f0-af01-4f1f-b9de-ee9a6e941889" }, { "k": "PACKAGE_ID", "v": "a262337f-
d715-4b1a-98ba-6e41c59fb939" } ] }
```

In the above request samples:

- REJECTED_DOCUMENT_COMMENT is optional.
- PACKAGE_ID is required in the user context and optional in the signer context.

The rejected documents can be retrieved using:

1. GET /rest/v8/packages/{packageId}
   Returns both user- and signer-rejected documents.
   Rejected documents by signers will include a `signerId`.
   Documents rejected by owners will not include a `signerId`.
2. GET /rest/v8/packages/{packageId}/signers/{signerId}
3. GET /rest/v8/packages/{packageId}/documents/{documentId}
   Each document includes a rejected flag and, if rejected, a `rejectedDocument` object.

Once a document is rejected, no further changes are allowed, and it can no longer be signed or reviewed. A signing package is marked as **completed** when all its documents are either withdrawn or declined. If a package is **canceled** (voided) by the owner, all documents within it are automatically rejected. Email notifications vary based on the recipient and the specific documents that were rejected.

It is also possible to register a webhook to get informed, when a document is declined/withdrawn. The configuration properties for webhook support for the document rejection feature are available in the *Tungsten SignDoc Administrator's Guide*.

Changed requests in v8

The following requests are changed in v8.

Signer search and Get a list of signers

Starting with SignDoc 3.1.0 release the "Signer search" and the "Get a list of signers" requests will have a new parameter in the `RestSignerListEntry` response body namely

`tspSignerInfoAvailable`. This parameter in the response is set to true if the signer has a TSP plug-in type and TSP signature type available.

GET Account(s)

Starting with SignDoc 3.4.0 release, GET rest/v8/accounts and the GET rest/v8/account endpoints will include details pertaining to SignDoc licenses available in the licensing server, alongside information regarding account-specific or global licenses.

Information regarding licenses from the TotalAgility Licensing server will be encapsulated within the new `totalAgilityLicenseInfo` attribute, while details concerning SignDoc licenses will be presented in the already existing `licenseInfo` parameter. Users can identify the active license through the new `licenseType` attribute. In the event of an error with the TotalAgility Licensing server, the new `totalAgilityLicenseInfo` attribute will return `totalAgilityLicensingServerError` along with appropriate error 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
2.1.0.1	Changing the email address of a user or server administrator requires the password of the authenticated user. Links: Update a user Update a server administrator
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 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.

Since SignDoc version	Change
3.0.0	A user can now save a binary configuration along with a password for type ENCRYPTED and subtype PKCS12. This is mostly the 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.
3.0.0	A new signing method "Sign with a stamp" is introduced as part of SignDoc 3.0.0 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 the 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 SignDoc 3.0.0 release. The signer can now sign a signature field using TSP using a TSP plug-in that supports the same. When a 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 a 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 APIs v2, v3, v5 are no longer supported.
3.1.0	The final PDF document sent when the package is completed will not be sent as a container document by default. By default, the recipient will receive all the documents of the signing package as multiple attachments along with the final signed and combined audit trail for all the signers and the documents. The recipients can still view the container document by using the "Open Documents" link in the email. If the recipient still wants a container document as in SignDoc 3.0, then it needs to be configured explicitly in the administration section.

Since SignDoc version	Change
3.1.0	As the documents in the final email sent after the package is complete are required to be PDFs, the file names of the uploaded documents are modified to add pdf extension. In SignDoc 3.0, if the user uploaded a file with name abc.png, the name was changed to abc.png.pdf. In SignDoc 3.1, abc.png will be changed to abc.pdf. If the user added documents to a signing package using REST, and if the file name is abc.cde this will be replaced with abc.pdf as opposed to SignDoc 3.0 where it was abc.cde.pdf. From SignDoc 3.1, when creating the file name for the final signed PDF, it is assumed that the index of the last "." character in the original file name is the file extension.
3.1.0	A package template could be created for an expiry date in the past than the current date. If a copy is made of an expired package or an expired template is used the expiry date of the new package will be set to null without throwing any error.
3.2.0	From SignDoc 3.2 release `auditTrailOptions`, the audit trail options of the final compound PDF document after package is completed in the `RestSigningPackageInput` object will understand either 0 or 1, unlike previously where values from 0-3 were allowed. 0 means no audit trail 1 means both package and document audit trail is added. For backward compatibility other values are accepted but anything greater than 0 will be considered as 1.
3.2.0	From SignDoc 3.2 release, a user has an option to enforce a signer to sign all the signature fields in a document with the signing mode it used to sign the first signature field. For this the package creator should set the `enforceSigningMode` flag in the `RestSigningPackageInput` when creating the signing package otherwise the default value from configuration setting `cirrus.package.enforceSigningMode.default` is used.
3.2.0	From SignDoc 3.2 release, the GET /rest/v8/packages/{packageid}/finaldocument also tries to create the final document if it is not available and if the package is already completed. Users can also create the final container document of the signing package if the creation failed due to some error using the new endpoint POST /rest/v8/packages/{packageid}/createFinalDocument This endpoint initiates the creation and would take time until the document is available for download, so users must wait between the two operations.
3.3.0	From SignDoc 3.3 release, the 'Remove user from team', DELETE /rest/v8/teams/{teamid}/users/{userid} will also work for users with the role USER. However, a user with only role USER can only remove itself from the team and not anyone else.
3.3.0	From SignDoc 3.3 release, a user can update an already STARTED package using the PUT signing package endpoint. The update allows only updates of signer related entries (not started or final state signers), updates containing other entities will be rejected. A new state STARTED is introduced for fully authenticated signers. For SEQ and STAGED signing packages if the signing order of a signer is changed, a new signing invitation email is sent to the signer when the signers turn arrive again. The signer who already received an invitation email, with the order changed will not be able to proceed until its new order is valid.

Since SignDoc version	Change
3.4.0	The SignDoc 3.4.0 release introduces support for the TotalAgility Licensing server. Now, both the GET rest/v8/accounts and the GET rest/v8/account endpoints will include details pertaining to SignDoc licenses available in the licensing server, alongside information regarding account-specific or global licenses. Users will not be able to import account-specific and global licenses if the configuration setting to use TotalAgility Licensing server is enabled. It is only possible again when the user switches off this setting. After the setting is switched off, users can use the old account-specific or global licenses without needing to upload a new one if the existing licenses are valid.
3.4.0	SignDoc 3.4.0 allows package creators to update access code and phone number of a recipient that has already started but not completed their signing session. This will alleviate issues related to lost access codes and changes in recipients' phone numbers.
3.4.0	Version 3.4.0 introduces the capability for users to assign a list of authentication providers to signers. In versions 3.3.0 and earlier, external authentication was configured using the `authenticationMode` attribute in the signer object. While this method is still supported for backward compatibility, it is marked for deprecation and will be removed in newer REST versions (v9). Users who wish to utilize the new features, such as using Microsoft and Google as authentication providers, should utilize the new attribute `authenticationProviders`. For example, "authenticationProviders" :[{"id":"MICROSOFT"}] for a single provider, or "authenticationProviders" :[{"id":"MICROSOFT"}, {"id":"GOOGLE"}, {"id":"EXTAUTH"}] for multiple providers. For backward compatibility, the REST signer response includes the `authenticationMode` attribute populated with appropriate values like CODE, EXTAUTH, and NONE. If `authenticationProviders` are employed to create a signer with multiple authentication providers, and it contains a value not available in `authenticationMode`, the response is populated as "authenticationMode": "UNSUPPORTED".
3.4.0	In SignDoc version 3.4.0, when the requesting user has the ADMIN role and no filter or TEAMS filter option is provided in the request, the GET endpoint /account will return all teams associated with the account for that user.
3.4.0.1	In SignDoc 3.4.0.1, users can upload a document with already signed signature fields. Such fields are maintained and are not cleared. Prior releases cleared such fields. The copy signing package feature also maintains such fields that were signed in the original document. RestDocumentOutput and RestDocumentListEntry returns permittedDocumentActions argument which states what actions are permitted on the document without compromising the document's validity.

Export via API

SignDoc offers REST web services for exporting data programmatically without accessing the user interface.

Access the API documentation through the Swagger UI on your SignDoc server. For example, navigate to:

`https://<host_server>:<port_number>/cirrus/swagger-ui/index.html`

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

This request returns the latest, that is, the current API version of the REST API along with the REST API's Base URL.

ℹ For this request no authentication is necessary.

URL

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

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

Produces

JSON

Header

Accept: application/json

Method

GET

Example request

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

Example request headers

```
Accept: application/json
```

Responses

Status 200 (OK): The latest REST API version was queried successfully. See RestID. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Example response body (JSON)

```
{
"id": "v8",
"url": "https://xxx/cirrus/rest/v8"
}
```

Get all active APIs

This request returns a list of all 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

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

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)

```
{
  "api": http://localhost:6611/cirrus/rest/api,
  "v8": http://localhost:6611/cirrus/rest/v8
}
```

Chapter 3

REST API reference v8

This chapter describes the REST API requests and schemas in detail.

Account requests

The following requests are related to SignDoc accounts.

- Delete an account
- Delete a signing certificate
- Delete a public key
- Get a single account
- Get account personalization
- Get all accounts
- Get number of accounts
- Get status information of account entities
- Get account logo
- Get notification type descriptions
- Get registered authentication providers for the account
- Create an account
- Update account personalization
- Import a license to an account
- Update an account

Delete an account

This request deletes an account as well as all users and signing packages associated with the specified account.

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

URL

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

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

Produces

JSON

Header

Accept: application

Method

DELETE

Example request

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

Example header

```
Accept: application/json
```

Parameter

- **accountid** (string, path, required): The ID of the account.

Responses

Status 200 (OK): The account 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.

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

URL

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

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

Produces

JSON

Header

Accept: application

Method

DELETE

Example request

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

Example header

```
Accept: application/json
```

Parameter

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

Responses

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

Delete a public key

This request deletes a public key for the specified account.

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

URL

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

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

Produces

JSON

Header

Accept: application

Method

DELETE

Example request

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

Example header

```
Accept: application/json
```

Parameter

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

Responses

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

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 you to customize the response body of the request. Depending on a comma separated list of filter options the response body can return additional information.

For this request, a valid authentication with a USER, ADMIN or SUPERUSER role is necessary. If the requesting user has only role USER, 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

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
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, query, optional): The ID of the account. 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]), query, optional]: The account filter option. Valid values: USERS, TEAMS, NONE
- **useIntId** (boolean, query, optional): The provided account ID is the internal ID which was returned by the "Get audit trail" method.

Responses

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

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"
}
```


Get account personalization

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

For this request, no authentication is necessary.

URL

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

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

Method

GET

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, query, optional): The ID of the account. Either `accountid` or the query parameter `usedefaultaccount` must be specified for retrieving account-specific personalization data.
- **defaultvalues** (string, query, 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, query, optional): If only one account is available in the current installation, this parameter can be set to true to get the personalization data from this account without specifying `accountid`. Default value: false

Responses

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

Get all accounts

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

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

URL

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

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

Method

GET

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Responses

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

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"
    },
 "totalAgilityLicenseInfo": [
       {
         "id": 224,
         "licenseName": "SignDoc Signing package",
         "expiryDate": "1899-12-30T00:00:00-08:00",
         "availableVolume": 18495,
         "periodStartDate": "2024-01-01T00:00:00-08:00",
         "licenseState": "VALID"
       },
 "licenseType": "TOTALAGILITY",
 "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"
    },
 "totalAgilityLicenseInfo": [
      {
        "totalAgilityLicensingServerError": {
          "statusCode": "403",
          "errorMessage": "Forbidden"
        }
      }
    ],
    "licenseType": "TOTALAGILITY",
 "url": "http://localhost:6611/cirrus/rest/v8/account?accountid= account2"
  }
]
```


Get number of accounts

This request returns the number of accounts.

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

URL

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

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

Method

GET

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Responses

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

Example response body (JSON)

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

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

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
```

Parameter

- **accountid** (string, query, optional): The ID of the account. A user with role SUPERUSER can provide the account ID to get account-specific notification types (enabled) in addition. If the account ID is omitted, only account-independent notification type descriptions (enabled) are returned.

Responses

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

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.

For this request no authentication is necessary.

URL

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

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,

Header

Accept: application

Method

GET

Example request

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

Parameters

- **accountid** (string, query, optional): The ID of the account. Either `accountid` or the query parameter `usedefaultaccount` must be specified for retrieving an account-specific logo.
- **usedefaultaccount** (boolean, query, optional): If only one account is available in the current installation this parameter can be set to true to get the logo image for this account without specifying `accountid`. Default value: false
- **defaultimage** (boolean, query, 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
- **logoType** (string, path, required): The logo type. Possible values: loginlogo, headerlogo
- **imageFormat** (string, path, 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

Responses

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

The response body returns the binary content of the logo.

Get notification type descriptions

This request retrieves notification type descriptions of loaded plug-ins which support `NotificationParametersEvent`.

Notification types can be retrieved for globally enabled plug-ins (without specifying an account ID), or for account-specific plug-ins (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

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

Method

GET

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, query, optional): The ID of the account. If omitted, notification type descriptions for plug-ins that are enabled account independent are returned.
- **languagetag** (string, query, optional): The language tag for retrieving the locale-specific configuration setting description (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, en-US will be used.

Responses

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

Get registered authentication providers for the account

This request provides a list of authentication providers for the account of the user, along with OAuth registrations that are allowed for the account.

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

URL

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

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, application

Method

GET

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **accountid** (string, query, required): The ID of the account. The ID should match the account of the user making the request. A user with role SUPERUSER can request providers for any account.

Responses

Status 200 (OK): The authentication providers for the account were successfully queried.

Example response body (JSON)

```
[
  "CODE",
  "EXTAUTH",
  "GOOGLE",
  "MICROSOFT"
]
```

Create an account

To create an account, an input JSON 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.

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

URL

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

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

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Request body

- RestAccountInput (required): The input JSON of the account.

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!"
   }]
}
```

Responses

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

Update account personalization

This request updates account-specific personalization information.

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

Method

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, optional): The ID of the account. 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 of 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 of 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, such as 31708f.
- **loginLogoType** (string, optional): The login logo type as an image media type, such as image/png.
- **headerBackground** (string, optional): The header background color as RGB hex value, such as 0079C1.
- **headerForeground** (string, optional): The header foreground color as RGB hex value, such as ffffff.
- **footerBackground** (string, optional): The footer background color as RGB hex value, such as f5f5f5.

- **footerForeground** (string, optional): The footer foreground color as RGB hex value, such as 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, https://www.tungstenautomation.com/contact-us/product-inquiry or here, https://knowledge.tungstenautomation.com/bundle/z-kb-articles-salesforce8/page/34366.html

Responses

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

Import a license to an account

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

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

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

Method

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Request body

- RestAccountLicenseInput (required): The input JSON of account license.

Responses

Status 201 (Created): The license was successfully imported. See RestAccountLicenseOutput. 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. Each attribute of the existing account can be changed by adding the corresponding attribute name and the new value to the input JSON. Each attribute included in 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.

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

URL

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

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

PUT

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

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

Request body

- RestAccountInput (required): The input JSON of the account.

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!"
   }]
}
```

Responses

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

AI requests

The following requests can be used for AI related functionality.

- Return the main language used in a document
- Generate document description using an AI
- Get the enabled status of an AI task

Return the main language used in a document

This request returns an AI generated analysis of the main language (expressed in the English language) used in document. It is required that a working AI Provider is configured and enabled for the SignDoc account.

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

URL

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

Consumes and produces

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/{packageid}/documents/
{documentid}/ai/lang/main
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

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

Responses

Status 200 (OK): The main language was successfully analyzed.

Example response body (JSON)

```
{
  "response": "English"
}
```

Generate document description using an AI

This request returns an AI generated description of the specified document. To use this feature, an active AI provider must be configured and enabled in the SignDoc account. For very large documents, the final description may not match the exact word count requested, as it depends on the capabilities of the underlying AI model

A valid authentication with a USER role is required to make this request.

URL

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

Consumes and produces

JSON

Header

Accept: application

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/packages/{packageid}/documents/
{documentid}/ai/gen/desc
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the signing document.
- **language** (string, query, optional): The language to be used for generating the document description (preferably English). If not specified, will be determined by the AI document description language settings from the administration section.
- **numberOfWords** (integer, query, optional): The approximate number of words to be used to generate the document description, between 10 and 100. If not specified, will be determined by the AI document description settings for words from the administration section.

Responses

Status 200 (OK): The document description was successfully generated.

Example response body (JSON)

```
{
  "response": "Bewertungsformular für technischen Support mit
 Leistungsbeurteilung und Zielsetzung für Mitarbeiter. \n\nTranslated into German:
\nLeistungsbeurteilungsformular für technischen Support mit Mitarbeiterbewertung und
 Zielsetzung."
}
```

Get the enabled status of an AI task

This request returns if the specified AI Task is enabled.

A valid authentication with a USER role is required to make this request.

URL

http://*host_server:port_number*/cirrus/rest/v8/ai/status/task/{aitask}

Consumes and produces

JSON

Header

Accept: application

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/ai/status/task/{aitask}
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **aitask** (string, path, required): The AI Task

Responses

Status 200 (OK): The status was successfully returned in the response.

Example response body (JSON)

```
{
  "enabled": true
}
```

Configuration requests

Use the following requests for configuration settings.

- Delete configuration settings
- Get configuration settings
- Get a binary configuration
- Export configuration settings and (or) document types
- Get configuration setting descriptions
- Set configuration settings
- Import configuration settings and (or) document types
- Set a binary configuration

Delete configuration settings

This request deletes one or more configuration settings. The `accountid` parameter is necessary if the requesting user is a server administrator which needs to delete account-specific configuration settings. For all other users, the `accountid` parameter is ignored.

```
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.

Parameter

- **accountid** (string, query, optional): The ID of the account. 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.

Request body

- **restConfigEntries** (string, required): Input JSON 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 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."
    }
  ]
}
```

Get configuration settings

Global system settings and account-specific settings can be retrieved as server administrator, as account administrator, as user or as signing session (depending on 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}]
```

Parameters

- **accountid** (string, query, optional): The ID of the account. 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], query, optional): The prefix of the requested configuration keys. Multiple `startswith` parameters can be specified.
- **endswith** (Array[string], query, optional): The suffix of the requested configuration keys. Multiple `endswith` parameters can be specified.
- **effective** (boolean, query, optional): The 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, query, optional): The locale for retrieving the locale-specific audit logs (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/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 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 to 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 account ID 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}

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

Parameters

- **configid** (string, path, required): The ID of the setting.
- **accountid** (string, query, optional): The ID of the account. 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, query, 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, query, optional): Requests locale specific configuration data. Format must be compatible with Locale.languageTag.
- **includeoriginalcontenttype** (boolean, query, optional): Includes the original content type of the binary data in the response.

Responses

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

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

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

Parameters

- **accountid** (string, query, optional): The ID of the account. This parameter can be null if global configuration is exported.
- **effective** (boolean, query, optional): The 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, query, optional): The indicator whether a hash of the content must be calculated or not. Default value: false
- **plugin** (boolean, query, optional): The indicator whether plug-in configurations must be exported. Default value: true
- **types** (Array[string], query, required): Defines what kind of settings are exported. Valid types are CONFIGURATION and DOCTYPE. Multiple types are comma separated.

Responses

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

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}]
```

Parameters

- **accountid** (string, optional): The ID of the account. 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): The locale for retrieving the locale-specific configuration setting description (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/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. See RestConfigurationDescription.

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."
    }
  ]
}
```

Set configuration settings

This request sets one or more existing configuration settings. The request body contains a `RestDataList` object with a list of `RestConfigEntry` items. `RestConfigEntry` has a key "k" and a value "v". The `accountid` parameter is necessary if the requesting user is a server administrator who wants to update account-specific configuration settings. For all other users, the `accountid` parameter is ignored.

URL

```
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".

Parameter

- **accountid** (string, query, optional): The ID of the account. 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 ADMIN rolr can only update their own account.

Request body

- **restConfigEntries** (string, required): Input JSON 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 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."
    }
  ]
}
```

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 who 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

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

Request body

- **file** (string, required): File
- **accountid** (string, query, optional): The ID of the account. 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.

Response status

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

Set a binary configuration

This request sets a binary configuration.

URL

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

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

Parameter

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

Request body

- **confdata_bin** (string, file, required): The binary configuration data.
- **accountid** (string, query, optional): The ID of the account. 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, query, optional): Set for locale-specific configuration data. Format must be compatible with Locale.languageTag.
- **confdata_password** (string, query, optional): The password of the configuration binary data, in case the type is ENCRYPTED and subType is PKCS12.

Responses

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

Plain PDF document requests

The following requests are related to plain PDF documents.

- Validate the SecureID of a personal certificate signature
- Validate the C2S (click-to-sign) SecureID of a personal certificate signature
- Get information on a PDF document
- Sign a plain PDF document

Validate the SecureID of a personal certificate signature

Validate the C2S (click-to-sign) SecureID of a personal certificate signature

Get information on a PDF document

This request returns properties of the provided PDF document.

URL

http://*host_server:port_number*/cirrus/rest/v8/document/info

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

Example request

```
GET http://localhost:6611/cirrus/rest/v8/document/info
```

Parameter

- **restPlainDocumentInput** (object, query, required):

Responses

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

Sign a plain PDF document

With this request a user signs an existing signature field or document. This request will no persist any data.

URL

http://*host_server:port_number*/cirrus/rest/v8/document/signature

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

Example request

```
GET http://localhost:6611/cirrus/rest/v8/document/signature
```

Parameters

- **fieldname** (string, query, optional): The name of the field. If this parameter is not set the whole document will be signed instead.
- **autoprepare** (string, boolean, optional): If true, a field will be inserted in the upper left corner of the first page. This request will no persist any data.

Request body

RestPlainDocumentSigningInput (required): The input string.

Responses

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

Document requests

The following requests can be used for the documents of the signing packages.

- Delete a document type instance
- Delete a document
- Delete a specified document type
- Delete supplemental document
- Get a single document type instance
- Get a single document
- Get a specified document type
- Get document type instance of a single signer
- Download supplemental document content
- Download the final document
- Download all documents as zip
- Find text in a single document
- Get a document page image
- Download a single document
- Get available document types
- Add supplemental document
- Create a document type instance for a signer
- Add document to signing package

- Create a document type
- Update a document type instance
- Update a document
- Update a specified document type

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}

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

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
```

Parameters

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

Responses

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

Delete a document

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

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

URL

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

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

Method

DELETE

Example request

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

Example header

```
Accept: application/json
```

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

Parameters

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

Responses

Status 200 (OK): The document was successfully deleted. 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.

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}

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

Produces

JSON

Header

Accept: application/json, application

Method

DELETE

Example request

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

Example headers

```
Accept: application/json
```

Parameters

- **accountid** (string, query, optional): The ID of the account. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **locale** (string, query, optional): The locale for retrieving the language-specific Locale for retrieving the locale-specific document type(s) (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, 'en' will be used.
- **doctypeid** (string, path, required): The ID of the document type.

Responses

Status 200 (OK): The document type was 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.

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}

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

Consumes

JSON

Header

Accept: application/json

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
```

Parameters

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

Responses

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

Get a single document type instance

This request returns the requested document type instance.

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}

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

Consumes

JSON

Header

Accept: application

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
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **signerid** (string, path, required): The ID of the signer.
- **doctypeinstid** (string, path, required): The ID of the document type instance.
- **effective** (boolean, query, optional): The parameter 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, query, optional): Locale for retrieving the language specific document types (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/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.

Responses

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

Get a single document

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

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}

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

Consumes

JSON

Header

Accept: application

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **resolution** (float, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default: 72 dpi.
- **fields** (string, query, 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, query, 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, query, optional): Whether the document content (as Base64 string) is returned or not.

- **thumbnail** (boolean, query, optional): Whether the thumbnail image of the first document page (as Base64 string) is returned or not.
- **useIntId** (boolean, query, optional): The provided IDs for package and document are the internal IDs which were returned by get audit trail method.

Responses

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

Get a specified document type

This request returns the requested document type.

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}

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

Consumes

JSON

Header

Accept: application

Method

GET

Example request

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

Parameters

- **accountid** (string, query, optional): The ID of the account. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **locale** (string, query, optional): The locale for retrieving the locale-specific document type(s) (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, en will be used.
- **doctypeid** (string, path, required): The ID of the document type.

Responses

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

Get document type instance of a single signer

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

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

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

Consumes

JSON

Header

Accept: application

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **signerid** (string, path, required): The ID of the signer.
- **effective** (boolean, query, required): This parameter 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, query, optional): Locale for retrieving the language specific document types (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/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.

Responses

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

Download supplemental document content

This request downloads a single supplemental document content.

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}

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

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **signerid** (string, path, required): The ID of the signer.
- **doctypeinstanceid** (string, path, required): The ID of the document instance.
- **supplementaldocid** (string, path, required): The ID of the supplemental document.
- **dispositiontype** (string, query, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog box in the 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, query, 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.

Responses

Status 200 (OK): The supplemental document was queried successfully. In case of a successful query the response body contains binary data with document content providing with proper HTTP headers. 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.

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

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

Consumes

JSON

Header

Accept: application

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package to query.
- **disposition_type** (string, query, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog box 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

- **filename** (string, query, 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

Responses

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.

Download all documents as zip

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

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

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

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

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

- **filename** (string, query, 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.

Responses

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

Find text in a single document

This request finds text in a document Find text occurrences in a document. The text to find must be on a single page and not extend over 2 or more pages.

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

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

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **resolution** (float, query, optional): The resolution in dpi for conversion of field coordinates to document coordinates. Default: 72 dpi
- **searchtext** (string, query, optional): The text to find and locate.
- **natural_order** (boolean, query, optional): Defines the sort order, when multiple matches are found. The natural sort order is applied, if this parameter is set to true and the reversed sort order if it is set to false.

Responses

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

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.

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

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

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **pageno** (string, path, required): The page number.
- **format** (string, query, 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, query, 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, query, 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, query, 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, query, 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, query, optional): The disposition type of the downloaded image.
- **filename** (string, query, optional): The file name of the downloaded image.

Responses

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

Download a single document

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

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

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

Consumes

application/pdf, JSON

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **disposition_type** (string, query, optional): There are situations (when downloading a PDF document) where you might want a hyperlink leading to a file to present a SaveAs dialog box in the 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, query, 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.

Responses

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

Get available document types

This request returns document types available for the account.

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

URL

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

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

Consumes

JSON

Header

Accept: application

Method

GET

Example request

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

Parameters

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

Responses

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

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
  }
]
```

Add supplemental document

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

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

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

Consumes

JSON

Header

Accept: application

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
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.

- **signerid** (string, path, required): The ID of the signer.
- **instanceid** (string, path, required): The ID of the document type instance.
- **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.

Responses

Status 201 (Created): The supplemental document was successfully added. See RestSupplementalDocumentInfo. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

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

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

Content-Type: application

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
```

Request body

- RestDocumentTypeInstanceInput (required): Input 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
}
```

Parameters

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

Responses

Status 201 (Created): The document type instance was created. See RestID.

Add document to signing package

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

An input JSON 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.

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

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

Method

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package the user wants to add a reminder.
- **resolution** (float, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default: 72 dpi.
- **autoprep** (boolean, query, optional): Autoprepare feature places signature fields in the document for each of the signers that have been defined. Default value: false

Request body

- RestDocumentInput (required): Represents a RestDocument object in JSON. Default value: null

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
        }
  }
 ]
}
```

Responses

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

Create a document type

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

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

URL

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

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

Produces

JSON,

Header

Accept: application/json

Content-Type: application

Method

POST

Example request

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

Example headers

```
Accept: application/json
```

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

Parameters

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

Request body

- RestDocumentType (required): The document type data.

Example request body (JSON)

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

```
}
```

Responses

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

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}

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

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
```

Parameters

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

Request body

- RestDocumentTypeInstanceInput (required): JSON of `RestDocumentTypeInstanceInput`.

Example request body

```
{
```

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

Responses

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

Update a document

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

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}

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

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"
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **resolution** (float, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default: 72 dpi.

Request body

- RestDocumentInput (required): Represents a RestDocument object in JSON . Default value: null

Update a specified document type

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

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}

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

Produces

JSON

Header

Accept: application/json

Content-Type: application

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"
}
```

Parameters

- **accountid** (string, query, optional): The ID of the account. This parameter can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **doctypeid** (string, path, required): The ID of the document type.

Request body

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

Responses

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

Field requests

The following request are related to interactive fields such as signature fields, text fields, checkboxes, initials fields.

- Clear a signature
- Clear an initials field
- Delete a field
- Remove a signer from field
- Get a single text field
- Get a single signature field
- Get a single checkbox
- Get a single initials field
- Get all fields
- Add text field to document
- Add signature field to document
- Get the biometric data of a signature field
- Add initials field to document
- Add checkbox to document
- Sign a signature field
- Add initials
- Update a text field

- Update a signature field
- Update an initials field
- Assign a field to 'Any' signer
- Update a checkbox field

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

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

Parameters

- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field (can be the ID of the field or the PDF field name). See also `fieldid_is_pdf_fieldname`.
- **fieldid_is_pdf_fieldname** (boolean, query, 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.

Responses

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.

Clear an initials field

This request clears an initials field.

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

URL

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

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

Parameters

- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the initials field.

Responses

Status 200 (OK): The initials were successfully cleared from the initials field.

Delete a field

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

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}

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

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
```

Parameters

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

Responses

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.

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

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

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
```

Parameters

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

Responses

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

Get a single text field

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

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}

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

Consumes

JSON

Header

Accept: application/json

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
    }
  ]
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field.
- **resolution** (float, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi.
- **useIntId** (boolean, query, optional): The provided IDs for signing package, document and field are the internal IDs which were returned by the "Get audit trail" method.

Responses

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

Get a single signature field

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

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}

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

Consumes

JSON

Header

Accept: application/json

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"
  ]
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field.
- **resolution** (float, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi.
- **useIntId** (boolean, query, optional): The provided IDs for signing package, document and field are the internal IDs which were returned by the "Get audit trail" method.

Responses

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

Get a single checkbox

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

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}

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

Consumes

JSON

Header

Accept: application/json

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
    }
  ]
}
```


Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field.
- **resolution** (number, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default: 72 dpi.
- **useIntId** (boolean, query, optional): The provided IDs for signing package, document and field are the internal IDs which were returned by the "Get audit trail" method.

Responses

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

Get a single initials field

This request returns a single initials field specified by a given initial field ID.

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

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


```
{
  "id": "initials1",
  "name": "Initials field 1",
  "description": "",
  "signerId": "signer1",
  "alternateName": "Initials field 1",
  "required": true,
  "readOnly": false,
  "added": false,
  "widgets": [
    {
      "index": 0,
      "pageNumber": 1,
      "top": 824.7549304962158,
      "left": 105.80211639404297,
      "right": 150.80211639404297,
      "bottom": 802.2549304962158
    }
  ]
}
```


Parameters

- **packageid** (string, path, required): The ID of the signing package.

- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field.
- **resolution** (number, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi
- **useIntId** (boolean, query, optional): The provided IDs for package, document and field are the internal IDs which were returned by the "Get audit trail" method.

Responses

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

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.

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

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

Consumes

JSON

Header

Accept: application/json

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"
  }
]
```


Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **fieldFilter** (string, query, optional): Whether the returned list should be filtered to a specific field type or not. Valid values: SignatureField, CheckBox, TextField

Responses

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

Add text field to document

This request adds a new text field to an existing document. An input JSON 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.

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

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

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
    }
  ]
}
```

Parameters

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

Request body

- RestTextFieldInput (required): Represents a `RestTextFieldInput` object in JSON. Default value: null

Responses

Status 201 (Created): The text field was created. See RestID. 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 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 plug-in is registered to a signer and if the TSP plug-in supports signing the signature field using TSP, the 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.

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

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

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"
  ]
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **resolution** (float, query, optional): The resolution in dpi for conversion of field coordinates to document coordinates. Default value: 72 dpi
- **searchtext** (string, query, optional): The text to find for locating the new field.
- **offset_horizontal** (double, query, 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, query, 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, query, 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, query, optional): Defines the sort order, when multiple text matches are found. The natural sort order is applied, if this parameter is set to true and the reversed sort order if it is set to false.
- **desired_width** (double, query, 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, query, 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, query, 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, query, 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.

Request body

- RestSignatureFieldInput (required): Input JSON string of signature field.

Responses

Status 201 (Created): The signature field was created. See RestID. 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.

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

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

type: multipart/form-data

Method

POST

Example request

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

Parameters

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

Request body

- **privkey** (string, required): The private key to decrypt the biometric data.
- **privkeypassword** (string, optional): The password for the private key.

Responses

Status 201 (Created): The requested data could be retrieved. See RestBiometricDataOutput. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Add initials field to document

This request adds a new initials field to an existing document. An input JSON string specifying the details of the initials 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. Please note that the page, coordinates and dimensions of the new initials field have to be in the bounds of the document.

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

URL

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

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

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Example request body (JSON)

```
{
  "alternateName": "Initials field 2",
  "description": "This is my new initials fiels",
  "id": "initialsfield2",
  "name": "InitialsField2",
  "readOnly": false,
  "required": false,
  "signerId": "signer-1",
  "widgets": [
```

```
    {
      "bottom": 300,
      "index": 0,
      "left": 200,
      "pageNumber": 1,
      "right": 220,
      "top": 320
    }
  ]
}
```

Parameters

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

Request body

- RestInitialsFieldInput (required): Input JSON string off initials field.

Responses

Status 201 (Created): The initials field was added to the documet. See RestID. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Add checkbox to document

This request adds a new checkbox to an existing document. An input JSON 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.

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

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

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
    }
  ]
}
```

Parameters

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

Request body

- RestCheckBoxFieldInput (required): Input JSON string of checkbox field.

Responses

Status 201 (Created): The checkbox was added to the document. See RestID. 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

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

Parameters

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

Request body

- **sigdata_bin** (file, optional): The signature data as binary data (file blob). Either `sigdata` or `sigdata_bin` must be set. Encoding must be set to BINARY.
- **sigtype** (string, optional): The signature type. Valid values: SIGNWARE, SIGNATURE_B, IMAGE, C2S, STAMP, TSP, PH
- **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
- **tz** (string, optional): Time zone. Default: UTC
- **personal_certificate_pubkey** (string, optional): Personal certificate public key.
- **personal_certificate** (string, optional): Personal certificate in DER format.
- **signing_appearance** (string, optional): Signing appearance used

Responses

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

Add initials

This request adds initials to an initials field. The allowable input is valid BASE64 encoded string of the initials PNG image without the image prefix.

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

URL

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

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

Parameters

- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the initials field.

Request body

- **initialsdata** (string, required): The BASE64 encoded PNG image initials data.

Responses

Status 201 (Created): The initials were successfully added to the initials field.

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 string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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}

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

Consumes

JSON

Header

Accept: application/json

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
```

Parameters

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

Request body

- RestTextFieldInput (required): Input JSON string of user.

Responses

Status 200 (OK): The text field was successfully updated. See RestID. 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 string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled. A user can also update signing mode options, required and readOnly attributes of a signature field not assigned to a stage, of an already 'STARTED' signing package, provided the signature field is not signed and the assigned signer has not signed any fields assigned to it.

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}

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

Consumes

JSON

Header

Accept: application/json

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
```

Parameters

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

Request body

- RestSignatureFieldInput (required): Input JSON string of signature field.

Responses

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

Update an initials field

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

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

Method

PUT

Example request

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

Example header

```
Accept: application/json

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

Parameters

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

Request body

- RestInitialsFieldInput (required): Input JSON string of user.

Responses

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

Assign a field to 'Any' signer

This request assigns a field to 'Any' signer.

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

URL

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

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

Header

Accept: */*

Method

PUT

Example request

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

Example header

Accept: */*

Parameters

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

Responses

Status 200 (OK): The field was successfully assigned to 'Any' signer.

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 string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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}

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

Consumes

JSON

Header

Accept: application/json

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
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **documentid** (string, path, required): The ID of the document.
- **fieldid** (string, path, required): The ID of the field.
- **resolution** (float, query, required): The resolution in dpi for conversion of field coordinates to document coordinates. Default value: 72 dpi
- RestCheckboxFieldInput (required): Input JSON string of checkbox field.

Responses

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

Signing package requests

Related to signing packages, the following requests can be used.

- Delete a signing package
- Delete start date of a signing package
- Delete expiration date of a signing package
- Get a single signing package
- Get a list of packages
- Get audit trail
- Prepare signing session
- Send email to one signer
- Send email to all signers
- Schedule a signing package
- Create the final document
- Create a new signing package
- Create a new express signing package
- Update a signing package

Delete a signing package

This request deletes a signing package as well as all signers, documents and fields associated with the specified signing package.


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

URL

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

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

DELETE

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package the user wants to delete.

Responses

Status 200 (OK): The 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.

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

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

Produces

JSON,

Header

Accept: application/json

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/4711/startdate
```

Example header

```
Accept: application/json
```

Parameter

- **packageid** (string, path, required): The ID of the signing package whose start date should be deleted.

Responses

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.

Delete expiration date of a signing package

This request deletes the expiration date of a signing package.

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

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

Produces

JSON,

Header

Accept: application/json

Method

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/packages/4711/expirationdate
```

Example header

```
Accept: application/json
```

Parameter

- **packageid** (string, path, required): The ID of the signing package whose expiration date should be deleted.

Responses

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.

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.

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

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": 1,
  "mailSubject": "MailSubject - A request via Tungsten 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",
      "authenticationProviders": [],
      "url": "http://localhost:6611/cirrus/rest/v8/packages/ebab2946-0abb-46c0-9a18-
c15c7eed6060/signers/signer-1",
      "gdprConsentRequired": false,
      "esignConsentRequired": true,
      "tspSignatureDocuments": []
    }
  ]
}
```


Parameters

- **packageid** (string, path, required): The ID of the signing package to query.
- **useIntId** (boolean, query, optional): The provided `packageid` is the internal ID which was returned by "Get audit trail" method. Default value: false

Responses

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 information as `RestSigningPackageOutput` structure. See RestSigningPackageOutput.

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}).

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageTypeFilter** (string, query, optional): Whether to filter the list for the types signing package or template or not. Possible values: PACKAGE, TEMPLATE
- **startdate** (string, query, optional): Start date filter with yyyy-MM-dd format and user-specific time zone.

- **enddate** (string, query, optional): End date filter with yyyy-MM-dd format and user-specific time zone.
- **useddate** (string, query, optional): Target of the date filters. Possible values: CREATION, LASTUPDATE, COMPLETION, EXPIRATION, STARTDATE. Default value: LASTUPDATE
- **searchtext** (string, query, 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], query, optional): The states filter of the signing package. More values can be provided. Possible values: draft, started, complete, rejected, expired, canceled, archived
- **allteams** (boolean, query, 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], query, 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, query, optional): The page number to return (first=1).
- **limit** (integer, query, optional): The number of results to be returned in one page.

Responses

Status 200 (OK): The signing packages were queried successfully. See RestPackageListEntry.

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

Get audit trail

This request returns the audit trail for a signing package specified by a given signing package ID.

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

URL

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

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

Consumes

JSON

Header

Accept: application/json

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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.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@tungstenautomation.com.",
    "workflowEvent": "PKG_SIGNING_SESSION_CLOSED",
    "creationTime": "2019-05-13T06:45:17.61Z",
    "intPackageId": 14313,
    "intAccountId": 51
  }
]
```


Parameters

- **packageid** (string, path, required): The ID of the signing package to query the audit trail from.
- **locale** (string, query, optional): Locale for retrieving the locale-specific audit logs (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, en-US will be used
- **names** (string, query, optional): The flag indicates whether performed by / document names should be returned.

Responses

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.

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.

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

URL

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

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

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package the user wants to create a signing session for.
- **signtype** (string, path, required): The signing type of the signing session. Valid values: common, remote
- **userid** (string, query, 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).

Request body

- **commonSigningSessionInput** (RestCommonSigningSessionInput, required): Represents a `RestCommonSigningSessionInput` object in JSON. Default value: null

Example request body (JSON)

```
{
  "manualSignerAuthentications": [
    {
      "signerId": "signer-1",
      "passport": true,
      "visualVerification": true,
      "other": "Signer is personally known"
    }
  ],
  "qrCodeSpecifications": {
    "imageType": "JPG",
    "width": 200,
    "height": 200
  }
}
```

Responses

Status 200 (OK): The signing session was successfully prepared. In case of a successful query the response body contains information as in RestCommonSigningSessionOutput shema. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

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"}
```

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.

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

URL

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

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

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."
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package for which the email shall be sent.
- **signerid** (string, path, required): The ID of the signer for which the email shall be sent.
- **includelink** (boolean, query, optional): Include a link to the signing package. Default value: false
- **includedownloadlink** (boolean, query, optional): Include a link to download the final documents of the completed signing package. Default value: false

Request body

- RestEmailNotification (required): Input JSON string.

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 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 to call the signing client for signing the documents of the package.

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

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

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

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."
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package for which the email shall be sent.
- **includelink** (boolean, query, optional): Include a link to the signing package. Default value: false
- **includedownloadlink** (boolean, query, optional): Include a link to download the final documents of the completed signing package. Default value: false

Request body

- RestEmailNotification (required): Input JSON string.

Responses

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.

Schedule a signing package

This request evaluates the specified signing package and schedules it if it is 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.

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

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

Consumes

JSON

Header

Accept: application/json

Method

POST

Example request

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

Example header

```
Accept: application/json
```

Parameter

- **packageid** (string, path, required): The ID of the signing package.

Responses

Status 200 (OK): The signing package was successfully started.

Create the final document

This request initiates the creation of the final package container document if not already available for the specified package ID. The document is not created immediately and will be created using asynchronous jobs and should be available within seconds to minutes after initiation of the request.

If the container document for the specified package is already available or if the signing package is not already completed, then an error is thrown to the user.

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

URL

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

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

Produces

JSON

Method

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/packages/{packageid}/
createFinalDocument
```

Example header

```
Accept: application/json
```

Parameter

- **packageid** (string, path, required): The ID of the signing package.

Responses

Status 200 (OK): The final document creation was initiated.

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 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 the 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.

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

If `schedule=true` is set as query parameter, 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

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

POST

Example request

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

Example header

```
Accept: application/json
```

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 text fields 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

Request body

- RestSigningPackageInput (required): Represents a `RestSigningPackageInput` object in JSON. See Signing package properties overview.

Responses

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 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": "1",
  "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 completely 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 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

For a detailed overview of the signing package properties, see RestSigningPackageInput.

Create signing package scenarios

The following scenarios describe how a signing package can be created.

- Create signing package from scratch
- Create signing package from template
- Create signing package from template with metadata
- Create signing package from Word document

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 property of the signing package a user is allowed to change (see Signing package properties overview).

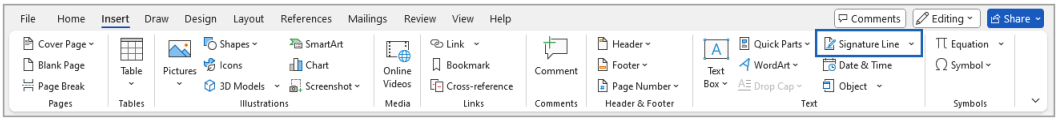
Also with create package on base of a template it is possible to schedule the package immediately (`schedule=true`) and to delete an already existing package with the same ID before package creation (`delete_existing=true`).

Create signing package from Word document

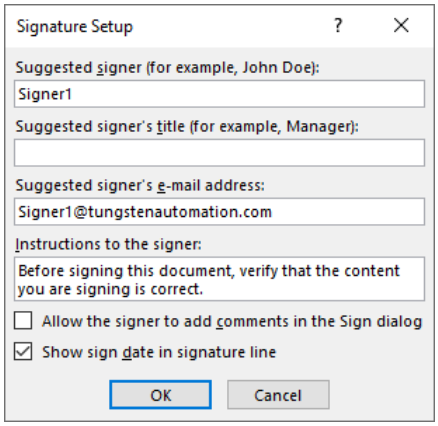
This scenario will use a Word document to create a signing package.

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

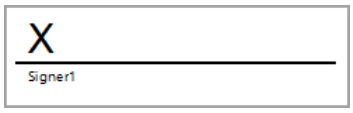
a signature field by coordinates alone can be quite a nuisance. The Signature Line Feature can be accessed in the Insert menu of Word on the right side of the submenu pane.



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@tungstenautomation.com).



After clicking OK 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 Create signing package from Word document.

MS Word specific configuration

SignDoc currently offers 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.

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	The ID of the signer.
Suggested signer	The name of the signer.
Suggested signer's title	Not used.
Suggested signer's email address	The email of the 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	The ID of the signer.
Suggested signer's title	Not used.
Suggested signer's email address	The email of the 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.

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.

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

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

Header

Accept: application/json

Method

POST

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **document** (string, 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): The 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://www.rfc-editor.org/info/bcp47). If omitted, en will be used. This setting will only be used if the signer name is not provided in the request.

Responses

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" }
```

Update a signing package

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

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

URL

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

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

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"
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package the user wants to update.
- **resolution** (number, query, optional): The resolution in dpi for conversion of document coordinates to required screen coordinates. Default value: 72 dpi

Request body

- RestSigningPackageInput (required): Represents a `RestSigningPackageInput` object in JSON.

Response status

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

Plug-in requests

The following request is used for plug-ins.

- Get a list of enabled plug-ins

Get a list of enabled plug-ins

This request returns a list (array) of plug-ins that are enabled for the specified account and implement the specified type.

A user with role SUPERUSER must specify the `accountid` parameter.

URL

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

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
```

Parameters

- **accountid** (string, query, optional): The account to use to list the enabled plug-ins. A user with role SUPERUSER must specify the account ID.
- **type** (string, query, required): The filter for plug-ins of the specified type. Possible value: TSP

Responses

Status 200 (OK): The plug-ins were queried successfully. See RestEnabledPluginInfo.

Reminder requests

For the signing package reminders the following requests are availabele.

- Delete a reminder
- Add reminder to signing package
- Update a reminder

Delete a reminder

This request deletes a reminder specified by ID.

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

URL

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

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

DELETE

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package containing the reminder the user wants to delete.
- **reminderid** (string, path, required): The ID of the reminder the user wants to delete.

Responses

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

Add reminder to signing package

This request adds a new reminder to an existing signing package. An input JSON string specifying the details of the reminder, has to be provided as a request parameter.

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

URL

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

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

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"
}
```

Parameters

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

Request body

- RestReminderInput (required): Represents a `RestReminderInput` object in JSON.

Responses

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 string. Each attribute of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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

URL

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

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

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"
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package containing the reminder the user wants to update.
- **reminderid** (string, path, required): The ID of the reminder the user wants to update.

Request body

- RestReminderInput (required): Represents a `RestReminderInput` object in JSON.

Responses

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

Signer requests

For the signers of the signing package, the following rwquests can be used.

- Delete a signer
- Delete signer email

Delete a signer

This request deletes a signer specified by ID.

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

URL

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

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

DELETE

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **signerid** (string, path, required): The ID of the signer the user wants to delete.

Responses

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.

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

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

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
```

Parameters

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

Responses

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.

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

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

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
```

Parameters

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

Responses

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.

Get a single signer

This request returns a signer specified by its ID and signing package ID.

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}

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Parameters

- **packageid** (string, path, required): The ID of the signing package.
- **signerid** (string, path, optional): The ID of the signer.
- **useIntId** (boolean, query, optional): The provided IDs for signing package and signer are the internal IDs which were returned by the "Get audit trail" method.

Responses

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

Process TSP result

This request processes the TSP result.

URL

http://*host_server:port_number*/cirrus/rest/v8/tsp/result/{result}/{tspkey}

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

Parameters

- **result** (string, path, required): The TSP result.
- **tspkey** (string, path, required): The TSP key.
- **did** (string, query, required): The ID of the document.
- **curl** (string, query, required): The client URL for redirection with the TSP result.

Responses

Status 200 (OK): The TSP result could be verified. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Get TSP info

This request provides information about the configured TSP (trusted service provider).

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

URL

http://*host_server:port_number*/cirrus/rest/v8/tsp/info

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

Parameter

- **locale** (string, query, optional): Locale for retrieving the locale-specific TSP information (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, en-US will be used.

Responses

Status 200 (OK): The TSP information is returned successfully. See RestTSPInfo. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

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.

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

URL

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

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

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

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. The parameters `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. The parameters `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.

Responses

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

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}

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/signers/johndoe%40kofax.com
```

The @ sign within the email address must be encoded with %40.

Example header

```
Accept: application/json
```

Parameter

- **email** (string, path, required): The email of the signer.

Responses

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

Get a list of signers

This request returns a list of signers which are found by the defined `SignerSearchEvent` plug-in. The SignDoc default implementation of the plug-in 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.

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

URL

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

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/signerlist
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **searchname** (string, query, optional): The signer name for searching. A signer is included only if his name starts with the provided text. This is case-insensitive. The parameters `searchname` and `searchemail` cannot be set at the same time.
- **searchemail** (string, query, optional): The signer email for searching. A signer is included only if his email address starts with the provided text. This is case-insensitive. The parameters `searchname` and `searchemail` cannot be set at the same time.
- **limit** (integer, query, 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, query, optional): Any custom character data which is passed through to the called plug-in.

Responses

Status code 200 (OK): The request was successful. The body content contains a list of RestSignerListEntry elements with the required fields.

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.

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

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
```

Parameters

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

Responses

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.

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 account ID, the package ID 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

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/signers/authentication
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **token** (string, query, required): The signing session token.
- **signtype** (string, query, required): The signing type. Valid values: REMOTE, COMMON or DOWNLOAD
- **signerid** (string, query, optional): The ID of the signer. It should be provided in case of Common Signing Session for document processing requests.
- **accesscode** (string, query, optional): The access code requested for 2-factor authentication of a remote signer.
- **locale** (string, query, optional): Locale for retrieving the locale-specific TSP information (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47). If omitted, en-US will be used.

Responses

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 2-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. Valid values: c, r (common, remote)

Add signer to signing package

This request adds a new signer to an existing signing package. An input JSON 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).

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

URL

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

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

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",
      "authenticationProviders" :[{"id":"MICROSOFT"}, {"id":"CODE"}]
      "authenticationParam": "qgzCVa",
      "accessCodeDeliveryPluginId": "NotificationSMSClickatell",
      "esignConsentRequired": true,
      "gdprConsentRequired": true,
      "preferredLanguage": "en",
      "accessCodeDeliveryParameters": [
        {
          "key": "NotificationSMSClickatell.phonenumber",
          "value": "+1-62293520272454"
        }
      ],
      "id": "signer-1"
    }
```

Parameters

- **packageid** (string, path, required): The ID of the signing package the user wants to add a signer.

Body parameters

- RestSignerInput (required): Represents a `RestSignerInput` object in JSON. Default value: null

Responses

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.

Update a signer

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

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

URL

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

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

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
}
```

Parameters

- **packageid** (string, path, required): The ID of the signing package containing the signer to be updated.
- **signerid** (string, path, required): The ID of the signer the user wants to update.

Request body

- RestSignerInput (required): Represents a `RestSignerInput` object in JSON.

Responses

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.

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

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

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"
 }
```

Parameters

- **packageid** (string, path, required): The ID of the signing package containing the signer who wants to perform the delegate operation.
- **signerid** (string, path, required): The ID of the signer the user wants to delegate the signing session.

Request body

- RestSignerInput (required): Represents a `RestSignerInput` object in JSON.

Responses

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.

Statistics requests

Use the following request for SignDoc statistics data.

Get statistics data

Get statistics data

This request returns statistics data pre-defined for the whole system based on the user, account and the time frame.

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

URL

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

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

Produces

JSON

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/statistics?accountid=signdoc
```

Parameters

- **accountid** (string, query, optional): Statistics for the specific account. If no account ID is specified for role SUPERUSER, the statistics data for all the accounts will be returned.
- **userid** (string, query, optional): Statistics for the specific user. If the user ID is specified for role SUPERUSER, then it is a must to provide the account ID. If no user ID is provided, for role ADMIN or SUPERUSER the statistics data for all users belonging to the account will be returned. For users with role USER, statistics related only to the logged in user is returned.
- **startdate** (long, query, optional): Start date in milliseconds for the statistics data. If not provided, the start date is the start of the 30th day from today.
- **enddate** (long, query, optional): End date for in milliseconds for the statistics data. If not provided, current time is used.

Responses

Status 200 (OK): Statistics data was returned successfully. See RestStatisticsList. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

System requests

For SignDoc system settings and information, the following requests are used.

- Get the global license information

Get the global license information

This request retrieves information about an installed global SignDoc license.

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

URL

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

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/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"
}
```

Responses

Status 200 (OK): The global license could be retrieved successfully. See RestAccountLicenseOutput. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Get license information from TotalAgility Licensing server

This request retrieves license information from TotalAgility Licensing server for a system administrator. To see the details in this request the use of TotalAgility Licensing server should be enabled for the system administrator. This information may or may not be shared across all accounts, based on, if the configuration for a particular account is different from the system.

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

URL

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

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/totalAgilityLicense
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)

```
{
  "totalAgilityLicenseInfo": [
    {
      "id": 224,
      "licenseName": "SignDoc Signing package",
      "availableVolume": 0,
```

```
      "periodStartDate": "2024-01-01T08:00:00Z",
      "state": "VALID"
    },
    {
      "id": 13033,
      "licenseName": "SignDoc Base",
      "state": "VALID"
    },
    {
      "id": 13035,
      "licenseName": "SignDoc Unlimited Users",
      "availableVolume": -1,
      "state": "VALID"
    }
  ]
}
```


Responses

Status 200 (OK): The license information was retrieved. See RestGlobalTotalAgilityLicenseOutput. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

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

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

Produces

JSON

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Responses

Status is 200 (OK): The versions were queried successfully. See RestModulesVersion.

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

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
```

Parameter

- **locale** (string, optional): The ETF BCP 47 language tag. If the value is unknown or invalid, English will be used.

Responses

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

Get the version of the SignDoc modules

This request returns the modules versions of the SignDoc distribution. For this request authentication is required.

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

URL

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

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

Produces

JSON

Header

Accept: application/json

Method

GET

Example request

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

Example header

```
Accept: application/json
```

Responses

Status 200 (OK): The version of the modules was queried successfully. See RestModulesVersion.

Example response body (JSON)

```
{
"signdoc_standard": "version-x",
"signdoc_sdk": "version-y",
"rest_api": "v1"
}
```

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://www.rfc-editor.org/info/bcp47.

The display name language can be defined by providing another language tag (IETF BCP 47 language tag) as query parameter `locale`.

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

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

Produces

JSON

Header

Accept: application/json

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"
}
```

Parameters

- **language_tag** (string, path, required): The language tag which describes the locale for which the name should be displayed in IETF BCP 47 language tag format, see https://www.rfc-editor.org/info/bcp47.
- **locale** (string, query, optional): The language tag which describes the locale of the display language in IETF BCP 47 language tag format, see https://www.rfc-editor.org/info/bcp47.

Responses

Status 200 (OK): The display name could be retrieved successfully. See RestModulesVersion. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Import a global license

This request imports a global SignDoc license.

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

URL

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

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

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"

}
```

Request body

- RestAccountLicenseInput (required): Input JSON string of license.

Responses

Status 200 (OK): The global license could be imported successfully. See RestAccountLicenseOutput. Otherwise a SignDoc Standard status code is returned together with the explaining messages.

Team requests

The following requests are related to SignDoc teams.

- Delete team from account
- Remove user from team
- Get a single team
- Add user to team
- Add team to account
- Update a team
- Reinvite a member to a team

Delete team from account

This request deletes an existing team from the account.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}

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

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **teamid** (string, path, required): The ID of the team the user wants to delete.

Responses

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.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}/users/{userid}

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

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/teams/team1/users/user1
```

Example header

```
Accept: application/json
```

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

Parameters

- **teamid** (string, path, required): The ID of the team.
- **userid** (string, path, required): The ID of the user who should be removed from the team.
- **manager** (boolean, query, 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

Responses

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

Get a single team

This request returns a single team of an account specified by a team ID.

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}

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json
```

Parameter

- **teamid** (string, path, required): The ID of the team.

Responses

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

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 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.

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}

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/teams/team1/users/user1
```

Example header

```
Accept: application/json
```

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

Parameters

- **teamid** (string, path, required): The ID of the team.
- **userid** (string, path, required): The ID of the user who should be added.
- **as_team_manager** (boolean, query, optional): Whether the added user should be team manager or not. Default value: false

Responses

Status 200 (OK): The user was successfully added to the team. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Add team to account

This request adds a team to an account. An input JSON 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.

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

URL

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

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

POST

Example request

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

Example header

```
Accept: application/json

Content-Type: application/json
```

Request body

- RestTeamInput (required): Input JSON string team. Default value: null

Example request body (JSON)

```
"id": "team1",
"name": "Team 1",
"managers": [
    "jdo", "ble"
],
"members": [
    "tla", "jcu"
]
```

Responses

Status 201 (Created): The team was created. See RestID. 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 string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}

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

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/teams/team1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **teamid** (string, path, required): The ID of the team the user wants to update.

Request body

- RestTeamInput (required): Represents a RestTeam object in JSON.

Example request body (JSON)

```
{
  "name": " Team "
}
```

Responses

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

Reinvite a member to a team

This request reinvites an inactive member to a team.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/teams/{teamid}/users/{userid}/reinvitation

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

PUT

Example request

```
PUT http://localhost:6611/cirrus/rest/v8/teams/team1/users/abc/reinvitation
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **teamid** (string, path, required): The ID of the team.
- **userid** (string, path, required): The ID of the user.

Responses

Status 200 (OK): The user was successfully reinvited to the team.

TSP signing requests

The following requests are related to TSP signing.

- Poll the staus of a TSP signing request initiated by the signer

Poll the status of a TSP signing request initiated by a signer

This request retrieves the status of a TSP signing request initiated by a signer.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/tsp/signing/request/{requestid}

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/tsp/signing/request/{requestid}
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **requestid** (string, path, required): The ID of the TSP signing request.

Responses

Status 200 (OK): The status of the TSP signing request was retrieved successfully.

Example response body (JSON)

```
[
```

```
  {
    "k": "string",
    "v": "string"
  }
]
```

User requests

The following requests are related to SignDoc users.

- Delete a user
- Delete a server administrator
- Remove role from user
- Get a single user
- Get a single server administrator
- Get current server administrator
- Get current user
- Get all users
- Get all server administrators
- Refresh an authentication token
- Get number of users
- Add role to user
- Reset password of a user
- Create 'password recovery' notification for a user
- Invite or reinvite a user
- Set user API key
- Reset password of a server administrator
- Invite or reinvite a server administrator
- Create server administrator
- Create an authentication token
- Add user to account
- Set a user password
- Set API key for a specified user
- Update a user
- Update a server administrator

Delete a user

This request deletes the existing user specified by account ID and user ID.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}

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

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/user1
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, query, 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, query, optional): If this parameter is set to true, then the user will be deleted although he is owner of any signing packages. In this case all assigned signing packages are also deleted.

  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.

- **userid** (string, path, required): The ID of the user who should be deleted.

Responses

Status 200 (OK): The user was successfully deleted. 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.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}

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

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/serveradmins/ksdadmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

- **userid** (string, path, required): The ID of the server administrator who should be deleted.

Responses

Status 200 (OK): The server administrator 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.

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

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

DELETE

Example request

```
DELETE http://localhost:6611/cirrus/rest/v8/users/user1/role
```

Example header

```
Accept: application/json
```

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

Parameters

- **userid** (string, path, required): The ID of the user.
- **accountid** (string, query, optional): The ID of the account. It can be omitted if the user is bound to an account. Users with the role SUPERUSER have to provide it.
- **role** (string, query, required): The name of the role. Valid values: ADMIN, TEAMMGR

Responses

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

Get a single user

This request returns a single user specified by a given user and account ID.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/user1
```

Example header

```
Accept: application/json
```

Parameters

- **accountid** (string, query, 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.
- **userid** (string, path, required): The ID of the user to query.
- **useIntId** (boolean, query, optional): The provided `userid` is the internal ID which was returned by "Get audit trail" method.

Responses

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

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"
}
```

Get a single server administrator

This request returns a single server administrator specified by a given ID.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}

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

Consumes

JSON

Header

Accept: application/json

Method

GET

Example request

```
GET http://localhost:6611/cirrus/rest/v8/users/serveradmins/ksdadmin
```

Example header

```
Accept: application/json
```

Parameters

- **userid** (string, path, required): The ID of the user.
- **useIntId** (boolean, query, optional): The provided `userid` is the internal ID which was returned by the "Get audit trail" method.

Responses

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

Example response body (JSON)

```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@tungstenautomation.com",
  "state": "ACTIVE",
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1448268442365,
  "url": "/rest/v8/users/serveradmins/ksdadmin"
  }
```

Get current server administrator

This request returns the current server administrator.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmin

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/users/serveradmin
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Example response body (JSON)


```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@tungstenautomation.com",
  "state": "ACTIVE",
  "settings": {
    "defaultTemplateName": "Template created by Tenants Administrator at 11/17/2021
 11:17 AM",
    "defaultPackageName": "Package created by Tenants Administrator at 11/17/2021 11:17
 AM",
    "defaultMailSubject": "An e-signing request via Tungsten SignDoc",
    "defaultMailMessage": "Tenants Administrator has invited you to sign documents with
 Tungsten SignDoc"
  },
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1444743172375,
  "url": "/rest/v8/users/serveradmins/ksdadmin"
  }
```


Responses

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

Get current user

This request returns the current user specified by the current authentication.

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

URL

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

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/user
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Responses

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

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}).

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

URL

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

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/users
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameter

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

Responses

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

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"
  }
]
```


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}).

For this request a valid session and SUPERUSER role is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins

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/users/serveradmins
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Responses

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

Example response body (JSON)

```
[
  {
    "id": "ksdadmin",
    "name": "Tenants Administrator",
    "email": "ksdadmin@tungstenautomation.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"
    ]
  }
]
```

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

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/users/refreshtoken
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Responses

Status 200 (OK): The authentication token was successfully refreshed. Otherwise, a SignDoc Standard status code is returned together with the explaining messages. See RestMsgList shema.

Get number of users

This request returns the number of users inside an account.

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

URL

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

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/users/count
```

Example header

```
Accept: application/json
```

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

Parameter

- **accountid** (string, query, 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.

Responses

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

Example response body (JSON)

```
{
  "count": 86
}
```

Add role to user

This request adds a role to a user specified by ID. Valid roles are ADMIN or TEAMMGR.

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

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/role
```

Example header

```
Accept: application/json
```

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

Parameters

- **userid** (string, path, required): The ID of the user.
- **accountid** (string, query, 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.
- **role** (string, query, required): Name of the new role. Valid values: ADMIN, TEAMMGR

Responses

Status 200 (OK): The role was successfully added. 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. Besides that anybody provides the `accountid` as a query parameter.

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

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

Consumes

JSON

Header

Accept: application/json

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
```

Parameters

- **userid** (string, path, required): The ID or email of the user.
- **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.

Responses

Status 200 (OK): The password was successfully reset. 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.

For this request no authentication is necessary.

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/pwrecovery

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/pwrecovery
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **accountid** (string, query, optional): This parameter can be omitted if the user is bound to an account.
- **usedefaultaccount** (string, query, optional): Server uses sole account ID implicitly, applicable if the SignDoc Standard system has only one account configured. Default value: false

- **userid** (string, path, required): Either the user ID or email address of the user.

Responses

Status 200 (OK): The recovery notification was sent successfully, or the notification was not sent because the ID or email address of the user is not correct. 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.

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

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/user1/invitation
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **userid** (string, path, required): The ID of the user.
- **accountid** (string, query, optional): The ID of the account. It is only necessary if a user with SUPERUSER role wants to invite or reinvite an account administrator.

Responses

Status 200 (OK): The user was successfully invited or reinvited. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

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.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/user/apikey

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/apikey
```

Example header

```
Accept: application/json

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

Parameters

- **uaerid** (string, path, required): The ID of the user.
- **password** (string, required): The password of the current user.
- **apikey** (string, required): The new API key for the user.

Responses

Status 200 (OK): The API key was successfully updated. 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).

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}/pwreset

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/serveradmins/kssdadmin/pwreset
```

Example header

```
Accept: application/json
```

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

Parameter

- **userid** (string, path, required): The ID or email address of the user.

Responses

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 server administrator

This request sends an invitation or a reinvitation to a server administrator specified by ID.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmins/{userid}/invitation

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/serveradmins/admin1/invitation
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Prameter

- **userid** (string, path, required): The ID of the user.

Responses

Status 200 (OK): The server administrator was successfully invited or reinvited. Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Create server administrator

To create a server administrator, an input JSON 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.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/serveradmin

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

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"]
}
```

Request body

- RestUserInput (required): Input JSON string of user.

Responses

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

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

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/authentication
```

Example header

```
Accept: application/json

Content-Type: application/json
```

Parameters

- **credentials** (string, query, required): Either the userid or email address of the user.
- **accountid** (string, query, optional) The ID of the account. Account users have to provide it. Users with the role SUPERUSER must leave this field empty.
- **password** (string, query, required): The password of the user.

- **authid** (string, query, optional): The authentication ID, needed for autologoff feature.
- **newpassword** (string, query, optional): This field can be set to specify a new password for the user.
- **usedefaultaccount** (boolean, optional): Server uses sole account ID implicitly, applicable if SignDoc Standard system has only one account configured. Default value: false

Responses

Status 200 (OK): The authentication token was successfully created. See RestAuthentication Otherwise, a SignDoc Standard status code is returned together with the explaining messages.

Add user to account

To add a user to an existing account specified by ID, an input JSON 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.

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

URL

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

 *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

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@tungstenautomation.com",
  "id": "use1",
  "name": "use1",
  "password": "2beUserU!",
  "roles": [
    "USER",
    "TEAMMGR",
    "ADMIN"
  ],
  "state": "ACTIVE"
}
```

Parameters

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

Request body

- RestUserInput (required): Represents a `RestUserInput` object in JSON.

Responses

Status 201 (Created): The user was successfully added to the account. 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 box

The invitation link (in the email) calls the Manage Client with the query parameter aptoken (access token for server administrators) 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.

Parameters

- **token** (string, optional): The token which identifies the user.
- **password** (string, optional): 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.

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

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

Produces

JSON

Header

Accept: application/json

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
```

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 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.

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

URL

http://*host_server:port_number*/cirrus/rest/v8/users/{userid}/apikey

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

POST

Example request

```
POST http://localhost:6611/cirrus/rest/v8/users/apikey
```

Example header

```
Accept: application/json

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

Parameters

- **password** string, required): The password of the user.
- **apikey** (string, required): The new API key for the user.

Responses

Status 200 (OK): The API key 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 string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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}

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

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"
    ]
}
```

Parameters

- **accountid** (string, query, optional): The ID of the account. 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.
- **userid** (string, path, required): The ID of the user to update.

Request body

- RestUserInput (required): Represents a `RestUserInput` object in JSON.

Responses

Status 200 (OK): The user was successfully updated. 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 string. Selected attributes of the existing user can be changed by adding the corresponding attribute name and the new value to the input JSON string. Each attribute included that way will be changed provided all requirements of the new attribute values are fulfilled.

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}

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

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"
}
```

Parameter

- **userid** (string, path, required): The ID of the user to update.

Request body

- RestUserInput (required): Input JSON string of user.

Responses

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

Schemas

DC3ServerObject

- **data** (string, optional)
- **sessionKey** (string, optional)

RestAccessCodeDeliveryChannelParameter

- **key** (string, optional): The parameter key.
- **value** (string, optional): The parameter value.

RestAccountInput

- **id** (string, optional): The ID of the account.
- **name** (string, optional): The name of the account.
- **company** (string, optional): The company of the account.
- **state** (string, optional): The state of the account. Valid values: INACTIVE, ACTIVE
- **notificationsEnabled** (boolean, optional): Whether notifications are sent or not.
- **dnsLabel** (string, optional): The DNS label of the account.
- **contactInformation** (string, optional): The contact information of the account.
- **license** (string, optional): The current license for the account.
- **signingCertificatePassword** (string, optional): The password of the signing certificate.
- **publicKey** (string, optional): The public key of the account.
- **signingCertificate** (string, optional): The signing certificate of the account.
- **pemCertificate** (string, optional): The certificate (PEM format, unencrypted) for the signing certificate.
- **pemCertificateKey** (string, optional): The certificate key (PEM format) for the signing certificate.
- **pemCertificateChain** (string, optional): The certificate chain (PEM format, unencrypted) for the signing certificate.
- **rootSigningCertificate** (string, optional): The public root (CA) certificate for signing, if required (PEM format, unencrypted).
- **users** (Array[RestUserInput], optional): A list of all users of the account.
- **teams** (Array[RestTeamInput], optional): A list of all teams of the account.

RestAccountLicenseInput

- **accountId** (string, optional): The ID of the account. This parameter can be omitted if the user is bound to an account.
- **license** (string, required): The account license as Base64-encoded string.

RestAccountLicenseOutput

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

RestAccountListEntry

- **id** (string, optional): The ID of the account.
- **name** (string, optional): The name of the account.
- **company** (string, optional): The company of the account.
- **state** (string, optional): The state of the account. Valid values: INACTIVE, ACTIVE
- **licenseInfo** (Array[RestAccountLicenseOutput], optional): The license information of the account.
- **totalAgilityLicenseInfo** (Array[RestAccountTotalAgilityLicenseOutput], optional): The TotalAgility Licensing server license information.
- **licenseType** (string, optional): Type of license used and is currently active. Valid values: ACCOUNT, GLOBAL, TOTALAGILITY, NONE
- **url** (string, optional): The URL of the account.

RestAccountOutput

- **id** (string, optional): The ID of the account.
- **name** (string, optional): The name of the account.
- **company** (string, optional): The company of the account.
- **state** (string, optional): The state of the account. Valid values: INACTIVE, ACTIVE
- **notificationsEnabled** (boolean, optional): Whether notifications are sent or not
- **dnsLabel** (string, optional): The DNS label 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)
- **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** (Array[RestAccountLicenseOutput], optional): The current account license information.
- **totalAgilityLicenseInfo** ([Array[RestAccountTotalAgilityLicenseOutput], optional): The TotalAgility licensing server license information.
- **licenseType** (string, optional): Type of license used and is currently active. Valid values: ACCOUNT, GLOBAL, TOTALAGILITY, NONE
- **url** (string, optional): The URL to query the account.
- **publicKeyInfo** (Array[RestPublicKeyInfo], optional): The public key information for the account.
- **signingCertificateInfo** (Array[RestSigningCertificateInfo], optional): The signing certificate information for the account.
- **users** (Array[RestUserListEntry], optional): A list of all users of the account.
- **teams** (Array[RestTeamListEntry], optional): A list of all teams of the account.

RestAccountTotalAgilityLicenseOutput

- **id** (string, optional): License ID.
- **licenseName** (string, optional): Name of the license.

- **expiryDate** (string, optional): Expiry date of the license. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **availableVolume** (int, optional): The total number of available volume units.
- **periodStartDate** (string, optional): License active start date. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)
- **state** (string, optional): Current TotalAgility license state. Valid values : INACTIVE, EXPIRED, VALID
- **totalAgilityLicensingServerError** (Array[RestTotalAgilityLicensingServerError], optional): The TotalAgility Licensing server error information including server connection errors.

RestAddSignatureResult

- **resultCode** (string, optional): The result code. Valid values: SUCCESS, SIGNATURE_TOO_SIMPLE_OR_NOT_USABLE, PERSONAL_CERTIFICATE
- **fieldsToUpdate** (Array[RestFieldInfo], optional): The fields to be updated.
- **personalCertificateMessage** (Array[DC3ServerObject], optional): The personal certificate message.

RestAuditTrailOutput

- **message** (string, optional): The message of the audit trail.
- **workflowEvent** (string, optional): The workflow event of the audit trail. Valid 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

- **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)
- **intUserId** (integer, optional): The internal ID of the user.
- **intSignerId** (integer, optional): The internal ID of the signer.
- **intDocumentId** (integer, optional): The internal ID of the document.
- **documentName** (string, optional): The name of the document.
- **intDocumentFieldId** (integer, optional): The internal ID of the document field.
- **intPackageId** (integer, optional): The internal ID of the package.
- **intAccountId** (integer, optional): The internal ID of the account.
- **performedBy** (string, optional): The person who performed this action.

RestAuthentication

- **exp** (integer, optional): The expiration date of the authentication token in number of milliseconds from the epoch
- **iat** (integer, optional): The 'issued At' date of the authentication token in number of milliseconds from the epoch
- **t** (string, optional): The 'type' of the authentication token. 0 mean normal token, 1 means one-time token
- **accountID** (string, optional)
- **accountName** (string, optional)
- **userId** (string, optional)
- **userName** (string, optional)
- **geteMail** (string, optional)
- **roles** (Array[string], optional)
- **globalLicense** (boolean, optional)
- **apikey** (string, optional)

RestAuthenticationProviderInput

- **id**(string, optional): Identifier of the authentication provider. Possible values: EXTAUTH,GOOGLE,MICROSOFT,CODE

RestAuthenticationProviderOutput

- **id**(string, optional): Identifier of the authentication provider. Possible values: EXTAUTH,GOOGLE,MICROSOFT,CODE
- **name** (string, optional): The name of the authentication service
- **url** (string, optional): The authentication URL
- **channel** (string, optional): The channel type by which authentication parameters will be sent to the signer

RestBiometricDataOutput

- **dataBase64** (string, optional): The binary data encode in Base64.

RestCheckboxFieldInput

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

RestCheckboxFieldOutput

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

RestCommonSigningSessionInput

- **qrCodeSpecifications** (RestQRCodeSpecifications, optional): The 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.
- **manualSignerAuthentications** (Array[RestManualSignerAuthentication]): A list of manual signer authentications.

RestCommonSigningSessionOutput

- **url** (string, optional): The URL of the common signing session.

- **qrcode** (string, optional): The Base64-encoded QR code of the common signing session.

RestConfigEntry

- **k** (string, required): The key of the configuration entry.
- **v** (string, required): The value of the configuration entry.
- **l** (string, optional): The locale string.
- **b** (Array[string], optional): The byte array in string representation

RestConfigurationDescription

- **id** (string, optional): The key of the configuration setting.
- **sortId** (string, optional): The optional sort key of the configuration setting.
- **title** (string, optional): The title of the configuration setting.
- **description** (string, optional): The description of the configuration setting.
- **type** (string, optional): The type of the configuration setting. Valid values: STRING, INTEGER, DATE, PASSWORD, BINARY, BOOLEAN, CATEGORY, LOCSTRING, LOCBINARY, ENCRYPTED.
- **subType** (string, optional): The subtype of the configuration setting. Valid values: MULTILINE, MULTILINE_HTML, MULTILINE_XML, PKCS12.
- **defaultValue** (string, optional): The default value of the configuration setting.
- **provider** (string, optional): The configuration provider of the configuration setting. Valid values: CIRRUS, PLUGIN, CONSTRAINT.
- **accountSpecific** (boolean, optional): Specifies whether the configuration setting can have account-specific values.
- **show** (boolean, optional): Specifies whether this setting should be shown in the configuration editor.
- **editRoles** (integer, optional): Defines the roles which are allowed to edit the setting.
- **viewRoles** (integer, optional): Defines the roles which are allowed to view the setting.
- **rangeMin** (integer, optional): Either the lowest allowed number or the minimum number of characters if the value is a string.
- **rangeMax** (integer, optional): Either the highest allowed number or the maximum number of characters if the value is a string.
- **regExp** (string, optional): The regular expression pattern for the value validation.
- **mediaType** (string, optional): The media type of the configuration option.
- **imagePreview** (boolean, optional): Specifies whether a binary configuration setting supports an image preview.
- **export** (boolean, optional): Specifies whether this setting should be part of the configuration export.

RestCount

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

RestDocumentFieldListEntry

- **id** (string, optional): The ID of the field.
- **label** (string, optional): The label of the field.
- **type** (string, optional): The type of the field. Valid values: SignatureField, CheckBox, TextField
- **signerID** (string, optional): The signer ID of the field.
- **url** (string, optional): The request URL for the entire field.
- **stage** (integer, optional): The stage of the signer allowed to sign the field.

RestDocumentInput

- **id** (string, optional): The ID of the document.
- **name** (string, required): The name of the document.
- **description** (string, optional): The description of the document.
- **format** (string, optional): The document format. Valid values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **content** (string, required): The actual document in Base64 format.
- **order** (integer, optional): The order of the document.
- **documentMessage** (string, optional): The message displayed when the recipient opens the document.
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a Base64-encoded string.
- **fileName** (string, optional): The file name of the document.
- **custom** (string): The custom field.
- **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.
- **checkboxFields** (Array[RestCheckboxFieldInput], optional): A list of all check boxes contained in the document.
- **initialsFields** (Array[RestInitialsFieldInput], optional): A list of all initials fields in the document.

RestDocumentListEntry

- **id** (string, optional): The ID of the document.
- **name** (string, optional): The name of the document.
- **fileName** (string, optional): The file name of the document.
- **description** (string, optional): The description of the document.
- **documentMessage** (string, optional): The document message.
- **url** (string, optional): The request URL for the entire document.
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a Base64-encoded string.
- **order** (integer, optional): The order of the document.

- **permittedDocumentActions** (Array[String, optional): The permitted actions for the document. Possible values: CLEAR_SIGNATURE, SIGN, FILL_FORMS, REGULAR_FORM_FIELDS, VIRTUAL_FORM_FIELDS, REGULAR_SIGNATURE_FIELDS, VIRTUAL_SIGNATURE_FIELDS, INITIALS_FIELDS
- **RejectedDocument** (RestRejectedPackageDocument): Rejected document details
- **IsRejected** (boolean): True if the document is rejected by a package owner or a signer

RestDocumentOutput

- **id** (string, optional): The ID of the document.
- **name** (string, required): The name of the document.
- **description** (string, optional): The description of the document.
- **format** (string, optional): The document format. Valid values: PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG. Default value: PDF
- **content** (string, required): The actual document in Base64 format.
- **order** (integer, optional): The order of the document.
- **documentMessage** (string, optional): The message displayed when the recipient opens the document.
- **thumbnail** (string, optional): The thumbnail (PNG format) of the first document page as a Base64-encoded string.
- **fileName** (string, optional): The file name of the document.
- **custom** (string): The custom field.
- **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.
- **checkboxFields** (Array[RestCheckboxFieldOutput], optional): A list of all check boxes contained in the document.
- **initialsFields** (Array[RestInitialsFieldOutput], optional): A list of all initials fields contained in the document.
- **permittedDocumentActions** (Array[String, optional): The permitted actions for the document . Possible values: CLEAR_SIGNATURE, SIGN, FILL_FORMS, REGULAR_FORM_FIELDS, VIRTUAL_FORM_FIELDS, REGULAR_SIGNATURE_FIELDS, VIRTUAL_SIGNATURE_FIELDS, INITIALS_FIELDS
- **RejectedDocument** (RestRejectedPackageDocument): Rejected document details
- **IsRejected** (boolean): True if the document is rejected by a package owner or a signer

RestDocumentType

- **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://www.rfc-editor.org/info/bcp47. Example: en for English or pt-BR for Brazilian Portuguese. If omitted, 'en' will be used.

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

RestDocumentTypeInstanceInput

- **id** (string, optional): The ID of the instance, must be unique for a signer. If omitted a random ID is created at the server.
- **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.
- **name** (string, optional): The name of the instance. Only allowed for document type GENERIC.
- **description** (string, optional): The description 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
- **maxFilesNumber** (integer, optional): The maximum number of files permitted for this document type. Only allowed for document type GENERIC.

RestDocumentTypeInstanceOutput

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

RestEmailNotification

- **subject** (string, required): The subject of the email to be sent.
- **message** (string, required): The message (body) of the email to be sent.

RestEnabledPluginInfo

- **id** (string, required): The ID of the plug-in.
- **name** (string, required): The name of the plug-in.

RestEntityStatus

- **id** (string, required): Defines uniquely the entity. Valid values: SIGNING_CERTIFICATE, SMTP_CONNECTION, BIOMETRIC_KEY, SMIME_CERTIFICATE, PLUGIN, WEBHOOK_CONNECTION
- **statusClass** (string, required): Status classification. INFO: [blue] Just for information. OK: [green] Everything is configured and running correctly. WARN: [yellow] The user must be aware of upcoming issues and possibly take action soon. PROBLEM: [red] The user must fix a problem.
- **statusCode** (string, required): The status code of the information. Valid values: WEBHOOK_ERROR, SMTP_ERROR, EXPIRED, NOT_YET_VALID, EXPIRES_SOON, NOT_SET, IS_VALID, INVALID, GENERAL_ERROR, MESSAGE, NO_DATA
- **statusIdString** (string, required): The localized description of the entity name.
- **statusClassString** (string, required): The localized description of the status class value.
- **statusCodeDescription** (array[string], required): The localized description of the status code value. Consists usually of multiple lines.
- **timestamp** (string, required): The timestamp in ISO format.

RestEntry

- **k** (string, required): The key of the entry.
- **v** (string, required): The value of the entry.

RestFieldInfo

- **signingPackageId** (string, required): The ID of the signing package.
- **documentId** (string, required): The ID of the document.
- **fieldId** (string, required): The ID of the field.

RestFindTextResult

- **resolution** (number, optional): The resolution in dpi of the coordinates.
- **page** (integer): The page of the result.
- **left** (number, optional): The left coordinate.
- **bottom** (number, optional): The bottom coordinate.
- **right** (number, optional): This attribute contains the rough (calculated) horizontal coordinate value from the right margin of the found text.

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.

RestGlobalTotalAgilityLicenseOutput

- **totalAgilityLicenseInfo** (Array[RestAccountTotalAgilityLicenseOutput], optional): SignDoc license information in the TotalAgility Licensing server.

- **state** (string, optional): Defines if the use of TotalAgility licensing server is enabled by the system administrator.

RestID

- **id** (string, required): The ID of the created resource.
- **url** (string, required): The URL to access the created resource.
- **commonSigningUrl** (string, optional): The URL to access the common signing session of the created resource.
- **messages** (RestMsgList, optional): Any informational messages.

RestInitialsFieldInput

- **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!).
- **description** (string, optional): The description of the field.
- **signerId** (string, optional): The ID of the signer assigned to the field.
- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc).
- **required** (boolean, optional): Whether the field is required or not.
- **readOnly** (boolean, optional): Whether the field is read only or not.
- **stage** (integer, optional): The stage of the signer allowed to sign the field.
- **added** (boolean, optional): The state of the initials field, true is added, false is not added (default).
- **widgets** (Array[RestWidget], required): A list of all widgets of the field.

RestInitialsFieldOutput

- **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!)
- **description** (string, optional): The description of the field.
- **signerId** (string, optional): The ID of the signer assigned to the field.
- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc).
- **required** (boolean, optional): Whether the field is required or not.
- **readOnly** (boolean, optional): Whether the field is read only or not.
- **stage** (integer, optional): The stage of the signer allowed to sign the field.
- **added** (boolean, optional): The state of the initials field, true is added, false is not added (default).
- **widgets** (Array[RestWidget], required): A list of all widgets of the field.

RestLocaleInfo

- **languageTag** (string, optional): The language tag (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47).

- **displayName** (string, optional): The name for the locale that is appropriate for display to the user.

RestManualSignerAuthentication

- **signerId** (string, required): The 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.

RestModulesVersion

- **signdoc_standard** (string, optional)
- **signdoc_sdk** (string, optional)
- **rest_api** (string, optional)

RestMsg

- **code** (integer, required): The code of the message.
- **type** (string, required): The type of the message. Valid values: ERROR, WARNING, INFO
- **message** (string, required): The description of the message.

RestMsgList

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

RestNotificationTargetParameterDescription

- **name** (string, optional): The name of the parameter.
- **description** (string, optional): The description of the parameter. This description will be displayed by the GUI when the user enters a notification target.
- **validationRegExp** (string, optional): The Java regular expression used by the GUI to validate the parameter value entry.
- **helpText** (string, optional): The help text to be displayed as a tooltip for the parameter.
- **placeholder** (string, optional): The placeholder to be displayed in the parameter field, can be null.

RestNotificationTypeDescription

- **notificationType** (string, optional): The notification type, such as SMS.
- **pluginId** (string, optional): The ID of the notification plug-in.
- **maxMessageSize** (integer, optional): The maximum size of the notification message.
- **targetParameterList** (Array[RestNotificationTargetParameterDescription], optional): The notification target parameters, such as phone number for SMS.

RestPackageListEntry

- **id** (string, optional): The ID of the package.
- **name** (string, optional): The name of the package.
- **description** (string, optional): The description of the package.
- **owner** (string, optional): The owner ID of the package.
- **type** (string, optional): The type of the package. Possible values: PACKAGE, TEMPLATE
- **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)
- **state** (string, optional): The state of the package. Possible values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED
- **documentEntries** (Array[RestDocumentListEntry], optional): A list of all documents contained in the package (short info).
- **currentSigner** (string, optional): The ID of the signer which is currently locking 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)
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package.
- **enforceSigningMode** (boolean, optional): The flag to enforce the signing mode used to sign the first signature field by a signer of a specific document in a signing package.
- **signerEntries** (Array[RestSignerListEntry], optional): A list of all signers contained in the package.
- **RejectedDocuments** (Array[RestRejectedPackageDocument): The rejected documents of the signing package.

RestPageInfo

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

RestPersonalization

- **loginLogoUrl** (string, optional): The login logo URL.
- **loginLogoType** (string, optional): The login logo image type.
- **headerLogoUrl** (string, optional): The header logo URL.
- **headerLogoType** (string, optional): The login logo image type.
- **headerBackground** (string, optional): The header background color as RGB hex value, such as f1f1f1.
- **headerForeground** (string, optional): The header foreground color as RGB hex value.
- **footerBackground** (string, optional): The footer background color as RGB hex value.
- **footerForeground** (string, optional): The footer foreground color as RGB hex value.
- **headingsTitleForeground** (string, optional): The heading title foreground color as RGB hex value.

- **footerLinks** (Array[RestFooterLink], optional): A list with footer links.

RestPlainDocumentInfo

- **pageCount** (integer, optional)
- **signed** (boolean, optional)
- **pages** (number, required)
- **fields** (string, required)
- **isSigned** (boolean, optional)

RestPlainDocumentInput

- **documentBase64** (string, optional)

RestPlainDocumentOutput

- **pageCount** (integer, optional)
- **documentBase64** (string, optional)

RestPlainDocumentSigningInput

- **documentBase64** (string, optional)
- **signatureImage** (string, optional)

RestPublicKeyInfo

- **algorithm** (string, optional): The algorithm of the public key.
- **format** (string, optional): The format 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.
- **fingerprintSha256** (integer, optional): The SHA-256 fingerprint of the certificate.
- **fingerprintSha1** (string, optional): The SHA-1 fingerprint of the certificate.

RestQRCodeSpecifications

- **imageType** (string, required): Image type of the QR code. Valid values: JPG, GIF, PNG
- **width** (integer, required): The width of the QR code image (50 - 1000).
- **height** (integer, required): The height of the QR code image (50 - 1000).

RestReminderInput

RestReminderInput

- **id** (string, optional): The ID of the reminder.
- **type** (string, optional): The type of the reminder. Valid values: AFTER_SEND, BEFORE_EXPIRES
- **days** (integer, required): The days of the reminder.

RestReminderOutput

- **id** (string, optional): The ID of the reminder.
- **type** (string, optional): The type of the reminder. Valid values: AFTER_SEND, BEFORE_EXPIRES
- **days** (integer, required): The days of the reminder.
- **sendDate** (string, optional): Date when the reminder shall be sent. Format: date-time with a time-zone in UTC, such as '2020-12-03T10:15:30Z' (ISO-8601)

RestSignatureFieldInput

- **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!).
- **description** (string, optional): The description of the field.
- **signerId** (string, optional): The ID of the signer assigned to the field.
- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc).
- **required** (boolean, optional): Whether the field is required or not.
- **readOnly** (boolean, optional): Whether the field is readonly or not.
- **stage** (integer, optional): The stage of the signer allowed to sign the field.
- **widgets** (Array[RestWidget, required): A list of all widgets of 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 plug-in is registered to a signer and supports signing the signature field using a trusted service provider, 'TSP' signing mode is also added by default.
- **signingAppearanceOptions** (string, optional): A list of the signing appearances (currently respected by signingMode=PC (personal certificate)) 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 sign with (up-)loaded image 'STAMP' for sign with (up-)loaded stamp image. Default: HW, PH, C2S, IMG, STAMP and PLUGIN

RestSignatureFieldOutput

- **id** (string, optional): The ID of the field.
- **name** (string, optional): The name of the field.
- **description** (string, optional): The description of the field.
- **signerId** (string, optional): The ID of the signer assigned to the field.
- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc).
- **required** (boolean, optional): Whether the field is required or not.
- **readOnly** (boolean, optional): Whether the field is readonly or not.
- **stage** (integer, optional): The stage of the signer allowed to sign 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 plug-in is registered to a signer and supports signing the signature field using a trusted service provider, TSP signing mode is also added by default. Valid values: UNKNOWN, HW, PH, C2S, IMG, STAMP, TSP
- **signingAppearance** (string, optional): The signing appearance of the signature field if signed, can be HW for handwritten (Tablet or HTML5), PH for photo, captured by camera or C2S for Click2Sign, IMG for sign with (up-)loaded image, STAMP for sign with (up-)loaded stamp image.
- **signed** (boolean, required): Whether the signature field is signed or not.
- **widgets** (Array[RestWidget, required): A list of all widgets of 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 to sign with a trusted service provider. Default value: HW, PH, C2S, IMG, STAMP. If a TSP plug-in is registered to a signer and supports signing the signature field using a trusted service provider, TSP signing mode is also added by default.
- **signingAppearanceOptions** (string, optional): A list of the signing appearances (currently respected by signingMode=PC (personal certificate)) 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 sign with (up-)loaded image, STAMP for sign with (up-)loaded stamp image. Default: HW, PH, C2S, IMG, STAMP, PLUGIN

RestSignerInput

- **id** (string, optional): The ID of the signer.
- **name** (string, optional): The display name of the signer.
- **firstName** (string, optional): The given name of the signer.
- **lastName** (string, optional): The surname of the signer
- **email** (string, optional): The email 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.
- **role** (string, optional): The role of the signer. Valid values: SIGNER, REVIEWER
- **commentForDecline** (string, optional): The comment for decline of the signer.
- **authenticationMode** (string, optional): The authentication mode of the signer. Valid values: NONE, CODE, EXTAUTH
- **reasonForDecline** (string, optional): The reason for decline of the signer. Valid 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

- **accessCodeDeliveryPluginId** (string, optional): The ID of the plug-in for the access code delivery. An empty string in `RestSignerInput` deletes the plug-in ID entry and the related `RestAccessCodeDeliveryChannelParameter` entries. Note: The access code delivery plug-in can be used for sending an access code to a recipient for a two-factor authentication.
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed.
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed.
- **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).
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47) Note: The preferred signer language overwrites the account specific communication language. This setting determines the used language in the emails 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.
- **delegate** (boolean, optional): The flag to indicate if the signer can delegate the signing session to some other recipient.
- **authenticationParam** (string, optional): The authentication code for the signer.
- **tspSignerInfo** (RestTSPSignerInfoInput, optional): TSP-related info for the signer to sign documents using TSP (trusted service provider). With an appropriate TSP plug-in, it is possible to let the signer sign with a qualified certificate.
- **delegateMessage** (string, optional): The optional message from the original recipient to the delegated recipient.
- **authenticationProviders** (Array[RestAuthenticationProviderInput], optional): List of authentication providers for a signer.
- **accessCodeDeliveryParameters**(Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plug-in for the access code delivery. The parameters are only considered if the ID of the plug-in for the access code delivery is also provided.

RestSignerListEntry

- **id** (string, optional): The ID of the signer.
- **name** (string, optional): The name of the signer.
- **email** (string, required): The email of the signer.
- **order** (integer, optional): The order of the signer.
- **role** (string, optional): The role of the signer. Valid values: SIGNER, REVIEWER
- **state** (string, optional): The state of the signer. Valid values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE

- **authenticationMode** (string, optional): The ID authentication mode the signer: Valid values: NONE, CODE, EXTAUTH
- **url** (string, optional): The request URL for the entire signer info.
- **accessCodeDeliveryChannel** (string, optional): The channel name by which authentication code will be sent to the signer.
- **unreadDocuments** (Array[string], optional): The list of document IDs needed to be reviewed by the signer.
- **authenticationProviders** (Array[RestAuthenticationProviderOutput], optional): List of authentication providers for a signer.
- **externalAuthenticationUrl** (string, optional): The external authentication URL.
- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed.
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed.
- **relatedUser** (string, optional): The identifier of the related SignDoc user.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47).
- **signerColor** (string, optional): The display color assigned to the signer.
- **stage** (integer, optional): The stage of the signer.
- **delegate** (boolean, optional): The flag to indicate if the signer can delegate the signing session to some other recipient.
- **tspSignerInfoAvailable** (boolean, optional): Indicates if the signer has tsp signer info set.
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP.
- **RejectedDocuments** (Array[RestRejectedPackageDocument): The rejected documents by the signer

RestSignerOutput

- **id** (string, optional): The ID of the signer.
- **name** (string, optional): The display name of the signer.
- **firstName** (string, optional): The given name of the signer.
- **lastName** (string, optional): The surname of the signer.
- **email** (string, optional): The email of the signer.
- **order** (integer, optional): The order of the signer.
- **role** (string, optional): The role of the signer. Valid values: SIGNER, REVIEWER
- **commentForDecline** (string, optional): The comment for decline of the signer.
- **authenticationMode** (string, optional): The authentication mode of the signer. Valid values: NONE, CODE, EXTAUTH
- **reasonForDecline** (string, optional): The reason for decline of the signer. Valid values: R1, R2, R3, R4, R5
- **accessCodeDeliveryPluginId** (string, optional): The ID of the plug-in for the access code delivery. An empty string in `RestSignerInput` deletes the plug-in ID entry and the related `RestAccessCodeDeliveryChannelParameter` entries.

- **gdprConsentRequired** (boolean, optional): Whether the GDPR consent must be displayed and agreed.
- **esignConsentRequired** (boolean, optional): Whether the e-sign consent must be displayed and agreed.
- **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.
- **preferredLanguage** (string, optional): The preferred language of the signer (IETF BCP 47 language tag, see https://www.rfc-editor.org/info/bcp47).
- **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.
- **delegate** (boolean, optional): The flag to indicate if the signer can delegate the signing session to some other recipient
- **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)
- **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)
- **state** (string, optional): The state of the signer. Possible values: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **accessCodeDeliveryChannel** (string, optional): The channel type by which authentication code will be sent to the signer.
- **tspSignerInfo** (RestTSPSignerInfoOutput, optional): TSP-related info for the signer to sign documents using TSP.
- **externalAuthenticationUrl** (string, optional): The external authentication URL.
- **clientCertificateIV** (string, optional): The IV for encryption of a client side certification.
- **clientCertificatePubKeyPem** (string, optional): The public key (PEM format) for encryption of client side certificate (AES) key.
- **signerColor** (string, optional): The display color assigned to the signer.
- **authenticationProviders** (Array[RestAuthenticationProviderOutput], optional): List of authentication providers for a signer.
- **accessCodeDeliveryParameters** (Array[RestAcessCodeDeliveryChannelParameter], optional): The list of parameters of the plug-in for the access code delivery. The parameters are only considered if the ID of the plug-in for the access code delivery is also provided.
- **RejectedDocuments** (Array[RestRejectedPackageDocument): The rejected documents by the signer

RestSigningCertificateInfo

- **type** (string, optional): The type of the signing certificate.
- **version** (integer, optional): The version of the signing certificate.
- **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

- **validityDateNotAfter** (string, optional): The notAfter date from the validity period of the certificate. Format: date-time with a time-zone in UTC (ISO-8601)
- **validityDateNotBefore** (string, optional): The notBefore date from the validity period of the certificate. Format: date-time with a time-zone in UTC (ISO-8601)
- **fingerprintSha256** (string, optional): The SHA-256 fingerprint of the certificate
- **fingerprintSha1** (string, optional): The SHA-1 fingerprint of the certificate

RestSigningPackageEvent

- **eventId** (integer, optional): The signing package event ID. Defines the event, predefined numeric values for package states. Specifically, 'PREPARED'=1, 'COMPLETED'=2, 'REJECTED'=3, 'EXPIRED'=4, 'CANCELED'=5

  The states REJECTED and CANCELED of the packages correspond to the package states Declined and Voided displayed on the Statistics page of the Manage Client and Administration Center.

- **eventTime** (string, optional): The event time of the specific state of the signing package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the signing package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601)

RestSigningPackageInput

- **id** (string, optional): The ID of the package.
- **name** (string, optional): The name of the package.
- **custom** (string, optional): The custom field.
- **description** (string, optional): The description of the package.
- **type** (string, optional): The type of the package, can be PACKAGE (default) or TEMPLATE.
- **processingType** (string, optional): The processing type of the package. Valid values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default).
- **state** (string, optional): The state of the package. Valid 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).
- **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).
- **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).
- **auditTrailOptions** (integer, optional): The audit trail options of the final compound pdf document after package is completed. 0 - no audit trail is added, 1 - both document and signing package audit trail is added.
- **mailSubject** (string, optional): The mail subject of the package for signer notification.
- **mailMessage** (string, optional): The mail message of the package for signer notification.
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package.

- **enforceSigningMode** (boolean, optional): The flag to enforce the signing mode used to sign the first signature field by a signer of a specific document in a signing package.
- **documents** (Array[RestDocumentInput], optional): The list of all documents contained in the package.
- **signers** (Array[RestSignerInput], optional): The list of all package-specific signers.
- **reminders** (Array[RestReminderInput], optional): The list of package-specific email reminders for the 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. Valid values: UNKNOWN, HW, PH, C2S, IMG, STAMP, TSP, PC
- **signingAppearanceOptions** (Array[string], optional): The 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. Valid values: HW, PH, C2S, IMG, STAMP, PLUGIN

RestSigningPackageOutput

- **id** (string, optional): The ID of the package.
- **name** (string, optional): The name of the package.
- **custom** (string, optional): The custom field.
- **description** (string, optional): The description of the package.
- **type** (string, optional): The type of the package. Valid values: PACKAGE, TEMPLATE
- **processingType** (string, optional): The processing type of the package. Valid values: SEQ - sequential notification of signers, PAR - parallel notification of signers (default), STAGED - sequential and parallel notification of signers based on staged value.
- **state** (string, optional): The state of the package. Valid values: DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED
- **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)
- **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)
- **auditTrailOptions** (integer, optional): The audit trail options of the package.
- **mailSubject** (string, optional): The mail subject of the package.
- **mailMessage** (string, optional): The mail message of the package.
- **inPersonEnabled** (boolean, optional): The flag to enable the in-person signing of a signing package.
- **enforceSigningMode** (boolean, optional): The flag to enforce the signing mode used to sign the first signature field by a signer of a specific document in a signing package.
- **owner** (RestUserInfo, optional): The owner's information.
- **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)
- **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)

- **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)
- **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)
- **auditTrailUrl** (string, optional): The URL to retrieve audit trail for the package.
- **documentEntries** (Array[RestDocumentListEntry], optional): A list of all documents contained in the package (short info).
- **signerEntries** (Array[RestSignerListEntry], optional): A list of all signers contained in the package.
- **currentSigner** (RestSignerOutput, optional): The current signer of the package.
- **finalDocumentAvailable** (boolean, optional): Confirmation if final container document is available.
- **reminderEntries** (Array[RestReminderOutput], optional): A list of all reminders contained in the package.
- **finalDocumentAvailable** (boolean): The ID of the package.
- **RejectedDocuments** (Array[RestRejectedPackageDocument]): The rejected documents of the signing package

RestStatisticsList

- **restSigningPackageEvents** (Array[RestSigningPackageEvent], optional)

RestSupplementalDocumentInfo

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

RestTSPInfo

- **name** (string, optional)
- **validationText** (string, optional)
- **registrationURL** (string, optional)
- **signingText** (string, optional)
- **helpText** (string, optional)
- **validationFields** (Array[RestTSPValidationEntryField], optional)
- **url** (string, optional)

RestTSPSignatureDocument

- **documentId** (string, required): The ID of the document.
- **state** (string, required): The current state of the document in TSP signing process. Valid values: REQUIRED, DONE

RestTSPSignerInfoInput

- **tspPluginId** (string, required): The ID of the plug-in needed for an interaction with the required TSP.
- **tspSignatureType** (string, required): The type of signature which will be applied by the required TSP. Valid values: BASIC, ADVANCED, QUALIFIED
- **tspCredentials** (Array[RestEntry], required): The list of credentials needed for an authentication in the required TSP.
- **tspSignatureDocuments** (Array[string], optional): List of IDs of documents needed to be signed by the signer using the required TSP.

RestTSPSignerInfoOutput

- **tspPluginId** (string, required): The ID of the plug-in needed for an interaction with the required TSP.
- **tspSignatureType** (string, required): The type of signature which will be applied by the required TSP. Valid values: BASIC, ADVANCED, QUALIFIED
- **tspCredentials** (Array[RestEntry], required): The list of credentials needed for an authentication in the required TSP.
- **tspSignatureDocuments** (Array[RestTSPSignatureDocument], optional): List of documents needed to be signed by the signer using the required TSP.

RestTSPSigningInfo

- **name** (string, optional)
- **requestId** (string, optional)

RestTSPValidationEntryField

- **id** (string, optional)
- **label** (string, optional)
- **description** (string, optional)
- **type** (string, optional): Valid values: STRING, BOOLEAN
- **placeholder** (string, optional)
- **regex** (string, optional)

RestTeamInput

- **id** (string, optional): The ID of the team.
- **name** (string, optional): The name 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.

RestTeamListEntry

- **id** (string, optional): The ID of the team.

- **name** (string, optional): The name of the team.
- **url** (string, optional): The request URL to query the complete 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.

RestTeamOutput

- **id** (string, optional): The ID of the team.
- **name** (string, optional): The name of the team.
- **managers** (Array[RestUserListEntry], optional): A list of all team managers.
- **members** (Array[RestUserListEntry], optional): A list of all team members.
- **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)
- **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)
- **url** (string, optional): The URL to query the team.

RestTextFieldInput

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

RestTextFieldOutput

- **id** (string, optional): The ID of the field.
- **name** (string, optional): The name of the field.
- **description** (string, optional): The description of the field.
- **signerId** (string, optional): The ID of the signer assigned to the field.
- **alternateName** (string, optional): The alternate name of the field (used as label value in SignDoc).
- **required** (boolean, optional): Whether the field is required or not.
- **readOnly** (boolean, optional): Whether the field is readonly or not.

- **stage** (integer, optional): The stage of the signer allowed to sign the field.
- **value** (string, optional): The value of the field.
- **multiLine** (boolean, optional): Whether the field allows multiple lines or not.
- **maxLength** (integer, optional): The max length of the field.
- **widgets** (Array[RestWidget, required): A list of all widgets of the field.

RestTotalAgilityLicensingServerError

- **statusCode** (string, optional): Status code of the error from the TotalAgility Licensing server.
- **errorMessage** (string, optional): Error message from the TotalAgility Licensing server.

RestUserInfo

- **id** (string, optional): The ID of the user.
- **company** (string, optional): The company name of the user.
- **name** (string, optional): The name of the user.
- **email** (string, optional): The email of the user.

RestUserInput

- **id** (string, optional): The ID of the user.
- **name** (string, optional): The name 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.
- **phone** (string, optional): The phone of the user.
- **state** (string, optional): The state of the user. Valid values: ACTIVE, SUSPENDED, INVITED
- **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.
- **roles** (Array[string], optional): A list of all roles the user possesses.
- **currentContext** (string, optional): The current context 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.

RestUserListEntry

- **id** (string, optional): The ID of the user.
- **name** (string, optional): The name of the user.
- **email** (string, optional): The email of the user.
- **state** (string, optional): The state of the user. Valid values: ACTIVE, SUSPENDED, INVITED
- **teamState** (string, optional): The state of the user in a team. Valid values: ACTIVE, SUSPENDED, INVITED

- **url** (string, optional): The request URL to query the complete user.
- **roles** (Array[string], optional): A list of all roles the user possesses.
- **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)

RestUserOutput

- **id** (string, optional): The ID of the user.
- **name** (string, optional): The name 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.
- **phone** (string, optional): The phone of the user.
- **state** (string, optional): The state of the user. Valid values: ACTIVE, SUSPENDED, INVITED
- **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.
- **roles** (Array[string], optional): A list of all roles the user possesses.
- **currentContext** (string, optional): The current context of the user.
- **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.
- **accessTokenExpires** (string, optional): The expiration date of the access token of the user. Format: date-time with a time-zone in UTC, such as '2021-12-03T10:15:30Z' (ISO-8601)
- **lastSignInTime** (string, optional): The last sign in time of the user. Format: date-time with a time-zone in UTC, such as '2021-12-03T10:15:30Z' (ISO-8601)
- **lastSignOutTime** (string, optional): The last sign out time of the user. Format: date-time with a time-zone in UTC, such as '2021-12-03T10:15:30Z' (ISO-8601)
- **lastUpdateTime** (string, optional): The last update time of the user. Format: date-time with a time-zone in UTC, such as '2021-12-03T10:15:30Z' (ISO-8601)
- **creationTime** (string, optional): The creation time of the user. Format: date-time with a time-zone in UTC, such as '2021-12-03T10:15:30Z' (ISO-8601)
- **url** (string, optional): The URL to query the user.

RestUserSettings

- **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.
- **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.

RestWidget

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

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 (such as radio buttons). Currently only one widget is supported in SignDoc.

Chapter 4

Webhooks

Webhooks provide a simple option to integrate the SignDoc workflow with other systems without having to implement extension plug-ins that must be installed locally.

Whenever a supported event occurs, such as the state of a signing package changed, SignDoc can call an external URL (using HTTP POST) that gets called by SignDoc, with context information about the event, and handles the data, such as archiving documents. The application that processes the callback is called webhook handler further on.

Webhook events can be grouped by type. Currently these webhook types are supported:

- **State change:** The event of this type is fired whenever a state change of a supported entity occurs.
- **TSP:** This event is fired whenever SignDoc processes a TSP enabled signing package.

Webhook handler requirements

The only requirement to the webhook handler is to accept HTTP POST requests. Additionally, the handler should be able to read not only the request body but also the request headers. The implementation language does not matter.

Sample code

Sample code for a webhook receiver can be found in the directory:

```
signdoc_home/interfaces/webhooks
```

Supported events

Mandatory for all webhooks

- Health check: The webhook URL must return status code 200 so that SignDoc considers the webhook healthy.
  Event: `health-check`

State change type

- Package state change: A package's state changed.
  Event: `package-state-change`

Example: A signing package is completed.

- Signer state change: A signer's state changed.
  Event: `signer-state-change`
  Example: A signer finished a signing session.

TSP type

- Validate signer credentials
  Event: `tsp-is-valid-signer`
- Get TSP information
  Event: `tsp-info`
- Send document to TSP for signing
  Event: `tsp-push-document`
- Retrieve signed document
  Event: `tsp-get-signed-document`

Configuration options

Configuration options are generally account specific and each webhook type has its own URL setting and can be turned on/off individually.

The webhook related settings can be found in the **General** and **Type** tabs.

General

- **webhook.general.enabled** (boolean): Enable or disable webhook callbacks completely for an account. Default: off/false
- **webhook.general.event.post.blobs** (boolean): If set to 'on', the webhook request payload will also contain large data such as PDF documents. Default: off/false
- **webhook.general.event.post.auth** (boolean): If set to 'on', the webhook request headers will, when applicable for the event, contain a valid X-AUTH-TOKEN of the signing package owner that can be used for follow-up interaction with the REST API. Default: on/true

Types

- **webhook.type.<webhook_type>.url** (string): The webhook URL that consumes the webhook events of the specified type. If not set, webhooks will not be processed. Default: blank
  Example: `webhook.type.state_change.url`
- **webhook.type.<webhook_type>.event.enabled:** (boolean): Defines if the events associated with this type should be processed. Default: off/false
  Example: `webhook.type.state_change.enabled`

Webhook request format

Common HTTP headers

These HTTP headers are available with all webhook requests.

- **signdoc-base-url:** The SignDoc base URL which is also named CIRRUS_URL.
  Example: `http://myserver/cirrus`
- **signdoc-version:** The SignDoc version.
  Example: 3.3.0.0.0.1
- **signdoc-webhook-event:** The event causing the webhook request. Defines also the request body.
  Example: `package-state-change`

HTTP body

The HTTP body is sent as JSON object.

The data is very similar but not equivalent to the REST API data structures. For detailed meaning of the attributes consult the corresponding REST API or the SignDoc OpenAPI documentation.

State change webhook type

Using the state change webhook type enables customers to react on state changes of signer and signing package objects without having to implement a SignDoc plug-in that must run locally.

This protocol is uni-directional, that is, the webhooks only consume data and do not return data in the HTTP response except for the status code.

State change events

The protocol that SignDoc uses to interact with a TSP consists of 2 events:

Event `signer-state-change`: Checking the signers credentials

Event `package-state-change`: Providing information of the TSP that is presented to the signer in the SignDoc Signing Client

Event signer-state-change

This topic describes the `signer-state-change` webhook event.

HTTP headers:

- **x-auth-token:** Contains a valid x-auth-token of the package owner that can be used to authenticate against the SignDoc REST API.
- **signdoc-old-state:** The previous state of the signing package. One of: null, ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE

JSON body structure:

These attributes identify a signer and its state before state change.

- **account_id** (string): The account ID the signing package belongs to.
- **signingpackage_id** (string): The signing package ID.

- **id** (string): The signer ID.

Further interesting signer attributes (can also be retrieved via REST API):

- **name** (string): The signer's name.
- **email** (string): The signer's email address.
- **role** (string): The signer role. One of: SIGNER, REVIEWER
- **state** (string): The signer state. One of: null, ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **order** (integer): Numerical order of the signer.
- **stage** (integer): Numerical order of the stage.
- **authenticationMode** (string): Authentication mode of the signer. One of: NONE, CODE, EXTAUTH

Example of a `signer-state-change` webhook request:

Headers

```
"signdoc_base_url" : "http://localhost:6611/cirrus",
"signdoc_old_state" : "ASSIGNED",
"signdoc_version" : "3.3.0.0.0.1",
"signdoc_webhook_event" : "signer-state-change",
"x-auth-token" : "eyJleHAiOjE2NjYxj...6W9990DpHXBsjTXYj8Br1YUdiI="
```

Body

```
{
    "id": "signer-1",
    "state": "INFORMED",
    "signingPackageId": "9544b5fd-00bb-406a-90db-feb7abeeae77",
    "name": "Important Signer",
    "email": "signer@example.com",
    "accountId": 51,
    "role": "SIGNER",
    "order": 1,
    "stage": 1,
    "authenticationMode": "NONE",
    "authenticationProviders": [],
    "delegate": true
}
```

Event package-state-change

This topic describes the `package-state-change` webhook event.

HTTP headers:

- **x-auth-token:** Contains a valid x-auth-token of the package owner that can be used to authenticate against the SignDoc REST API.
- **signdoc-old-state:** The previous state of the signing package. One of: null, DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED

JSON body structure:

These attributes identify a signing package and its state before state change.

- **account_id** (string): The account ID the signing package belongs to.

- **id** (string): The signing package ID.

Further interesting signing package attributes (can also be retrieved via REST API):

- **custom** (string): A custom attribute that was set when creating the signing package via REST API.
- **state** (string): The current package state. One of: null, DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED
- **name** (string): The name of the signing package.
- **type** (string): The signing package type. One of: PACKAGE, TEMPLATE
- **documents** [Array(string)]: Array of document objects of the signing package (see below). Usually, this substructure contains a link to the document so it can be downloaded, but the webhook can also be configured to send the BLOBs in the JSON structure as Base64.
- **signers** [Array(string)]: Array of signer objects belonging to the signing package (see below).
- **currentSigner** [Array(string)]: This is an optional structure that is only available while there is an active signing session.
- **processingType** (string): The processing type of the package. One of: SEQ, PAR, STAGED
- **lastUpdateTime** (string): The last modified timestamp [ms since the epoch].

Substructure of `documents`:

- **id** (string): The document ID.
- **name** (string): The document name.
- **documentUrl** (string): The URL that can be used to download the document (using the x-auth-token for authentication).
- **document** (string): This attribute that contains the document in Base64 format is optional and only available if `webhook.event.post.blobs` is true.
- **order** (integer): Numerical order of the document.
- **fileName** (string): Original file name of the document.

Substructure of `signers` and `CurrentSigner`:

- **id** (string): The signer ID.
- **name** (string): The signer's name.
- **email** (string): The signer's email address.
- **role** (string): The signer role. One of: SIGNER, REVIEWER
- **state** (string): The signer state. One of: ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR, STAGE_COMPLETE
- **order** (integer): Numerical order of the signer.
- **stage** (integer): Numerical order of the stage.
- **authenticationMode** (string): Authentication mode of the signer. One of: NONE, CODE, EXTAUTH

Example of a `package-state-change` webhook request:

Headers

```
"signdoc_base_url" : "http://localhost:6611/cirrus",
"signdoc_old_state" : "DRAFT",
"signdoc_version" : "3.3.0.0.0.1",
"signdoc_webhook_event" : "package-state-change",
```

```
"x-auth-token" : "eyJleHAiOjE2NjYx...SDsFCNRig="
```

Body


```
{
    "id": "9544b5fd-00bb-406a-90db-feb7abeeae77",
    "state": "PREPARED",
    "account_id": "signdoc",
    "name": "Hello Template without TSP",
    "owner": "chi1",
    "custom": null,
    "description": "A simple template",
    "type": "PACKAGE",
    "documents": [
        {
            "id": "document-1",
            "name": "Main Doc",
            "documentUrl": "http://localhost:6611/cirrus/rest/v8/
packages/9544b5fd-00bb-406a-90db-feb7abeeae77/documents/document-1",
            "order": 1,
            "fileName": "Hello.pdf"
        }
    ],
    "signers": [
        {
            "id": "signer-1",
            "name": "Important Signer",
            "email": "signer@example.com",
            "role": "SIGNER",
            "state": "ASSIGNED",
            "order": 1,
            "stage": 1,
            "authenticationMode": "NONE",
            "authenticationProviders": [],
            "delegate": true
        }
    ],
    "currentSigner": null,
    "containerDocumentAvailable": false,
    "containerDocumentUrl": "http://localhost:6611/cirrus/rest/v8/
packages/9544b5fd-00bb-406a-90db-feb7abeeae77/finaldocument",
    "processingType": "PAR",
    "startDate": 1666124665432,
    "auditTrailOptions": 1,
    "inPersonEnabled": true,
    "lastUpdateTime": 1666124665499,
    "creationTime": 1666124661897
}
```


TSP webhook type

Using the TSP webhook type enables customers to integrate with external TSP providers without having to implement a SignDoc plug-in that must run locally. Technically the core plug-in "SignDoc TSP webhook plug-in" is used to handle the communication with the webhook implementation.

TSP protocol

The protocol that SignDoc uses to interact with a TSP consists of 4 stages. All stages must be passed for a signer of a signing package to complete the SignDoc TSP signing process. This protocol is bi-directional, that is, the webhooks return data in the HTTP response that is consumed by SignDoc.

- Event `tsp-is-valid-signer`: Checking the signers credentials.
- Event `tsp-info`: Providing information of the TSP that is presented to the signer in the SignDoc Signing Client.
- Event `tsp-push-document`: Sending the document to sign to the TSP and starting the TSP signing session via redirect.
- Event `tsp-get-signed-document`: Returning the signed document to SignDoc.

Event tsp-is-valid-signer

This topic describes the `tsp-is-valid-signer` webhook event.

Request

HTTP headers:

- no additional headers

JSON body structure:

```
{
"credentials": {},
"signatureType": ""
}
```

Attributes:

- **credentials** (dictionary): An optional key/value map containing signer associated credentials needed for the TSP.
- **signaturetype** (string): One of these values: BASIC, ADVANCED, QUALIFIED

Example:

```
{
"credentials": {"username":"sam", "password":"sample"},
"signatureType": "ADVANCED"
}
```

Response

HTTP headers:

- no additional headers

JSON body structure:

```
{
  "valid": true|false
}
```

Attributes:

- **valid** (boolean): If true, the signer is acceptable, if not the signer is rejected and the TSP process cannot continue.

Example:

```
{
  "valid": true
}
```

Event tsp-info

This topic describes the `tsp-info` webhook event.

Request

HTTP headers:

- no additional headers

JSON body structure:

```
{
"locale": ""
}
```

Attributes:

- **locale** (string): The current locale of the signer

Example:

```
{
"locale": "en"
}
```

Response

HTTP headers:

- no additional headers

JSON body structure:

```
{
  "provider_info": {
    "help_text": "",
    "signing_text": ""
  },
  "tsp_name": ""
}
```

Attributes:

- **provider_info** (object): Contains information to be displayed in the Signing Client.
- **provider_info.help_text** (string): Text that is displayed when clicking the help icon.

- **provider_info.signing_text** (string): Text that is displayed in the confirmation dialog box before the TSP signing process is started.
- **tsp_name** (string): The name of the TSP webhook.

Example:

```
{
  "provider_info": {
    "help_text": "Here is a detailed help text",
    "signing_text": "A simple TSP webhook implementation"
  },
  "tsp_name": "TSP Sample Webhook"
}
```

Event tsp-push-document

This topic describes the `tsp-push-document` webhook event.

Request

HTTP headers:

- **x-auth-token**: Contains a valid x-auth-token of the package owner that can be used to authenticate against the SignDoc REST API.

JSON body structure:

```
{
    "signer_oid": "",
    "signing_package_oid": "",
    "document": "JVBERi0....MTUKJSVFT0YNCg==",
    "redirect_url_ok": "",
    "redirect_url_cancel": "",
    "redirect_url_error": "",
    "signature_field_name": "",
    "credentials": {},
    "signature_type": "",
    "signer_name": "",
    "signer_email": "",
    "signer_first_name": "",
    "signer_last_name": "",
    "signer_preferred_language": "",
    "auth_token_expiry": "",
    "document_name": "",
    "document_page_count": "",
    "document_filename": "",
    "document_description": "",
    "signing_package_description": "",
    "signing_package_name": "",
    "signing_package_owner_email": "",
    "signing_package_owner_name": "",
    "locale": "en"
}
```

Attributes:

- **signer_oid** (string): The signer's OID.
- **signing_package_oid** (string):The signing packages's OID.
- **document** (Base64 encoded string): The document to sign.

- **redirect_url_ok** (string): The redirect URL to use when the TSP signing process was successful.
- **redirect_url_cancel** (string): The redirect URL to use when the TSP signing process was cancelled by the user.
- **redirect_url_error** (string): The redirect URL to use when the TSP signing process was erroneous.
- **signature_field_name** (string): The name of the signature field to be signed. Can be null if the complete document should be signed without a signature field.
- **credentials** (object): Optional TSP credentials. See also Event `tsp-is-valid-signer`.
- **signature_type** (string): The type of the signature. One of: BASIC, ADVANCED, QUALIFIED
- **signer_name** (string): The full name of the signer.
- **signer_email** (string): The email of the signer.
- **signer_first_name** (string): The first name of the signer (can be null).
- **signer_last_name** (string): The last name of the signer (can be null).
- **signer_preferred_language** (string): The signer's preferred language (can be null).
- **auth_token_expiry** (string): The expiry date/time of the current signing session in epoch milliseconds. The TSP signing session must be finished before.
- **document_name** (string): The name of the document.
- **document_page_count** (string): The page count of the document.
- **document_filename** (string): The file name of the document (can be null).
- **document_description** (string): Full description of the document (can be null).
- **signing_package_description** (string): Full description of the signing package (can be null).
- **signing_package_name** (string): Name/Title of the signing package.
- **signing_package_owner_email** (string): Owner of the signing package.
- **signing_package_owner_name** (string): Email address of the signing package's owner.
- **locale** (string): The current locale used in the signing client.

Example:

```
{
   "signer_oid": "daacf32c-1286-4549-af19-a570e300b9ef",
   "signing_package_oid": "a8d5d15f-df6b-457d-9dd8-22316a9338c4",
   "document": "JVBERi0 ... MTUKJSVFT0YNCg==",
   "redirect_url_ok": "http://localhost:6611/cirrus/rest/v8/tsp/result/TSPO/6f5 ...
42524",
   "redirect_url_cancel": "http://localhost:6611/cirrus/rest/v8/tsp/result/TSPC/6 ...
24",
   "redirect_url_error": "http://localhost:6611/cirrus/rest/v8/tsp/result/TSPE/6f ...
2524",
   "signature_field_name": "signature-1",
   "credentials": {"credentials": {"username":"sam", "password":"sample"},
"signatureType":"ADVANCED"},
   "signature_type": "QUALIFIED",
   "signer_name": "Christoph Hipp 1",
   "signer_email": "chi@sdlabs.de",
   "signer_first_name": null,
   "signer_last_name": null,
   "signer_preferred_language": null,
   "auth_token_expiry": "1670576762504",
   "document_name": "Main contract",
   "document_page_count": "1",
   "document_filename": "contract.pdf",
   "document_description": "",
```

```
    "signing_package_description": null,
    "signing_package_name": "template 4 - tsp field and document",
    "signing_package_owner_email": "sam@example.com",
    "signing_package_owner_name": "Sam Sample",
    "locale": "en"
}
```

Response

HTTP headers:

- no additional headers

JSON body structure:

```
{
    "auth_token": "",
    "document_verification_url": "",
    "signature_process_token": ""
}
```

Attributes:

- **auth_token** (string): Optional token that is stored with the signing package and send in the request of tsp-get-signed-document event.
- **document_verification_url** (string): The redirect URL that the signing client will use to redirect the signer to the TSP's web application. If the TSP does not have an interactive web application, either `redirect_url_ok`, `redirect_url_cancel` or `redirect_url_error` URL must be used after signing the document.
- **signature_process_token** (string): A unique ID that identifies this TSP signing session. This token is stored with the signing package and sent in the request of `tsp-get-signed-document` event.

Example:

```
{
    "auth_token": "",
    "document_verification_url": "http://localhost:6611/cirrus/rest/v8/tsp/result/
TSPO/6f51 ... 2524",
    "signature_process_token": "b02f8e55-e90d-4ad6-8b0f-680685900406"
}
```

Event tsp-get-signed-document

This topic describes the `tsp-get-signed-document` webhook event.

Request

HTTP headers:

- **x-auth-token**: Contains a valid x-auth-token of the package owner that can be used to authenticate against the SignDoc REST API.

JSON body structure:

```
{
"signature_process_token": "",
"auth_token": ""
}
```

Attributes:

- **signature_process_token** (string): The token (returned in `tsp-push-document` event response) for the document to retrieve.
- **auth_token** (string): An additional access token (returned in `tsp-push-document` event response). Can be null.

Example:

```
{
"signature_process_token": "0275bd97-fade-4fd0-91e3-19aebc794632",
"auth_token": null
}
```

Response

HTTP headers:

- no additional headers

JSON body structure:

```
{
  "audit_trail_entry": "My custom Audit Trail line",
  "document": "JVBERi ... LRg0K"
}
```

Attributes:

- **audit_trail_entry** (string): A custom audit trail entry that is added to the signing packages's audit trail.
- **document** (Base64 string): The signed document in Base64 format.

Example:

```
{
  "audit_trail_entry": "My custom Audit Trail line",
  "document": "JVBERi ... LRg0K"
}
```

Chapter 5

Support GDPR

This chapter explains methods to access personal data according to the EU Regulation 2016/679 (General Data Protection Regulation, GDPR), including support for the right to erasure ("right to be forgotten").

When SignDoc Standard is integrated in other business process management (BPM) systems, personal data is usually provided by them. By itself, SignDoc Standard does not transfer personal data to other entities, countries, or international organizations. Transferring finalized documents and audit trails is in the responsibility of the system or account owner.

Acknowledge personal data collection

Server administrator

The *Tungsten SignDoc Standard Administration Center Help* informs about where personal data is collected from the data subject.

Account administrator, team manager, SignDoc user

The *Tungsten SignDoc Standard Help* informs about where personal data is collected from the data subject.

Signer or reviewer

The e-sign consent form allows customers to provide information about which personal data is used, processed, or collected during the signing process. When they agree with the information, the signing process will continue. Signers and reviewers also have 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 signing package. In the audit trail all actions of SignDoc users, signers and reviewers performed for this specific signing package are logged.

Provide information about personal data (right of access by the data subject)

Server administrator

The 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

The 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.

The 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

The 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.

The SignDoc REST API call

```
GET /rest/v8/signers/{email}
```

returns a signer based on the email.

The 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

The SignDoc REST API call

```
PUT /rest/v8/users/serveradmins/{userid}
```

updates an existing server administrator.

Account administrator, team manager, SignDoc user

The 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

The 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 signing package are removed by default (configurable).